

Microsoft Official Course



AZ-301T04

Designing an Infrastructure Strategy

AZ-301T04

Designing an Infrastructure Strategy

MCT USE ONLY. STUDENT USE PROHIBITED



Contents

■	Module 0 Welcome	1
	Start Here	1
■	Module 1 Module Application Architecture Patterns in Azure	3
	Design Pattern Resources	3
	Performance Patterns	7
	Resiliency Patterns	9
	Scalability Patterns	11
	Data Patterns	14
	Review Questions	17
■	Module 2 Module Building Azure IaaS-Based Server Application	19
	High Availability	19
	Templated Infrastructure	23
	Domain-Joined Virtual Machines	28
	Online Lab - Building Azure IaaS-Based Server Applications by using Azure ARM Templates and Azure Building Blocks	33
	Review Questions	50
■	Module 3 Module Networking Azure Application Components	51
	Virtual Networks	51
	Load Balancing	55
	External Connectivity	61
	Secure Connectivity	64
	Design - Networking Case Study	66
	Online Lab - Deploying Network Infrastructure for Use in Azure Solutions	73
	Review Questions	84
■	Module 4 Module Integrating Azure Solution Components Using Messaging Services	85
	Event Messaging	85
	Integration	91
	Internet of Things (IoT)	94
	Online Lab - Deploying Messaging Components to Facilitate Communication Between Azure Resources	99
	Review Questions	106



Module 0 Welcome

Start Here

Welcome to Designing an Infrastructure Strategy

Welcome to *Designing an Infrastructure Strategy*. This course is part of a series of four courses to help students prepare for Microsoft's Azure Solutions Architect technical certification exam AZ-301: Microsoft Azure Architect Design. These courses are designed for IT professionals and developers with experience and knowledge across various aspects of IT operations, including networking, virtualization, identity, security, business continuity, disaster recovery, data management, budgeting, and governance.

This course contains the following four modules:

Module 1 - Application Architecture Patterns in Azure

This module introduces, and reviews common Azure patterns and architectures as prescribed by the Microsoft Patterns & Practices team. Each pattern is grouped into performance, resiliency, and scalability categories and described in the context of similar patterns within the category.

After completing this module, students will be able to:

- Locate and reference the Cloud Design Patterns documentation.
- Locate and reference the Azure Architecture Center.
- Describe various patterns pulled from the Cloud Design Patterns.

Module 2 - Building Azure IaaS-Based Server Applications (ADSK)

This module identifies workloads that are ideally deployed using Infrastructure-as-a-Service services in Azure. The module focuses on the VM Scale Sets and Virtual Machine services in Azure and how to best deploy workloads to these services using best practices and features such as Availability Sets.

After completing this module, students will be able to:

- Design an availability set for one or more virtual machines.
- Describe the differences between fault and update domains.

- Author a VM Scale Set ARM template.
- Join a virtualized machine to a domain either in Azure or on a hybrid network.

Module 3 - Networking Azure Application Components

This module describes the various networking and connectivity options available for solutions deployed on Azure. The module explores connectivity options ranging from ad-hoc connections to long-term hybrid connectivity scenarios. The module also discusses some of the performance and security concerns related to balancing workloads across multiple compute instances, connecting on-premise infrastructure to the cloud and creating gateways for on-premise data.

After completing this module, students will be able to:

- Describe DNS and IP strategies for VNets in Azure.
- Compare connectivity options for ad-hoc and hybrid connectivity.
- Distribute network traffic across multiple loads using load balancers.
- Design a hybrid connectivity scenario between cloud and on-premise.

Module 4 - Integrating Azure Solution Components Using Messaging Services

This module describes and compares the integration and messaging services available for solutions hosted on the Azure platform. Messaging services described include Azure Storage Queues, Service Bus Queues, Service Bus Relay, IoT Hubs, Event Hubs, and Notification Hubs. Integration services include Azure Functions and Logic Apps.

After completing this module, students will be able to:

- Compare Storage Queues to Service Bus Queues.
- Identify when to use Azure Functions or Logic Apps for integration components in a solution.
- Describe the differences between IoT Hubs, Event Hubs and Time Series Insights.

Prerequisites

This course requires that students have the following knowledge and skills:

- Create resources and resource group in Azure.
- Manage users, groups, and subscriptions in an Azure Active Directory instance.
- Build an Azure Virtual Machine with related resources.
- Manage containers and blobs stored in an Azure Storage account.
- Create App Service Plans and manage apps related to the plan.
- Configure an Azure Virtual Network and enable S2S and P2S connectivity.
- Protect networked application components using Network Security Groups.
- Automate everyday Azure resource tasks using Azure CLI or Azure PowerShell.
- Deploy an Azure SQL, MySQL, Postgres or Cosmos database instance.
- Monitor existing Azure solutions using built-in metrics, Application Insights, or Operational Insights.



Module 1 Module Application Architecture Patterns in Azure

Design Pattern Resources

Why Patterns?

This module introduces and reviews common Azure patterns and architectures as prescribed by the Microsoft Patterns & Practices team. Each pattern is grouped into performance, resiliency, and scalability categories and described in the context of similar patterns within the category.

Objectives

After completing this module, students will be able to:

- Locate and reference the Cloud Design Patterns documentation.
- Locate and reference the Azure Architecture Center.
- Describe various patterns pulled from the Cloud Design Patterns.

This lesson introduces the resources available on Microsoft's websites that deal with the architecture and design of solutions hosted on the Azure platform.

Many common problem areas in Computer Science have been explored and experienced by a wide variety of professionals. As with any discipline, best practices have been conceptualized, proven and prescribed as the most effective or efficient ways to solve common problems. Many professionals rely on these best practices so that they can work more efficiently and focus on obstacles unique to their actual problem space. In software engineering, these best practices are commonly referred to as design patterns.

In software engineering, a design pattern is a general reusable solution to a commonly occurring problem within a given context in software design. A design pattern is not a finished design that can be transformed directly into source or machine code. It is [instead] a description or template for how to solve a problem that can be used in many different situations.

"Software Design Pattern." Wikipedia.

There are many different patterns that already exist in the industry. In this course, we will focus on patterns that are Domain-specific to the cloud. While examples of these patterns may be implemented in Azure or using the .NET framework, most cloud patterns are language-agnostic in their design and can be implemented universally across a wide variety of cloud providers, programming frameworks or operating systems.

Microsoft Patterns & Practices

Microsoft Patterns & Practices is a long-standing group at Microsoft that collects, authors and shares best practice documentation for the benefit of the community. The team's intention is to provide anyone a stable technical foundation to start from when building elegant solutions built in any language, hosted on any platform.

Most solutions from patterns & practices incorporates many different Microsoft products and technologies as part of the overall solution. The team has been responsible for well-known solution libraries like Prism, Unity and Enterprise Library.

Cloud Design Patterns

The patterns & practices team at Microsoft has collected twenty-four design patterns that are relevant when designing the architecture of a cloud application. Each pattern includes a brief discussion of the benefits, considerations and implementation of each pattern. The collection of patterns is not meant to be comprehensive and is instead focused on the most popular design patterns for cloud applications.

The guide consists of a collection of web pages each individually focused on a pattern. The pages are broken down into sections describing the problem domain, a high-level technical solution, how these patterns solve a problem, considerations when using the pattern, an example implementing the pattern and when the pattern is suitable.

The patterns included in this guide can be used in many ways including:

- Implement the sample code as a starting point for your own cloud application.
- Use the Context, Problem and Solution sections of a pattern's web page as discussion points during an architectural design session.
- Use the Example section of a pattern's web page to teach colleagues about a design pattern.
- Use the Issues and Considerations section of a pattern's web page to identify common issues in your problem space.

The remainder of this module will focus on a subset of patterns from the Cloud Design Patterns documentation and explain why they are important to understand and how they relate to the decisions you will make as an Azure architect.

Microsoft Patterns & Practices shares much of their documentation, projects and findings today on **GitHub**:

<https://github.com/mspnp>

For example, the Microservices Reference Implementation shares best practices when designing a microservices solution running on Azure using Kubernetes:

<https://github.com/mspnp/microservices-reference-implementation>












Azure Architecture Center

The Azure Architecture Center is a landing page that contains links to documentation written by Microsoft patterns & practices, the Azure product groups and Azure subject-matter experts on architecting solutions for the Azure platform.

The center contains links to resources such as:

- Reference Azure Architectures
- Cloud Design Patterns
- Best Practices for Cloud Applications
- Azure Building Blocks
- Running SharePoint Server on Azure Guidance
- Performance Antipatterns
- Azure and SQL Server Customer Advisory
- Identity Management for Multitenant Applications
- Azure for AWS Professionals
- Microservice Guidance for Azure using Kubernetes and Azure Container Service

Azure Architecture Center

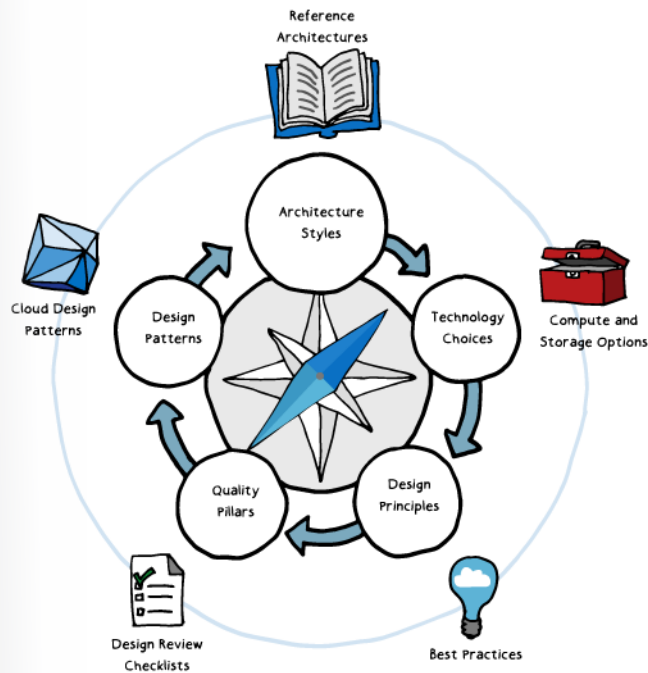
 <p>Azure Application Architecture Guide A guide to designing scalable, resilient, and highly available applications, based on proven practices that we have learned from customer engagements.</p>	 <p>Reference Architectures A set of recommended architectures for Azure. Each architecture includes best practices, prescriptive steps, and a deployable solution.</p>	 <p>Cloud Design Patterns Design patterns for developers and solution architects. Each pattern describes a problem, a pattern that addresses the problem, and an example based on Azure.</p>
 <p>Build Microservices on Azure This design guide takes you through the process of designing and building a microservices architecture on Azure. A reference implementation is included.</p>	 <p>Azure Data Architecture Guide A structured approach to designing data-centric solutions on Microsoft Azure.</p>	
 <p>Cloud Best Practices Best practices for cloud applications, covering aspects such as auto-scaling, caching, data partitioning, API design, and others.</p>	 <p>Design for Resiliency Learn how to design resilient applications for Azure.</p>	
 <p>Azure Building Blocks Simplify deployment of Azure resources. With a single settings file, deploy complex architectures in Azure.</p>	 <p>Design Review Checklists Checklists to assist developers and solution architects during the design process.</p>	
 <p>Azure Virtual Datacenter When deploying enterprise workloads to the cloud, organizations must balance governance with developer agility. Azure Virtual Datacenter provides models to achieve this balance with an emphasis on governance.</p>	 <p>Azure for AWS Professionals Leverage your AWS experiences in Microsoft Azure.</p>	

Azure Architecture Center: <https://docs.microsoft.com/en-us/azure/architecture/>

Architecture Center Guide

This guide presents a structured approach for designing applications on Azure that are scalable, resilient, and highly available. It is based on proven practices that we have learned from customer engagements.

The guide is intended for application architects, developers, and operations teams. It's not a how-to guide for using individual Azure services. After reading this guide, you will understand the architectural patterns and best practices to apply when building on the Azure cloud platform. You can also download an **e-book version of the guide**¹.



Architecture Center Guide: <https://docs.microsoft.com/azure/architecture/guide/>

¹ <https://azure.microsoft.com/campaigns/cloud-application-architecture-guide/>

Performance Patterns

Stateless Applications

Partitioning Workloads

This lesson introduces patterns related to the performance of applications and workloads on Azure.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe the Valet Key pattern.
- Describe the Command-Query Responsibility Segregation pattern.
- Describe the Throttling pattern.

A modular application is divided into functional units, also referred to as modules, which can be integrated into a larger application. Each module handles a portion of the application's overall functionality and represents a set of related concerns. Modular applications make it easier to design both current and future iterations of your application. Existing modules can be extended, revised or replaced to iterate changes to your full application.

Modules can also be tested, distributed and otherwise verified in isolation. Modular design benefits are well understood by many developers and architects in the software industry.

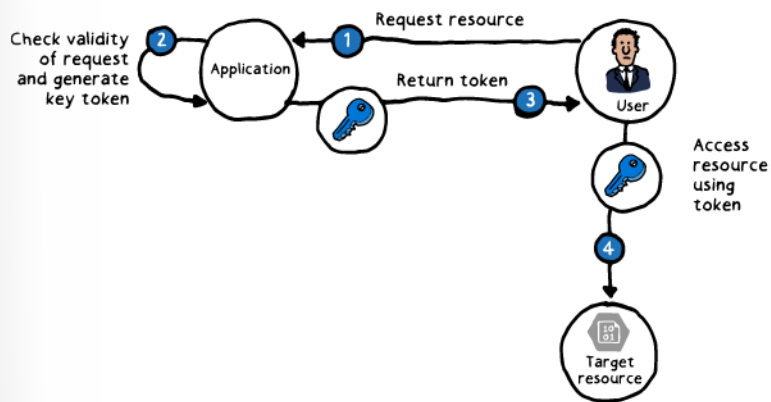


Note: What are some of the ideas that you have for modular and scalable applications? These ideas do not necessarily have to be web applications.

The Valet Key Pattern

Cloud applications might allow some resources to be downloaded by the end users. This can result in a tight dependency on a storage mechanism and apart from being an overhead to the cloud application itself. Consider a scenario where a hosted web service offers an expansive set of functionality, which includes downloading and streaming images to the user. In such a scenario, the application could spend a considerable amount of its processing power and memory just to download the images. Because of specific load patterns the application may limit resources to other tasks to ensure that it handles all of the download requests. This is not an ideal situation and can occur if you have a scenario where your application needs to validate the user before downloading protected images.

The Valet Key pattern introduces a mechanism where your application can validate the user's request without having to manage the actual download of the media. Both can occur while keeping the media private and locked away from anonymous access.



The pattern specifies that the client needs to send a request to the web service or application if it wants to access a media asset. The application then validates the request. If the request is valid, the application goes to the external storage mechanism, generates a temporary access token, and then provides the URI of the media asset to the client, along with the temporary access token. The application has now processed this request and can now move on to other requests. It is up to the client application or browser to use the URI and temporary access token to download the media asset. Now, the storage service is responsible for scaling up to manage all the requests for media while the application can focus solely on other functionality.



Reference Link: <https://docs.microsoft.com/azure/architecture/patterns/valet-key>

Resiliency Patterns

Transient Errors

This lesson introduces patterns related to the resiliency of workloads in Azure.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe the Circuit Breaker pattern.
- Describe the Retry pattern.
- Describe the Queue-Based Load Leveling pattern.

One of the primary differences between developing applications on-premises, and in the cloud, is the way you design your application to handle transient errors. Transient errors are as errors that occur due to temporary interruptions in the service or due to excess latency. Many of these temporary issues are self-healing and can be resolved by exercising a retry policy.

Retry policies define when and how often a connection attempt should be retried when a temporary failure occurs. Simply retrying in an infinite loop can be just as dangerous as infinite recursion. A break in the circuit must eventually be defined so that the retries are aborted if the error is determined to be of a serious nature and not just a temporary issue.

Transient Fault Handling is a pattern that makes your application more resilient by handling temporary issues in a robust manner. This is done by managing connections and implementing a retry policy. This pattern is already implemented in many common .NET libraries such as Entity Framework and the Azure software development kit (SDK). This pattern is also implemented in the Enterprise Library in such a generic manner that it can be brought into a wide variety of application scenarios.



Reference Link: <https://docs.microsoft.com/aspnet/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/transient-fault-handling>

The Retry Pattern

Applications can experience transient errors when connecting to external services irrespective of whether they are hosted in the cloud or they use external cloud-hosted services. A retry pattern can be used to implement a limited number of retries if the connection error implies that it is a temporary issue.

When the initial connection fails, the failure reason is analyzed first to determine if the fault is transient or not. If the failure reason or the error code indicates that this request is unlikely to succeed even after multiple retries, then retries are not performed at all.

Retries are performed until either the connection succeeds or a retry limit is reached. The limit is in place to avoid a situation where the retries might go on infinitely. The retries are typically performed after a timed delay and the delay might remain constant or increase linearly or exponentially after each retry. If the retry limit is reached, the connection is said to have failed in a non-transient manner.

There are many external libraries that implement the Retry Policy pattern including:

- Entity Framework

- Enterprise Library - Transient Fault Handling Application Block



Reference Link:<https://docs.microsoft.com/azure/architecture/patterns/retry>

Queues

Queueing is both a mathematical theory and also a messaging concept in computer science. In cloud applications, queues are critical for managing requests between application modules in a manner such that it provides a degree of consistency regardless of the behavior of the modules.

Applications might already have a direct connection to other application modules using direct method invocation, a two-way service, or any other streaming mechanism. If one of the application modules experiences a transient issue, then this connection is severed and it causes an immediate application failure. You can use a third-party queue to persist the requests beyond a temporary failure. Requests can also be audited independent of the primary application as they are stored in the queue mechanism.

Scalability Patterns

Asynchronous Messaging

Many software curriculums teach that separating your application into smaller modules will make it easier to manage the application in the long term. Modules can be swapped, modified and updated without having to update the entire application. Partitioning your workloads into modules also carries another benefit of allowing each logic center in your application to scale in isolation.

If you have a web application that allows people to upload images for processing, your image processing module can become CPU intensive and easily account for the majority of your CPU time and disk usage. By separating the image processing module out to another distinct server (or set of servers), you can scale this module in isolation without having to modify, scale or change the module that serves the web pages. It then becomes very important to figure out how to communicate between these modules.

Messaging is a key strategy employed in many distributed environments such as the cloud. It enables applications and services to communicate and cooperate, and can help to build scalable and resilient solutions. Messaging supports asynchronous operations, enabling you to decouple a process that consumes a service from the process that implements the service.

Problem: Handling Variable Quantities of Requests

An application running in the cloud may be expected to handle a large number of requests. The number of requests could vary significantly over time for many reasons. A sudden burst in user activity or aggregated requests coming from multiple tenants may cause unpredictable workload. At peak hours a system might need to process many hundreds of requests per second, while at other times the number could be very small. Additionally, the nature of the work performed to handle these requests might be highly variable.

Using a single instance of the consumer service might cause that instance to become flooded with requests or the messaging system may be overloaded by an influx of messages coming from the application.

Solution: Asynchronous Messaging with Variable Quantities of Message Producers and Consumers

Rather than process each request synchronously, a common technique is for the application to pass them through a messaging system to another service (a consumer service) that handles them asynchronously. This strategy helps to ensure that the business logic in the application is not blocked while the requests are being processed.

A message queue can be used to implement the communication channel between the application and the instances of the consumer service. To handle fluctuating workloads, the system can run multiple instances of the consumer service. The application posts requests in the form of messages to the queue, and the consumer service instances receive messages from the queue and process them. This approach enables the same pool of consumer service instances to handle messages from any instance of the application.

Example Implementation in Azure

In Azure, the competing consumers pattern can be easily implemented using either Storage Queues or Service Bus Queues.

1. The IoT device sends an HTTP request to the distributed Azure API App instances with temperature, barometric pressure and humidity information collected throughout the day.
2. The individual application instance that receives the request uses an HTTP request to add a message to the Storage Queue with the weather metadata included as part of the message body.

3. The consumer services use the Azure SDK for Ruby to poll the queue to see if any messages are available.

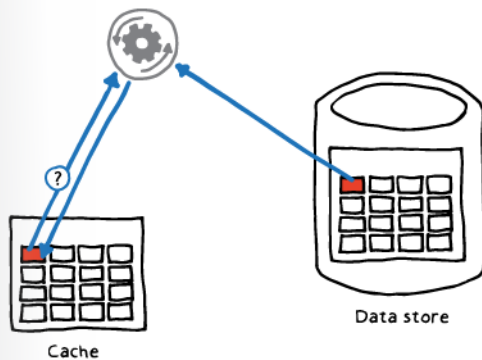
4. One of the consumer services' instances will receive the previously generated message and can now process the message. If the message has been successfully processed, the instance will use the SDK to mark the message as deleted in the queue. If the instance crashes or times out, the queue message will eventually be available to other instances after a visibility timeout period has elapsed.

Cached Data Consistency

Applications use a cache to optimize repeated access to information held in a data store. However, it is usually impractical to expect that cached data will always be completely consistent with the data in the data store. Applications developers should consider a strategy that helps to ensure that the data in the cache is up to date as far as possible, but can also detect and handle situations that arise when the data in the cache has become stale.

Solution: Read/Write-Through Caching

Many commercial caching systems provide read-through and write-through/write-behind operations. In these systems, an application retrieves data by referencing the cache. If the data is not in the cache, it is transparently retrieved from the data store and added to the cache. Any modifications to data held in the cache are automatically written back to the data store as well.



For caches that do not provide this functionality, it is the responsibility of the applications that use the cache to maintain the data in the cache. An application can emulate the functionality of read-through caching by implementing the cache-aside strategy. This strategy effectively loads data into the cache on demand if it's not already available in the cache.

Example Implementation in Azure

This pattern can be used with a variety of cache mechanisms. In Azure, the most popular cache mechanism is Azure Redis Cache. Redis Cache is a key-value store that is wildly popular for caching in many web applications.

SCENARIO: Your online game web application shows the featured player on the homepage. Since the homepage is accessed often, it is important to cache this value. Azure Redis Cache is used for the cache and Document DB is used for the data store. The featured player is typically updated using a separate application where the admins can specify a new featured player.

1. When the web application loads for the first time it makes a request (GET) to the Redis Cache instance for the name of the featured player using the key: player:featured.

2. The cache will return a nil value indicating that there is not a corresponding value stored in the cache for this key.
3. The web application can then go to the Document DB data store and gets the name of the featured player.
4. The name of the featured player will be stored in the Redis Cache using a request (SET) and the key player:featured.
5. The web application returns the name of the featured player and displays it on the homepage.
6. Subsequent requests for the home page will use the cached version of the value as Redis Cache will successfully return a value for the name of the featured player.
7. If an administrator updates the featured player, the web application will replace the value in both Redis Cache and Document DB to maintain consistency.

Load Balancing

Load balancing is a computing concept where the application traffic or load is distributed among various endpoints by using algorithms. By using a load balancer, multiple instances of your website can be created and they can behave in a predictable manner. This provides the flexibility to grow or shrink the number of instances in your application without changing the expected behavior.

Load Balancing Strategy

There are a couple of things to consider when choosing a load balancer. First, you must decide whether you wish to use a physical or a virtual load balancer. In Azure infrastructure as a service (IaaS), it is possible to use virtual load balancers, which are hosted in virtual machines, if a company requires a very specific load balancer configuration.

