

About this Course

Course Description

This course teaches Azure Solution Architects how to design infrastructure solutions. Course topics cover governance, compute, application architecture, storage, data integration, authentication and identity, networks, high availability, business continuity, and migrations. The course combines lecture with case studies to demonstrate basic architect design principles.

Level: Advanced

Audience

Successful students have experience and knowledge in IT operations, including networking, virtualization, identity, security, business continuity, disaster recovery, data platforms, and governance. Students also have experience designing and architecting solutions.

Prerequisites

Before attending this course, students must have previous experience deploying or administering Azure resources and conceptual knowledge of:

- Azure Active Directory
- Azure compute technologies such as VMs, containers and serverless solutions
- Azure virtual networking to include load balancers
- Azure Storage technologies (unstructured and databases)
- General application design concepts such as messaging and high availability

You can gain the prerequisites and a better understanding of Azure by taking [AZ-104: Prerequisites for Azure Administrators](#). This free online training will give you the experience you need to be successful in this course.

Expected learning

- Design a governance solution.
- Design a compute solution.
- Design a data storage solution.
- Design a data integration solution.
- Design an app architecture solution.
- Design authentication, authorization, and identity solutions.
- Design monitoring solutions.
- Design a networking solution.
- Design backup and disaster recovery solutions.
- Design migration solutions.

About this Course

Course Description

This course teaches Azure Solution Architects how to design infrastructure solutions. Course topics cover governance, compute, application architecture, storage, data integration, authentication and identity, networks, high availability, business continuity, and migrations. The course combines lecture with case studies to demonstrate basic architect design principles.

Level: Advanced

Audience

Successful students have experience and knowledge in IT operations, including networking, virtualization, identity, security, business continuity, disaster recovery, data platforms, and governance. Students also have experience designing and architecting solutions.

Prerequisites

Before attending this course, students must have previous experience deploying or administering Azure resources and conceptual knowledge of:

- Azure Active Directory
- Azure compute technologies such as VMs, containers and serverless solutions
- Azure virtual networking to include load balancers
- Azure Storage technologies (unstructured and databases)
- General application design concepts such as messaging and high availability

You can gain the prerequisites and a better understanding of Azure by taking [AZ-104: Prerequisites for Azure Administrators](#). This free online training will give you the experience you need to be successful in this course.

Expected learning

- Design a governance solution.
- Design a compute solution.
- Design a data storage solution.
- Design a data integration solution.
- Design an app architecture solution.
- Design authentication, authorization, and identity solutions.
- Design monitoring solutions.
- Design a networking solution.
- Design backup and disaster recovery solutions.
- Design migration solutions.

About this Course

Course Description

This course teaches Azure Solution Architects how to design infrastructure solutions. Course topics cover governance, compute, application architecture, storage, data integration, authentication and identity, networks, high availability, business continuity, and migrations. The course combines lecture with case studies to demonstrate basic architect design principles.

Level: Advanced

Audience

Successful students have experience and knowledge in IT operations, including networking, virtualization, identity, security, business continuity, disaster recovery, data platforms, and governance. Students also have experience designing and architecting solutions.

Prerequisites

Before attending this course, students must have previous experience deploying or administering Azure resources and conceptual knowledge of:

- Azure Active Directory
- Azure compute technologies such as VMs, containers and serverless solutions
- Azure virtual networking to include load balancers
- Azure Storage technologies (unstructured and databases)
- General application design concepts such as messaging and high availability

You can gain the prerequisites and a better understanding of Azure by taking [AZ-104: Prerequisites for Azure Administrators](#). This free online training will give you the experience you need to be successful in this course.

Expected learning

- Design a governance solution.
- Design a compute solution.
- Design a data storage solution.
- Design a data integration solution.
- Design an app architecture solution.
- Design authentication, authorization, and identity solutions.
- Design monitoring solutions.
- Design a networking solution.
- Design backup and disaster recovery solutions.
- Design migration solutions.

Syllabus

The course content includes a mix of content, case studies, knowledge check questions, and reference links.

Module 01 - Design governance and compute solutions

In this module, you will be introduced to governance and compute solutions.

- **Lesson 01 - Design for governance** including management groups, subscriptions, resource groups, Azure policy, Azure RBAC, and Azure Blueprints.
- **Lesson 02 - Design a compute solution** including virtual machines, Batch solutions, Azure Container Instances, Azure App Services, Azure Kubernetes, Azure Functions, and Azure Logic Apps.

Module 02 – Design storage and data integration solutions

In this module, you will be introduced to storage and data integration solutions.

- **Lesson 01 - Design a data solution storage** for non-relational data including storage accounts, redundancy, blobs, files, disks, and storage security.
- **Lesson 02 - Design a data storage solution** for relational data including Azure SQL, database scalability, SQL Edge, Azure Cosmos DB, Azure tables, and database security.
- **Lesson 03 - Design data integration solutions** including Azure Data Factory, Azure Data Lake, Azure Databricks, Azure Synapse Analytics, and Azure Stream Analytics.

Module 03 - Design app architecture and monitoring

In this module, you will be introduced to application architecture, app access, and monitoring

- **Lesson 01 - Design an application architecture** including message and event scenarios, messaging solutions, event solutions, app automation, and app lifecycle.
- **Lesson 02 - Design authentication and authorization solutions** including Active Directory, B2B, B2C, conditional access, identity protection, access reviews, and service principals.
- **Lesson 03 - Design a solution to log and monitor** Azure resources including Azure Monitor, Log Analytics, Azure Insights and Azure Data Explorer.

Module 04: Design for networks and business continuity solutions

- **Lesson 01 - Design network solutions** including network architecture, on-premises connections, application delivery services, and application protection services.
- **Lesson 02 - Design a solution for backup and disaster recovery** including Azure Backup, blob backup, file backup, virtual machine backup, SQL backup, and Azure Site Recovery.
- **Lesson 03 - Design migrations** including the Cloud Migration Framework, assessing workloads, migration tools, online migration, offline migrations, and storage migrations.

AZ-305 Certification Exam

The AZ-305, [Designing Microsoft Azure Infrastructure Solutions](#), certification exam is geared towards Azure Solution Architects.

The exam includes four study areas. The percentages indicate the relative weight of each area on the exam. The higher the percentage, the more questions the exam will contain.

AZ-305 Study Areas	Weights
Design identity, governance, and monitoring solutions	25-30%
Design data storage solutions	25-30%
Design business continuity solutions	10-15%
Design infrastructure solutions	25-30%

References - additional study resources

There are a lot of resources to help you and the student learn about Azure. We recommend you bookmark these pages. The list is included in the Welcome section of the student materials.

- [Microsoft Learn](#). Provides a wealth of self-paced learning including hands-on experiences.
- [Azure Architecture Center](#). Is guidance for architecting solutions on Azure using established patterns and practices.
- [Azure Documentation](#). Stay informed on the latest products, tools, and features. Get information on pricing, partners, support, and solutions.

Introduction

The term governance describes the general process of establishing rules and policies. Governance ensures those rules and policies are enforced.

A good governance strategy helps you maintain control over the applications and resources that you manage in the cloud. Maintaining control over your environment ensures that you stay compliant with:

- Industry standards, such as information security management.
- Corporate or organizational standards, such as ensuring that network data is encrypted.

Governance is most beneficial when you have:

- Multiple engineering teams working in Azure.
- Multiple subscriptions to manage.
- Regulatory requirements that must be enforced.
- Standards that must be followed for all cloud resources.

Meet Tailwind Traders



Tailwind Traders is a fictitious home improvement retailer. It operates retail hardware stores across the globe and online. The Tailwind Traders CTO is aware of the opportunities offered by Azure, but also understands the need for strong governance. Without strong governance, the company may end up with a difficult to manage Azure environment and costs, which are hard to track and control. The CTO is interested in understanding how Azure manages and enforces governance standards.

Learning objectives

In this module, youâ€™ll learn how to:

- Design for governance.
- Design for management groups.
- Design for Azure subscriptions.
- Design for resource groups.
- Design for Azure policies.
- Design for resource tags.
- Design for Azure blueprints.

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions. The module concepts are covered in:

Design Identity, Governance, and Monitoring

- Design Governance

Prerequisites

- Conceptual knowledge of governance policies, resource organization, and subscription management.
- Working experience with organizing resources, applying governance policies, and enforcing compliance requirements.

Introduction

The term governance describes the general process of establishing rules and policies. Governance ensures those rules and policies are enforced.

A good governance strategy helps you maintain control over the applications and resources that you manage in the cloud. Maintaining control over your environment ensures that you stay compliant with:

- Industry standards, such as information security management.
- Corporate or organizational standards, such as ensuring that network data is encrypted.

Governance is most beneficial when you have:

- Multiple engineering teams working in Azure.
- Multiple subscriptions to manage.
- Regulatory requirements that must be enforced.
- Standards that must be followed for all cloud resources.

Meet Tailwind Traders



Tailwind Traders is a fictitious home improvement retailer. It operates retail hardware stores across the globe and online. The Tailwind Traders CTO is aware of the opportunities offered by Azure, but also understands the need for strong governance. Without strong governance, the company may end up with a difficult to manage Azure environment and costs, which are hard to track and control. The CTO is interested in understanding how Azure manages and enforces governance standards.