After you select a specific load balancer you need to select a load balancing algorithm. You can use various algorithms such as round robin or random choice. For example, round robin selects the next instance for each request based upon a predetermined order that includes all of the instances.

Other configuration options exist for load balancers such as affinity or stickiness. For example, stickiness allows you determine whether a subsequent request from the same client machine should be routed to the same service instance. This might be required in scenarios where your application servers have a concept of state.

Data Patterns

Redis Cache

There are two primary cache mechanisms available in Azure, Azure Cache and Redis Cache. Azure cache is deprecated and only exists to support existing cloud applications. All new applications should use the Redis Cache.

Azure Managed Cache

Azure Cache is a managed cache service that is based on the App Fabric platform. You can create the Cache instances by using third-party applications or Windows PowerShell cmdlets. This cache mechanism is typically seen when working with custom cloud service roles.

Redis Cache

Redis Cache is an open-source NoSQL storage mechanism that is implemented in the key-value pair pattern common among other NoSQL stores. Redis Cache is unique because it allows complex data structures for its keys.



Redis: <http://go.microsoft.com/fwlink/?LinkID=525523>

Azure Redis Cache is a managed service based on Redis Cache that provides you secure nodes as a service. There are only two tiers for this service currently available:

- **Basic.** Single node
- **Standard.** Two nodes in the Master/Replica configuration. Replication support and Service Level Agreement (SLA) is included

Azure Redis Cache provides a high degree of compatibility with existing tools and applications that already integrate with Redis Cache. You can use the Redis Cache documentation that already exists on the open source community for Azure Redis Cache.



Reference Link: <https://docs.microsoft.com/azure/redis-cache/>

Database Partitioning

Most cloud applications and services store and retrieve data as part of their operations. The design of the data stores that an application uses can have a significant bearing on the performance, throughput, and scalability of a system. One technique that is commonly applied in large-scale systems is to divide the data into separate partitions.

Partitioning refers to the physical separation of data for scale. Modern data stores understand that data may be spread across many different instances as the size of the sum data for the application can be larger than any individual physical store can handle in an efficient manner.

Partitioning can improve performance, availability and scalability. Any data access operation will only occur on a smaller subset (volume) of data which in turn ensures that the operation will be more efficient when compared to a query over the entire superset of data for your application. Individual databases can hit physical limits which are also overcome when portioning data across many different individual databases. Partitioning also spreads your data across multiple nodes which can individually be replicated or scaled. When a node is unavailable or being maintained, the application can use replica nodes.

Database Partitioning: Scenario

Problem: Hosting Large Volumes of Data in a Traditional Single-Instance Store

A data store hosted by a single server may be subject to the following limitations:

- **Storage space.** A data store for a large-scale cloud application may be expected to contain a huge volume of data that could increase significantly over time. A server typically provides only a finite amount of disk storage, but it may be possible to replace existing disks with larger ones, or add further disks to a machine as data volumes grow. However, the system will eventually reach a hard limit whereby it is not possible to easily increase the storage capacity on a given server.
- **Computing resources.** A cloud application may be required to support a large number of concurrent users, each of which run queries that retrieve information from the data store. A single server hosting the data store may not be able to provide the necessary computing power to support this load, resulting in extended response times for users and frequent failures as applications attempting to store and retrieve data time out. It may be possible to add memory or upgrade processors, but the system will reach a limit when it is not possible to increase the compute resources any further.
- **Network bandwidth.** Ultimately, the performance of a data store running on a single server is governed by the rate at which the server can receive requests and send replies. It is possible that the volume of network traffic might exceed the capacity of the network used to connect to the server, resulting in failed requests.
- **Geography.** It may be necessary to store data generated by specific users in the same region as those users for legal, compliance, or performance reasons, or to reduce latency of data access. If the users are dispersed across different countries or regions, it may not be possible to store the entire data for the application in a single data store.

Scaling vertically by adding more disk capacity, processing power, memory, and network connections may postpone the effects of some of these limitations, but it is likely to be only a temporary solution. A commercial cloud application capable of supporting large numbers of users and high volumes of data must be able to scale almost indefinitely, so vertical scaling is not necessarily the best solution.

Solution: Partitioning Data Horizontally Across Many Nodes

Divide the data store into horizontal partitions or shards. Each shard has the same schema, but holds its own distinct subset of the data. A shard is a data store in its own right (it can contain the data for many entities of different types), running on a server acting as a storage node.

Sharding physically organizes the data. When an application stores and retrieves data, the sharding logic directs the application to the appropriate shard. This sharding logic may be implemented as part of the data access code in the application, or it could be implemented by the data storage system if it transparently supports sharding.

Abstracting the physical location of the data in the sharding logic provides a high level of control over which shards contain which data, and enables data to migrate between shards without reworking the business logic of an application should the data in the shards need to be redistributed later (for example, if the shards become unbalanced). The tradeoff is the additional data access overhead required in determining the location of each data item as it is retrieved.

To ensure optimal performance and scalability, it is important to split the data in a way that is appropriate for the types of queries the application performs. In many cases, it is unlikely that the sharding scheme will exactly match the requirements of every query. For example, in a multi-tenant system an application may need to retrieve tenant data by using the tenant ID, but it may also need to look up this data based on some other attribute such as the tenant's name or location. To handle these situations, implement a sharding strategy with a shard key that supports the most commonly performed queries.

Database Partitioning: Example

Microsoft Azure supports Sharding for the Azure SQL Database either by manually creating shards or using the Elastic Scale automated functionality. This example assumes that you are using the Elastic Scale feature.

SCENARIO: *Your music catalog application uses elastic scale to spread the large amount of data across many partitions. Each partition is an instance of Azure SQL Database. The artist's unique ID number is used as the shard key.*

1. A shard map manager is created and shards are added to the manager. ID ranges for are associated to each shard so that artists are evenly spread across shards (databases). The first three shards are configured in the following manner.

Shard Name	ID range
DataStore_I	[(0-1000)]
DataStore_II	[(1001-2000)]
DataStore_III	[(2001-3000)]
DataStore_IV	[(3001-4000)]

There are more shards with similar ranges. The application has approximately 40 shards to start.

2. Your application needs a list of albums for an artist. The application uses the shard map manager to get a connection string to the database associated with the shard. The application can now query that database directly for all albums with the artist ID.

3. One of the artists in a shard is getting very popular. This artist has an ID of 1245 and is affecting the resources of other artists in the same shard. Elastic Scale uses the "Split" feature to create two new shards from the original DataStore_II shard. The new DataStore_II shard has a disjointed ID range of [(1001-1244),(1246-2000)]. This means that all IDs from 1001-2000 except 1245 are included in this range. The new DataStore_XLI shard has an ID range of just [1245].

4. The application already uses the smallest database size for two of its shards, DataStore_IV and DataStore_III. Even with this small size, the databases are not fully utilized. Elastic Scale uses the "Merge" feature to create a new shard from these two shards. The new DataStore_XLII shard has a continuous ID range of [(2001-4000)].

Review Questions

Module 1 Review Questions

Design patterns

You develop a web application that allows users to sign in and download images and videos. You need to ensure the application performs well during peak traffic hours. What design pattern should you use for the solution?

Suggested Answer ↓

There are many design patterns that you can use to more effectively plan a solution. The Valet key pattern introduces a mechanism where your application can validate the user's request without having to manage the actual download of the media. View Microsoft's patterns and practices resource center on GitHub for more information.

Design patterns

A web application is experiencing transient errors when connecting to an external database. What design pattern should you use for the solution?

Suggested Answer ↓

The Retry pattern is used to implement a limited number of retries if an external service connection is the issue. View Microsoft's **patterns and practices resource center**² on GitHub for more information.

Design patterns (applications)

You are developing an application that connects with a web app to process information and then displays the results.

You need to manage requests between the two applications without depending on the behavior of either application. What design pattern should you use for the solution?

Suggested Answer ↓

The Queue design pattern is a resiliency pattern. You can use this pattern to manage requests between application modules in a way that provides consistency regardless of the behavior of the modules. View Microsoft's **patterns and practices resource center**³ on GitHub for more information.

² <https://github.com/mspnp>

³ <https://github.com/mspnp>



Module 2 Module Building Azure IaaS-Based Server Application

High Availability

Azure Availability

This module identifies workloads that are ideally deployed using Infrastructure-as-a-Service services in Azure. The module focuses on the VM Scale Sets and Virtual Machine services in Azure and how to best deploy workloads to these services using best practices and features such as Availability Sets.

After completing this module, students will be able to:

- Design an availability set for one or more virtual machines.
- Describe the differences between fault and update domains.
- Author a VM Scale Set ARM template.
- Join a virtualized machine to a domain either in Azure or on a hybrid network.

High Availability

Azure has provided a money-backed Service Level Agreement (SLA) for a long time and with the recent product releases and enhancements now provides several different services and levels of service. This lesson will explain what is available and the best way to ensure the availability of your resources. This section will cover Availability Sets and Availability Zones.

After completing this section you will be able to:

- Describe Availability features of Microsoft Azure.
- Understand difference between an Availability Set and an Availability Zone.
- Deploy resources into Availability Sets correctly.
- Decide which applications and resources should not use Availability Sets.

Azure Availability

Microsoft Azure provides a Service Level Agreement (SLA) that is backed by a financial service credit payment for infrastructure as a Service (IaaS) Virtual Machines. The SLA depends entirely upon the deployment of the virtual machine and what resources it uses. The aim is to prevent virtual machine reboots.

The method Azure uses to ensure that the SLA can be provided is an Availability Set. An availability set ensures that all virtual machines that are added to the set are placed in such a way as to ensure that neither hardware faults or Azure fabric updates that is unplanned and planned maintenance events can bring down all of the virtual machines.

Application Availability

An Azure virtual machine can be impacted for one of three reasons:

- Unplanned hardware maintenance event
- An unexpected downtime
- Planned maintenance events

To reduce or remove the impact of downtime related to these events there are several steps to take, these include:

- Place virtual machines in an availability set for redundancy.
- Use managed disks for all VMs placed in an availability set.
- Use Scheduled Events to respond to events.
- Place each tier of your application in a separate availability set.
- Use a load balancer in combination with availability sets.

All the above steps provide additional high availability for your application and can be used in varying situations. The fundamental building block is the Availability Set.

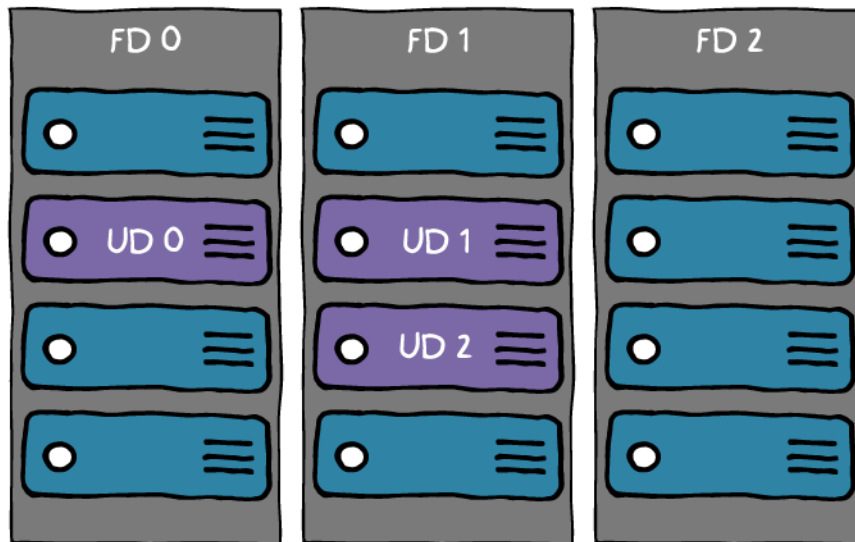
Availability Sets

The availability set is the recommended solution for placing multiple instance VMs, this configuration will allow for at least one VM being available at least 99.95% of the time.



Best Practice: Avoid single instance VMs in an availability set. These are not subject to any SLA unless all the Operating System and Data disks are using Premium storage.

Remember Availability Sets comprise of Update Domains and Fault Domains, as shown in the image below.



UPDATE AND FAULT DOMAINS

Each machine in the Availability set is placed in an Update Domain and a Fault domain.

An Availability Set is a logical grouping service that you can create and use to ensure that the VMs you deploy within an Azure datacenter are isolated from each other. Azure places the VMs you deploy within an Availability Set across multiple physical servers, racks and network switches.

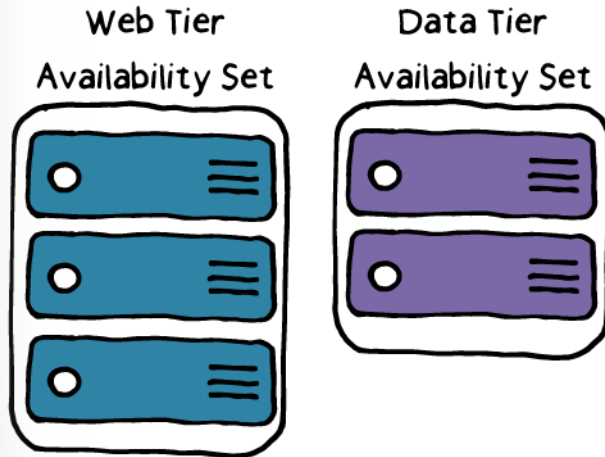
In the event of an Azure hardware or software failure, only a proportion of the VMs deployed to your Availability set are impacted. The application running on the VMs will remain available. Architecturally availability sets are essential to most cloud-based solutions.

Multiple Availability Sets

An extension of the availability set model is used logically to place individual tiers of an application into separate Availability Sets.

In this example we have a two-tier VM-based application which contains three load-balanced front-end Web servers and two back-end VMs that host a SQL Server data store. In Azure, it would be best practice to create two availability sets in preparation for the application. AVSet1 for the Web tier and AVSet2 for the data tier. Each time you create a VM for the application, you deploy it to the correct Availability set for the function it will perform.

Azure will then ensure that the VMs you create within the availability set are isolated across multiple physical hardware resources. If the physical hardware that one of your Web Server or Database Server VMs is running on has a problem, you know that the other instances of your Web Server and Database VMs remain running because they are on different hardware.

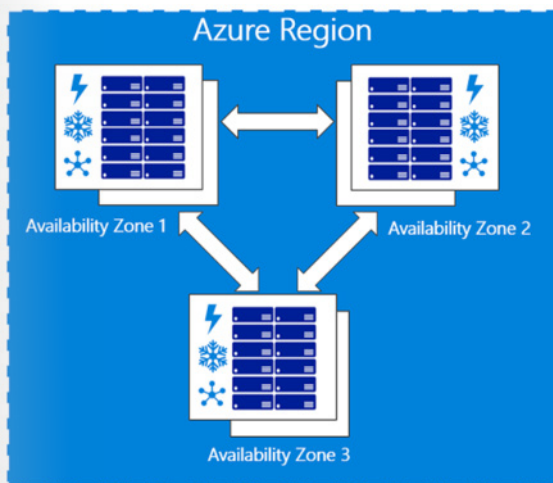


N-TIER AVAILABILITY SETS

Availability Zones

A new preview service further extending highly available VMs is the Availability Zone. Whilst Availability Sets can protect from individual hardware faults and even to rack-based faults, the advent of a data-center-wide fault would prevent the Availability set from functioning.

To extend the capability further, Microsoft has released a preview service which as an alternative to Availability Sets can provide highly available VMs across datacenter buildings or Zones. In this instance, the Availability Zone feature will allow for a complete data center failure and keep your VM based application running. The title Zone indicates a separate zone or building within a single Azure region.



There is a maximum of three Availability Zones per supported Azure region. Each Zone operates on an entirely isolated power source, cooling system, and network infrastructure.

The use of Availability Zones to architect your applications to use VMs replicated in Zones provides an additional level of protection.

There are few scenarios where the use of either Availability Zones or Availability Sets would not be used in a production IaaS based infrastructure or application. The use of single VMs or VM's without availability sets whilst not attracting an SLA would still be suitable for dev and test scenarios.

Templated Infrastructure

Templated Infrastructure

Azure provides a compute resource capable of true Autoscale without pre-provisioning Virtual machines. Azure VM Scale Sets allow for templated deployment of marketplace images and custom images in highly scalable and highly available infrastructure to provide a platform for big compute and big data IaaS based applications. This section provides an overview of Scale Sets and discusses the differences between deployment of multiple VMs and scale sets in your application.

After completing this section you will be able to:

- Describe Azure VM Scale Set features.
- Decide how to deploy VM Scale sets.
- Decide whether VM Scale Sets or standalone VMs provide the features you require.
- Deploy an application to a VM Scale Set using VSTS.

Scalable architecture in cloud platforms is nothing new, but always comes with a trade-off between complexity and control. Complexity is defined here as the difficulty in how to define and deploy the resources you need to control the individual resources once deployed. The tradeoff leads typically to a choice between a PaaS solution and an IaaS solution, neither really providing everything that is required.

Azure VM Scale Sets are an Azure compute resource that provides both a high degree of infrastructure resource control without the need to invest time and energy in managing the attendant networking, storage and compute. In addition, the built-in load balancing allows it to be controlled like IaaS but Scaled like PaaS.

It is usual when building cloud infrastructure to create storage, compute and network resources and create the dependencies between them. Azure VM Scale sets handle this resource creation for you and go one step further by managing the necessary configurations and optimizations when the scale set scales up or down, further reducing the workload required.

Virtual Machine Scale Sets

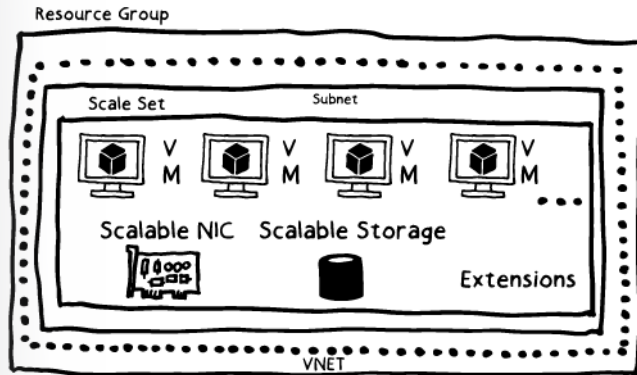
An Azure VM Scale Set has several features that make it attractive to the Azure architect. The ability to define a VM Scale Set by JSON template and deploy it using any of the standard deployment methods enables their use in

many automated solutions. This extends into continuous deployment scenarios with Visual Studio Team Services.

An Azure VM Scale Set allows a Virtual machine to deploy up to 1000 times in the same subnet in a controlled and automated manner with accurate auto-scaling.

An Azure VM Scale Set also requires no pre-provisioning of the Virtual Machine before adding to the scale set. The network and load balancer are created, configured and managed automatically, including the Network Address Translation (NAT) for access to and from the VM Instances.

These features added to the ease of deployment through the portal, Azure PowerShell or Azure CLI make the Azure VM Scale Set a powerful tool for the Azure cloud architect.



Virtual Machines vs. Virtual Machine Scale Sets

Azure Virtual Machines and Azure VM Scale Sets have several unique features that allow the architect to choose between their suitability for an application or deployment.

Azure VM Scale Sets

The Scale Set capacity property allows you to deploy more VMs in parallel. This is easier than writing a scripting the orchestration required to deploy individual VMs in parallel. With Azure VM Scale Sets:

- You can use Azure Autoscale to automatically scale a scale set but not individual VMs.
- You can reimage scale set VMs but not individual VMs.
- You can overprovision scale set VMs for increased reliability if a faster deployment time is required. To do this with individual VMs custom code must be written.
- Can take advantage of an upgrade policy. This makes it easy to upgrade all the VMs in your Scale. With individual VMs, this must be orchestrated.

Azure Virtual Machines

In contrast, with Azure Virtual Machines:

- You can attach data disks to individual VMs, but attached data disks in VM Scale Sets apply to all instances in the set. You can also attach non-empty data disks to individual VMs but not to VMs in a scale set.
- You can snapshot an individual VM but not a VM in a scale set. You can also capture an image from an individual VM but not from a VM in a scale set.
- You can migrate an individual VM to use managed disks from native disks, this cannot be done in a VM Scale Set.
- You can assign IPv6 public IP addresses to individual network Interface Cards in a VM but cannot do so for VMs in a scale set.



Note: You can, however, assign an IPv6 public IP address to load balancers. It does not matter if the load balancer is in front of a VM or a VM Scale Set.

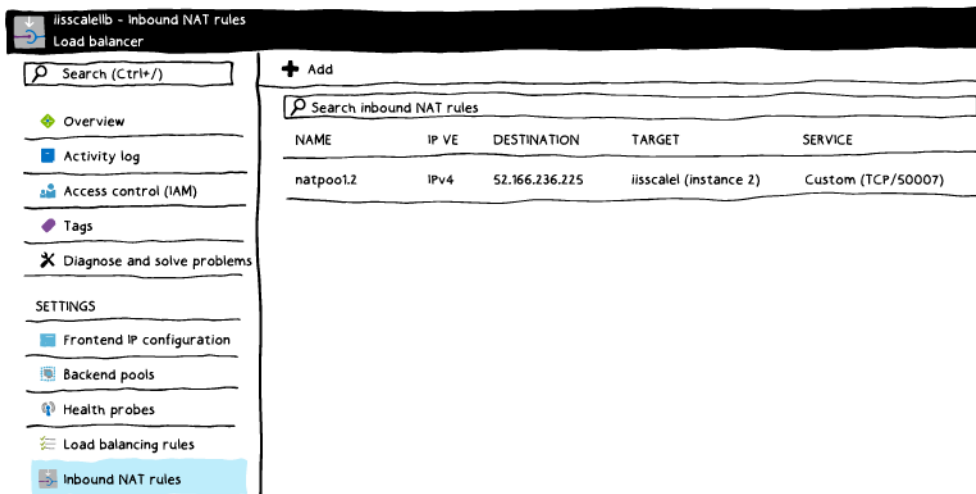
Virtual Machine Scale Set Considerations

When working with VM Scale Sets there are design considerations which will affect the ease of use and performance of the resultant Scale Set.

Connecting to a VM Scale Set Instance VM

If using a Windows VM in the Scale Set, it is possible to connect to a specific VM instance by accessing the Load balancer inbound NAT rules and using the correct IP address and custom port. In the instance below the RDP client would be pointed at 52.166.236.225:50007.

Once you enter the correct admin user credentials access will be granted to the Virtual Machine instance.



INBOUND NAT RULES



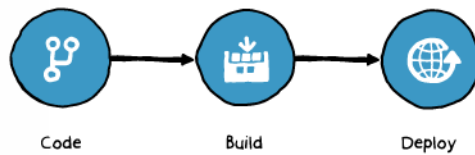
Note: For a Linux VM Scale Set, there is a choice between SSH key or username and password.

Continuous Delivery in VMSS

Use Continuous delivery to maintain an application in a VMSS with Visual Studio Team Services. Continuous delivery in Visual Studio Team Services simplifies setting up a robust deployment pipeline for your application. By default, the pipeline builds code, and updates VM scale set with the latest version of your application. The update to VM Scale set can be done either by creating an image and using that to create/update VM scale set or by using custom script VM extension to install/update your application on VM scale set. You can quickly add another VM Scale Set to the pipeline to validate your changes before they ever get to production.

Need to provision additional Azure resources, run scripts, upgrade your database or run additional validation tests? You can easily extend this deployment automation to handle any other operations your application needs to do during deployment.

Visual Studio Team Services can be used to automate the deployment of code to a VMSS Instance.



When deploying an application to a VM Scale Set, there are usually two ways to achieve the goal:

1. Use of VM extensions to install software to each instance at deployment time.
2. Create a custom image that already contains the OS and application in a single VHD.

Option 2 is also known as an immutable deployment. This method has many advantages:

- Predictability
- Easy to Scale
- Easy to roll-back
- Faster to scale (no code to install on each VM as it is deployed)

Having chosen option 2, the benefits can be further enhanced by taking advantage of the Visual Studio Team Services continuous integration toolset and the Continuous Delivery preview service in the VM Scale Set blade.

✖ Disconnect ✎ Edit

i Continuous delivery on this virtual machine scale set (which uses OS image from gallery) will use custom Azure VM extension to deploy application

Deploy with confidence

- ✔ Automate your deployments to Virtual Machine Scale Set
- ✔ Update your application by creating immutable images or by using custom script VM extension
- ✔ Setup approvals for deployment to production
- ✔ Extend and customize your deployment automation

By enabling and configuring VSTS continuous delivery, the application code and deployment scripts can be deployed from either GitHub or the Visual Studio Team Service repository.

The added benefit is that any new versions of the application can be tested on a similar VM Scale set and then deployed directly into the production instances without any downtime.

Large VM Scale Sets

Azure VM Scale Sets can handle up to 1000 VMs in each Scale Set. Azure classifies a Scale Set that can scale beyond 100 VMs as a large VM Scale Set. The large VM Scale Set capability is marked by a **single-PlacementGroup= false** property setting. The large VM Scale set has unique requirements as well as changing the way specific aspects of a Virtual Machine deployment behave, such as load balancing and fault domains.

To decide whether your application can make efficient use of large scale sets, consider the following requirements:

- Large scale sets require Azure Managed Disks.

- Scale sets created from Azure Marketplace images can scale up to 1,000 VMs.
- Scale sets created from custom images can scale up to 300 VMs.
- Layer-4 load balancing with scale sets composed of multiple placement groups requires Azure Load Balancer Standard SKU.
- Layer-7 load balancing with the Azure Application Gateway is supported for all scale sets.
- Scale sets are defined with a single subnet – ensure subnet is large enough to handle all potential VM instances.
- Ensure your compute limits are high enough, the requirement for compute cores will prevent a successful deployment if not.
- Fault Domains and Update Domains relate to a single placement group, to maintain high availability ensure there are at least two VM instances in each Fault Domain and Update Domain.

Domain-Joined Virtual Machines

Domain and IaaS Applications

The use of Active Directory is widespread throughout the on-premises and cloud-based Windows infrastructure world. The advent of Azure AD brings many options for the Azure Architect to choose between. This lesson will examine the benefits of and differences between cloud only and hybrid solutions comprised of on-premises Active Directory Domain Services, Azure AD, and Azure AD Domain Services.

After completing this section you will be able to:

- Describe Azure AD Domain Services and Hybrid AD options.
- Decide when to use Azure AD Domain Services, AD DS in an Azure VM or Hybrid on premises.
- Create an Azure AD Domain Services Managed domain.

Domain and IaaS Applications

Authentication and Authorization of users in a cloud or hybrid infrastructure require careful planning and consideration. There are several options when using Azure AD. Azure AD is a multi-tier Identity as a Service offering that provides a complete Identity and Access Management solution for your cloud or hybrid environment.

In its basic form, Azure AD is a free service that provides the ability for Single Sign-On into cloud applications. The various tiers basic, premium 1 and Premium 2 each provide additional levels of service such as Multi-Factor Authentication and additional reporting. This lesson covers the various options available to Azure AD administrators to provide domain services to their hybrid or cloud-based networks:

- Azure AD Connect
- Azure AD Domain Services
- Azure AD pass-through Authentication

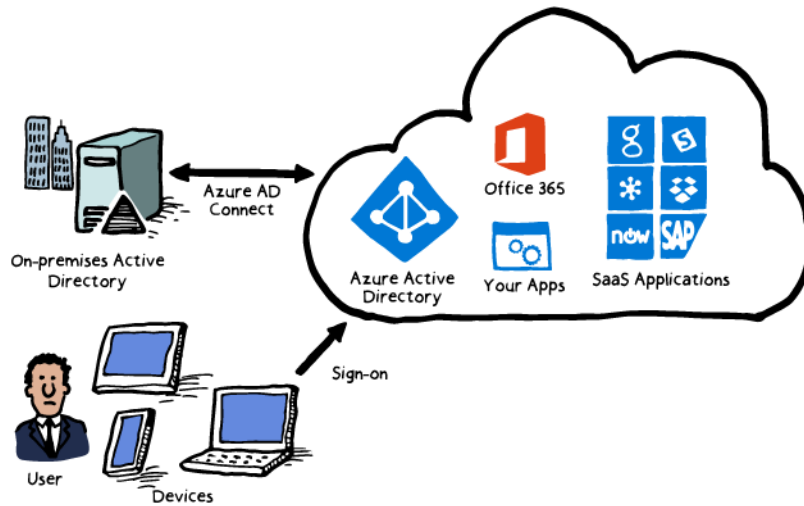
Azure AD also provides services to allow connection with consumers (Azure AD B2C) and business partners (Azure AD B2B) without deploying complex infrastructure or federation.

Hybrid Connectivity

Azure AD Connect

Azure AD Connect provides the ability to integrate your on-premises directories with Azure AD. Having deployed AD Connect, you can provide single sign-on to cloud applications such as Office 365, Azure and other SaaS applications.

Why use AD Connect? Having a common identity to access both cloud and on-premises applications and resources enables users to take advantage of a Single identity, one set of credentials to remember and a Single tool, single sign in, easy for administrators to deploy.



AZURE AD HYBRID

Azure AD Connect provides the choice of:

- Password Synchronization only, the ability to synchronize users and groups.
- ADFS, to allow on-premises authentication, 3rd party MFA, etc.
- Pass-through authentication, provides on-premises authentication without deploying ADFS.

Whichever architecture you decide to use, if on-premises directories are included, you will need Azure AD Connect to provide the synchronization engine. The Microsoft Identity Manager (MIM) client is installed on-premises and used to configure the users, groups and attributes to be synchronized.

Azure AD Domain Services

Azure provides the ability to deploy Infrastructure solutions in many ways. To support this Azure AD Domain Services provides managed domain services such as domain join, group policy, LDAP and Kerberos authentication, all of which are entirely Windows Server Active Directory compatible.

Azure AD Domain Services allows you to take advantage of these services without having to deploy a Domain Controller to an IaaS VM in the cloud. Azure AD Domain Services is compatible with both Cloud only tenants and hybrid tenants using Azure AD Connect.

Cloud Only Tenant

By enabling Azure AD Domain Services on an Azure AD tenant, Azure creates a highly-available domain service which is connected to the virtual network, and all the Azure AD objects are available within this domain, but all user identities, credentials and groups, including group memberships, are created and managed in Azure AD.

The advantages of this solution are:

- The domain administrator does not need to manage this domain or any domain controllers.
- AD replication for this domain does not require management. All objects are automatically available.
- Azure manages this Domain, so the Azure AD tenant administrator has no domain or enterprise admin privileges.

Hybrid Cloud Tenant

By enabling Azure AD Domain Services on an Azure AD tenant that is synchronized to an on-premises directory, an additional stand-alone Domain is created by the managed service. All objects from the on-premises domain and the Azure AD tenant are available to the managed service domain. Tenant identities are still created and managed within Azure AD, and on-premises identities are still created and managed on-premises. This solution allows users to sign in to cloud services with their on-premises identities. For this to work with the Azure AD Domain Services, Azure AD Connect must be configured to allow password synchronization; this is required so resources in the cloud connected to the managed domain can use Kerberos to authenticate. The managed domain is a standalone domain and not an extension to the on-premises directory.

The advantages of this solution are:

- The domain administrator does not need to manage this domain or any domain controllers for the managed domain.
- AD replication for this domain does not require management. All objects are automatically available.
- Azure manages this Domain, so the Azure AD tenant administrator has no domain or enterprise admin privileges.

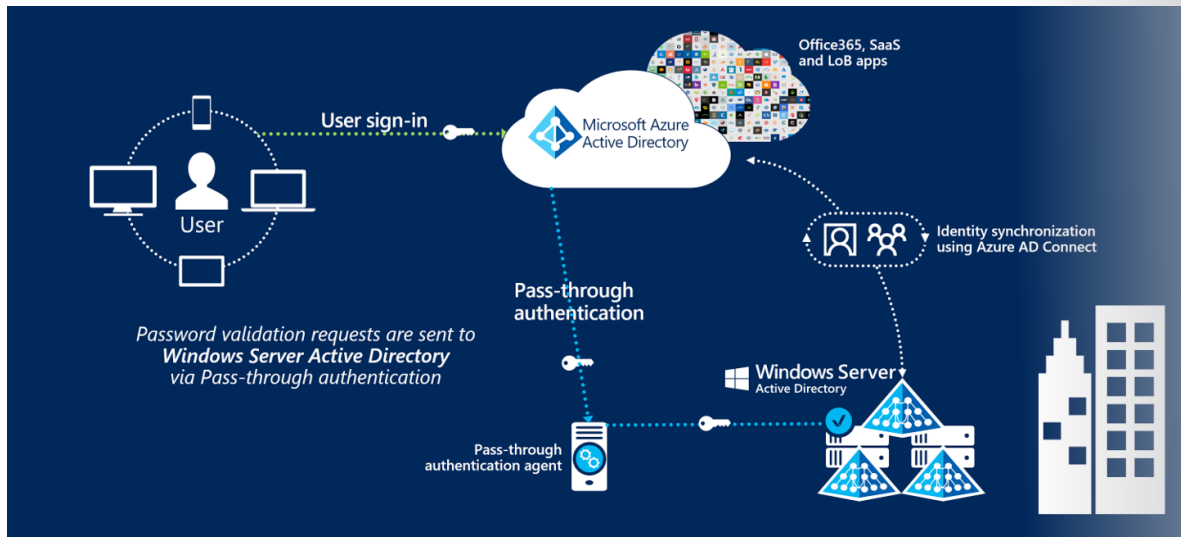
The benefits of the Azure AD Domain Services managed domain offering are:

- **Simple to deploy in a few clicks**—No IaaS infrastructure required to provide authentication to Virtual Machines.
- **Deep integration within your Azure AD tenant.**
- **Compatible with Windows Server Active Directory**—Not all features available in Windows Server AD are available in Azure AD Domain Services. The following are compatible, LDAP, Kerberos, NTLM, Group Policy, and domain join capabilities.
- **Cost-effective**—No need to pay for Azure IaaS Virtual Machines.

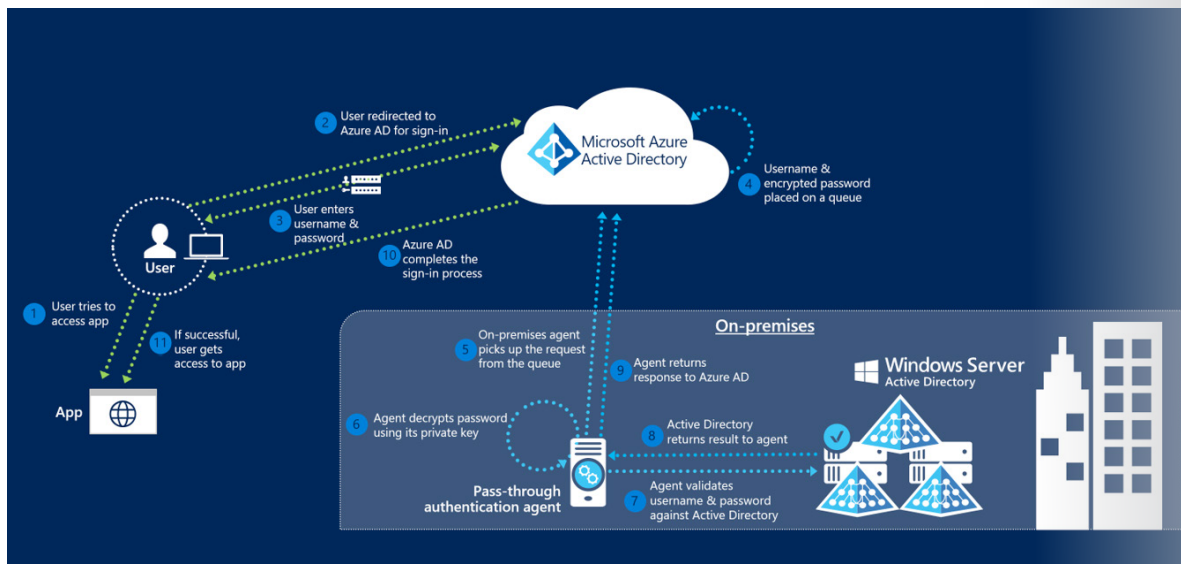
Azure AD Pass-Through Authentication

Azure AD Connect provides a new service that permits on-premises authentication without the need to deploy ADFS infrastructure. This is a considerable cost and time saver. With Azure AD Connect, you have no need for complicated certificates or trusts.

Azure AD Pass-through Authentication enables users to sign in to both on-premises and cloud applications using the same credentials. When users sign in using Azure AD, this feature validates users' passwords against your on-premises Active Directory, the same as an ADFS based solution would do.



Pass-through authentication:



The highlighted benefits of this solution are:

- A great user experience, the user uses the same passwords to sign into both on-premises and cloud-based applications.
- Users spend less time resolving password-related issues.
- Users can be enabled to use self-service password management from the Azure AD directly.
- Easy to deploy, there is no requirement for on-premises deployments or network configuration. Only a small agent needed on-premises.
- Secure storage of passwords since on-premises passwords are never stored in the cloud.
- No additional ports or network configuration is required since the agent communicates outbound, so no perimeter network is required.
- Takes advantage of Azure AD Conditional Access policies, including Multi-Factor Authentication (MFA).

- By installing additional agents, the service can become highly available.

This is a free feature available to all tiers of Azure AD and supports all web-based apps and those supporting modern authentication. Multi-forest environments are supported, although some routing changes may be required.

Online Lab - Building Azure IaaS-Based Server Applications by using Azure ARM Templates and Azure Building Blocks

Lab Steps

Online Lab - Building Azure IaaS-Based Server Applications by using Azure ARM Templates and Azure Building Blocks

NOTE: For the most recent version of this online lab, see: <https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign>

Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
 - Username: **Admin**
 - Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
 - Microsoft Edge
 - File Explorer
 - **Visual Studio Code**¹
 - **Microsoft Azure Storage Explorer**²
 - Bash on Ubuntu on Windows
 - Windows PowerShell
3. **Note:** You can also find shortcuts to these applications in the **Start Menu**.

¹ <https://code.visualstudio.com/>

² <https://azure.microsoft.com/features/storage-explorer/>

3

Exercise 1: Deploy an Azure VM by using Azure Resource Manager templates with PowerShell Desired State Configuration (DSC) extension from the Azure portal.

4

Task 1: Open the Azure Portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (<https://portal.azure.com>).
3. If prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

5

Task 2: Create an Azure VM running Windows Server 2016 Datacenter.

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Windows Server 2016** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Windows Server 2016 Datacenter**.
4. On the **Windows Server 2016 Datacenter** blade, click the **Create** button.
5. On the **Basics** tab, perform the following tasks:
 - Leave the **Subscription** drop-down list entry set to its default value.
 - In the **Resource group** section, click **Create new**, in the text box, type **AADesignLab0301-RG**, and click **OK**.
 - In the **Name** text box, enter the value **lab03vm0**.
 - In the **Region** drop-down list, select an Azure region to which you want to deploy resources in this lab.
 - In the **Availability options** drop-down list, select **Availability set**.
 - In the **Availability set** section, click **Create new**, box, enter the value **lab03avset0**, set **Fault domains** to the maximum value, leave **Update domains** with its default value, and click **OK**.

3 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#exercise-1-deploy-an-azure-vm-by-using-azure-resource-manager-templates-with-powershell-desired-state-configuration-dsc-extension-from-the-azure-portal

4 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-1-open-the-azure-portal

5 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-2-create-an-azure-vm-running-windows-server-2016-datacenter

- Leave the entry in the **Image** drop-down list set to its default value.
 - Ensure that the size is set to **Standard DS1 v2**
 - In the **Username** text box, enter the value **Student**.
 - In the **Password** and **Confirm password** text boxes, enter the value **Pa55w.rd1234**.
 - In the **Public inbound ports** section, select the **Allow selected port** option and, in the **Select inbound ports** drop-down list, select **HTTP**.
 - Leave the **Already have a Windows license?** option set to **No**.
 - Click **Next: Disks** >
6. On the **Disks** tab, perform the following tasks:
 - Ensure that the **OS disk type** dropdown list entry is set to **Premium SSD**
 - Click **Next: Networking** >
 7. On the **Networking** tab, perform the following tasks:
 - In the **Virtual network** section, click **Create new**.
 - On the **Create virtual network** blade, specify the following settings and click **OK**:
 - In the **Name** text box, enter the value **lab03vnet0**.
 - In the **Address range** text box, enter the value **10.3.0.0/16**.
 - In the **Subnet name** text box, enter the value **subnet-0**.
 - In the **Subnet address range** text box, enter the value **10.3.0.0/24**, and click **OK**.
 - Leave the **Public IP** entry set to its default value.
 - Leave the **NIC network security group** option set to **Basic**.
 - Leave the **Public inbound ports** option set to **Allow selected ports**
 - Leave the **Select inbound ports** entry set to **HTTP**
 - Leave the **Accelerated networking** entry set to its default value.
 - Click **Next: Management** >
 8. On the **Management** tab, perform the following tasks:
 - Leave the **Boot diagnostics** option set to its default value.
 - Leave the **OS guest diagnostics** option set to its default value.
 - Leave the **Diagnostics storage account** entry set to its default value.
 - Leave the **System assigned managed identity** option set to its default value.
 - Leave the **Enable auto-shutdown** option set to its default value.
 - Leave the **Enable backup** option set to its default value.
 - Click the **Review + create** button.
 9. On the **Create a virtual machine** blade, review the settings of your new virtual machine and click the **Create** button.
 10. Do not wait for the deployment to complete and proceed to the next task.

6

Task 3: View DSC configuration

1. On the Taskbar, click the **File Explorer** icon.
2. In the **File Explorer** window that appears, navigate to the `\allfiles\AZ-301T04\Module_02\LabFiles\Starter\` folder.
3. Right-click the **IISWebServer.zip** file and select the **Extract All...** option.
4. In the **Extract Compressed (Zipped) Folders** dialog, perform the following tasks:
 - In the **Files will be extracted to this folder:** field, enter the name of the folder into which you want to extract the files.
 - Ensure that the **Show extracted files when complete** checkbox is selected.
 - Click the **Extract** button.
5. In the new **File Explorer** window that appears, right-click the **IISWebServer.ps1** file and select the **Open with Code** option to start the **Visual Studio Code** application.
6. In the **Visual Studio Code** window that appears, review the content of the PowerShell script.
7. At the top of the **Visual Studio Code** window, click the **File** menu and select the **Close Window** option.
8. Close both **File Explorer** windows.
9. Return to the **Microsoft Edge** window with the **Azure Portal** open.

7

Task 4: Create an Azure Storage account

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Storage account** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Storage account - blob, file, table, queue**.
4. On the **Storage account - blob, file, table, queue** blade, click the **Create** button.
5. On the **Create storage account** blade, perform the following tasks:
 - In the **Name** text box, type a unique name consisting of a combination of between 3 and 24 characters and digits.
 - In the **Deployment model** section, ensure that the **Resource manager** option is selected.
 - In the **Account kind** drop-down list, ensure that the **Storage (general purpose v1)** option is selected.
 - Leave the **Location** entry set to the same Azure region you selected earlier in this exercise.

6 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-3-view-dsc-configuration

7 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-4-create-an-azure-storage-account

- In the **Replication** drop-down list, select the **Locally-redundant storage (LRS)** entry.
 - In the **Performance** section, ensure that the **Standard** option is selected.
 - In the **Secure transfer required** section, ensure that the **Disabled** option is selected.
 - Leave the **Subscription** drop-down list entry set to its default value.
 - In the **Resource group** section, ensure that the **Use existing** option is selected and, in the drop-down list below, select the resource group you created earlier in this exercise.
 - Leave the **Configure virtual networks** option set to its default value.
 - Leave the **Hierarchical namespaces** option set to its default value.
 - Click the **Create** button.
6. Wait for the deployment to complete before you proceed to the next task.
 7. **Note:** This operation can take about 2 minutes.

8

Task 5: Upload DSC configuration to Azure Storage

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the entry representing the resource group into which you deployed the storage account.
3. On the resource group blade, click the entry representing the newly created storage account.
4. With the **Overview** selection active, on the storage account blade, click **Blobs**.
5. Click the **Container** button at the top of the blade.
6. In the **New container** pane that appears, specify the following settings and click **OK**:
 - In the **Name** text box, enter the value **config**.
 - In the **Public access level** list, select the **Blob (anonymous read access for blobs only)** option.
7. Back on the **Blob service** blade, click the entry representing the new **config** container.
8. On the **config** blade, click the **Upload** button at the top of the blade.
9. In the **Upload blob** pane, perform the following tasks:
 - In the **Files** field, click the blue folder button to the right of the field.
 - In the **Open file** dialog that appears, navigate to the **\\allfiles\\AZ-301T04\\Module_02\\LabFiles\\Starter** folder.
 - Select the **IISWebServer.zip** file.
 - Click the **Open** button to close the dialog box and return to the **Upload blob** popup.
 - Click the **Upload** button.
10. Navigate to the **config** blade and click the entry representing the **IISWebServer.zip** blob.

8 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-5-upload-dsc-configuration-to-azure-storage

11. In the **Blob properties** popup that appears, locate and record the value of the **URL** property. This URL will be used later in this lab.

9

Task 6: Deploy an Azure VM by using an Azure Resource Manager template with PowerShell DSC extension from the Aure portal.