Learning objectives

In this module, youâ€™ll learn how to:

- Design for governance.
- Design for management groups.
- Design for Azure subscriptions.
- Design for resource groups.
- Design for Azure policies.
- Design for resource tags.
- Design for Azure blueprints.

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions. The module concepts are covered in:

Design Identity, Governance, and Monitoring

- Design Governance

Prerequisites

- Conceptual knowledge of governance policies, resource organization, and subscription management.
- Working experience with organizing resources, applying governance policies, and enforcing compliance requirements.

Introduction

The term governance describes the general process of establishing rules and policies. Governance ensures those rules and policies are enforced.

A good governance strategy helps you maintain control over the applications and resources that you manage in the cloud. Maintaining control over your environment ensures that you stay compliant with:

- Industry standards, such as information security management.
- Corporate or organizational standards, such as ensuring that network data is encrypted.

Governance is most beneficial when you have:

- Multiple engineering teams working in Azure.
- Multiple subscriptions to manage.
- Regulatory requirements that must be enforced.
- Standards that must be followed for all cloud resources.

Meet Tailwind Traders



Tailwind Traders is a fictitious home improvement retailer. It operates retail hardware stores across the globe and online. The Tailwind Traders CTO is aware of the opportunities offered by Azure, but also understands the need for strong governance. Without strong governance, the company may end up with a difficult to manage Azure environment and costs, which are hard to track and control. The CTO is interested in understanding how Azure manages and enforces governance standards.

Learning objectives

In this module, youâ€™ll learn how to:

- Design for governance.
- Design for management groups.
- Design for Azure subscriptions.
- Design for resource groups.
- Design for Azure policies.
- Design for resource tags.
- Design for Azure blueprints.

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions. The module concepts are covered in:

Design Identity, Governance, and Monitoring

- Design Governance

Prerequisites

- Conceptual knowledge of governance policies, resource organization, and subscription management.
- Working experience with organizing resources, applying governance policies, and enforcing compliance requirements.

Design for governance

Governance provides mechanisms and processes to maintain control over your applications and resources in Azure. Governance involves determining your requirements, planning your initiatives, and setting strategic priorities.

To effectively apply your governance strategies, you must first create a hierarchical structure. This structure lets you apply governance strategies exactly where they're needed. The governance strategies we'll cover in this module are Azure policy and resource tags.



A typical Azure hierarchy has four levels: management groups, subscriptions, resource groups, and resources. We'll examine each level in more detail, but here's an overview.

- **Management groups** help you manage access, policy, and compliance for multiple subscriptions.
- **Subscriptions** are logical containers that serve as units of management and scale. Subscriptions are also billing boundaries.
- **Resource groups** are logical containers into which Azure resources are deployed and managed.
- **Resources** are instances of services that you create. For example, virtual machines, storage, and SQL databases.

Note: The **tenant root group** contains all the management groups and subscriptions. This group allows global policies and Azure role assignments to be applied at the directory level.

Design for management groups

Management groups are containers that help you manage access, policy, and compliance across **multiple subscriptions**. You can use management groups to:

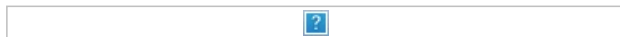
- Limit in which regions, across several subscriptions, virtual machines can be created.
- Provide user access to multiple subscriptions by creating one role assignment that will be inherited by other subscriptions.
- Monitor and audit, across subscriptions, role, and policy assignments.

Things to know about management groups

- Management groups can be used to aggregate policy and initiative assignments via Azure Policy.
- A management group tree can support up to **six levels of depth**. This limit doesn't include the tenant root level or the subscription level.
- Azure role-based access control authorization for management group operations isn't enabled by default.
- By default, all new subscriptions will be placed under the root management group.

Things to consider when creating management groups

Tailwind Traders has Sales, Corporate, and IT departments. The Sales department manages offices in the West and in the East. The Corporate main office includes Human Resources (HR) and Legal. The Information Technology (IT) department handles research, development, and production. There are currently two apps hosted in Azure. Here's a proposed management group hierarchy.



- **Design management groups with governance in mind.** For example, apply Azure policies at the management group level for all workloads that require the same security, compliance, connectivity, and feature settings.
- **Keep the management group hierarchy reasonably flat.** Ideally have no more than three or four levels. A hierarchy that is too flat doesn't provide flexibility and complexity for large organizations. A hierarchy with too many levels will be difficult to manage.
- **Consider a top-level management group.** This management group supports common platform policy and Azure role assignments across the whole organization. For example, the Tailwinds management group is a top-level management group for all organizational-wide policies.
- **Consider an organizational or departmental structure.** An organizational structure will be easy to understand. For example, the Sales, Corporate, and IT management groups.
- **Consider a geographical structure.** A geographical structure allows for compliance policies in different regions. For example, the West and East management groups in Sales.
- **Consider a production management group.** A production management group creates policies that apply to all corporate products. In our example, the Production management group provides product-specific policies for corporate apps.
- **Consider a sandbox management group.** A sandbox management group lets users experiment with Azure. The sandbox provides isolation from your development, test, and production environments. Users can experiment with resources that might not yet be allowed in production environments.
- **Consider isolating sensitive information in a separate management group.** In our example, the Corporate management group provides more standard and compliance policies for the main office.

Design for subscriptions

Azure Subscriptions are logical containers that serve as units of management and scale and billing boundaries. Limits and quotas can be applied, and each organization can use subscriptions to manage costs and resources by group.

Things to know about subscriptions

Using Azure requires an Azure subscription. A subscription provides you with a logical container to provision and pay for Azure products and services. There are **several types of subscriptions**, such as Enterprise Agreement and Pay-as-You-Go.



You can use subscriptions to:

- Organize specialized workloads that need to scale outside the existing subscription limits.
- Provide different billing environments such as development, test, and production.
- Achieve compliance by applying policies to a subscription.
- Manage and track costs for your organizational structure.

Things to consider when creating subscriptions

Tailwind Traders has established their management group structure. Now they need to determine where to assign subscriptions. Here's one possible solution.



- **Treat subscriptions as a democratized unit of management.** Align your subscriptions with business needs and priorities.
- **Group subscriptions together under management groups.** Grouping ensures that subscriptions with the same set of policies and Azure role assignments can inherit them from a management group. For example, both the West and East subscriptions will inherit policy from the Sales management group.
- **Consider a dedicated shared services subscription.** A shared services subscription ensures that all common network resources are billed together and isolated from other workloads. For example, Azure ExpressRoute and Virtual WAN.
- **Consider subscription scale limits.** Subscriptions serve as a scale unit for component workloads. For example, large, specialized workloads like high-performance computing, IoT, and SAP are all better suited to use separate subscriptions. Separate subscriptions will avoid **resource limits** (such as a limit of 50 Azure Data Factory integrations).
- **Consider administrative management.** Subscriptions provide a management boundary, which allows for a clear separation of concerns. Will each subscription need a separate administrator, or can you combine subscriptions? In our example, the Corporate management group could have a single subscription for both the HR and Legal departments.
- **Consider how you'll assign Azure policies?** Both management groups and subscriptions serve as a boundary for the assignment of Azure policies. For example, workloads such as Payment Card Industry (PCI) workloads typically require additional policies to achieve compliance. Instead of using a management group to group workloads that require PCI compliance, you can achieve the same isolation with a subscription. These types of decisions ensure you don't have too many management groups with only a few subscriptions.
- **Consider network topologies.** Virtual networks can't be shared across subscriptions. Resources can connect across subscriptions with different technologies such as virtual network peering or Virtual Private Networks (VPNs). Consider which workloads must communicate with each other when you decide whether a new subscription is required.
- **Consider making subscription owners aware of their roles and responsibilities.** For example, conduct an access review using Azure AD Privileged Identity Management quarterly or twice a year. Access reviews ensure privileges don't proliferate as users move within the customer organization.

Note: One size doesn't fit all for subscriptions. What works for one business unit might not work for another.

Design for resource groups

Resource groups are logical containers into which Azure resources are deployed and managed. These resources can include web apps, databases, and storage accounts. You can use resource groups to:

- Place resources of similar usage, type, or location in logical groups.
- Organize resources by life cycle so all the resources can be created or deleted at the same time.
- Apply role permissions to a group of resources or give a group access to administer a group of resources.
- Use resource locks to protect individual resources from deletion or change.

Things to know about resource groups

- Resource groups have their own location (region) assigned. This region is where the metadata is stored.
- If the resource group's region is temporarily unavailable, you can't update resources in the resource group because the metadata is unavailable. The resources in other regions will still function as expected, but you can't update them.
- Resources in the resource group can be in different regions.
- A resource can connect to resources in other resource groups. For example, you can have a web app that connects to a database in a different resource group.
- Resources can be moved between resource groups.
- You can add a resource to or remove a resource from a resource group at any time.
- Resource groups can't be nested.
- Each resource must be in one, and only one, resource group.
- Resource groups cannot be renamed.

Things to consider when creating resource groups


Tailwind Traders has two Azure-based apps (App1 and App2). Each app has a web service with SQL database, virtual machines, and storage. Tailwind Traders needs to decide how to organize their resource groups.