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
6. On the **Edit template** blade, click **Load file**.
7. In the **Choose File to Upload** dialog box, navigate to the `\allfiles\AZ-301T04\Module_02\LabFiles\Starter\` folder, select the `dsc-extension-template.json` file, and click **Open**. This will load the following content into the template editor pane:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "virtualMachineName": {
      "type": "string",
      "defaultValue": "lab03vm0"
    },
    "configurationModuleUrl": {
      "type": "string"
    },
    "extensionFunction": {
      "type": "string",
      "defaultValue": "IISWebServer.ps1\\IISWebServer"
    }
  },
  "resources": [
    {
      "apiVersion": "2018-06-01",
      "type": "Microsoft.Compute/virtualMachines/extensions",
      "name": "[concat(parameters('virtualMachineName'), '/dscExtension')]",
      "location": "[resourceGroup().location]",
```

9 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-6-deploy-an-azure-vm-by-using-an-azure-resource-manager-template-with-powershell-dsc-extension-from-the-aure-portal

```

    "properties": {
      "publisher": "Microsoft.Powershell",
      "type": "DSC",
      "typeHandlerVersion": "2.75",
      "autoUpgradeMinorVersion": true,
      "settings": {
        "ModulesUrl": "[parameters('configurationModuleUrl')]",
        "ConfigurationFunction": "[parameters('extensionFunc-
tion')]",
        "Properties": {
          "MachineName": "[parameters('virtualMachineName')]"
        }
      },
      "protectedSettings": null
    }
  ]
}

```

8. Click the **Save** button to persist the template.
9. Back on the **Custom deployment** blade, perform the following tasks:
 - Leave the **Subscription** drop-down list entry set to its default value.
 - In the **Resource group** section, select the **Use existing** option and, in the drop-down list, select the resource group you created earlier in this exercise.
 - Leave the **Location** drop-down list set to its default value.
 - Leave the **Virtual Machine Name** field set to its default value: **lab03vm0**.
 - In the **Configuration Module Url** field, enter the URL value that you recorded in the previous task.
 - Leave the **Extension Function** field set to its default value: **IISWebServer.ps1\IISWebServer**.
 - In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
 - Click the **Purchase** button.
10. Wait for the deployment of the DSC configuration to complete before you proceed to the next task.
11. **Note:** DSC configuration deployment can take up to ten minutes.

10

Task 7: Validate that the Azure VM is serving web content

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the entry representing the resource group into which you deployed the virtual machine.

10 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-7-validate-that-the-azure-vm-is-serving-web-content

3. On the resource group blade, click the entry representing the **Virtual Machine** you deployed.
4. On the **Virtual machine** blade, locate the **Public IP address** entry, and identify its value.
5. Open a new Microsoft Edge tab and navigate to the IP address you identified in the previous step.
6. Verify that you are able to access the default Internet Information Services webpage.
7. Close the new browser tab.

Review: In this exercise, you deployed an **Virtual Machine** from the Azure portal and then used the **PowerShell DSC** extension to apply changes to the virtual machine in an unattended manner.

11

Exercise 2: Deploy an Azure Virtual Machine Scale Set (VMSS) by using Azure Resource Manager templates with PowerShell Desired State Configuration (DSC) extension from the Azure portal.

12

Task 1: View an Azure Resource Manager template.

1. On the Taskbar, click the **File Explorer** icon.
2. In the **File Explorer** window that appears, navigate to the `\allfiles\AZ-301T04\Module_02\LabFiles\Starter\` folder.
3. Right-click the `vmss-template.json` file and select the **Open with Code** option to start the **Visual Studio Code** application.
4. In the **Visual Studio Code** window that appears, review the content of the JSON file.
5. At the top of the **Visual Studio Code** window, click the **File** menu and select the **Close Window** option.
6. Close the **File Explorer** window.
7. Return to the **Microsoft Edge** window with the **Azure Portal** open.

11 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#exercise-2-deploy-an-azure-virtual-machine-scale-set-vmss-by-using-azure-resource-manager-templates-with-powershell-desired-state-configuration-dsc-extension-from-the-azure-portal

12 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-1-view-an-azure-resource-manager-template

13

Task 2: Deploy a VMSS using ARM

1. In the hub menu of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click **Build your own template in the editor**.
6. On the **Edit template** blade, click **Load file**.
7. In the **Open** file dialog that appears, navigate to the **F:\Labfiles\Mod03\Starter** folder.
8. Select the **vmss-template.json** file.
9. Click the **Open** button.
10. Back on the **Edit template** blade, click the **Save** button to persist the template.
11. Back on the **Custom deployment** blade, perform the following tasks:
 - Leave the **Subscription** drop-down list entry set to its default value.
 - In the **Resource group** section, select the **Create new** option and, in the text box, type **AADesign-Lab0302-RG**.
 - Leave the **Location** entry set to its default value.
 - In the **Admin User Name** text box, enter the value **Student**.
 - In the **Admin Password** text box, enter the value **Pa55w.rd1234**.
 - In the **Instance Count** text box, enter the value **2**.
 - Leave the **Overprovision** text box set to its default value: **true**.
 - In the **Configuration Module Url** text box, enter the URL that you recorded for the uploaded blob in the previous exercise of this lab.
 - In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
 - Click the **Purchase** button.
12. Wait for the deployment to complete before you proceed to the next task.

14

Task 3: Validate that VMSS instances are serving web content

1. In the hub menu in the Azure portal, click **Resource groups**.

13 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-2-deploy-a-vmss-using-arm

14 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-3-validate-that-vmss-instances-are-serving-web-content

2. On the **Resource groups** blade, click the entry representing the resource group into which you deployed the virtual machine scale set.
3. On the resource group blade, click the resource of the **Public IP address** type.
4. On the Public IP address resource blade, in the **Essentials** section, identify the value of **IP address** entry.
5. Open a new Microsoft Edge tab and navigate to the IP address you identified in the previous step.
6. Verify that you are able to access the default Internet Information Services webpage.
7. Close the new browser tab and return to the browser tab with the **Azure Portal** currently active.

Review: In this exercise, you created a Virtual Machine scale set and configured the individual instances using PowerShell DSC.

15

Exercise 3: Deploy Azure VMs running Windows Server 2016 and Linux by using Azure Building Blocks with PowerShell Desired State Configuration (DSC) extension from the Azure Cloud Shell.

16

Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
2. **Note:** The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
3. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.
4. **Note:** If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.
5. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
 - Leave the **Subscription** drop-down list entry set to its default value.
 - In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you intend to deploy resources in this exercise.
 - In the **Resource group** section, ensure that the **Create new** option is selected and then, in the text box, type **AADesignLab0303-RG**.

15 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#exercise-3-deploy-azure-vms-running-windows-server-2016-and-linux-by-using-azure-building-blocks-with-powershell-desired-state-configuration-dsc-extension-from-the-azure-cloud-shell

16 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-1-open-cloud-shell

- In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
 - In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
 - Click the **Create storage** button.
6. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

17

Task 2: Install the Azure Building Blocks npm package in Azure Cloud Shell

1. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to create a local directory to install the Azure Building Blocks npm package:

```
mkdir ~/.npm-global
```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to update the npm configuration to include the new local directory:

```
npm config set prefix '~/.npm-global'
```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to open the `~/.bashrc` configuration file for editing:

```
vi ~/.bashrc
```

4. At the **Cloud Shell** command prompt, in the vi editor interface, scroll down to the bottom of the file (or type **G**), scroll to the right to the right-most character on the last line (or type **\$**), type **a** to enter the **INSERT** mode, press **Enter** to start a new line, and then type the following to add the newly created directory to the system path:

```
export PATH="$HOME/.npm-global/bin:$PATH"
```

5. At the **Cloud Shell** command prompt, in the vi editor interface, to save your changes and close the file, press **Esc**, press **:**, type **wq!** and press **Enter**.
6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to install the Azure Building Blocks npm package:

```
npm install -g @mspn/azure-building-blocks
```

7. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to exit the shell:

17 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-2-install-the-azure-building-blocks-npm-package-in-azure-cloud-shell


```
exit
```

8. In the **Cloud Shell timed out** pane, click **Reconnect**.
9. **Note:** You need to restart Cloud Shell for the installation of the Building Blocks npm package to take effect.

18

Task 3: Deploy a Windows Server 2016 Azure VM from Cloud Shell by using Azure Building Blocks

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to download the GitHub repository containing the Azure Building Blocks reference architecture files:

```
git clone https://github.com/mspnp/reference-architectures.git
```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to view the content of the Azure Building Block parameter file you will use for this deployment:

```
cat ./reference-architectures/virtual-machines/single-vm/parameters/windows/single-vm.json
```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

```
SUBSCRIPTION_ID=$(az account list --query "[0].id" | tr -d '"')
```

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you created earlier in this exercise:

```
RESOURCE_GROUP='AADesignLab0303-RG'
```

5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

```
LOCATION=$(az group list --query "[?name == 'AADesignLab0301-RG'].location" --output tsv)
```

6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminUsername** parameter with the value **Student** in the Building Blocks parameter file:

```
sed -i.bak1 's/"adminUsername": ""/"adminUsername": "Student"/' ./reference-architectures/virtual-machines/single-vm/parameters/windows/single-vm.json
```

18 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-3-deploy-a-windows-server-2016-azure-vm-from-cloud-shell-by-using-azure-building-blocks

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminPassword** parameter with the value **Pa55w.rd1234** in the Building Blocks parameter file:

```
sed -i.bak2 's/"adminPassword": ""/"adminPassword": "Pa55w.rd1234"/' ./reference-architectures/virtual-machines/single-vm/parameters/windows/single-vm.json
```

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the parameter values were successfully changed in the Building Blocks parameter file:

```
cat ./reference-architectures/virtual-machines/single-vm/parameters/windows/single-vm.json
```

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy a Windows Server 2016 Azure VM by using the Azure Building Blocks:

```
azbb -g $RESOURCE_GROUP -s $SUBSCRIPTION_ID -l $LOCATION -p ./reference-architectures/virtual-machines/single-vm/parameters/windows/single-vm.json --deploy
```

- Wait for the deployment to complete before you proceed to the next task.

19

Task 4: Validate that the Windows Server 2016 Azure VM is serving web content

- In the hub menu in the Azure portal, click **Resource groups**.
- On the **Resource groups** blade, click the entry representing the resource group into which you deployed the Windows Server 2016 Datacenter virtual machine earlier in this exercise.
- On the resource group blade, click the entry representing the virtual machine you deployed.
- On the **Virtual machine** blade, locate the **Public IP address** entry, and identify its value.
- Open a new Microsoft Edge tab and navigate to the IP address you identified in the previous step.
- Verify that you are able to access the default Internet Information Services webpage.
- Close the new browser tab.

¹⁹ https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20Using%20ARM.md#task-4-validate-that-the-windows-server-2016-azure-vm-is-serving-web-content

20

Task 5: Deploy a Linux Azure VM from Cloud Shell by using Azure Building Blocks

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to view the content of the Azure Building Block parameter file you will use for this deployment:

```
cat ./reference-architectures/virtual-machines/single-vm/parameters/linux/single-vm.json
```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to generate the SSH key pair that you will use to authenticate when accessing the Linux VM:

```
ssh-keygen -t rsa -b 2048
```

- When prompted to enter the file in which to save the key, press **Enter** to accept the default value (`~/.ssh/id_rsa`).
- When prompted to enter passphrase, press **Enter** twice.

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the public key of the newly generated key pair:

```
PUBLIC_KEY=$(cat ~/.ssh/id_rsa.pub)
```

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the public key of the newly generated key pair and which takes into account any special character the public key might include:

```
PUBLIC_KEY_REGEX="$(echo $PUBLIC_KEY | sed -e 's/\\/\\\\\\\\/g; s/\\/\\\\\\\\\\\\/g; s/&/\\\\\\\\&/g') "
```

5. **Note:** This is necessary because you will use the **sed** utility to insert this string into the Azure Building Blocks parameter file. Alternatively, you could simply open the file and enter the public key string directly into the file.

6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

```
SUBSCRIPTION_ID=$(az account list --query "[0].id" | tr -d '"')
```

7. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you will use for the deployment:

```
RESOURCE_GROUP='AADesignLab0304-RG'
```

8. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

20 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-5-deploy-a-linux-azure-vm-from-cloud-shell-by-using-azure-building-blocks

```
LOCATION=$(az group list --query "[?name == 'AADesignLab0301-RG'].location"
--output tsv)
```

9. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminUsername** parameter with the value **Student** in the Building Blocks parameter file:

```
sed -i.bak1 's/"adminUsername": ""/"adminUsername": "Student"/' ./reference-architectures/virtual-machines/single-vm/parameters/linux/single-vm.json
```

10. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **sshPublicKey** parameter with the value of the **\$PUBLIC_KEY_REGEX** variable in the Building Blocks parameter file:

```
sed -i.bak2 's/"sshPublicKey": ""/"sshPublicKey": ""$PUBLIC_KEY_REGEX""/' ./reference-architectures/virtual-machines/single-vm/parameters/linux/single-vm.json
```

11. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the parameter values were successfully changed in the Building Blocks parameter file:

```
cat ./reference-architectures/virtual-machines/single-vm/parameters/linux/single-vm.json
```

12. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new resource group:

```
az group create --name $RESOURCE_GROUP --location $LOCATION
```

13. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy a Linux Azure VM by using the Azure Building Blocks:

```
azbb -g $RESOURCE_GROUP -s $SUBSCRIPTION_ID -l $LOCATION -p ./reference-architectures/virtual-machines/single-vm/parameters/linux/single-vm.json --deploy
```

14. Wait for the deployment to complete before you proceed to the next task.

21

Task 7: Validate that the Linux Azure VM is serving web content

1. In the hub menu in the Azure portal, click **Resource groups**.

21 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-7-validate-that-the-linux-azure-vm-is-serving-web-content

2. On the **Resource groups** blade, click the entry representing the resource group into which you deployed the virtual machine.
3. On the resource group blade, click the entry representing the virtual machine you deployed.
4. On the **Virtual machine** blade, locate the **Public IP address** entry, and identify its value.
5. Open a new Microsoft Edge tab and navigate to the IP address you identified in the previous step.
6. Verify that you are able to access the default Apache2 Ubuntu webpage.
7. Close the new browser tab.
8. Close the **Cloud Shell** pane.

Review: In this exercise, you deployed Azure VMs running Windows Server 2016 Datacenter and Linux from Cloud Shell by using Azure Building Blocks.

22

Exercise 4: Remove lab resources

23

Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

```
az group list --query "[?starts_with(name, 'AADesignLab03')].name" --output tsv
```

3. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

24

Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

```
az group list --query "[?starts_with(name, 'AADesignLab03')].name" --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'
```

22 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#exercise-4-remove-lab-resources

23 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-1-open-cloud-shell-1

24 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod02_Building%20Azure%20IaaS-Based%20Server%20Applications%20by%20using%20ARM.md#task-2-delete-resource-groups

2. Close the **Cloud Shell** prompt at the bottom of the portal.

Review: In this exercise, you removed the resources used in this lab.

Review Questions

Module 2 Review Questions

Hardware faults

You design a cloud solution that uses multiple Azure virtual machines (VMs).
What can you do to protect from hardware or rack-based faults? What can you do to protect from datacenter faults?

Suggested Answer ↓

You can create an Availability Set to ensure that all virtual machines which are added to the set are safe from hardware faults or rack-based faults. Availability Zones keep your VMs running during a complete datacenter failure.

VM Scale sets

You design a cloud solution that uses multiple Azure virtual machines (VMs).
You need to ensure the VMs can auto-scale in parallel during peak workload periods.
What can you use to auto-scale the VMs? What can you use to maintain the deployment of applications on the VMs?

Suggested Answer ↓

An Azure VM scale set is used to horizontally auto-scale Azure VMs. You can use continuous delivery processes to maintain an application in a scale set by using Visual Studio Team Services.

Authentication

You design a cloud solution that uses on-premises and Azure-based resources.
You need to design a hybrid solution for authenticating resources.
What Azure service should you use?

Suggested Answer ↓

You can use Azure AD Connect and Active Directory Federation Services (AD FS) to connect Azure and on-premises resources to Azure AD.



Module 3 Module Networking Azure Application Components

Virtual Networks

Azure Virtual Network (VNET) Architecture

This module describes the various networking and connectivity options available for solutions deployed on Azure. The module explores connectivity options ranging from ad-hoc connections to long-term hybrid connectivity scenarios. The module also discusses some of the performance and security concerns related to balancing workloads across multiple compute instances, connecting on-premise infrastructure to the cloud and creating gateways for on-premise data.

After completing this module, students will be able to:

- Describe DNS and IP strategies for VNETs in Azure.
- Compare connectivity options for ad-hoc and hybrid connectivity.
- Distribute network traffic across multiple loads using load balancers.
- Design a hybrid connectivity scenario between cloud and on-premise.

Virtual Networks

Azure Virtual Networking is the baseline for all Azure networking resources and features. Starting from a high-level topology view and detailed walkthrough of what an Azure VNET is, we drill down into all aspects of Subnet designing, how to integrate networks across Azure regions, as well as between on-premises corporate networks and Azure regions. Next, we describe the different configuration settings within a VNET, like what DNS options are available and how IP-addressing is working for both Public and Private IP's.

After this section you will be able to:

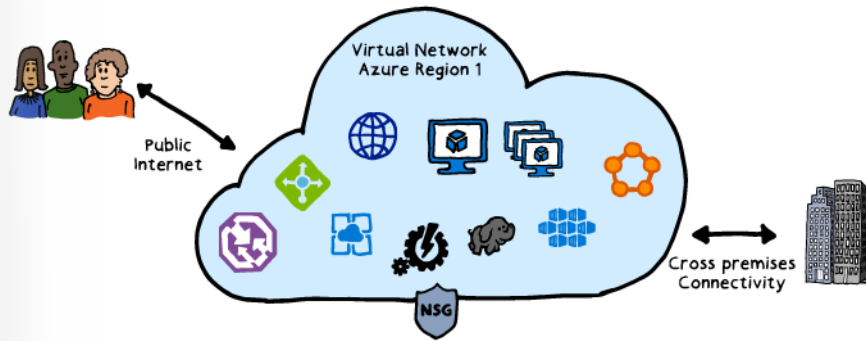
- Understand Azure Virtual Networks.
- Architect multi-region networking across Azure regions as well as between Azure and on-premises networks.

- Understand several Azure VNET configuration options like DNS, IP addressing and alike.

Multi-Region Virtual Network Architecture

An Azure Virtual Network (or VNET), is the logical unit of multiple or all network resources in an Azure region.

On the highest level of the network topology in an Azure Region, you define a Virtual Network. An Azure Region can have one or multiple Virtual Networks defined. Within a Virtual Network, you create one or more subnets. Like in a typical on-premises network, all traffic within a subnet is allowed, but communication across different subnets is blocked by default. Separating your workloads in multiple subnets within a VNET is a best practice and highly recommended.

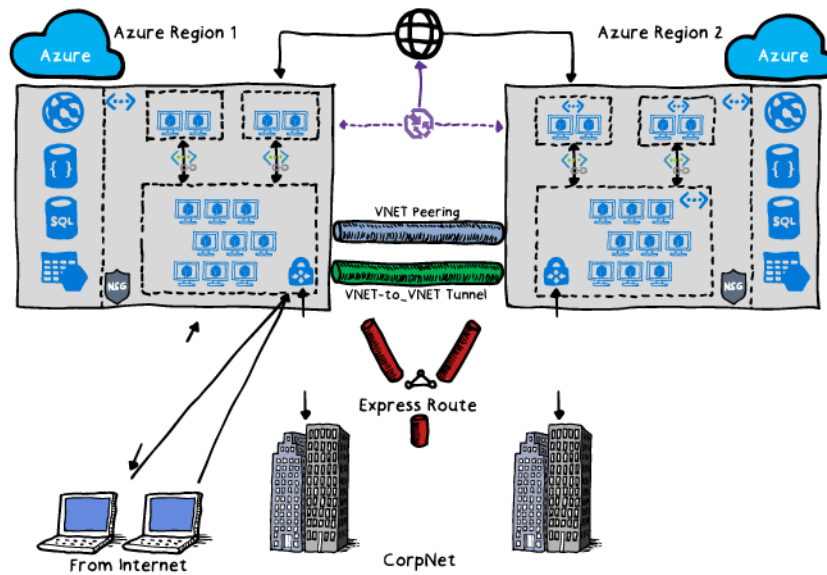


From a high-level perspective, Azure Virtual Networks allow for the following communication flows:

- From the Public Internet, incoming traffic is routed to Public IP addresses of Azure Resources, secured by Network Security Groups. (think of these as software-defined firewalling for now).
- Several Azure Resources are making use of VNETs and subnets for IP addressing. These Azure Resources can be IAAS related (Virtual Machines, VM Scale Sets, Load Balancers, Azure Traffic Manager,...) as well as PAAS related (Service Fabric, Azure Container Services, Hadoop, Azure Application Services,...).
- From a back-end perspective, Azure allows for a hybrid network integration between multiple Azure Regions, or with on-premises datacenters. In a next module, we will detail the different capabilities and configuration options.

We can assume an organization wants to leverage the capabilities of the Azure Public Cloud, by deploying workloads across multiple Azure Regions, or across Azure Regions and on-premises data centers. This scenario is entirely achievable in Azure:

- From the Public Internet, Azure allows for load balancing across multiple Azure Regions by deploying Azure Traffic Manager.



Interconnecting Azure Regions with each other is possible in three different ways:

- Configuring Azure Site-to-Site VPN between both regions
- Configuring Azure ExpressRoute communication tunnels
- A newer capability that allows for interconnecting multiple Azure regions is called Azure VNET Peering



Note: VNET Peering provides typical network communication, where VPN provides tunnel encryption—so it is up to the business requirements to make your decision.

Interconnecting Azure Regions with on-premises datacenters is possible in 2 different ways:

- Configuring Azure Site-to-Site VPN between Azure and on-premises
- Configuring Azure ExpressRoute communication tunnels between Azure and on-premises

Network Security Groups are active on Azure VNET layer or on individual NIC layer; however, NSGs cannot span Azure Regions. This means that you need to define the configuration within each VNET in each Azure Region.

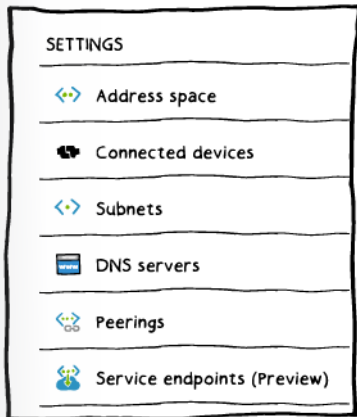
VNETs & Subnets

Networking Topology:

- Define one or more VNETs within an Azure Region, and configure an address space for each.
- Define one or more SubNets within a VNET, and configure address space within the VNET range.
- VNETs and SubNets are using CIDR notation (x.x.x.x/24, x.x.x.x/16,...).
- Configure Network Security Group settings on VNET level.
- Attach a NIC to a SubNet.

SubNet IP Addressing:

- IP-address gets allocated to a NIC during provisioning of the NIC
- First available IP-address in a SubNet range is x.x.x.4
- Azure SubNets support dynamic (=default) and static IP addressing
- Subnets can be configured directly in the Azure Portal.

**SUBNET CONFIGURATION****Public IP-addressing**

- Used for all public internet-facing communication
- Required parameter when creating a VM from the portal

Private IP-addressing

- Used for all inter-VNET communication
- Used for all communication between an Azure VNET and an on-premises VNET
- Azure DNS Resolving

DNS Server settings are configured on VNET level

Using Azure DNS is the default configuration setting, but this can be modified

Or use your custom DNS configuration:

- Azure DNS Appliance (from Azure Marketplace)
- Azure VM (e.g. Windows ADDS with DNS)
- On-premises DNS solution (requires connectivity)

Public DNS names (available for VMs and App Services) must be unique across Azure regions

An example of such Public DNS name is <host.region.cloudapp.azure.com>

Load Balancing

Load Balancing Solutions

Azure Load Balancing refers to several different Azure Resources that are available on the Azure Platform, offering application workload load balancing capabilities, much similar to traditional on-premises load balancing solutions. This lesson starts with describing the different flavors, zooming in on the characteristics of each of the available flavors.

After completing this section, you will be able to:

- Understand Azure Load Balancing.
- Recognizing the use case for each of the in-Azure provided Load Balancing solutions.
- Deciding between the different Azure Load Balancing options.

Load Balancing Solutions

Azure provides several built-in Azure Load Balancing Solutions:

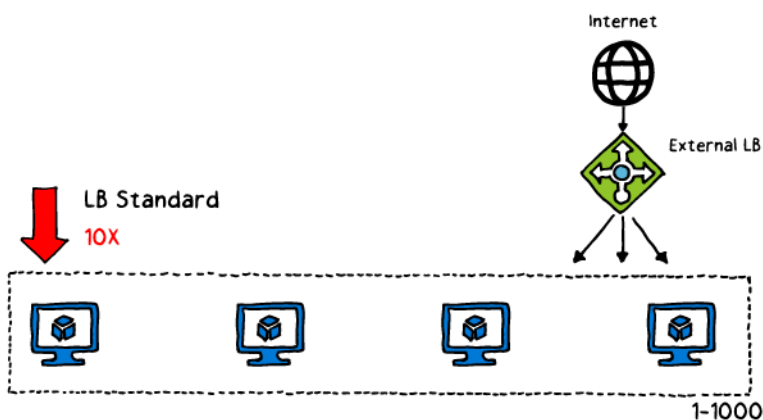
- Azure Load Balancer
- Azure Application Gateway
- Azure Marketplace Load Balancing Appliance
- Azure Traffic Manager

Azure Load Balancer

Starting with Azure Load Balancer, it is important to point out it can be configured both as an “external load balancer,” or as an “internal load balancer,” where one cannot act as both external and internal at the same time.

In the scenario of an Azure external load balancer, the load balancer front-end is configured with a Public-facing IP-address, sending all traffic along to the back-end pool servers, using their internal IP-addresses.

Azure Load Balancers can handle almost any TCP or UDP traffic; use case scenarios include RDS (Remote Desktop Services Farm), Linux SSH server connectivity load balancing, or any other application-specific traffic.



In relation to Azure Virtual Machine Availability Sets, where you deploy multiple instances of the same Virtual Machine, it is thanks to Azure Load Balancer functionality; an Availability Set can grow to 10x the number of instances (100 to 1000 in a single Availability Set).

Pretty similar in functionality is an Azure Internal Load Balancer. The main difference is that this solution doesn't have a Public-facing IP-address, and all communication is based on internal IP-addressing and IP-routing.

A typical use case for this setup is like the diagram shown, where you want to load balance incoming traffic between Azure Subnets or Azure VM Availability Sets. The external Load Balancer takes care of the incoming web traffic to the Web Server AVSet, where any Web Server VM can communicate with the Database Server backend, using the internal Azure Load Balancer option.

Azure Load Balancer existed in a Basic edition, which gave you the following characteristics:

- Up to 100 backend instances
- Single Availability Set
- Basic NAT and Probe health status
- No HA Ports
- Network Security Groups (NSG) are optional
- No cost related to using this Azure Load Balancer solution

While Azure Load Balancer was and still is a viable solution, it also had some technical limitations. Some of these are resolved in the Standard SKU of the Azure Load Balancer, giving you the following characteristics:

- Up to 1000 backend instances
- Availability Sets are not required; providing support for Availability Zones
- Integrated Frontend and Backend health metrics
- Support for HA Ports
- Network Security Groups are required during configuration and deployment

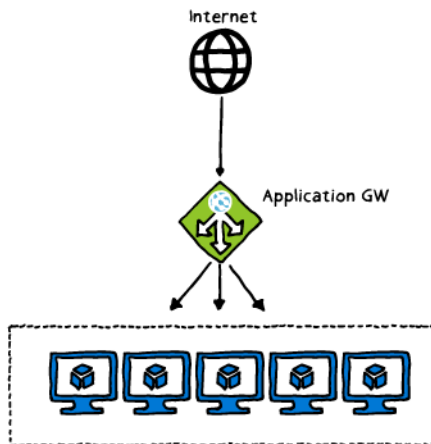
Azure Application Gateway

Another Azure Load Balancing solution on the platform is Azure Application Gateway. While mostly similar in functionality than the Azure Load Balancer, we like to focus on its specifics:

Load Balancing, active on Layer 7 of the network stack; this mainly means it is "application intelligent."

Main features Application Gateway provides, compared to Azure Load Balancer, are:

- HTTP/HTTPS traffic only, no other ports allowed
- SSL Offloading
- Cookie Affinity
- Web Application Firewall (WAF)
- URL Based Routing



In the scenario of URL based Routing, Application Gateway recognized the incoming web request and based on the URL information; it will redirect traffic to the correct destination. Besides main URL redirection, this can also be used on subheaders. For example, in the given scenario, web requests to the **<http://www.domain2.com/finance>** web app will be redirected to WebAV1, where all requests to the **<http://www.domain2.com/sales>** web app, will be redirected to another WebAV2 availability set.

In regard to SSL Termination, Azure App Gateway provides the following capabilities:

- SSL Offloading, by importing the SSL Certificate onto the App Gateway; traffic to the backend servers don't require HTTPS communication, also that would still be an option.
- HTTP to HTTPS redirect; this means that, whenever a user is connecting to the web app using HTTP, the request will be redirected to HTTPS, forcing SSL Tunneling for this given request.







The last feature of Azure Application Gateway we want to discuss here is Web Application Firewall (WAF). Based on industry-standard rules for WAF, CRS 2.2.9 and CRS 3.0, Azure Application Gateway provides protection against several common attacks and threats on application workloads:








- SQL Injection
- Cross-site scripting
- Protocol violations
- Generic attacks
- HTTP rate limiting
- Scanner detection
- Session fixation
- LFI/RFI

Azure Load Balancing Marketplace Appliances

Where both Azure Load Balancer and Azure Application Gateway are an interesting option, directly built into the Azure Platform and offered "as a service," the capabilities around management, monitoring and control might be too limited for certain organizations. In that scenario, one can deploy a third-party Azure Marketplace appliance.

The “common” vendors from the on-premises world are present in an Azure Appliance alternative. Support is initially provided by Microsoft, backed by SLA’s, and acting as a SPOC.

	Barracuda NextGen Firewall F-Series (BYOL) Barracuda Networks, Inc.
	FortiGate NGFW High Availability (HA) Fortinet
	FortiGate NGFW Single VM Fortinet
	Fortinet Web Application Firewall - FortiWeb Fortinet
	A10 vThunder ADC BYOL A10 Networks
	BYOL Load Balancer, ADC & WAF - Trial & Perpetual KEMP Technologies Inc

Appliances	
	VM-Series Next Generation Firewall (BYOL) Palo Alto Networks, Inc.
	Cisco ASA v - BYOL 4 NIC Cisco Systems, Inc.
	F5 WAF Solution for ASC F5 Networks
	Riverbed SteelCentral AppInternals APM Riverbed Technology
	Barracuda Web Application Firewall (WAF) - PAYG Barracuda Networks, Inc.
	KEMP 360 Central (BYOL) KEMP Technologies Inc
	Cisco CSR 1000v - XE 16.4 Deployment with 2 NICs Cisco Systems, Inc.

Most flavors support 2 different licensing models to choose from:

- **BYOL:** Bring Your Own License; this is an ideal candidate if you are removing or downsizing on your on-premises running third-party load balancer. Depending on the specific licensing terms of the vendor, one can reuse the license key on the Azure VM Appliance.
- **Pay-Per-Use:** In this model, the monthly Azure VM consumption cost is based on the VM Size allocation to the Appliance, as well as a monthly licensing fee for the third party load balancing application within the VM.

Azure Traffic Manager

Microsoft Azure Traffic Manager allows you to control the distribution of user traffic to your specified endpoints, which can include Azure cloud services, websites, and other endpoints. Traffic Manager works by applying an intelligent policy engine to Domain Name System (DNS) queries for the domain names of your internet resources. Your Azure cloud services or websites can be running in different datacenters across the world.

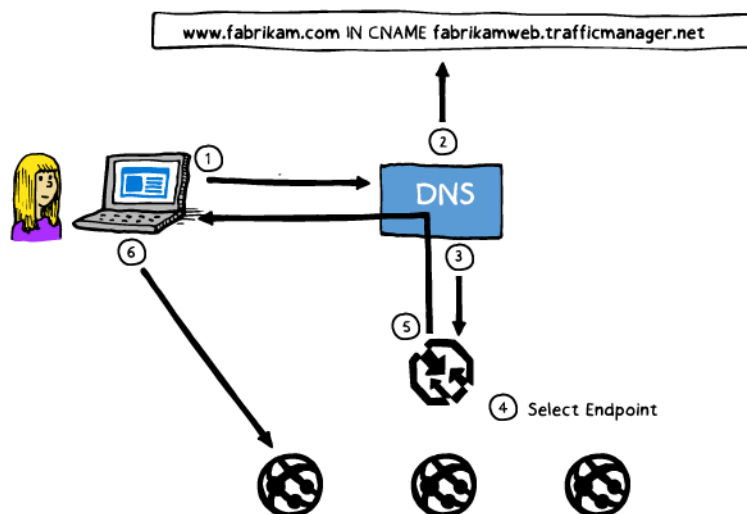
Traffic Manager is very flexible because it allows you to mix various endpoints behind the same DNS name. Traffic Manager can be used in a variety of scenarios but most use cases fall in the following scenarios:

- **Failover:** Traffic Manager can poll to determine if an endpoint is online or offline. The endpoints are then ordered in a priority list. By default, traffic routes to the first endpoint. If the first endpoint is down, traffic routes to the next endpoint (2) in the list. Traffic Manager will route requests to the endpoint that is the highest in the priority list and is still online. Using this method, you can have Traffic Manager route traffic to primary or backup datacenters/services for a simple failover scenario.
- **Geography:** Traffic Manager uses an Internet Latency Table to determine the endpoint that is "closest" to the client that is making a request. Using this method, an application can be hosted in West Europe and West US. A user from Denmark can reasonably expect to be served by the endpoint residing in the West Europe datacenter and should experience lower latency and higher responsiveness.
- **Distribution:** Traffic Manager can distribute traffic in a near-random way to distribute traffic evenly across a set of endpoints. If a specific endpoint is down, the traffic is distributed evenly across the remaining endpoints. The distribution can optionally be weighted so that certain endpoints receive more requests than others. The weighted distribution is especially useful if you want to distribute a small subset of your traffic to a hot disaster recovery site that is using smaller service tiers but keep the majority of your traffic to a primary site that is using larger service tiers.

Using Traffic Manager with Web Apps

Traffic Manager can be integrated with Web Apps easily. When you configure a Traffic Manager profile, the settings that you specify provide Traffic Manager with the information needed to determine which endpoint should service the request based on a DNS query. No actual endpoint traffic routes through Traffic Manager.

The below diagram shows how Traffic Manager directs users to one of a set of endpoints:



TRAFFIC MANAGER WORKFLOW

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

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

External Connectivity

On-Premises to Azure Connectivity

Azure Hybrid Connectivity discusses several options available within the Azure Platform, to establish end-to-end hybrid connectivity. This can be between an on-premises network and one or more Azure regions, or between multiple Azure regions. This lesson discusses some reference architectures, the different solutions available for each challenge, and some overall guidelines and concepts.

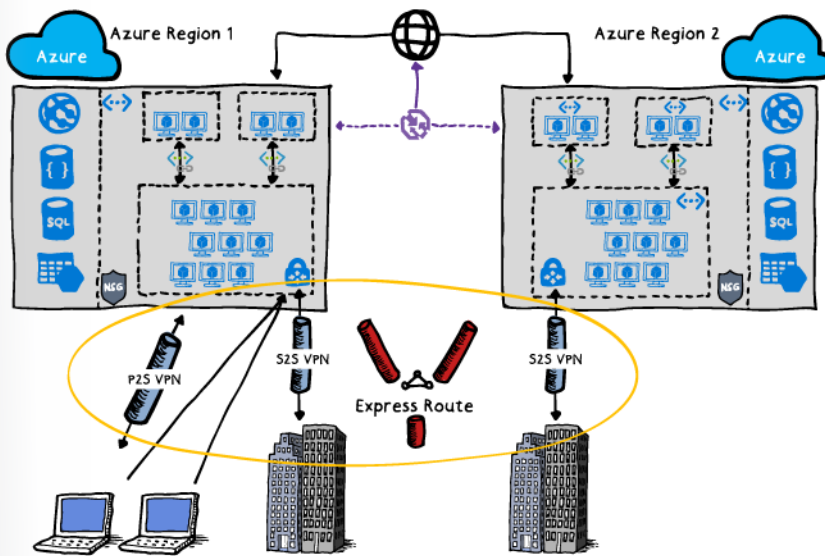
After completing this lesson, you will be able to:

- Understand Azure Hybrid Connectivity.
- Architecting Azure VNET Peering and understanding the use cases.
- Understanding the differences between Site-to-Site VPN and Point-to-Site VPN.
- Understand the key capabilities and characteristics of Azure ExpressRoute.

On-Premises to Azure Connectivity

There are three primary options to connect your on-premises data center to Azure, described in the following table.

Connectivity	Benefits
ExpressRoute	<ul style="list-style-type: none"> • ExpressRoute as primary cross-premises connectivity • Multiple circuits for redundancy & better routing • ExpressRoute-VPN co-existence for highly available, redundant paths
Site-to-Site VPN	<ul style="list-style-type: none"> • S2S VPN over internet for remote branch locations • BGP & active-active configuration for HA and transit
Point-to-Site VPN	<ul style="list-style-type: none"> • P2S VPN for mobile users & developers to connect from anywhere with macOS & Windows • AD/radius authentication for enterprise grade security



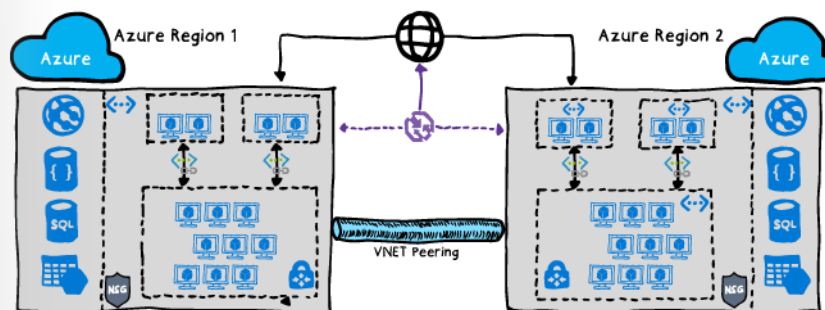
VNET Peering

VNET peering allows you to interconnect 2 Azure Regions with each other, using the Microsoft Backbone (not the public internet). Communication relies on internal IP addressing.

Here are some of the primary features of VNET Peering:

- VNET Peering allows you to interconnect 2 Azure VNET as if they are 1 large VNET.
- VNET Peering is possible within the same Azure region, or across Azure regions (using MS Backbone, no public internet).
- VNET Peering is supported to interconnect an Azure Classic VNET with an ARM VNET (e.g., For migrating workloads).

If VNET peering is not an option, because you might want to encrypt your traffic within the VNET tunnel, one can still deploy a VPN Gateway on both Azure Regional VNETs and creating a Site-to-Site VPN tunnel across those regions.



Multi-Region VPN Connectivity

Forced Tunneling

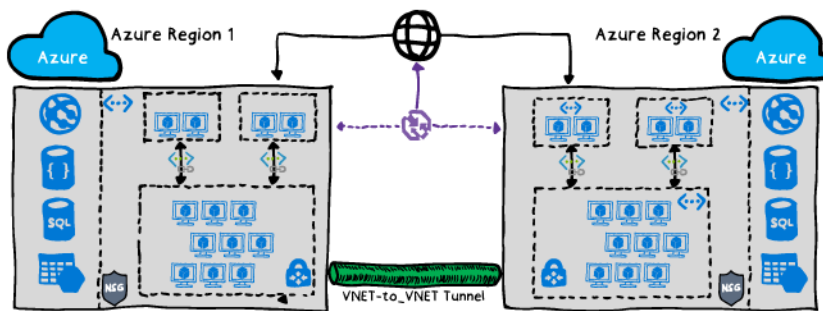
By using Forced Tunneling, Azure traffic can be rerouted to an on-premises virtual network, to be routed through an existing Site-to-Site VPN or ExpressRoute, into the internal Azure VNET.

This is a massive improvement from a security standpoint, as your internal Azure VMs are no longer accessible through the public internet.

Securing Access to PaaS Services

A similar concept now exists, to limit network access to PaaS Services in Azure. In a typical scenario, these PaaS services are/were accessible from the public internet. But that's not what all customers want. Maybe you want your application services endpoints in PaaS to be only accessed from the internal Azure VNETs.

The solution to this is using VNET service endpoints, where you define which PaaS services are no longer accessible through the public internet.



Note: For now, this feature is only available to Azure Storage Accounts, SQL DB Services in PaaS and Web Apps. More PaaS Services will be integrated with this feature soon enough.

Secure Connectivity

Network Security Groups

This section briefly talks about Network Security Groups and how they can be used to secure connections in a hybrid or external connectivity scenario.

After completing this lesson, you will be able to:

- Determine how and when to use Network Security Groups to secure access to and from a VM.

Network Security Groups

Network security groups are different than endpoint-based ACLs. Endpoint ACLs work only on the public port that is exposed through the input endpoint. An NSG works on one or more VM instances and controls all the traffic that is inbound and outbound.

You can associate an NSG to a VM, or to a subnet within a VNet. When associated with a VM, the NSG applies to all the traffic that is sent and received by the VM instance. When applied to a subnet within your VNet, it applies to all the traffic that is sent and received by ALL the VM instances in the subnet. A VM or subnet can be associated with only 1 NSG, and each NSG can contain up to 200 rules. You can have 100 NSGs per subscription on the VM.

Managing Network Security Groups

A NSG is a top level object that is associated to your subscription. An NSG contains access control rules that allow or deny traffic to VM instances. The rules of an NSG can be changed at any time, and changes are applied to all associated instances.

A network security group has a Name, is associated to a Region, and has a descriptive label. It contains two types of rules, Inbound and Outbound. The Inbound rules are applied on the incoming packets to a VM and the Outbound rules are applied to the outgoing packets from the VM. The rules are applied at the host where the VM is located. An incoming or outgoing packet has to match an Allow rule for it be permitted, if not it will be dropped.

Rules are processed in the order of priority. For example, a rule with a lower priority number (e.g. 100) is processed before rules with a higher priority numbers (e.g. 200). Once a match is found, no more rules are processed.

Default Network Security Group Rules

An NSG contains default rules. The default rules cannot be deleted, but because they are assigned the lowest priority, they can be overridden by the rules that you create. The default rules describe the default settings recommended by the platform. While connectivity to the internet is allowed for Outbound direction, it is by default blocked for Inbound direction. There is a default rule to allow Azure's load balancer (LB) to probe the health of the VM. You can override this rule if the VM or set of VMs under the NSG does not participate in the load balanced set.

Inbound

NAME	PRIORITY	SOURCE IP	SOURCE PORT	DESTINATION IP	DESTINATION PORT	PROTOCOL	ACCESS
ALLOW VNET IN-BOUND	65000	VIRTUAL_NETWORK	*	VIRTUAL_NETWORK	*	*	ALLOW
ALLOW AZURE LOAD BALANCER IN-BOUND	65001	AZURE_LOADBALANCER	*	*	*	*	ALLOW
DENY ALL IN-BOUND	65500	*	*	*	*	*	DENY

Outbound

NAME	PRIORITY	SOURCE IP	SOURCE PORT	DESTINATION IP	DESTINATION PORT	PROTOCOL	ACCESS
ALLOW VNET OUT-BOUND	65000	VIRTUAL_NETWORK	*	VIRTUAL_NETWORK	*	*	ALLOW
ALLOW INTERNET OUT-BOUND	65001	*	*	INTERNET	*	*	ALLOW
DENY ALL OUT-BOUND	65500	*	*	*	*	*	DENY

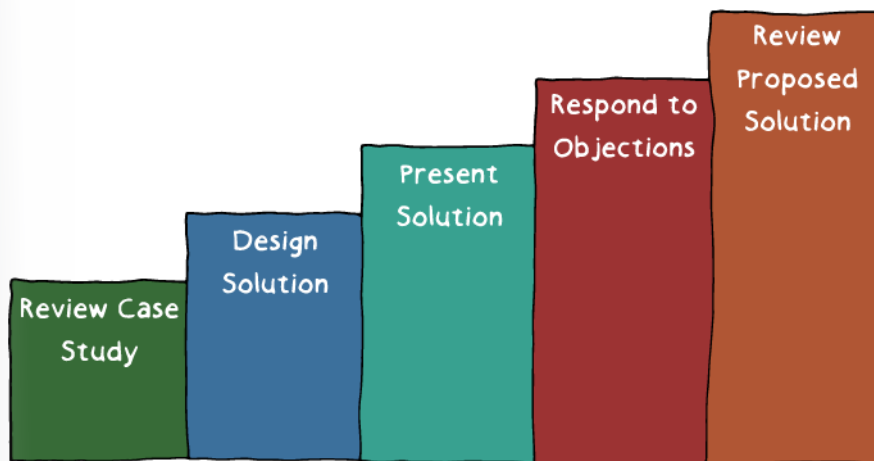
Design - Networking Case Study

Who Is the Customer?

In this case study, we will look at a customer problem that requires an architectural recommendation.

After this case study, you should:

- Identify customer problems as they are related to networking.
- Design a solution that will meet the customer's objectives.
- Ensure your designed solution accounts for customer objections.



Who Is the Customer?

Fabrikam Residences (<http://fabrikamresidences.com>) is a national real estate services group whose rapid growth was being slowed by an expensive and unresponsive datacenter infrastructure.

Fabrikam has two data centers in the United States, but it really doesn't want to be in the datacenter business. "We are a national real estate firm," says Craig Jones, Chief Information Officer for Fabrikam. "We want to make investments that support our core business, and buying and managing servers is not our core business. In fact, we have what we call a DOS strategy 'don't own stuff.' We were not an asset-intensive organization in any area but IT, where we had many underutilized assets."

What Does the Customer Already Have?

Fabrikam has about 250 servers in its datacenter in California, and another 110 in its Virginia datacenter, and hundreds of servers scattered across several branch offices throughout the United States. Fabrikam ended up overprovisioning servers each time it deployed an application to ensure that capacity would be there at peak times. This meant that millions of dollars' worth of hardware and software was sitting idle much of the time.

In addition to the primary data centers, Fabrikam also has several branch offices scattered across the United States that have connectivity to the primary data center through an MPLS based wide area network. Their partner is a Microsoft Azure ExpressRoute partner. To reduce costs, Fabrikam has made the decision to move its West coast datacenter to a colocation site in Silicon Valley and to virtualize the remainder of the servers in its branch offices and Virginia data center into the cloud. Fabrikam's current

virtualization and management solution is based on System Center so a solution that integrates well with these known tools is ideal.

What Is the Customer's Goal?

What Is the Customer's Goal?

Fabrikam Residences would like to eventually migrate the majority of their workloads to Azure. There are several workloads that will be migrated, but the most critical for Fabrikam is their CRM application.

The CRM application is a custom web application that runs on IIS 8 and SQL Server 2012 that stores sensitive documents for all of their customer's transactions. This application needs to perform well at peak time while mitigating the problem of overprovisioned capacity. The centralized nature of the application means that any downtime will block the activity of a significant portion of the company so the solution must be highly available. Due to the sensitive nature of this application security is key so access to the application is restricted to only authorized users from the corporate network including branch offices.

What Does the Customer Need?

- Reduce the number of existing on-premises servers through public cloud consolidation to reduce the costs of their current overprovisioned deployments. Servers running in the Virginia data center and remote branch offices will be moved to the closest Azure region. Due the sheer amount of servers being virtualized latency and performance of the network is a big concern.
- Because of the sensitive nature of the data that Fabrikam Real Estate works with ensuring the security and privacy of their infrastructure connects through is critical.
- As part of the migration efforts the CRM application must be deployed in a way that mitigates their current problem of overprovisioning capacity when not needed but able to scale to meet peak demand. The CRM application must be highly available and only accessible from the corporate intranet.

What Things Worry the Customer?

- We have a national business and we need connectivity that can accommodate connectivity from coast-to-coast.
- Our workloads are very seasonal. I do not want to pay for more resources than I need.
- The data that crosses our network is very confidential. Is Azure
- I need to deploy an intranet-based solution and I have heard that Azure requires an on-premises load-balancer for internal facing workloads.
- I have heard that the public IP address of an Azure deployment can change and break my application.
- My workloads require static IP addresses. I have heard Azure does not support this scenario.
- I have some workloads that require multiple network interfaces on my virtual machines.
- Some deployments require the segmenting of network traffic. Does Azure support this?

Case Study Solution

Preferred Target Audience: Craig Jones, Chief Information Officer for Fabrikam.

The primary audience is the business decision makers and technology decision makers. From the case study scenario, this would include the IT Director, Network Administrator and Security Lead.

Preferred Solution

Fabrikam Residences went with Azure ExpressRoute. They already had a relationship with a Network Service Provider and their MPLS WAN was already in place so it made logical sense to extend their network with Azure. ExpressRoute provides the secure and private connection they need to ensure the privacy of their customer records along with the high speed and low latency connectivity their workloads require.

The first step was to configure ExpressRoute by first contacting AT&T and start the onboarding process for ExpressRoute.

After a circuit was in place with their provider, the next step was to implement Azure Virtual Networks in each of the regions where they would be migrating workloads to the cloud.

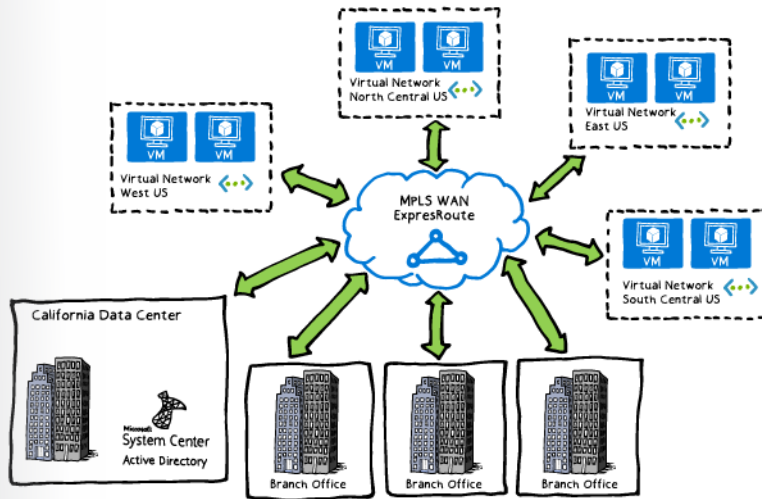
They connected the virtual networks across several regions where their branch offices are located to the ExpressRoute circuit using the PowerShell **New-AzureDedicatedCircuitLink** cmdlet.

The next step was to deploy Active Directory in each of the regions. Each Active Directory DC should be deployed with a static IP into a subnet that does not contain non-static IP based VMs. There should be at least two DCs for redundancy and deployed into an availability set. An Active Directory site should be configured so authentication requests stay local.

The final step was to architect the solution for their CRM solution to ensure it met the requirements of being secure, highly available, and easily scalable for peak demand. The CRM web servers are deployed into an availability set and auto scale is configured to avoid over provisioning. Access to CRM is through an internally load-balanced endpoint and the SQL Server deployment uses **AlwaysOn** availability groups as well as an internal load balanced IP for the listener.

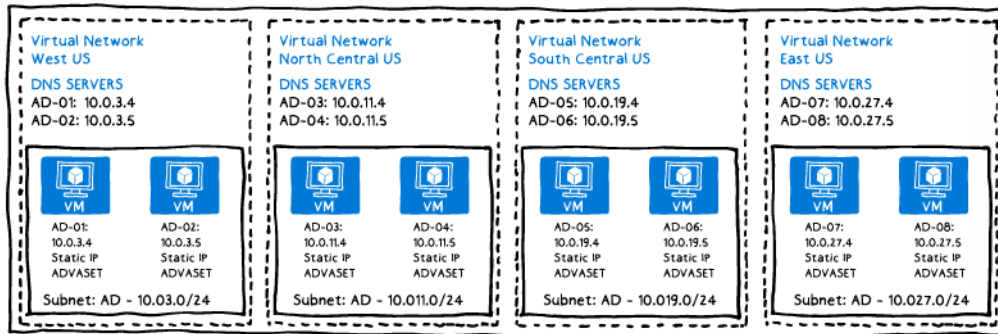
Example of a Preferred Solution

Deploying ExpressRoute with MPLS Network



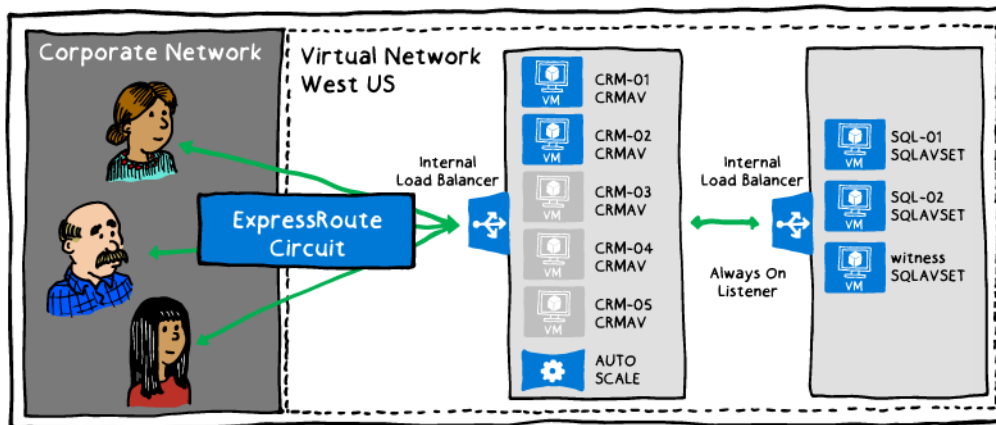
DEPLOYING EXPRESSROUTE

Each region should have at least two domain controllers configured within an availability set. Active Directory sites and site links should be configured to authentication to the local DCs first.



DEPLOYMENT OF ACTIVE DIRECTORY

The CRM webservers are deployed with auto scale enabled and configured using the internal load balancer. SQL Always On is configured using the internal load balancer.



CUSTOM CRM APPLICATION DEPLOYMENT

Checklist of Potential Benefits

Increased Security/Privacy and Network Performance for Enterprise Connectivity

Azure ExpressRoute provides a dedicated connection between an organizations network to Microsoft Azure through an ExpressRoute partner. Your connection is dedicated bandwidth between your infrastructure and Microsoft Azure. This committed bandwidth and additional control gives you predictable network performance.

Your connection is private. Traffic that flows between your on-premises network and Microsoft Azure does not traverse the public internet and is isolated using industry standard VLANs. ExpressRoute connections support high throughput and low latency for some of the most bandwidth hungry applications and workloads.

For workloads where large amounts of data are leaving the Microsoft Azure data center ExpressRoute can save significant amounts of money due to the included bandwidth and lower bandwidth costs that come with an ExpressRoute circuit.

Public Peering

In addition to providing private connectivity between your on-premises network and your Azure virtual networks you can enable public peering which provides private connectivity to a number of Azure public services.

Cross Region Connectivity

Azure ExpressRoute makes it simple to connect multiple virtual networks to the same ExpressRoute circuit as long as the virtual networks are on the same continent. This allows you to extend your on-premises network to multiple Azure regions.

Network Service Provider Model of ExpressRoute Provides a Simple Integration Point

Fabrikam already has an existing MPLS VPN with a Network Service Provider. With ExpressRoute choosing the Network Service Provider model the provider is responsible for onboarding your network into Azure. They take care of the routing configuration and ensuring everything works. You are still responsible for choosing your service tier and creating the circuit in Azure that they will setup. Bandwidth options for a Network Service Provider range from 10 Mbps all the way up to 1 Gbps. With a network service provider bandwidth is unlimited and not separately charged.

When a virtual network in Azure is configured in this manner the virtual network will be accessible to you just like any other site on your wide area network.

Checklist of Preferred Proof of Concept Potential Flow/Scope

Checklist of Preferred Proof of Concept Potential Flow/Scope

Objectives:

- Identify connectivity and latency requirements for proof of concept (regions and site connectivity).
- Configure AD sites/subnets to ensure efficient replication and authentication for the sites.
- Deploy custom CRM intranet based workload securely and without over provisioning.
- Demonstrate that Azure Virtual Machines and Virtual Networks can deliver the connectivity requirements for the solution.
- Address and resolve technical issues involved with connecting and deploying the virtual machines in the proof of concept

Flow/Scope of the proof of concept (a list of 2-3 bullets):

- Contact Network Service Provider and sign up for Azure ExpressRoute.
- Provision a circuit in Azure and work with Network Service Provider to connect the new circuit to the existing MPLS network.
- Identify services (virtual machines), regions and on-premises sites to design the network architecture.
- Configure Azure Virtual Networks for connectivity:
 - Design subnets for each virtual network to accommodate growth but not overlap any of the on-premises sites or other virtual networks.
 - Define the local network sites that the virtual networks will connect to. This may be other virtual networks (connecting across regions) or connecting to one or more of the branch offices.
 - Create the dynamic gateways at each virtual network site.
- Connect each virtual network to the ExpressRoute circuit using the **New-AzureDedicatedCircuitLink** PowerShell cmdlet.

- Deploy the virtual machines for Active Directory. There should be two domain controllers in an availability set for each virtual network. The virtual network should reference the IP addresses of both DCs. Sites and site links should be configured per virtual network to ensure Active Directory traffic stays local.
- Deploy the virtual machines for the custom CRM intranet based workload into a virtual network using an internal load balancer configuration for the web servers and another internal load balancer for the SQL Server Always On listener.

Conditions of satisfaction / success criteria for the PoC:

- Demonstrate that Azure Virtual Networks and Virtual Machines can provide the connectivity requirements and capacity for their virtualization.
- Ensure that AD replication and authentication occurs.
- Ensure the CRM intranet solution scales without overprovisioning and connectivity is secure.

Resources / Bill of Materials that you would use:

- Azure Virtual Networks and Virtual Machines
- Azure ExpressRoute and Network Service Provider partner
- Partner / MCS

Checklist of Preferred Objection Handled

We have a national business and we need connectivity that can accommodate coast-to-coast.

- Microsoft Azure ExpressRoute can provide connectivity to virtual networks on the same continent. The virtual networks do not even have to reside in the same Azure Subscription.

I need to deploy an intranet-based solution and I have heard that Azure requires an on-premises load balancer for internal facing workloads.

- Microsoft Azure now supports configuring an internal load-balancer using an internal IP address from your virtual network. You can load balance up to 50 virtual machines in a single load-balanced set.

I have heard that the public IP address of an Azure deployment can change and break my application.

- Azure virtual machines now support reserved IP addresses. Reserved IPs allow you to assign an IP address to a virtual machine deployment as the public IP. Even if you shut down all of the virtual machines or delete them and recreate them you can re-use the reserved IP address.

My workloads require static IP addresses. I have heard Azure does not support this scenario.

- Microsoft Azure now supports deploying virtual machines with static IP addresses in virtual networks.

Network security is critical to our business. Is Azure secure?

- Customers often make broad statement that we cannot use public cloud because of the security concerns. We need to make sure we understand their specific concerns. Usually it falls in either of the four buckets:

- **Trust**—To build trust make sure customer is aware of Microsoft history & experience of delivering cloud services at scale, take them to datacenter tours and take accountability of their success.
- **Privacy**—Privacy is one of the foundations of Microsoft's Trustworthy Computing. Microsoft has a longstanding commitment to privacy, which is an integral part of our product and service lifecycle. Share the Microsoft Azure Privacy Statement that describes the specific privacy policy and practices that govern customers' use of Microsoft Azure.
- **Compliance**—Microsoft partners with customers to help them address a wide range of international, country/region, and industry-specific regulatory requirements. Microsoft provides Microsoft Azure customers with detailed information about our security and compliance programs, including audit reports and compliance packages, to help customers assess our services against their own legal and regulatory requirements.
- **Security of Infrastructure and services**—Microsoft Azure runs in geographically dispersed datacenters that comply with key industry standards, such as ISO/IEC 27001:2005, for security and reliability. They are managed, monitored, and administered by Microsoft operations staff (Global foundation Services) that have years of experience in delivering the world's largest online services with 24 x 7 continuity.



For more information, visit Microsoft Azure Trust Center and familiarize yourself with Microsoft Azure Security practices. (Links are provided further in this section.)

AWS supports setting ACLs within subnets of their VPC.

- Azure supports setting ACLs on endpoints. As a workaround you can isolate networks in this manner. You can also use the firewall within the guest OS (if Windows you can use Group Policy).



Reference Link: <https://azure.microsoft.com/blog/new-windows-azure-network-security-white-paper/>



Reference Link: <http://azure.microsoft.com/support/trust-center/>

Online Lab - Deploying Network Infrastructure for Use in Azure Solutions

Lab Steps

Deploying Network Infrastructure for Use in Azure Solutions

NOTE: For the most recent version of this online lab, see: <https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign>

Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
 - Username: **Admin**
 - Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
 - Microsoft Edge
 - File Explorer
 - **Visual Studio Code**¹
 - **Microsoft Azure Storage Explorer**²
 - Bash on Ubuntu on Windows
 - Windows PowerShell
3. **Note:** You can also find shortcuts to these applications in the **Start Menu**.

3

Exercise 1: Configure the lab environment

4

Task 1: Open the Azure Portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (<https://portal.azure.com>).

¹ <https://code.visualstudio.com/>

² <https://azure.microsoft.com/features/storage-explorer/>

³ https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod03_Deploying%20Network%20Infrastructure%20for%20use%20in%20Azure%20Solutions.md#exercise-1-configure-the-lab-environment

⁴ https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod03_Deploying%20Network%20Infrastructure%20for%20use%20in%20Azure%20Solutions.md#task-1-open-the-azure-portal

3. When prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

5

Task 2: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
2. **Note:** The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
3. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.
4. **Note:** If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.
5. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
 - Leave the **Subscription** drop-down list entry set to its default value.
 - In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you intend to deploy resources in this lab
 - In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab0901-RG**.
 - In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
 - In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
 - Click the **Create storage** button.
6. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

6

Task 3: Install the Azure Building Blocks npm package in Azure Cloud Shell

1. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to create a local directory to install the Azure Building Blocks npm package:

```
mkdir ~/.npm-global
```

5 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod03_Deploying%20Network%20Infrastructure%20for%20use%20in%20Azure%20Solutions.md#task-2-open-cloud-shell

6 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod03_Deploying%20Network%20Infrastructure%20for%20use%20in%20Azure%20Solutions.md#task-3-install-the-azure-building-blocks-npm-package-in-azure-cloud-shell

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to update the npm configuration to include the new local directory:

```
npm config set prefix '~/.npm-global'
```

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to open the ~/.bashrc configuration file for editing:

```
vi ~/.bashrc
```

- At the **Cloud Shell** command prompt, in the vi editor interface, scroll down to the bottom of the file (or type **G**), scroll to the right to the right-most character on the last line (or type **\$**), type **a** to enter the **INSERT** mode, press **Enter** to start a new line, and then type the following to add the newly created directory to the system path:

```
export PATH="$HOME/.npm-global/bin:$PATH"
```

- At the **Cloud Shell** command prompt, in the vi editor interface, to save your changes and close the file, press **Esc**, press **:**, type **wq!** and press **Enter**.
- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to install the Azure Building Blocks npm package:

```
npm install -g @mspn/azure-building-blocks
```

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to exit the shell:

```
exit
```

- In the **Cloud Shell timed out** pane, click **Reconnect**.
- Note:** You need to restart Cloud Shell for the installation of the Building Blocks npm package to take effect.

7

Task 3: Prepare Building Blocks Hub and Spoke parameter files

- In the **Cloud Shell** pane, click the **Upload/Download files** icon and, in the drop-down menu, click **Upload**.
- In the **Open** dialog box, navigate to the \allfiles\AZ-301T04\Module_03\LabFiles\Starter\ folder, select the **hub-nva.json** file, and click **Open**.
- Repeat the previous step to upload to **Cloud Shell** the remaining files in the \allfiles\AZ-301T04\Module_03\LabFiles\Starter\ folder, including **hub-vnet.json**, **hub-vnet-peering.json**, **spoke1.json**, and **spoke2.json**.

⁷ https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod03_Deploying%20Network%20Infrastructure%20for%20Use%20in%20Azure%20Solutions.md#task-3-prepare-building-blocks-hub-and-spoke-parameter-files

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminUsername** parameter with the value **Student** in the **hub-vnet.json** Building Blocks parameter file:

```
sed -i.bak1 's/"adminUsername": ""/"adminUsername": "Student"/' ~/hub-vnet.json
```

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminPassword** parameter with the value **Pa55w.rd1234** in the **hub-vnet.json** Building Blocks parameter file:

```
sed -i.bak2 's/"adminPassword": ""/"adminPassword": "Pa55w.rd1234"/' ~/hub-vnet.json
```

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the parameter values were successfully changed in the **hub-vnet.json** Building Blocks parameter file:

```
cat ~/hub-vnet.json
```

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminUsername** parameter with the value **Student** in the **hub-nva.json** Building Blocks parameter file:

```
sed -i.bak1 's/"adminUsername": ""/"adminUsername": "Student"/' ~/hub-nva.json
```

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminPassword** parameter with the value **Pa55w.rd1234** in the **hub-nva.json** Building Blocks parameter file:

```
sed -i.bak2 's/"adminPassword": ""/"adminPassword": "Pa55w.rd1234"/' ~/hub-nva.json
```

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the parameter values were successfully changed in the **hub-nva.json** Building Blocks parameter file:

```
cat ~/hub-nva.json
```

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminUsername** parameter with the value **Student** in the **spoke1.json** Building Blocks parameter file:

```
sed -i.bak1 's/"adminUsername": ""/"adminUsername": "Student"/' ~/spoke1.json
```

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminPassword** parameter with the value **Pa55w.rd1234** in **spoke1.json** the Building Blocks parameter file:

```
sed -i.bak2 's/"adminPassword": ""/"adminPassword": "Pa55w.rd1234"/' ~/spoke1.json
```

12. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the parameter values were successfully changed in the **spoke1.json** Building Blocks parameter file:

```
cat ~/spoke1.json
```

13. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminUsername** parameter with the value **Student** in the **spoke2.json** Building Blocks parameter file:

```
sed -i.bak1 's/"adminUsername": ""/"adminUsername": "Student"/' ~/spoke2.json
```

14. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminPassword** parameter with the value **Pa55w.rd1234** in the **spoke2.json** Building Blocks parameter file:

```
sed -i.bak2 's/"adminPassword": ""/"adminPassword": "Pa55w.rd1234"/' ~/spoke2.json
```

15. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the parameter values were successfully changed in the **spoke2.json** Building Blocks parameter file:

```
cat ~/spoke2.json
```

8

Task 4: Implement the hub component of the Hub and Spoke design

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

```
SUBSCRIPTION_ID=$(az account list --query "[0].id" | tr -d ' ')
```

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that will contain the hub virtual network:

```
RESOURCE_GROUP_HUB_VNET='AADesignLab08-hub-vnet-rg'
```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment (replace the placeholder `<Azure region>` with the name of the Azure region to which you intend to deploy resources in this lab):

```
LOCATION='<Azure region>'
```

8 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod03_Deploying%20Network%20Infrastructure%20for%20use%20in%20Azure%20Solutions.md#task-4-implement-the-hub-component-of-the-hub-and-spoke-design

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the hub component of the Hub-and-Spoke topology by using the Azure Building Blocks:

```
azbb -g $RESOURCE_GROUP_HUB_VNET -s $SUBSCRIPTION_ID -l $LOCATION -p ~/hub-vnet.json --deploy
```

- Do not wait for the deployment to complete but proceed to the next task.

9

Task 5: Implement the spoke components of the Hub and Spoke design

- On the Taskbar, click the **Microsoft Edge** icon.
- In the open browser window, navigate to the **Azure Portal** (<https://portal.azure.com>).
- If prompted, authenticate with the same user account account that you used earlier in this lab.
- At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

```
SUBSCRIPTION_ID=$(az account list --query "[0].id" | tr -d '')
```

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that will contain the first spoke virtual network:

```
RESOURCE_GROUP_SPOKE1_VNET='AADesignLab08-spoke1-vnet-rg'
```

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

```
LOCATION=$(az group list --query "[?name == 'AADesignLab08-hub-vnet-rg'].location" --output tsv)
```

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the first spoke component of the Hub-and-Spoke topology by using the Azure Building Blocks:

```
azbb -g $RESOURCE_GROUP_SPOKE1_VNET -s $SUBSCRIPTION_ID -l $LOCATION -p ~/spoke1.json --deploy
```

- Do not wait for the deployment to complete but proceed to the next step.
- On the Taskbar, click the **Microsoft Edge** icon.
- In the open browser window, navigate to the **Azure Portal** (<https://portal.azure.com>).
- If prompted, authenticate with the same user account account that you used earlier in this lab.

9 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod03_Deploying%20Network%20Infrastructure%20for%20use%20in%20Azure%20Solutions.md#task-5-implement-the-spoke-components-of-the-hub-and-spoke-design

13. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
14. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

```
SUBSCRIPTION_ID=$(az account list --query "[0].id" | tr -d '')
```

15. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that will contain the second spoke virtual network:

```
RESOURCE_GROUP_SPOKE2_VNET='AADesignLab08-spoke2-vnet-rg'
```

16. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

```
LOCATION=$(az group list --query "[?name == 'AADesignLab08-hub-vnet-rg'].location" --output tsv)
```

17. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the second spoke component of the Hub-and-Spoke topology by using the Azure Building Blocks:

```
azbb -g $RESOURCE_GROUP_SPOKE2_VNET -s $SUBSCRIPTION_ID -l $LOCATION -p ~/spoke2.json --deploy
```

18. Do not wait for the deployment to complete but proceed to the next task.

10

Task 6: Configure the VNet peering of the Hub and Spoke design

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (<https://portal.azure.com>).
3. If prompted, authenticate with the same user account that you used earlier in this lab.
4. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

```
SUBSCRIPTION_ID=$(az account list --query "[0].id" | tr -d '')
```

6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that contains the hub virtual network:

```
RESOURCE_GROUP_HUB_VNET='AADesignLab08-hub-vnet-rg'
```

¹⁰ https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod03_Deploying%20Network%20Infrastructure%20for%20Use%20in%20Azure%20Solutions.md#task-6-configure-the-vnet-peering-of-the-hub-and-spoke-design

7. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