- **Consider group by type.** Group by type is most appropriate for on-demand services that aren't associated with an app. In our example, a resource group for the SQL databases (SQL-RG) and a separate resource group (WEB-RG) for the web services.



- **Consider group by app.** Group by app is appropriate when all the resources have the same policies and life cycle. This method could also be applied to test or prototype environments. For Tailwind Traders, App1 and App2 would have separate resource groups. Each group would have all the resources for that application.



- **Consider group by department, group by location (region), and group by billing (cost center).** These grouping strategies aren't common but may be useful in your situation.
- **Consider a combination of organizational strategies.** Don't restrict your thinking to one strategy. Often a combination of different strategies is best.
- **Consider resource life cycle.** Do you want to deploy, update, and delete resources at the same time? If so, you may want to place all those resources in one resource group.
- **Consider administration overhead.** How many resource groups would you like to manage? Do you have centralized or decentralized Azure administrators?
- **Consider resource access control.** At the resource group level, you can assign Azure policies, Azure roles, and resource locks. **Resource locks** prevent unexpected changes to critical resources.
- **Consider compliance requirements.** Do you need to ensure your resource group metadata is stored in a particular region?

Design for resource tagging

Resource tags are another way to organize resources. Tags provide extra information, or metadata, about your resources.

Tip: Before starting a resource tagging project, ask yourself what you want to accomplish. Will the tags be used for reporting or billing? Perhaps you will use the tags to enable more effective searching? Maybe the tags will be used in automated scripts. Be sure to clearly define your goals.

Things to know about resource tags

- A resource tag consists of a name-value pair. For example, env: production or env: test.
- You can assign one or more tags to each Azure resource, resource group, or subscription.
- You can add, modify, or delete resource tags. These actions can be done with PowerShell, the Azure CLI, Azure Resource Manager templates, the REST API, or the Azure portal.
- You can **apply tags** to a resource group. However, the resources in the resource group don't inherit those tags by default.

Things to consider when creating resource tags

Tailwind Traders has created their organization hierarchy and now needs to determine which resource tags to apply.



- **Consider your organization's taxonomy.** Has your organization already defined terms for compliance or cost reporting? Aligning tags with accepted department nomenclature will make it easier to understand. For example, are office locations, confidentiality levels, and programs already defined?
- **Consider whether you need IT-aligned or business-aligned tagging.** Generally, you can choose IT-aligned tagging or business-aligned tagging. A combination of the two approaches can be effective. Many organizations are shifting from IT-aligned to business-aligned tagging strategies.

Tagging scheme	Description	Example
IT-aligned tagging	Focuses on workload, application, function, or environment criteria. Reduces the complexity of monitoring assets. Simplifies making management decisions based on operational requirements.	Our printers are busy 80% of the time. We have five high-speed color printers and should buy more.
Business-aligned tagging	Focuses on accounting, business ownership, cost responsibility, or business criticality. Provides improved accounting for costs and value of IT assets to the overall business. Shifts the focus from an asset's operational cost to an asset's business value.	Our marketing department's promotional literature has increased sales revenue 10%. We should invest in more printing capabilities.

Consider the type of tagging that is required.

Resource tags generally fall into five categories functional, classification, accounting, partnership, and purpose.

Tag type	Examples	Description
Functional	app = catalogsearch1 tier = web webserver = apache env = prod, dev, staging	Categorize resources by their purpose within a workload, what environment they've been deployed to, or other functionality and operational details.
Classification	confidentiality = private SLA = 24hours	Classifies a resource by how it's used and what policies apply to it.

Accounting	department = finance Program = business-initiative region = northamerica	Allows a resource to be associated with specific groups within an organization for billing purposes.
Partnership	owner = jsmith contactalias = catsearchowners â€Žstakeholders = user1;user2;user3	Provides information about what people (outside of IT) are related or otherwise affected by the resource.
Purpose	businessprocess = support businessimpact = moderate â€Žrevenueimpact = high	Aligns resources to business functions to better support investment decisions.

- **Consider starting with a few tags and then scaling out.** Your resource tagging approach can be simple or complex. Rather than identifying all the possible tags your organization needs, prototype using a few important or critical tags. Determine how effective the tagging is before adding more resource tags.
- **Consider using Azure policy to apply tags and enforce tagging rules and conventions.** Resource tagging is only effective if itâ€™s used consistently across an organization. You can use Azure policy to require certain tags be added to new resources as they're provisioned. You can also define rules that reapply tags that have been removed.
- **Consider which resources require tagging.** Keep in mind that you don't need to enforce that a specific tag is present on all your resources. For example, you might decide that only mission-critical resources have the Impact tag. All non-tagged resources would then not be considered as mission critical.

Note: Youâ€™ll need support from many different stakeholders to implement an effective resource tagging structure.

Design for Azure Policy

Azure Policy is a service in Azure that enables you to create, assign, and manage policies that control or audit your resources. These policies enforce different rules over your resource configurations so that those configurations stay compliant with corporate standards.

Things to know about Azure policy

- Azure policy lets you define both individual policies and groups of related policies, called initiatives. Azure Policy comes with many **built-in policy** and **initiative definitions**.
- Azure policy lets you scope and enforce your policies at different levels in the organizational hierarchy.
- Azure policies are inherited down the hierarchy.
- Azure policy evaluates your resources and highlights resources that aren't compliant with the policies you've created.
- Azure policy can prevent noncompliant resources from being created.
- Azure Policy can automatically remediate noncompliant resources.
- Azure Policy evaluates all resources in Azure and Arc enabled resources (specific resource types hosted outside of Azure).
- Azure policy integrates with Azure DevOps by applying pre-deployment and post-deployment policies.

Things to consider when using Azure policy

Tailwind Traders is now ready to consider applying Azure policy to their apps. Some policies will be applied at the Production management group level. Other policies will be assigned at the app level.



- **Consider using the Azure policy compliance dashboard.** The Azure policy compliance dashboard provides an aggregated view to help evaluate the overall state of the environment. You can drill down to a per-resource, or per-policy level granularity. You can also use capabilities like bulk remediation for existing resources and automatic remediation for new resources, to resolve issues rapidly and effectively.
- **Consider when Azure policy evaluates resources.** Azure policy evaluates resources at specific times. It's important to understand when an evaluation is triggered. There may be a lag in identifying non-compliant resources. The following events or times will trigger an evaluation.
 - A resource has been created, deleted, or updated in scope with a policy assignment.
 - A policy or an initiative is newly assigned to a scope.
 - A policy or an initiative that's been assigned to a scope is updated.
 - The standard compliance evaluation cycle (happens once every 24 hours).
- **Consider what you'll do if a resource is non-compliant.** Organizations will vary in how they respond to non-compliant resources. Your strategy may be different depending on the resource. Here's some examples of what to do if a resource is non-compliant.
 - Deny a change to a resource.
 - Log changes to a resource.
 - Alter a resource before or after a change.
 - Deploy related compliant resources.
- **Consider when to automatically remediate non-compliant resources.** In some cases, Azure policy can automatically remediate noncompliant resources. Remediation is especially useful in resource tagging. Azure policy can tag resources and reapply tags that have been removed. For example, Azure policy can ensure all resources in a certain resource group should be tagged with the Location tag.
- **Consider how Azure policy is different from role-based access control (RBAC).** It's important not to confuse Azure Policy and Azure RBAC. Azure RBAC and Azure Policy should be used together to achieve full scope control.
 - You use Azure Policy to ensure that the resource state is compliant to your organization's business rules. Compliance doesn't depend on who made the change or who has permission to make changes. Azure Policy will evaluate the state of a resource, and act to ensure the resource stays compliant.

- You use Azure RBAC to focus on user actions at different scopes. Azure RBAC manages who has access to Azure resources, what they can do with those resources, and what areas they can access. If actions need to be controlled, then use Azure RBAC. If an individual has access to complete an action, but the result is a non-compliant resource, Azure Policy still blocks the action.

Once you have determined your identity management solution, it's time to think about resource access. What resources should these identities be able to access? How will you enforce that access? How will you monitor and review the access?

A user's identity goes through several phases. Initially, the user will have no access. Access can then be granted through role-based access control and verified with Azure AD conditional access. Azure AD Identity Protection can be used to monitor the user's access. And then periodically Azure AD access reviews will confirm the access is still required.

Design for Azure role-based access control (RBAC)

Azure RBAC allows you to grant access to Azure resources that you control. Suppose you need to manage access to resources in Azure for the Tailwind Trader’s development, engineering, and marketing teams. Here are some scenarios you can implement with Azure RBAC.

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

How does role-based access control work?

Azure RBAC evaluates each request for access. The evaluations will determine if access should be blocked, not allowed, or allowed.



Things to consider when using role-based access control

- **Remember RBAC is an allow model.** An allow model means when you’re assigned a role, Azure RBAC allows you to perform certain actions. For example, a role assignment could grant you read permissions to a resource group. To have write permissions the role would need to explicitly allow write access.
- **Assign at the highest scope level that meets the requirements.** Your first step is to accurately define the role definition and related permissions. Next assign roles to users, groups, and service principals. Lastly, scope the roles to management groups, subscriptions, resource groups, and resources. Be sure to assign at the highest scope level that meets the requirements.



- **Only grant users the access they need.** When planning your access control strategy, it's a best practice to grant users the least privilege to get their work done. This allows you to segregate duties within your team. By limiting roles and scopes, you limit what resources are at risk if the security principal is ever compromised. Creating a diagram like this, might help to explain Azure RBAC roles.



- **Assign roles to groups, not users.** To make role assignments more manageable, avoid assigning roles directly to users. Instead, assign roles to groups. Assigning roles to groups helps minimize the number of role assignments.
- **Know when to use Azure policies.** Azure policies focuses on resource properties. For example, during deployment, a policy can ensure users can only deploy certain virtual machines in a resource group. A combination of Azure policies and Azure RBAC can provide effective access control.

Area	Azure Policy	Role-based Access Control
Description	Ensure resources are compliant with a set of rules.	Authorization system to provide fine-grained access controls.
Focus	Focused on the properties of resources.	Focused on what resources the users can access.
Implementation	Specify a set of rules.	Assign roles and scopes.
Default access	By default, rules are set to allow.	By default, all access is denied.

- **Know when to create a custom role.** Sometimes, the built-in roles don't grant the precise level of access you need. Custom roles allow you to define roles that meet the specific needs of your organization. Custom roles can be shared between subscriptions that trust the same Azure Active Directory.
- **Consider what happens if you have overlapping role assignments.** Azure RBAC is an additive model, so your effective permissions are the sum of your role assignments. Consider a user is granted the Contributor role at the subscription scope and the Reader role on a resource group. The sum of the Contributor permissions and the Reader permissions is effectively the Contributor role for the subscription. Therefore, in this case, the Reader role assignment has no impact.

Design for Azure Blueprints

Azure Blueprints lets you define a repeatable set of governance tools and standard Azure resources that your organization requires. Azure Blueprints are used to scale governance practices throughout an organization.

A blueprint is a package related to the implementation of Azure cloud services, security, and design. A blueprint can be reused to maintain consistency and compliance.



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

Resource	Hierarchy Options	Description
Resource Groups	Subscription	Create a new resource group for use by other artifacts within the blueprint. These placeholder resource groups enable you to organize resources exactly the way you want them structured and provides a scope limiter for included policy and role assignment artifacts and templates. Templates, including nested and linked templates, are used to compose complex environments. Example environments: a SharePoint farm, Azure Automation State Configuration, or a Log Analytics workspace.
Azure Resource Manager Template	Subscription, Resource group	Allows assignment of a policy or initiative to the subscription the blueprint is assigned to. The policy or initiative must be within the scope of the blueprint definition location. If the policy or initiative has parameters, these parameters are assigned at creation of the blueprint or during blueprint assignment.
Policy Assignment	Subscription, Resource group	Add an existing user or group to a built-in role to make sure the right people always have the right access to your resources. Role assignments can be defined for the entire subscription or nested to a specific resource group included in the blueprint.
Role Assignment	Subscription, Resource group	

How are Azure Blueprints different from Azure Policy

A policy is a default allow and explicit deny system focused on resource properties during deployment and for already existing resources. It supports cloud governance by validating those resources within a subscription adhere to requirements and standards.

A policy can be included as one of many artifacts in a blueprint definition. Including a policy in a blueprint enables the creation of the right pattern or design during assignment of the blueprint. The policy inclusion makes sure that only approved or expected changes can be made to the environment to protect ongoing compliance to the intent of the blueprint.

Summary and resources

In this module, youâ€™ve learned how to design with governance in mind. You have learned different strategies for implementing an organizational hierarchy and enforcing rules and policies. You should now be able to:

- Design for governance.
- Design for management groups.
- Design for Azure subscriptions.
- Design for resource groups.
- Design for Azure policy.
- Design for resource tags.
- Design for Azure blueprints.

Learn more with Azure documentation

- [Governance in the Microsoft Cloud Adoption Framework for Azure - Cloud Adoption Framework | Microsoft Docs](#)
- [Organize your resources with management groups - Azure Governance - Azure governance | Microsoft Docs](#)
- [Organize and manage multiple Azure subscriptions - Cloud Adoption Framework | Microsoft Docs](#)
- [Resource naming and tagging decision guide - Cloud Adoption Framework | Microsoft Docs](#)
- [Recommended policies for Azure services - Azure Policy | Microsoft Docs](#)

Learn more with self-paced training

- [Build a cloud governance strategy on Azure - Learn | Microsoft Docs](#)
- [Describe core Azure architectural components - Learn | Microsoft Docs](#)
- [Microsoft Cloud Adoption Framework for Azure - Learn | Microsoft Docs](#)
- [Introduction to Azure Blueprints - Learn](#)
- [Secure your Azure resources with Azure role-based access control \(Azure RBAC\) - Learn](#)

Learn more with optional hands-on exercises

- [List access using Azure RBAC and the Azure portal](#)

Introduction

Let's suppose you work as an Architect at Tailwind Traders. Tailwind Traders is a company that specializes in hardware manufacturing with online sales. Your management team tells you several development projects need to migrate to the cloud. There are also several new projects that should be optimized for the cloud.



You know the department's budget is tight. It will be important to select the right compute technology for each project. Ideally, you would like to create compute resources, configure the resources, and pay for only what you use.

Learning objectives

In this module, you'll learn how to:

- Choose a compute service.
- Design for Azure virtual machines solutions.
- Design for Azure Batch solutions.
- Design for Azure Function solutions.
- Design for Azure Logic App solutions.
- Design for Azure Container Instances solutions.
- Design for Azure App Services solutions.
- Design for Azure Kubernetes Service solutions.

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions.

Design for compute solutions

- Recommend an appropriately sized compute solution based on workload requirements.
- Recommend a Container-based compute solution.
- Recommend a Serverless-based compute solution
- Recommend a Virtual Machine-based compute solution

Prerequisites

- Conceptual knowledge of Azure compute solutions.
- Working experience with virtual machines, containers, and app service.

Choose a compute service

Compute refers to the hosting model for the computing resources that your applications run on. Azure offers several compute services, which we will cover in this module. Here's a short summary.

- **Virtual machines (IaaS).** Deploy and manage VMs inside an Azure virtual network.
- **Azure Batch (PaaS).** A managed service for running large-scale parallel and high-performance computing (HPC) applications.
- **Azure Functions (FaaS).** A managed service for running code in the cloud, without worrying about the infrastructure.
- **Azure Logic Apps (PaaS).** A cloud-based platform for creating and running automated workflows.
- **Container Instances (PaaS).** A fast and simple way to run a container in Azure. You don't provision any virtual machines and don't need a higher-level service.
- **App Service (PaaS).** A managed service for hosting web apps, mobile app back ends, RESTful APIs, or automated business processes.
- **Azure Kubernetes Service (PaaS).** A managed Kubernetes service for running containerized applications.
- **Azure Service Fabric.** A distributed systems platform that makes it easy to package, deploy, and manage scalable and reliable microservices and containers.

This [flowchart](#) provides high-level guidance on when to select each compute option. You'll want to refer to this diagram as we go through the choices.

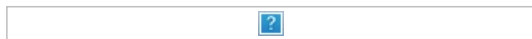


On the diagram, **Cloud optimized** is a strategy for migrating to the cloud. Cloud optimized refactors an application to take advantage of cloud-native features and capabilities. A **lift and shift** strategy migrates workloads without redesigning the application or making code changes. Lift-and-shift lets organizations keep running their applications with minimal changes and disruption.

Tip: The output from this flowchart is a **starting point** for consideration. You'll need to do a more detailed evaluation of the service to determine if it meets your needs. The next sections will help with this analysis.

Review the compute hosting options

The compute solution has three hosting options: Infrastructure as a Service, Platform as a Service, and Function as a Service? There's also Software-as-a-Service which isn't a compute solution. The **hosting option** determines the developer and cloud provider responsibilities. This hosting decision will influence your design.



- **Infrastructure-as-a-Service (IaaS)** lets you create individual VMs along with the associated networking and storage components. Then you deploy whatever software and applications you want onto those VMs. This model is the closest to a traditional on-premises environment, except that Microsoft manages the infrastructure. You still manage the individual VMs.
- **Platform-as-a-Service (PaaS)** provides a managed hosting environment, where you can deploy your application without needing to manage VMs or networking resources. Azure App Service is a PaaS service.
- **Functions-as-a-Service (FaaS)** goes even further in removing the need to worry about the hosting environment. In a FaaS model, you deploy your code, and the service automatically runs it. Azure Functions is a FaaS service.

Design for Azure virtual machine solutions

Whether you're building new or migrating using a lift and shift pattern, [Azure virtual machines](#) (VMs) might be a choice for you. Azure VMs are the basis of the Azure [Infrastructure-as-a-Service \(IaaS\) model](#). Azure VMs can be used for the development, testing, deployment of applications in the cloud, or extension of your data center. Azure VMs provide a fast, scalable, flexible way to add more compute power to your enterprise.

There are two main scenarios for deciding to use virtual machines.

- **Build new** because demand for your application can fluctuate. It makes economic sense to run it on a VM in Azure.
- **Lift and shift (rehosting)** migration strategy that involves moving data and applications from an on-premises location to Azure-based virtual machines in the cloud.