```
LOCATION=$(az group list --query "[?name == 'AADesignLab08-hub-vnet-rg']".location) --output tsv)
```

8. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to provision peering of the virtual networks in the Hub-and-Spoke topology by using the Azure Building Blocks:

```
azbb -g $RESOURCE_GROUP_HUB_VNET -s $SUBSCRIPTION_ID -l $LOCATION -p ~/hub-vnet-peering.json --deploy
```

9. Do not wait for the deployment to complete but proceed to the next task.

11

Task 7: Configure routing of the Hub and Spoke design

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (<https://portal.azure.com>).
3. If prompted, authenticate with the same user account account that you used earlier in this lab.
4. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

```
SUBSCRIPTION_ID=$(az account list --query "[0].id" | tr -d '')
```

6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that will contain the hub Network Virtual Appliance (NVA) functioning as a router:

```
RESOURCE_GROUP_HUB_NVA='AADesignLab08-hub-nva-rg'
```

7. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

```
LOCATION=$(az group list --query "[?name == 'AADesignLab08-hub-vnet-rg']".location) --output tsv)
```

8. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the NVA component of the Hub-and-Spoke topology by using the Azure Building Blocks:

```
azbb -g $RESOURCE_GROUP_HUB_NVA -s $SUBSCRIPTION_ID -l $LOCATION -p ~/hub-nva.json --deploy
```

9. Wait for the deployment to complete before you proceed to the next task.

¹¹ https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod03_Deploying%20Network%20Infrastructure%20for%20use%20in%20Azure%20Solutions.md#task-7-configure-routing-of-the-hub-and-spoke-design

10. **Note:** The deployment can take about 10 minutes.

12

Exercise 2: Review the Hub-spoke topology

13

Task 1: Examine the peering configuration

1. In the hub menu in the Azure portal, click **All services**.
2. In the **All services** menu, in the **Filter** text box, type **Virtual networks** and press **Enter**.
3. In the list of results, click **Virtual networks**.
4. On the **Virtual networks** blade, click **hub-vnet**.
5. On the **hub-vnet** blade, click **Peerings**.
6. On the **hub-vnet - Peerings** blade, review the list of peerings and their status.
7. Navigate back to the **Virtual Networks** blade and click **spoke1-vnet**.
8. On the **spoke1-vnet** blade, click **Peerings**.
9. On the **spoke1-vnet - Peerings** blade, review the existing peering and its status.
10. Navigate back to the **Virtual Networks** blade and click **spoke2-vnet**.
11. On the **spoke2-vnet** blade, click **Peerings**.
12. On the **spoke2-vnet - Peerings** blade, review the existing peering and its status.

14

Task 2: Examine the routing configuration

1. In the **All services** menu, in the **Filter** text box, type **Route tables** and press **Enter**.
2. In the list of results, click **Route tables**.
3. On the **Route tables** blade, click **hub-dmz-rt**.
4. On the **hub-dmz-rt** blade, review the list of routes. Note the **NEXT HOP** entry for the routes **toSpoke1** and **toSpoke2**.
5. Navigate back to the **Route tables** blade and click **spoke1-rt**.
6. On the **spoke1-rt** blade, click **Peerings**.
7. On the **spoke1-rt** blade, review the list of routes. Note the **NEXT HOP** entry for the route **toSpoke2**.

¹² https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod03_Deploying%20Network%20Infrastructure%20for%20Use%20in%20Azure%20Solutions.md#exercise-2-review-the-hub-spoke-topology

¹³ https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod03_Deploying%20Network%20Infrastructure%20for%20Use%20in%20Azure%20Solutions.md#task-1-examine-the-peering-configuration

¹⁴ https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod03_Deploying%20Network%20Infrastructure%20for%20Use%20in%20Azure%20Solutions.md#task-2-examine-the-routing-configuration

8. Navigate back to the **Route tables** blade and click **spoke2-rt**.
9. On the **spoke2-rt** blade, click **Peerings**.
10. On the **spoke2-rt** blade, review the list of routes. Note the **NEXT HOP** entry for the route **toSpoke1**.

15

Task 3: Verify connectivity between spokes

1. In the hub menu in the Azure portal, click **All services**.
2. In the **All services** menu, in the **Filter** text box, type **Network Watcher** and press **Enter**.
3. In the list of results, click **Network Watcher**.
4. On the **Network Watcher** blade, in the **NETWORK DIAGNOSTIC TOOLS** section, click **Connection troubleshoot**.
5. On the **Network Watcher - Connection troubleshoot** blade, perform the following tasks:
 - Leave the **Subscription** drop-down list entry set to its default value.
 - In the **Resource group** drop-down list, select the **AADesignLab08-spoke1-vnet-rg** entry.
 - In the **Virtual machine** drop-down list, leave the default entry.
 - Leave the **Port** text box blank.
 - Ensure that the **Destination** option is set to **Specify manually**.
 - In the **URI, FQDN, or IPv4** text box, type **10.2.0.68** entry.
 - In the **Port** text, type 3389.
 - Click the **Check** button.
6. Wait until results of the connectivity check are returned and verify that the status is **Reachable**.
7. **Note:** If this is the first time you are using Network Watcher, the check can take up to 5 minutes.

16

Exercise 3: Remove lab resources

17

Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.

15 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod03_Deploying%20Network%20Infrastructure%20for%20use%20in%20Azure%20Solutions.md#task-3-verify-connectivity-between-spokes

16 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod03_Deploying%20Network%20Infrastructure%20for%20use%20in%20Azure%20Solutions.md#exercise-3-remove-lab-resources

17 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T04_Lab_Mod03_Deploying%20Network%20Infrastructure%20for%20use%20in%20Azure%20Solutions.md#task-1-open-cloud-shell

2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

```
az group list --query "[?starts_with(name, 'AADesignLab08')].name" --output tsv
```

3. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

18

Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