Let's walk through a checklist of things to think about when designing for Azure VMs.

- Start with the network
- Name the VM
- Decide the location for the VM
- Determine the size of the VM
- Review the pricing model
- Review the storage options
- Select an operating system

Start with the network

The first thing to think about isn't the virtual machine at all - it's the network. So, spend some time thinking about your network configuration. Network addresses and subnets aren't trivial to change once you have them set up. If you have an on-premises network, you'll want to carefully consider the network topology before creating any virtual machines.

Name the virtual machine

One thing people don't put much thought into is the **name** of the VM. This name defines a manageable **Azure resource**, and it's also not easy to change. Choose names that are meaningful and consistent so you can easily identify what the VM does. For example, devusc-webvm01 might represent the first development web server hosted in the US South Central location.

Decide the location for the VM

Azure has data centers all over the world filled with servers and disks. These datacenters are grouped into geographic regions ('West US', 'North Europe', 'Southeast Asia' etc.) to provide redundancy and availability.

Each virtual machine is in a region where you want the resources (CPU, storage etc.) to be allocated. Regional location lets you place your VMs as close as possible to your users. This location can improve performance and ensure you meet any legal, compliance, or tax requirements.

Two other things to think about the location choice. First, the location can limit your available options. Each region has different hardware available, and some configurations aren't available in all regions. Second, there are price differences between locations. To find the most cost-effective choice, check for your required configuration in different regions.

Determine the size of the VM

Once you have the name and location set, you need to decide on the size of your VM. Azure offers different memory and storage options for different [VM sizes](#).

The best way to determine the appropriate VM size is to consider the type of workload your VM needs to run. Based on the workload, you're able to choose from a subset of available VM sizes. Azure virtual machine workloads are classified as follows.

Option	Description
General purpose	General-purpose VMs are designed to have a balanced CPU-to-memory ratio. Ideal for testing and development, small to medium databases, and low to medium traffic web servers. Compute optimized VMs are designed to

Compute optimized	have a high CPU-to-memory ratio. Suitable for medium traffic web servers, network appliances, batch processes, and application servers.
Memory optimized	Memory optimized VMs are designed to have a high memory-to-CPU ratio. Great for relational database servers, medium to large caches, and in-memory analytics.
Storage optimized	Storage optimized VMs are designed to have high disk throughput and IO. Ideal for VMs running databases.
GPU	GPU VMs are specialized virtual machines targeted for heavy graphics rendering and video editing. These VMs are ideal options for model training and inferencing with deep learning.
High performance computes	High performance compute is the fastest and most powerful CPU virtual machines with optional high-throughput network interfaces.

Tip: Try the [Virtual machines selector tool](#) to find other sizes that best fit your workload.

Review the pricing model

There are two separate costs the subscription will be charged for every VM: compute and storage. By separating these costs, you can scale them independently and only pay for what you need.

- **Compute costs** - Compute expenses are priced on a per-hour basis but billed on a per-minute basis. For example, you're only charged for 55 minutes of usage if the VM is deployed for 55 minutes. You're not charged for compute capacity if you stop and deallocate the VM. The [hourly price](#) varies based on the VM size and OS you select.
- **Storage costs** - You're charged separately for the storage the VM uses. The status of the VM has no relation to the storage charges that will be incurred. You are always charged for storage used by the disks.

Review the storage options

Managed disks handle Azure storage account creation and management in the background for you. You specify the disk size and the performance tier (Standard or Premium), and Azure creates and manages the disk. As you add disks or scale the VM up and down, you don't have to worry about the storage being used.

Select an operating system

Azure provides various OS images that you can install into the VM, including several versions of Windows and flavors of Linux. Azure bundles the cost of the OS license into the price.

If you're looking for more than just base OS images, you can search the [Azure Marketplace](#). There are various install images that include not just the OS but popular software tools. For example, there is an image for WordPress. The image stack consists of a Linux server, Apache web server, a MySQL database, and PHP. So, instead of setting up and configuring each component, you can install a Marketplace image and get the entire stack all at once.

Lastly, if you can't find a suitable OS image, you can create your own disk image. Your disk image can then be uploaded to Azure storage and used to create an Azure VM. Keep in mind that Azure only supports 64-bit operating systems.

Important: There's a lot to think about when planning for virtual machines. Take a few minutes to think through what you have learned? Will you need virtual machines? If so, what decisions will you make on size, pricing, and operating systems?

Design for Azure Batch solutions

Azure Batch runs large-scale applications efficiently in the cloud. You can schedule compute-intensive tasks and dynamically adjust resources for your solution without managing infrastructure. Azure Batch can create and manage a pool of compute nodes (virtual machines). Azure Batch can also install the application that you want to run, and schedule jobs to run on the compute nodes.



When to use Azure Batch

Azure Batch works well with applications that run independently (parallel workloads). Azure Batch is also effective for applications that need to communicate with each other (tightly coupled workloads). For example, you can use Batch to build a service that runs a Monte Carlo simulation for a financial services company or a service to process images.

Azure Batch enables large-scale parallel and high-performance computing (HPC) batch jobs with the ability to scale to tens, hundreds, or thousands of VMs. When you're ready to run a job, Batch does the following.

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

How Azure Batch works

As shown in the following diagram, a typical real-world scenario for Azure Batch requires data and application files. The Batch workflow begins with uploading the data and application files to an Azure storage account. Based on the demand, you create a Batch pool with as many Windows or Linux virtual compute nodes as needed. If the demand increases, compute nodes can be automatically scaled.



You can think of the diagram in two parts:

- **Your service** that uses Azure as the platform. The platform is for completing computationally intensive work and then retrieving results. You can also monitor jobs and task progress.
- **Batch as the compute platform behind your service.** Batch uses Azure Storage to fetch applications or data needed to complete a task. Azure Batch writes output to Azure storage. Behind the scenes, there are collections (pools) of virtual machines. Pools are the resources that jobs, and tasks are executed on.

Best practices and useful tips for using the Azure Batch service

Best practices for Azure Batch are grouped into pools, nodes, and jobs.

- **Pools.** If your jobs consist of short-running tasks, don't create a new pool for each job. The overhead to create new pools will diminish the run time of the job. Also, it's best to have your jobs use pools dynamically. If your jobs use the same pool for everything, there's a chance that jobs won't run if something goes wrong with the pool.
- **Nodes.** Individual nodes aren't guaranteed to always be available. If your Batch workload requires deterministic, guaranteed progress, you should allocate pools with multiple nodes. Consider using isolated VM sizes for workloads with compliance or regulatory requirements.
- **Jobs.** Uniquely name your jobs so you can accurately monitor and log the activity. Consider grouping your tasks into efficiently sized jobs. For example, it's more efficient to use a single job containing 1000 tasks rather than creating 100 jobs that contain 10 tasks each.

Tip: We've covered just a few best practices. Take a few minutes to read more about [Best practices - Azure Batch | Microsoft Docs](#).

Design for Azure App Services solutions

Azure App Service is an HTTP-based service that lets you build and host web apps, background jobs, mobile backends, and RESTful APIs. App Service lets you use the programming language of your choice. Azure App Services offers automatic scaling and high availability. App Service enables automated deployments from GitHub, Azure DevOps, or any Git repo.



Important: Azure App Service is platform as a service (PaaS) environment. You focus on the website development and API logic. Azure handles the infrastructure to run and scale your web applications.

Types of app services

With Azure App Service, all your apps share **common benefits** including:

- Development in multiple languages and frameworks.
- Integrated deployment and management with secured endpoints.
- Global scale with high availability.
- Built-in load balancing and traffic management.

These benefits make App Service the ideal choice for any hosted web application.

Azure App Service costs

You pay for the Azure compute resources your app uses while it processes requests. The cost is based on the **App Service plan** you choose. The App Service plan determines how much hardware is devoted to your host. For example, the plan determines whether it's dedicated or shared hardware and how much memory is reserved. You can have different app service plans for different apps.

Your App Service plan can be scaled up and down at any time. For example, you can start testing your web app in a Free App Service plan and pay nothing. When you want to add your custom DNS name to the web app, just scale your plan up to the Shared tier.

Use App Services deployment slots for continuous deployment

Azure DevOps provides developer services for support teams to plan work, collaborate on code development, and build and deploy applications. Whenever possible when continuously deploying your code, use **deployment slots** for a new production build.



When using a Standard App Service Plan tier or better, you can deploy your app to a staging environment, validate your changes, and do smoke tests. When you're ready, you can swap your staging and production slots. The swap operation warms up the necessary worker instances to match your production scale, thus eliminating downtime.

Consider authentication and authorization options

Implementing a secure solution for authentication (signing-in users) and authorization (providing access to secure data) can take significant effort. Azure App Service provides **built-in authentication and authorization capabilities** (sometimes referred to as "Easy Auth"). So, you can sign in users and access data by writing minimal or no code. Here are some benefits.

- Azure App Service provides built-in auth capabilities for your web app or API. You don't need to implement the authentication yourself.
- It's built directly into the platform. You don't need any language, SDK, security expertise, or even any code to utilize.
- You can integrate with multiple sign-in providers. For example, Azure AD, Facebook, Google, and Twitter.

Tip: The built-in authentication features for App Service is the same for Azure Functions.