```
az group list --query "[?starts_with(name, 'AADesignLab08')].name" --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'
```

2. Close the **Cloud Shell** prompt at the bottom of the portal.

Review: In this exercise, you removed the resources used in this lab.

Review Questions

Module 3 Review Questions

Virtual networks

You are designing a line of business solution for a company. The solution connects to Azure-based resources by using the public Internet.

You need to recommend a solution to protect network traffic.

What should you use? What are the capabilities and limitations of the technology?

Suggested Answer ↓

An Azure Virtual Network (VNET) is the logical unit of multiple or all network resources in an Azure region. From the public Internet, incoming traffic is routed to Public IP addresses of Azure Resources.

Load balancing

A company has an Azure environment that contains a VNET with two subnets.

You need to recommend a solution to handle all incoming traffic between the subnets.

What should you recommend?

Suggested Answer ↓

Azure Load Balancers can handle almost any TCP or UDP traffic. An internal load balancer can load balance incoming traffic between Azure Subnets.

Network types

You are designing a solution for a company.

You need to recommend a solution to connect on-premises resources with resources in Azure.

What should you recommend?

Suggested Answer ↓

Azure supports multiple options to connect your on-premises datacenter to Azure including Express-Route, Site-to-Site VPN, and Point-to-Site VPN.

Module 4 Module Integrating Azure Solution Components Using Messaging Services

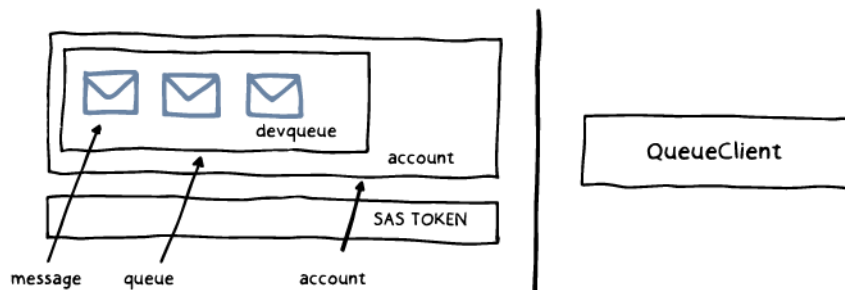
Event Messaging

Storage Queues

This module describes and compares the integration and messaging services available for solutions hosted on the Azure platform. Messaging services described include Azure Storage Queues, Service Bus Queues, Service Bus Relay, IoT Hubs, Event Hubs, and Notification Hubs. Integration services include Azure Functions and Logic Apps.

After completing this module, students will be able to:

- Compare Storage Queues to Service Bus Queues.
- Identify when to use Azure Functions or Logic Apps for integration components in a solution.
- Describe the differences between IoT Hubs, Event Hubs and Time Series Insights.



Event Messaging

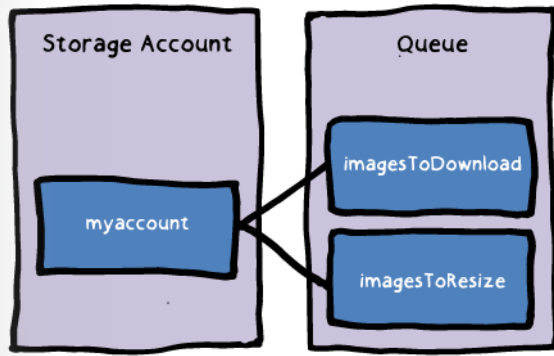
This lesson introduces Service Bus and Event Grid as key components of an event messaging architecture hosted in Azure.

After completing this lesson, you will be able to:

- Describe various Service Bus offerings and compare them to equivalent offerings in other services.
- Identify when to use Event Grid as a messaging component of a solution in Azure.

Queue Service Components

The Queue service contains the following components:



URL format: Queues are addressable using the following URL format:

`http://[account].queue.core.windows.net/<queue>`

The following URL addresses one of the queues in the diagram:

`http://[account].queue.core.windows.net/imagesToDownload`

Queue: A queue contains a set of messages. All messages must be in a queue.

Message: A message, in any format, of up to 64KB.

Storage Queue Message Handling

Storage Queue Message Handling

Queue messages can be managed in a variety of different ways. The following is a list of actions that you can perform on a queue or its messages:

- **Create/Delete Queue:** Client SDKs, PowerShell or the REST API can be used to create a new queue instance or remove an existing one.
- **Measure Queue Length:** You can get an estimate of the number of messages in the queue. Due to race conditions and technical implementation, this count is not guaranteed and should be treated in your application as an approximate queue length.
- **Insert Message into Queue:** New messages can be added to the queue. These messages have a body that are either a string (in UTF-8 format) or a byte array.
- **Retrieve the Next Message:** A copy of the next available message is retrieved and the message is made invisible for a specific duration. During this time, you can process the message. Other queue consumers will not see this message in the queue when they retrieve messages while it is invisible. After a specific amount of time, the invisibility duration will elapse and the message is available to other queue consumers.

- **Extend Message Lease:** If you need more time to process a retrieved queue message, you can return to the queue and update the invisibility duration for the queue message. This will ensure that the message is not prematurely available to other queue consumers.
- **Peek at the Next Message:** You can retrieve a copy of the next available message in the queue without making the message invisible to other queue consumers.
- **Update a Message:** The contents of a retrieved message can be updated in the queue if you need to have a concept of checkpoints or state for queue messages.
- **Delete a Message:** Once you have completed processing a message, you must delete the message from the queue if you do not wish for it to be processed by any more queue consumers. Otherwise the message invisibility will timeout and the message will be processed again by other queue consumers.

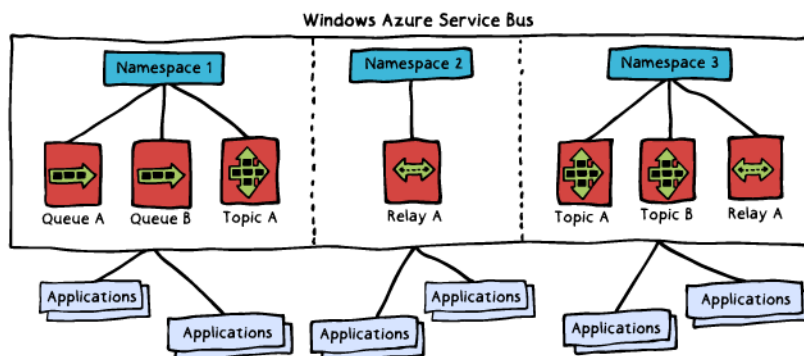
Because of the workflow, queue messages and their processors/consumers must be designed to be idempotent and should not have side effects from processing the same message multiple times. For example, if a queue message contains data that needs to be stored in a SQL database, your processor should check to see if an existing record exists and then update that record or create a new record. Otherwise, you may end up with false duplicate data from your queue processors processing the same message multiple times.

Service Bus

Azure Service Bus provides a hosted, secure, and widely available infrastructure for widespread communication, large-scale event distribution, naming, and service publishing. Service Bus provides connectivity options for Windows Communication Foundation (WCF) and other service endpoints—including REST endpoints—that would otherwise be difficult or impossible to reach. Endpoints can be located behind network address translation (NAT) boundaries, or bound to frequently-changing, dynamically-assigned IP addresses, or both.

Service Bus provides both “relayed” and “brokered” messaging capabilities. In the relayed messaging pattern, the relay service supports direct one-way messaging, request/response messaging, and peer-to-peer messaging. Brokered messaging provides durable, asynchronous messaging components such as Queues, Topics, and Subscriptions, with features that support publish-subscribe and temporal decoupling: senders and receivers do not have to be online at the same time; the messaging infrastructure reliably stores messages until the receiving party is ready to receive them.

Service Bus services are typically partitioned into namespaces as each namespace provide both a service and security boundary.



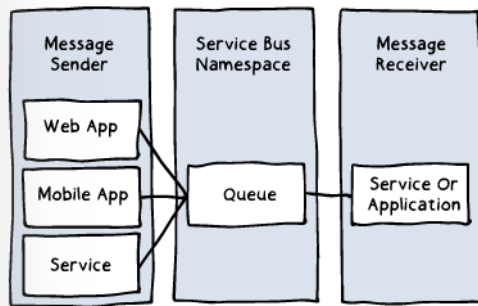
When you create a queue, topic, relay, or Event Hub, you give it a name. Combined with whatever you called your namespace, this name creates a unique identifier for the object. Applications can provide this name to Service Bus, then use that queue, topic, relay, or Event Hub to communicate with one another.

To use any of these objects, Windows applications can use Windows Communication Foundation (WCF). For queues, topics, and Event Hubs Windows applications can also use Service Bus-defined messaging APIs. To make these objects easier to use from non-Windows applications, Microsoft provides SDKs for Java, Node.js, and other languages. You can also access queues, topics, and Event Hubs using REST APIs over HTTP.

Service Bus Queues

Service Bus Queue is a brokered messaging system, which is similar to the Queue service Azure Storage. By using these queues, application modules that are distributed do not need to communicate directly with each other. These application modules can instead communicate by using the queues. This ensures that there is a separation between message generators and message processors. This separation provides the flexibility of having one or more application component instances that are generating messages and one or more application component instances that are processing the same messages. If an instance encounters an irrecoverable exceptional condition, other instances can continue processing the messages. If the workload for the entire application is increased, new instances can be created to handle the load. These scenarios are common and critical when developing and designing cloud applications.

Service Bus queues do implement a familiar first in, first out (FIFO) message delivery strategy. Service Bus queues can also guarantee that a message is received and processed both at least and at most once by the message consumers.



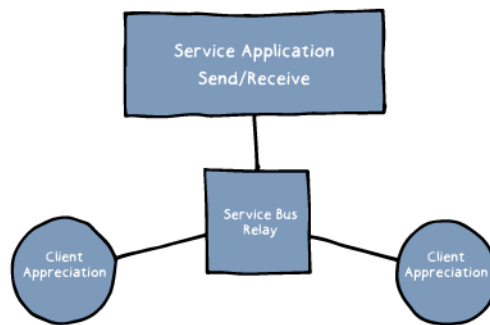
Service Bus Relay

Service Bus includes a relay component that can connect existing services to new client applications without exposing the true location or address of the service. Some of the advantages of using a relay include:

- Services that need to communicate directly with external client applications or devices (For example, mobile devices) typically need to be placed in a special subnet or a virtual network with a unique NAT or firewall configuration. The address for the service endpoint will need to be publicly addressable in order for client devices to connect. In some enterprises, this can be considered dangerous or unacceptable. With Service Bus Relay, the service makes an outbound connection to the relay and bypasses many of the complex network configurations that are necessary for inbound connections.
- Although mobile applications are deployed and updated regularly, end users might not update their applications as regularly as you want them to. If your service needs to be migrated to a new network or moved to a new IP address, this can cause a lapse of connectivity for your mobile applications. Using

Service Bus Relay, your mobile applications address a publicly accessible and permanent uniform resource identifier (URI). You are then free to make changes and migrate your service within your organization's infrastructure. The new service instance or location simply needs to connect to the relay for client devices to access it. This enables more mobility for services that are connected to the applications that are already deployed.

Service Bus Relay also supports direct peer-to-peer communication. This is negotiated if the relay determines that the connecting application can easily and directly address the service.



Event Grid

Azure Event Grid is a single service designed to manage and route systemic events from any source service in your Azure subscription. Event Grid is designed to eliminate the need to poll existing services and enable applications to run in an event-driven manner where applications are triggered only when needed, and consume compute only when necessary.

Event Grid can be used to publish custom events from your workloads or pre-determined events designed with each Azure service. Event Grid can be used in a variety of ways including:

- Integrate various components of your workloads together using a publish-subscribe mode.
- Enable your solution to listen to events from third-party B2B services or publish events for the third-party services to consume.
- Create serverless compute that is triggered by a specific Azure service event such as the creation of a database or VM.
- Automate the deployment of resources by subscribing to ARM events.

Azure Event Grid allows users to build applications with event-based architectures with ease. You merely select the Azure resource you wish to use, then select the WebHook endpoint or event handler to send the event. Event Grid has integrated support for events coming from Azure services, as well as support for third-party events. You can even use filters to route specific events to different or even multiple endpoints.

An Event Grid instance can use point-and-click to configure events, moving them from your Azure resource to an event handler or endpoint. It also has reliability, allowing the use of a 24-hour retry with exponential backoff so users can ensure events are delivered on time. Event Grid can also build high-volume workloads with support for millions of these events per second, and even custom events can be routed, filtered, and delivered. With all the power of Event Grid, you can rest assured that it is pay per event, meaning that you only pay for what you use with Event Grid.

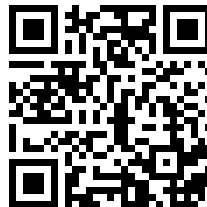
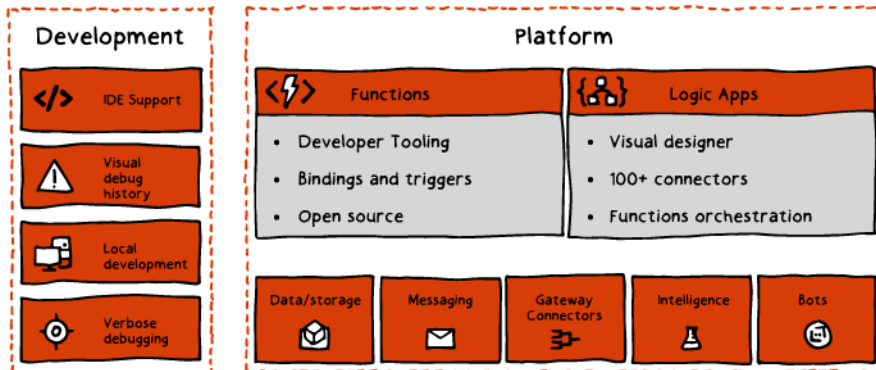
Event Grid can provide users with several options to improve serverless, integration, and ops automation work. Event Grid can connect data sources and event handlers to provide a serverless application archi-

ture. For example, you can use Event Grid to trigger a function to run image analysis every time a new item becomes added to a blob storage container. Event Grid can also connect an application with other services. As an example, a user can create a storage blob to send your app's event data to Event grid, while taking advantage of its reliable delivery, direct integration, and advanced routing with Azure. Lastly, it also allows users to speed an automation process. A user can request Event Grid to notify Azure Automation when a virtual machine starts or a SQL Database is spun up as one of many examples. Said events can be used to tag virtual machines, file work items, put metadata into operations tools, or even automatically check to ensure services configurations are correct.

Integration

Serverless Integration

Serverless computing promises agility and power in building the next generation of solutions. See how Azure Logic Apps and Azure Functions provide powerful tools to build serverless applications faster than ever. In video below, you'll see how building an automated email system using only serverless technologies. As the video progresses, it will walk through the steps to integrate the solution with source control and continuous delivery tools.



Notification Hubs

Notification Hubs is a combination of a service infrastructure and a set of client libraries that allows you to publish push notifications to your mobile applications from any application service component. With Notification Hubs, you can send notifications that are personalized to a specific user, notifications that are distributed to many users across various platforms, and notifications that are filtered to a specific set of users. The Notification Hubs infrastructure abstracts the implementation of the various Platform Notification Systems (PNS) for each mobile platform. By using a single method call, you can send notifications to various device platforms without having to implement a different message structure or communication mechanism for each platform.

You can use notification hubs in a variety of scenarios including:

- Send wide-reaching news notifications to all devices with your mobile application installed.
- Send a notification to a subset of your users that is determined based on a tag, label, or location.
- Send specific notifications to a user for the activities that are related to their specific account.

Benefits

Notification hubs eliminate the challenges that are involved in managing push notifications. Notification Hubs use a full multiplatform, scaled-out push notification infrastructure, and considerably reduce the

push-specific code that runs in the app. Notification hubs implement all the functionality of a push infrastructure. Devices are only responsible for registering PNS handles, and the backend is responsible for sending platform-independent messages to users or interest groups.

Notification hubs provide a push infrastructure with the following advantages:

Multiple platforms:

- Support for all major mobile platforms—Windows, Windows Phone, iOS, and Android.
- No platform-specific protocols. The application only communicates with Notification Hubs.
- Device handle management. Notification Hubs maintains the handle registry and feedback from PNSs.

Works with any backend - Works with Cloud or on-premises applications that are written in .NET, PHP, Java, or Node.

- **Scale:** Notification hubs scale to millions of devices without the need of rearchitecting or sharding.

Rich set of delivery patterns: Associate devices with tags, representing logical users or interest groups:

- **Broadcast:** Allows for near-simultaneous broadcast to millions of devices with a single Application Programming Interface (API) call.
- **Unicast/Multicast:** Push to tags representing individual users, including all their devices; or a wider group. For example, a user could use the app on separate devices (tablet, phone, etc.) and would require push notifications to either be pushed to all devices or a specific device.
- **Segmentation:** Push to a complex segment that is defined by tag expressions (For example, devices in New York following the Yankees).
- **Personalization:** Each device can have one or more templates to achieve per-device localization and personalization without affecting the backend code.

Platform Notifications

At a high level, all platform notification systems follow the same pattern:

1. The client application contacts the PNS to retrieve its handle. The handle type depends on the system. For Windows Notification Service (WNS), it is a URI or notification channel. For Apple Push Notification Service (APNS), it is a token.
2. The client application stores this handle in the app backend for later usage. For WNS, the backend is typically a cloud service. For APNS, the system is called a provider.
3. To send a push notification, the app backend contacts the PNS by using the handle to target an instance of a specific client application.
4. The PNS forwards the notification to the device specified by the handle.

Notification Hubs can be used in flexible ways to register devices and eventually send a message to the devices. Devices can register themselves and receive notifications using the following method:

1. The client device reaches out to the PNS by using the Notification Hubs SDK. It registers a unique PNS handle that is used by the service to send notifications to this device whether the application is running or not.

2. The client device can alternatively send its PNS handle to the application backend to have the application register the device.
3. When the application backend sends a message to the Notification Hubs service, the service handles sending the message to the appropriate target clients by using their registered PNS handles. The application backend simply requests the message is sent and the Notification Hubs service and the PNS handle the actual distribution of messages to client devices.

Internet of Things (IoT)

Event Hubs

This section covers the two most common real-time message streaming options available in Azure, Event Hubs and IoT Hubs.

After completing this section you will be able to:

- Identify the various streaming queue services in Azure.
- Compare and contrast Event Hubs and IoT Hubs.
- Architect a real-time streaming solution using Event or IoT Hubs and Azure data processing components.

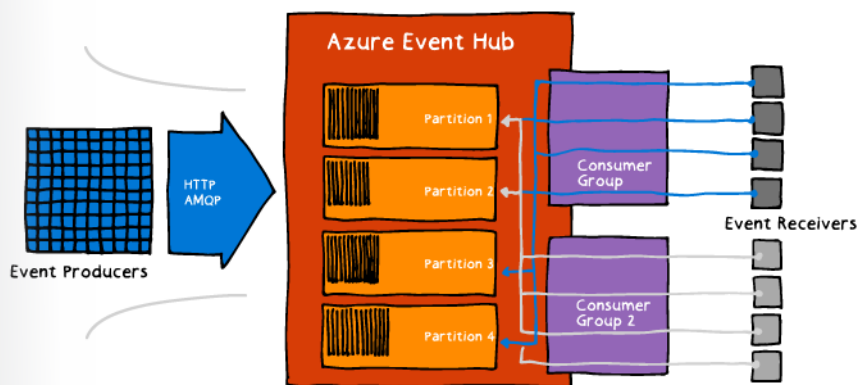
Event Hubs

As a scalable data streaming platform with an event ingestion service, Azure Event Hubs can be used to receive and process millions of events per second. It can process telemetry, events, or even information produced by distributed software and devices as well as store said information. Information sent to an event hub can become converted or stored using any real-time analytics provider or batching/storage adapter.

Azure Event Hubs Features and Role

Azure Event Hubs provides various features that can assist with storing and processing events. It can provide an entity to send data to event hubs, capable of publishing via AMQP 1.0 or HTTPS. Also, it can capture Event Hubs streaming data to store in an Azure Blob storage account. It can allow you to partition the data, allowing each consumer to only read a specific subset of the event stream, along with the capabilities for each consumer to act independently.

Also, they can identify and authenticate the event publisher through Shared Access Signature Token, or SAS Token. Azure Event Hubs use SAS tokens generated from a SAS key to regenerate the hash and authenticate the sender.

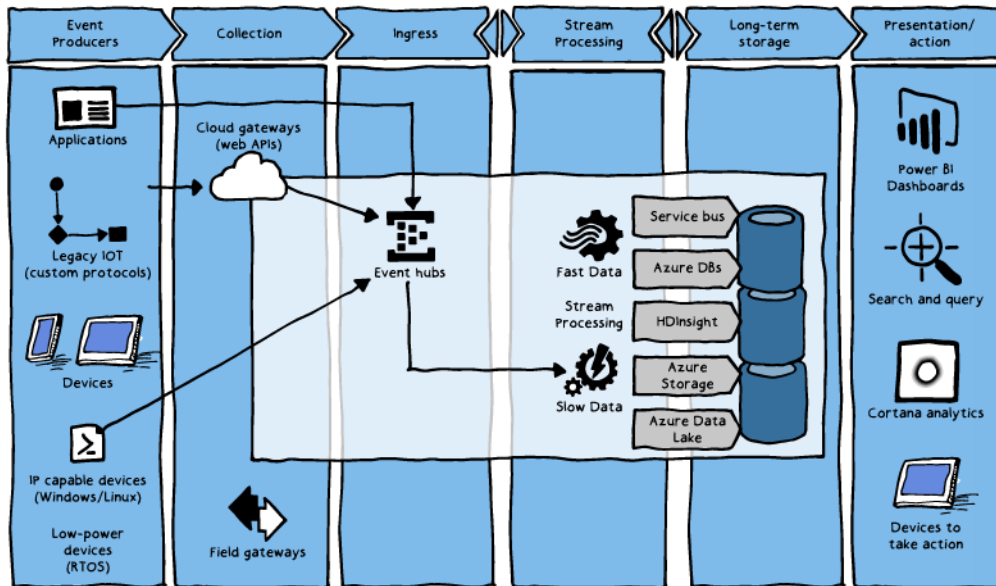


Also, Azure Event Hubs also contains Throughput units or Pre-purchased units of capacity. Throughput units can include the capacity of either 1 MB per second or 100 events per second, whichever comes first if its Ingress while maintaining up to 2 MB per second if its egress. These units are billed hourly for a minimum of one hour, and up to a maximum of 20 throughput units per Event Hubs namespace.

Azure Event Hubs commonly can be used as a "front line" for an event pipeline of a solution architecture, sometimes known as an event investor. An event investor is a service or component that sits between the

event consumers and publishers to separate the production of an event stream that obtained said events. Azure Event Hubs can provide message stream handling, but the capabilities of said service are different from traditional enterprise messaging.

The following figure depicts this architecture:



IoT Hubs

Azure IoT hub service is a managed service that can provide reliable and secure bi-directional communication between a multitude of IoT devices with a solution back end. The service can provide a multitude of functions such as providing multiple device-to-cloud and cloud-to-device communication options, a declarative message routing option built in that sends the message to other Azure services, and more.

In addition to its features, Azure IoT Hub provides a way to clarify message routes based on routing rules to control where the hub can send device-to-cloud messages. These rules do not require you to write any code to implement. You can also receive detailed logs for identifying connectivity events.

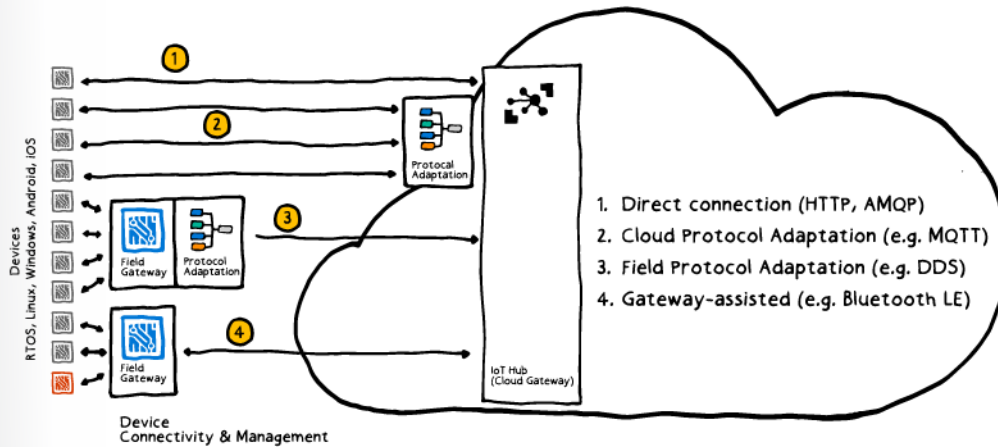
Azure IoT Hub Service and Azure Event Hubs are two separate products despite the similar description. IoT can provide additional features such as Device twins, which can be used to store and research device state information. While IoT provides the device-to-cloud and cloud-to-device communication options mentioned in the previous paragraph, Event hubs can only provide event ingress communication.

When using Azure IoT Hub, it implements a service-assisted communication pattern to act as a mediator between your devices and the solution in the back end. The communication provided to establish bi-directional, reliable communication path between a control system.

To maintain such a communication path, Azure IoT follows a set of principles. It focuses first and foremost on Security, not allowing it to accept any unsolicited network information. It even uses a device dedicated to this, which establish all connections and routes in an outbound direction only. The path between both device and service or even between a device and gateway must be secured at the application protocol layer.

In an IoT solution, a gateway can typically be either a protocol gateway deployed in a cloud or a field gateway deployed locally with a device. With a field gateway, you can make time-sensitive decisions, run

analytics on edge, provide device management services or even enforce security and privacy constraints. With a protocol gateway protocol, translations are carried out, such as MQTT to AMQP.



Comparing Event Hubs and IoT Hubs

Event Hubs and IoT Hubs are similar services but they each have unique differences detailed in this table.

Area	IoT Hub	Event Hubs
Communication patterns	Enables device-to-cloud communications (https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-d2c-guidance) (messaging, file uploads, and reported properties) and cloud-to-device communications (https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-c2d-guidance) (direct methods, desired properties, messaging).	Only enables event ingress (usually considered for device-to-cloud scenarios).
Device state information	Device twins can store and query device state information.	No device state information can be stored.
Device protocol support	Supports MQTT, MQTT over WebSockets, AMQP, AMQP over WebSockets, and HTTPS. Additionally, IoT Hub works with the Azure IoT protocol gateway, a customizable protocol gateway implementation to support custom protocols.	Supports AMQP, AMQP over WebSockets, and HTTPS.

Area	IoT Hub	Event Hubs
Security	Provides per-device identity and revocable access control. See the Security section of the IoT Hub developer guide.	Provides Event Hubs-wide shared access policies, with limited revocation support through publisher's policies. IoT solutions are often required to implement a custom solution to support per-device credentials and anti-spoofing measures.
Operations monitoring	Enables IoT solutions to subscribe to a rich set of device identity management and connectivity events such as individual device authentication errors, throttling, and bad format exceptions. These events enable you to quickly identify connectivity problems at the individual device level.	Exposes only aggregate metrics.
Scale	Is optimized to support millions of simultaneously connected devices.	Meters the connections as per Azure Event Hubs quotas. On the other hand, Event Hubs enables you to specify the partition for each message sent.
Device SDKs	Provides device SDKs for a large variety of platforms and languages, in addition to direct MQTT, AMQP, and HTTPS APIs.	Is supported on .NET, Java, and C, in addition to AMQP and HTTPS send interfaces.

Area	IoT Hub	Event Hubs
File upload	Enables IoT solutions to upload files from devices to the cloud. Includes a file notification endpoint for workflow integration and an operations monitoring category for debugging support.	Not supported.
Route messages to multiple endpoints	Up to 10 custom endpoints are supported. Rules determine how messages are routed to custom endpoints. For more information, see Send and receive messages with IoT Hub.	Requires additional code to be written and hosted for message dispatching.

IoT Remote Monitoring

You can use an IoT Hub to receive messages from a device and then push them to a Logic App for processing. The Logic App can send notifications for messages that cross specific thresholds and store aggregate data about all messages to a database.

Time Series Insights

The Azure Time Series Insights service is a product built for visualizing, storing, and querying vast amounts of time series information, such as information generated by IoT devices. If you have plans to store, manage query, or even visualize time series data in the cloud, Azure Time Series Insights may be the product for you.

For those unsure, whether their information could be considered time series a few factors can help determine the difference for you. By definition, time series data shows how an asset or process changes over time. It has a timestamp that is unique to its kind which is most meaningful as an axis. The time series data arrives in time order and can usually be an insert rather than an update to your database. Because time series data obtains and stores every new event as a row, the changes can be measured over a time frame, enabling one to not only look back into the past but also predict the future with the data.

You may want to consider using Azure Time Series Insights in various situations, including:

- **Storing and maintaining time series data in a scalable format:** Azure Time Series Insights provides a database with time series data in mind. It handles the work of storing and maintaining events due to its nature as a scalable and fully managed database.
 - **Near real-time data visualization:** Azure Time Series Insights can provide an explorer to visualize data streaming into an environment. When you connect an event source, the event data can be viewed, queried, and explored with Azure Time Series Insights.
 - **Producing customer applications:** You can build applications that use time series data using the exposed REST Query APIs provided by Azure Time Series Insights.
 - **Needing a global view of time series data streaming from different locations for multiple sites or assets:** Azure Time Series Insights allows you to connect multiple event sources to the environment. It means that data streaming in different or multiple locations can be viewed together in near real-time. Users of Azure Time Series Insights can use this visibility to share data with business leaders as well as provide better collaboration with domain experts who can apply their expertise to help solve problems.
- Azure Time Series also has a multitude of capabilities. It requires no upfront data preparation, making setup a quick process. It can obtain and store millions of sensor events per day with a one-minute latency, giving it the ability to provide near real-time insights. One can even embed Azure Time Series Insights data into existing applications, allowing one to create custom solutions with sharing capabilities for others to explore your insights.

Online Lab - Deploying Messaging Components to Facilitate Communication Between Azure Resources

Lab Steps

Online Lab: Deploying Messaging Components to Facilitate Communication Between Azure Resources

NOTE: For the most recent version of this online lab, see: <https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign>

Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
 - Username: **Admin**
 - Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
 - Microsoft Edge
 - File Explorer
 - **Visual Studio Code**¹
 - **Microsoft Azure Storage Explorer**²
 - Bash on Ubuntu on Windows
 - Windows PowerShell
3. **Note:** You can also find shortcuts to these applications in the **Start Menu**.

Exercise 1: Deploy a Service Bus namespace

Task 1: Open the Azure portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** (<https://portal.azure.com>).

¹ <https://code.visualstudio.com/>

² <https://azure.microsoft.com/features/storage-explorer/>

3. When prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

Task 2: Create a Service Bus namespace

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Service Bus** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Service Bus**.
4. On the **Service Bus** blade, click the **Create** button.
5. On the **Create namespace** blade, perform the following tasks:
 - In the **Name** text box, enter a globally unique name.
 - In the **Pricing tier** drop-down list, select the **Basic** option.
 - Leave the **Subscription** drop-down list entry set to its default value.
 - In the **Resource group** section, ensure that the **Create new** option is selected and then, in the text box, type **AADesignLab1101-RG**.
 - In the **Location** drop-down list, select the Azure region to which you intend to deploy resources in this lab.
 - Click the **Create** button.
6. Wait for the provisioning to complete before you proceed to the next step.

Task 3: Create a Service Bus Queue

1. In the hub menu of the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab1101-RG**.
3. On the **AADesignLab1101-RG** blade, click the newly created Service Bus namespace.
4. On the Service Bus namespace blade, in the **ENTITIES** section, click **Queues**.
5. On the Service Bus namespace blade, click the **+ Queue** button.
6. In the **Create queue** pane, perform the following tasks:
 - In the **Name** text box, type **messages**.
 - Leave all remaining settings with their default values.
 - Click the **Create** button.

Task 4: Get Service Bus Connection String

1. Back on the Service Bus namespace blade, click **Shared access policies**.
2. On the Service Bus namespace blade, click the **RootManageSharedAccessKey** policy.
3. In the **SAS Policy: RootManageSharedAccessKey** pane, locate and record the value of the **Primary Connection String** field. You will use this value later in this lab.

Review: In this exercise, you created a new Service Bus namespace and recorded a connection string to access queues in the namespace.

Exercise 2: Create a logic app

Task 1: Create an Azure Storage account

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Storage Account** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Storage Account - blob, file, table, queue**.
4. On the **Storage Account - blob, file, table, queue** blade, click the **Create** button.
5. On the **Create storage account** blade, perform the following tasks:
 - In the **Name** text box, type a unique name consisting of a combination of between 3 and 24 characters and digits.
 - In the **Deployment model** section, ensure that the **Resource manager** option is selected.
 - In the **Account kind** drop-down list, ensure that the **Storage (general purpose v1)** option is selected.
 - Leave the **Location** entry set to the same Azure region you selected earlier in this exercise.
 - In the **Replication** drop-down list, select the **Locally-redundant storage (LRS)** entry.
 - In the **Performance** section, ensure that the **Standard** option is selected.
 - In the **Secure transfer required** section, ensure that the **Disabled** option is selected.
 - Leave the **Subscription** drop-down list entry set to its default value.
 - In the **Resource group** section, ensure that the **Use existing** option is selected and, in the drop-down list below, select the resource group you created earlier in this exercise.
 - Leave the **Configure virtual networks** option set to its default value.
 - Leave the **Hierarchical namespaces** option set to its default value.
 - Click the **Create** button.

6. Wait for the provisioning to complete before you proceed to the next step.
7. In the hub menu of the Azure portal, click **Resource groups**.
8. On the **Resource groups** blade, click **AADesignLab1101-RG**.
9. On the **AADesignLab1101-RG** blade, click the newly created Azure Storage account.
10. On the Storage account blade, click the **Blobs** tile.
11. On the Storage account blade, click the **+ Container** button.
12. In the **New container** pane, perform the following tasks:
 - In the **Name** text box, type **messageoutput**.
 - In the **Public access level** drop-down list, select the **Blob (anonymous read access for blobs only)** option.
 - Click the **OK** button.

Task 2: Create a logic app

1. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Logic App** and press **Enter**.
2. On the **Everything** blade, in the search results, click **Logic App**.
3. On the **Logic App** blade, click the **Create** button.
4. On the **Create logic app** blade, perform the following tasks:
 - In the **Name** text box, type **ServiceBusWorkflow**.
 - Leave the **Subscription** drop-down list entry set to its default value.
 - In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab1101-RG**.
 - In the **Location** drop-down list, select the same Azure region you chose in the previous task.
 - In the **Log Analytics** section, ensure that the **Off** button is selected.
 - Click the **Create** button.
5. Wait for the provisioning to complete before you proceed to the next task.

Task 3: Configure logic app steps.

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab1101-RG**.
3. On the **AADesignLab1101-RG** blade, click the entry representing the logic app you created in the previous task.
4. On the **Logic Apps Designer** blade, scroll down and click the **Blank Logic App** tile in the **Templates** section.

5. On the **Logic Apps Designer** blade, perform the following tasks:
 - In the **Search connectors and triggers** text box, type **Service Bus**.
 - In the search results, select the trigger named **When a message is received in a queue (auto-complete) - Service Bus**.
 - In the **Connection Name** text box, type **ServiceBusConnection**.
 - In the list of **Service Bus namespaces**, select the namespace you created earlier in this lab.
 - In the list of policies, select the **RootManageSharedAccessKey** policy.
 - Click the **Create** button.
6. In the **When a message is received in a queue (auto-complete)** step, perform the following tasks:
 - In the **Queue name** drop-down list, select the **messages** entry.
 - In the **Interval** text box, type **30**.
 - In the **Frequency** drop-down list, select the **Second** entry.
7. On the **Logic Apps Designer** blade, click the **+ New Step** button.
8. On the **Logic Apps Designer** blade, perform the following tasks:
 - In the **Search connectors and actions** text box, type **Storage blob**.
 - In the search results, select the action named **Create blob - Azure Blob Storage**.
 - In the **Connection Name** text box, type **StorageConnection**.
 - In the list of *Storage accounts*, select the account you created earlier in this lab.
 - Click the **Create** button.
9. In the **Create Blob** step, perform the following tasks:
 - In the **Folder path** text box, type **/messageoutput**.
 - In the **Blob name** text box, type **@concat(triggerBody()?['MessageId'], '.txt')**.
 - In the **Blob content** text box, type **@string(decodeBase64(triggerBody()?['ContentData']))**.
10. At the top of the **Logic Apps Designer** blade, click the **Save** button to persist your workflow.

Task 2: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
2. **Note:** The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
3. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.
4. **Note:** If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.

5. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
 - Leave the **Subscription** drop-down list entry set to its default value.
 - In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you deployed resources in this lab.
 - In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab1101-RG**.
 - In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
 - In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
 - Click the **Create storage** button.
6. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

Task 4: Validate Logic App using Node.js

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
2. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to install the **azure** package using NPM:

```
npm install azure
```

3. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to open the interactive node terminal:

```
node
```

4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to import the **azure** module in Node:

```
var azure = require('azure');
```

5. At the **Cloud Shell** command prompt, type in the following command (replacing the placeholder `<Service Bus namespace connection string>` with the value of your url you recorded earlier in this lab) and press **Enter** to create a new variable for your Service Bus namespace connection string:

```
var connectionString = '<Service Bus namespace connection string>';
```

6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new client to connect to the Service Bus namespace:

```
var serviceBusService = azure.createServiceBusService(connectionString);
```

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to send a message to Service Bus namespace queue using the client.

```
serviceBusService.sendMessage('messages', { body: 'Hello World' },
function(error) { console.log(error) });
```

- In the hub menu of the Azure portal, click **Resource groups**.
- On the **Resource groups** blade, click **AADesignLab1101-RG**.
- On the **AADesignLab1101-RG** blade, click the Azure Storage account you created earlier in this lab.
- On the Storage account blade, click the **Blobs** tile.
- On the Storage account container blade, click the **messageoutput** container.
- Note the newly created blob in your container.

Review: In this exercise, you created a logic app that is triggered by messages from a queue in a Service Bus namespace.

Exercise 3: Remove lab resources

Task 1: Open Cloud Shell

- At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

```
az group list --query "[?starts_with(name,'AADesignLab11')].name" --output
tsv
```

- Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

Task 2: Delete resource groups

- At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

```
az group list --query "[?starts_with(name,'AADesignLab11')].name" --output
tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'
```

- Close the **Cloud Shell** prompt at the bottom of the portal.

Review: In this exercise, you removed the resources used in this lab.

Review Questions

Module 4 Review Questions

Azure Service Bus

You are designing a mobile solution for your company. Users search for products and create an order by adding the product to a virtual shopping cart.

You need to recommend a messaging infrastructure to reliably store the orders as messages while the orders are waiting to be processed.

What Azure service should you recommend?

Suggested Answer ↓

Microsoft Azure Service Bus is a host for queues holding jobs of critical business value, such as ordering products online. It allows for the creation of routes for messages that need to travel between applications and application modules. It is a solid platform for workflow and transaction handling and has robust facilities for dealing with many application fault conditions. A strength of Service Bus is also its function as a bridge between elements of hybrid cloud solutions and systems that include branch-office or work-site systems. Systems that sit “behind the firewall”, are roaming across networks, or are occasionally offline can’t be reached directly via “push” messaging but require messages to be sent to an agreed pickup location from where the designated receiver can obtain them.

Azure Logic Apps

You are designing a solution that will integrate several on-premises and Azure-based services. The solution will include workflows to perform operations in a specific order.

What Azure service should you recommend?

Suggested Answer ↓

Azure Logic Apps is a serverless solution that allows you to build automated scalable workflows. The workflows can integrate apps and data across cloud services and on-premises systems. Azure Logic Apps include several, built-in connectors to help your apps communicate with other apps and services, control the workflow through your logic apps, and manage or manipulate data. Organizations can also exchange messages through industry-standard protocols, including AS2, X12, and EDIFACT, using Azure Logic Apps enterprise integration features.

Azure Event Hub

Your company has manufacturing facilities worldwide. Each facility has several machines that produce products. The machines generate millions of messages daily to report progress, quality control metrics, and alerts.

You need to design a solution to receive and process the messages from the machines.

What Azure service should you include in the design?

Suggested Answer ↓

Azure Event Hubs is a highly scalable data streaming platform and ingestion service capable of receiving and processing millions of events per second. Event Hubs can process and store events, data, or telemetry produced by distributed software and devices. Data sent to an event hub can be transformed and stored using any real-time analytics provider or batching/storage adapters. Event Hubs sits between event publishers (the manufacturing machinery) and event consumers to decouple the production of an event stream from the consumption of those events. Event Hubs provides a unified streaming platform with time retention buffer, decoupling the event producers from event consumers.