When to use web apps

App Service supports web apps using ASP.NET, ASP.NET Core, Java, Ruby, Node.js, PHP, or Python. You can choose either Windows or Linux as the host operating system.

When to use API apps

Much like hosting a website, you can build REST-based web APIs by using your choice of language and framework. You get full Swagger support and the ability to package and publish your API in Azure Marketplace. The produced apps can be consumed from any HTTP- or HTTPS-based client.

When to use WebJobs

You can use the [WebJobs](#) feature to run a program or script. Program examples include Java, PHP, Python, or Node.js. Script examples include cmd, bat, PowerShell, or Bash. WebJobs can be scheduled or run by a trigger. WebJobs are often used to run background tasks as part of your application logic.

When to use Mobile apps

Use the Mobile Apps feature of App Service to quickly build a back end for iOS and Android apps. With just a few steps in the Azure portal, you can:

- Store mobile app data in a cloud-based SQL database.
- Authenticate customers against common social providers, such as MSA, Google, Twitter, and Facebook.
- Send push notifications.
- Execute custom back-end logic in C# or Node.js.

On the mobile app side, there's SDK support for native iOS and Android, Xamarin, and React native apps.

Design for Azure Container Instance solutions



Virtual machines are an excellent way to reduce costs versus the investments that are necessary for physical hardware. However, each virtual machine is still limited to a single operating system. If you want to run multiple instances of an application on a single host machine, containers are an excellent choice.

Azure Container Instances are a fast and simple way to run a container on Azure. Azure Container Instance scenarios include simple applications, task automation, and build jobs. Here are some benefits of containers.

- **Fast startup.** Launch containers in seconds.
- **Per second billing.** Incur costs only while the container is running.
- **Hypervisor-level security.** Isolate your application as completely as it would be in a VM.
- **Custom sizes.** Specify exact values for CPU cores and memory.
- **Persistent storage.** Mount Azure Files shares directly to a container to retrieve and persist state.
- **Linux and Windows.** Schedule both Windows and Linux containers using the same API.

What are container groups?

The top-level resource in Azure Container Instances is the container group. A container group is a collection of containers that get scheduled on the same host machine. The containers in a container group share a lifecycle, resources, local network, and storage volumes.



Multi-container groups are useful in cases where you want to divide a single functional task into several container images. These images can then be delivered by different teams and have separate resource requirements. Example usage could include:

- A container serving a web application and a container pulling the latest content from source control.
- An application container and a logging container. The logging container collects the logs and metrics output by the main application and writes them to long-term storage.
- An application container and a monitoring container. The monitoring container periodically makes a request to the application to ensure that it's running and responding correctly and raises an alert if it's not.
- A front-end container and a back-end container. The front end might serve a web application, with the back end running a service to retrieve data.

Security considerations for container instances

When working with container instances, consider these security best practices.

- **Use a private registry.** Containers are built from images that are stored in one or more repositories. These repositories can belong to a public registry or to a private registry. An example of a private registry is the **Docker Trusted Registry**, which can be installed on-premises or in a virtual private cloud. Another example is **Azure Container Registry** to build, store, and manage container images and artifacts.
- **Ensure the integrity of images throughout the lifecycle.** Part of managing security throughout the container lifecycle is to ensure the integrity of the container images. Images with vulnerabilities, even minor, shouldn't be allowed to run in a production environment. Keep the number of production images small to ensure that they can be managed effectively.
- **Monitor container resource activity.** Monitor your resource activity, like files, network, and other resources that your containers access. Monitoring resource activity and consumption is useful both for performance monitoring and as a security measure.

Tip: Read more about [Security considerations for container instances](#)

When to choose containers instead of virtual machines

Feature	Containers	Virtual Machines
	Typically provides lightweight isolation from the host and other	Provides complete isolation from the host operating system and other VMs. Isolation is useful when a

Isolation	containers but doesn't provide as strong a security boundary as a virtual machine.	strong security boundary is critical, such as hosting apps from competing companies on the same server or cluster.
Operating system	Runs the user mode portion of an operating system and can be tailored to contain just the needed services for your app, using fewer system resources.	Runs a complete operating system. Typically, requires more system resources (CPU, memory, and storage).
Deployment	Deploy individual containers by using Docker via command line; deploy multiple containers by using an orchestrator such as Azure Kubernetes Service.	Deploy individual VMs by using Windows Admin Center or Hyper-V Manager; deploy multiple VMs by using PowerShell or System Center Virtual Machine Manager.
Persistent storage	Use Azure Disks for local storage for a single node, or Azure Files (SMB shares) for storage shared by multiple nodes or servers.	Use a virtual hard disk (VHD) for local storage for a single VM, or an SMB file share for storage shared by multiple servers.
Fault tolerance	If a cluster node fails, any containers running on it are rapidly recreated by the orchestrator on another cluster node.	VMs can fail over to another server in a cluster, with the VM's operating system restarting on the new server.

Design for Azure Kubernetes solutions

Kubernetes is a portable, extensible open-source platform for automating deployment, scaling, and the management of containerized workloads. This orchestration platform gives us the same ease of use and flexibility as with Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) offerings. Kubernetes provides both container management and container orchestration.

- Container management is the process of organizing, adding, removing, or updating a significant number of containers. Most of these tasks are manual and error prone.
- Container orchestration is a system that automatically deploys and manages containerized apps. For example, the orchestrator can dynamically increase or decrease the deployed instances of the managed app. Or it can ensure all deployed container instances get updated if a new version of a service is released.



What is Azure Kubernetes Services (AKS)?

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment and makes it simple to deploy and manage containerized applications in Azure. Your AKS environment is enabled with features such as automated updates, self-healing, and easy scaling.



- The Kubernetes cluster is managed by Azure and is free. You manage the agent nodes in the cluster and only pay for the VMs on which your nodes run.
- When you create the cluster, you can use Resource Manager templates to automate cluster creation. With these templates, you specify features such as advanced networking, Azure Active Directory (AD) integration, and monitoring.
- With AKS, we get the benefits of open-source Kubernetes. You don't have the complexity or operational overhead running your own custom Kubernetes cluster.

When to use Azure Kubernetes Service

Here, we'll discuss how you can decide whether Azure Kubernetes Service (AKS) is the right choice for you.

You'll either approach your decision from a green field or a lift-and-shift project point of view. A green fields project will allow you to evaluate AKS based on default features. A lift-and-shift project will help you determine which features are best suited to support your migration. Here are a few factors to consider.

Factor	Things to consider
Identity and security management	Do you already use existing Azure resources and make use of Azure AD? If so, you can configure an AKS cluster to integrate with Azure AD and reuse existing identities and group membership.
Integrated logging and monitoring	Are you using Azure Monitor? If so, Azure Monitor provides performance visibility of the cluster.
Automatic cluster node and pod scaling	Do you need to scale up or down a large containerization environment? If so, AKS supports two auto cluster scaling options. The horizontal pod autoscaler watches the resource demand of pods and will increase pods to match demand. The cluster autoscaler component watches for pods that can't be scheduled because of node constraints. It will automatically scale cluster nodes to deploy scheduled pods.
Cluster node upgrades	Do you want to reduce the number of cluster management tasks? If so, AKS manages Kubernetes software upgrades and the process of cordoning off nodes and draining them.
Storage volume support	Does your application require persisted storage? If so, AKS supports both static and dynamic storage volumes. Pods can attach and reattach to these storage volumes as they're created or

	rescheduled on different nodes.
Virtual network support	Do you need pod to pod network communication or access to on-premises networks from your AKS cluster? If so, an AKS cluster can be deployed into an existing virtual network with ease.
Ingress with HTTP application routing support	Do you need to make your deployed applications publicly available? If so, the HTTP application routing add-on makes it easy to access AKS cluster deployed applications.
Docker image support	Do you already use Docker images for your containers? If so, AKS by default, supports the Docker file image format.
Private container registry	Do you need a private container registry? If so, AKS integrates with Azure Container Registry (ACR). You aren't limited to ACR though, you can use other container repositories, public, or private.

All the above features are configurable either when you create the cluster or following deployment.

Tip: Take a few minutes to read about how [Mercedes-Benz R&D is using AKS](#).

Design for Azure Function solutions



What are Azure Functions?

Azure Functions is a serverless application platform. Functions are used when you want to run a small piece of code in the cloud, without worrying about the infrastructure. Functions provide intrinsic scalability, and you're charged only for the resources used. You can write your function code in the language of your choice. Functions provide "compute on demand" in two significant ways.

- First, Azure Functions allows you to implement your system's logic into readily available blocks of code. These code blocks (functions) can run anytime you need to respond to critical events.
- Second, as requests increase, Azure Functions meets the demand with as many resources and function instances as necessary. As requests complete, any extra resources and application instances drop off automatically.

Scenarios for Azure Functions

Azure Functions are best when handling specific definable actions triggered by an event. For example, a function could process an API call and then store the processed data in Cosmos DB. Once the data transfer happens, another function could trigger a notification.



Tip: You can get some other ideas on how to use Azure Functions by visiting the [code samples](#) page.

Best practices and tips for using Azure Functions

- **Avoid long running functions.** Large, long-running functions can cause unexpected timeout issues. Whenever possible, refactor large functions into smaller function sets that work together and return responses faster. The default timeout is 300 seconds for Consumption Plan functions, 30 minutes for any other plan.
- **Know when to use durable functions.** **Durable functions** let you write stateful functions. So, behind the scenes, the function manages app state, checkpoints, and restarts. An example application pattern for durable functions is function chaining. Function chaining executes a sequence of functions in a specific order. The output of one function is applied to the input of another function. Do you understand how timeout issues can be overcome with durable functions and smaller function sets?



- **Organize functions for performance and scaling.** Consider how you want to group functions with different load profiles. For example, let's say you have two functions. One function processes many thousands of queued messages and has low memory requirements. The other function is only called occasionally but has high memory requirements. You might want to deploy separate function apps, so each function gets its own set of resources. Separate resources mean you can independently scale the functions.
- **Write defensive functions.** Design your functions assuming an exception could occur at any time. Downstream services, network outages, or memory limits could cause the function to fail. Plan out how you continue from a failure point.
- **Avoid sharing storage accounts.** When you create a function app, you must associate it with a storage account. To maximize performance, use a separate storage account for each function app. This is important if your function generates a high volume of storage transactions.

Tip: Take a few minutes to read about other [Azure Function best practices](#).

Note: Relativity, an e-discovery company, is using Azure functions to identify and resolve performance issues. Take a few minutes to [read about their successes](#).

Design for Azure Logic Apps

Azure Logic Apps is another type of serverless compute solution. Azure Logic Apps is a cloud-based platform for creating and running automated workflows. Workflows are step-by-step processes that integrate your apps, data, services, and systems. With Azure Logic Apps, you can quickly develop highly scalable integration solutions for your enterprise and business-to-business (B2B) scenarios.

Logic Apps is a member of **Azure Integration Services**. Logic Apps simplifies the way that you connect legacy, modern, and cutting-edge systems across cloud, on premises, and hybrid environments. The following list describes just a few example tasks, business processes, and workloads that you can automate using the Logic Apps service.

- Schedule and send email notifications using Office 365 when a specific event happens. For example, a new file is uploaded.
- Route and process customer orders across on-premises systems and cloud services.
- Move uploaded files from an SFTP or FTP server to Azure Storage.
- Monitor tweets, analyze the sentiment, and create alerts or tasks for items that need review.

How are Azure Logic Apps and Azure Functions different?

Azure Logic Apps and Azure Functions may seem similar but there are basic differences. Azure Functions is a code-first technology. Azure Logic Apps is a design-first technology.



Here are some other differences.

Comparison area	Durable Functions	Logic Apps
Development	Code-first	Designer-first
Method	Write code and use the durable functions extension	Create orchestrations by using a GUI or editing configuration files
Connectivity	Large selection of built-in binding types , write code for custom bindings	Large collection of connectors, Enterprise Integration Pack for B2B scenarios, build custom connectors
Monitoring	Azure Application Insights	Azure portal, Azure Monitor logs
Execution context	Can be run locally or in the cloud	Runs only in the cloud

Tip: You can mix and match services when you build an orchestration. You can call functions from logic apps and call logic apps from functions. Build each orchestration based on the services' capabilities or your personal preference.

Decision criteria for Logic Apps

When designing for Logic Apps consider integration, performance, conditionals, and connectors.

- **Integration.** The key question to ask when you're considering Logic Apps is "do I need to integrate services?" Logic Apps work well when you need to get multiple applications and systems to work together. That's what they were designed to do. If you're building an app with no external connections, Logic Apps is probably not the best option.
- **Performance.** The next consideration is performance. The Logic Apps execution engine scales your apps automatically. Logic Apps can process large datasets in parallel to let you achieve high throughput. However, fast activation time is not always guaranteed, nor enforcement of real-time constraints on execution time.
- **Conditionals.** Logic Apps provides control constructs like Boolean expressions, switch statements, and loops so your apps can make decisions based on your data. You can build highly complex and deeply nested conditionals into your Logic Apps.
- **Connectors.** The last consideration is whether there are pre-built connectors for all the services you need to access. If so, then you're ready to go. If not, then you'll need to create a custom connector. If the service has an existing REST or SOAP API, you can make the custom connector in a few hours without writing any code. If not, then you'll need to create the API first before making the connector.

Tip: Knowing when not to use Logic Apps is also important. The cases where Logic Apps might not be the best option include real-time requirements, complex business rules, or use of non-standard services.

Summary of design criteria for logic apps



Note: Take a few minutes to learn about how [Cramo is using Logic Apps](#) in their new integration platform.

Introduction



The Tailwind Trader™'s CTO asks, "What is our storage solution for non-relational data?" The CTO's question is a reasonable and an area Azure Architects can help. In this module, we'll explore different storage strategies. Storage strategies will include data types, storage accounts, blob storage, file storage, disk storage, storage security, and data protection.

Learning objectives

In this module, you'll learn how to:

- Design for data storage.
- Design for Azure storage accounts.
- Design for Azure blob storage.
- Design for Azure files.
- Design an Azure disk solution.
- Design for storage security.

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions. The module concepts are covered in:

Design a data storage solution for non-relational data

- Recommend a data storage solution to balance features, performance, and cost
- Design a data solution for protection and durability
- Recommend access control solutions to data storage

Prerequisites

- Conceptual knowledge of storage accounts, blobs, files, disks, and data protection.
- Working experience with creating and securing storage systems.

Introduction



The Tailwind Trader™'s CTO asks, "What is our storage solution for non-relational data?" The CTO's question is a reasonable and an area Azure Architects can help. In this module, we'll explore different storage strategies. Storage strategies will include data types, storage accounts, blob storage, file storage, disk storage, storage security, and data protection.

Learning objectives

In this module, you'll learn how to:

- Design for data storage.
- Design for Azure storage accounts.
- Design for Azure blob storage.
- Design for Azure files.
- Design an Azure disk solution.
- Design for storage security.

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions. The module concepts are covered in:

Design a data storage solution for non-relational data

- Recommend a data storage solution to balance features, performance, and cost
- Design a data solution for protection and durability
- Recommend access control solutions to data storage

Prerequisites

- Conceptual knowledge of storage accounts, blobs, files, disks, and data protection.
- Working experience with creating and securing storage systems.

Design for data storage

To design Azure storage, you first must determine what type of data you have.



- **Structured data** includes relational data and has a shared schema. Structured data is often stored in database tables with rows, columns, and keys. Structured data is often used for application storage like an ecommerce website.
- **Semi-structured** is less organized than structured data and isn't stored in a relational format. Semi-structured data fields don't neatly fit into tables, rows, and columns. Semi-structured data contains tags that clarify how the data is organized. The expression and structure of the data in this style is defined by a serialization language. Examples of semi-structured data include Hypertext Markup Language (HTML) files, JavaScript Object Notation (JSON) files, and Extensible Markup Language (XML) files
- **Unstructured data** is the least organized type of data. The organization of unstructured data is ambiguous. Examples of unstructured data include:
 - Media files, such as photos, videos, and audio files
 - Office files, such as Word documents
 - Text files

Important: This module will only cover unstructured data. These data types are often referred to as non-relational data.

Azure non-relational storage objects

In Azure, non-relational data is contained in several different storage data objects. There are four data storage objects we'll focus on.



Azure Blob storage is an object store used for storing vast amounts unstructured data. Blob stands for Binary Large Object, which includes objects such as images and multimedia files.

Azure Files is a shared storage service. You can access files with Server Message Block (SMB) on Windows or Network File Share (NFS) on Linux.

Azure managed disks are block-level storage volumes that are managed by Azure and used with Azure virtual machines. Managed disks are like a physical disk in an on-premises server but, virtualized.

Azure Queue Storage is a service for storing large numbers of messages. Queues are commonly used to create a backlog of work to process asynchronously.

Tip: Before beginning your study, think about which non-relational data types are of most interest to you or your organization.

Design for Azure storage accounts

Once you've determined your data storage requirements, you need to create storage accounts. An **Azure storage account** group together all the Azure Storage services you need. The storage account provides a unique namespace that's accessible from anywhere (assuming you have the correct permissions) in the world over HTTPS. Data in your storage account is durable and highly available, secure, and massively scalable.

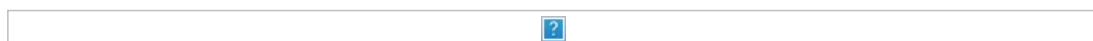
Select the appropriate storage account kind

Azure Storage offers several kinds of storage accounts. Each kind supports different features and has its own pricing model. Consider these differences to determine the kind of account that is best for your applications. The **types of storage accounts** are:

Storage Account	Supported Services	Recommended usage
Standard general-purpose v2	Blob (including Data Lake Storage), Queue, and Table storage, Azure Files	Supports all the storage services: Blob, Azure Files, Queue, Disk (Page Blob), and Table. Premium block blobs are ideal for applications that require high transaction rates. Also ideal for situations that use smaller objects or require consistently low storage latency. This storage is designed to scale with your applications.
Premium block blobs	Blob storage (including Data Lake Storage)	Recommended for enterprise or high-performance scale applications. Use Premium file shares if you need a storage account that supports both SMB and NFS file shares.
Premium file shares	Azure Files	Premium high-performance page blob scenarios. Page blobs are ideal for storing index-based and sparse data structures like OS and data disks for virtual machines and databases.
Premium page blobs	Page blobs only	

Determine the number of storage accounts

A storage account represents a collection of settings like location, replication strategy, and subscription owner. Organizations often have multiple storage accounts so they can implement different sets of requirements. The following illustration shows two storage accounts that differ in one setting. That one difference is enough to require separate storage accounts.



Considerations when deciding how many storage accounts to create

- **Location.** Do you have data that is specific to a country or region? For performance reasons, you might want to locate the data close to your users. You may need one storage account for each location.
- **Compliance.** Does your company have regulatory guidelines for keeping data in a specific location? Does your company have internal requirements for auditing or storing data?
- **Cost.** A storage account by itself has no financial cost; however, the settings you choose for the account do influence the cost of services in the account. Geo-redundant storage costs more than locally redundant storage. Premium performance and the hot access tier increase the cost of blobs. Do you need to keep track of expenses or billing by department or project? Are you working with partners where storage costs need to be separated?
- **Replication.** Does your data storage have different replication strategies? For example, you could partition your data into critical and non-critical categories. You could place your critical data into a storage account with geo-redundant storage. You could put your non-critical data in a different storage account with locally redundant storage.
- **Administrative overhead.** Each storage account requires some time and attention from an

administrator to create and maintain. It also increases complexity for anyone who adds data to your cloud storage. Everyone in this role needs to understand the purpose of each storage account so they add new data to the correct account.

- **Data sensitivity.** Do you have some data that is proprietary and some for public consumption? If so, you could enable virtual networks for the proprietary data and not for the public data. This may require separate storage accounts.
- **Data isolation.** Regulatory, compliance, or local policies may require data to be segregated. Perhaps data from one application should be separated from data in another application?

Tip: Take a few minutes to think about your organization's storage accounts. Are the storage accounts already in place? Would you make any changes? What type of storage accounts will you need, and why?

Design for data redundancy

Azure Storage always stores multiple copies of your data. This redundancy ensures the data is protected from planned and unplanned events. These events can include transient hardware failures, network or power outages, and massive natural disasters. **Storage redundancy** ensures that your storage account meets its availability and durability targets.

When deciding which redundancy option is best for your scenario, consider the tradeoffs between lower costs and higher availability. The factors that help determine which redundancy option you should choose include:

- How your data is replicated in the primary region.

- Whether your data is replicated to a second region. The secondary region is geographically distant to the primary region. This helps to protect against regional disasters.

- Whether your application requires read access to the replicated data in the secondary region if the primary region becomes unavailable for any reason.

Redundancy in the primary region

Azure Storage offers two options for how your data is replicated in the primary region.



Locally redundant storage (LRS) is the lowest-cost redundancy option and offers the least durability compared to other options. LRS protects your data against server rack and drive failures. However, if the data center fails, all replicas of a storage account using LRS may be lost or unrecoverable. LRS is a good choice for the following scenarios:

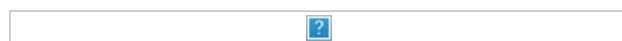
- If your application stores data that can be easily reconstructed if data loss occurs.

- If your application is restricted to replicating data only within a country or region due to data governance requirements.

Zone-redundant storage (ZRS) replicates synchronously across three Azure availability zones in the primary region. With ZRS, your data is still accessible for both read and write operations even if a zone becomes unavailable. ZRS provides excellent performance, low latency, and resiliency for your data if it becomes temporarily unavailable. However, ZRS by itself may not protect your data against a regional disaster where multiple zones are permanently affected.

Redundancy in a secondary region

For applications requiring high durability, you can choose to copy the data in your storage account to a secondary region. When you create a storage account, you select the primary region for the account. The paired secondary region is determined based on the primary region and can't be changed. Azure Storage offers two options for copying your data to a secondary region: Geo-redundant storage (GRS), and Geo-zone-redundant storage (GZRS).



- The primary difference between GRS and GZRS is how data is replicated in the primary region. Within the secondary region, data is always replicated synchronously with LRS.

- If the primary region becomes unavailable, you can choose to fail over to the secondary region. After the failover has completed, the secondary region becomes the primary region, and you can again read and write data.

- Data is replicated to the secondary region asynchronously. A failure that affects the primary region may result in data loss if the primary region cannot be recovered.

- With GRS or GZRS, the data in the secondary region isn't available for read or write access unless there is a failover to the secondary region. For read access to the secondary region, configure your storage account to use read-access geo-redundant storage (RA-GRS) or read-access geo-zone-redundant storage (RA-GZRS).

Tip: If your storage account is configured for read access to the secondary region, then you can design your applications to seamlessly shift to reading data from the secondary region if the primary region becomes unavailable for any reason.

Design for Azure blob storage

When designing **blob storage**, there are two main areas weâ€™ll highlight. The first is determining the blob access tier. Blob storage type affects storage availability, latency, and cost. The second area is deciding if immutable storage is needed.

Availability, Latency, and Cost

Premium blob storage
Hot, cool, and archive access tiers

Immutable Storage

Legal hold policies
Time-based retention policies

Determine the Azure blob access tier

Optimize storage costs by placing your data in the appropriate access tier.

Feature	Premium	Hot tier	Cool tier	Archive tier
Availability	99.9%	99.9%	99%	Offline
Availability (RA-GRS reads)	N/A	99.99%	99.9%	Offline
Usage charges	Higher storage costs, lower access, and transaction cost	Higher storage costs, lower access, and transaction costs	Lower storage costs, higher access, and transaction costs	Lowest storage costs, highest access, and transaction costs
Minimum storage duration	N/A	N/A	30 days	180 days
Latency (time to first byte)	Single-digit milliseconds	milliseconds	milliseconds	hours

- **Premium blob storage.** The **premium blob storage account types** are best suited for I/O intensive workloads that require low and consistent storage latency. Premium blob storage uses solid-state drives (SSDs) for fast and consistent response times. This storage is best for workloads that perform many small transactions. An example would be a mapping app that requires frequent and fast updates.
- **Hot access tier.** By default, new storage accounts are created in the hot access tier. The hot tier is optimized for frequent reads and writes of objects in the storage account. The hot tier has higher storage costs than cool and archive tiers, but the lowest access costs. A good usage case is data that is actively being processed.
- **Cool access tier.** The cool access tier is optimized for storing large amounts of data that is infrequently accessed. This tier is intended for data that will remain in the cool tier for at least 30 days. The cool access tier has lower storage costs and higher access costs compared to hot storage. A usage case for the cool access tier is short-term backup and disaster recovery datasets and older media content. This content wouldnâ€™t be viewed frequently but must be available immediately.
- **Archive access tier.** The **archive access tier** is optimized for data that can tolerate several hours of retrieval latency. Data must remain in the archive tier for at least 180 days or be subject to an early deletion charge. The archive tier is the most cost-effective option for storing data. But, accessing that data is more expensive than accessing data in the other tiers. Data for the archive tier includes secondary backups, original raw data, and legally required compliance information.

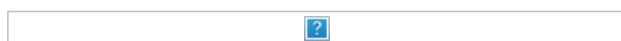
Tip: Take a few minutes to determine the data sets and access tiers your organization will need. How did you decide on the access tier? How will you manage the data sets?

Determine requirements for Azure blob immutable storage

Immutable storage for Azure Blob Storage enables users to store business-critical data in a WORM (Write Once, Read Many) state. While in a WORM state, data canâ€™t be modified or deleted for a user-specified interval. By configuring immutability policies for blob data, you can protect your data from overwrites and deletes. Policies are applied at the container level and audit logs are available.



The next diagram shows how time-based retention policies and legal holds prevent write and delete operations.



Immutable storage for Azure Blob storage supports two types of immutability policies.

- **Time-based retention policies:** With a **time-based retention policy**, users can set policies to store data for a specified interval. When a time-based retention policy is in place, objects can be created and read, but not modified or deleted. After the retention period has expired, objects can be deleted but not overwritten.

- **Legal hold policies:** A **legal hold** stores immutable data until the legal hold is explicitly cleared. When a legal hold is set, objects can be created and read, but not modified or deleted.

Tip: Take a few minutes to determine if your organization will need immutable blob storage policies. Which data sets and policies would be most helpful?

Design for Azure files

Moving a company's shared files into the cloud-based [Azure Files](#) requires an analysis of the options and a plan for the implementation. There's an important decision to make. How are you going to access and update the files? You could choose to replace your existing Server Message Block (SMB) file shares with their equivalent in Azure Files. The other option is to set up an instance of Azure File Sync. If you choose to use Azure File Sync, there's more flexibility on how files are secured and accessed.

What are Azure Files?



You can think of Azure Files as a standard file share, hosted on Azure, that you can access with the industry standard SMB/CIFS protocol. You can mount or connect to an Azure file share at the same time on all the main operating systems.

Azure Files can be used to add to or replace a company's existing on-premises NAS devices or file servers. Some reasons why your organization will want to use Azure Files are:

- Developers can store apps and configuration files in a file share and connect new VMs to the shared files. This action reduces the time to get new machines into production.
- With file shares on Azure, a company doesn't need to buy and deploy expensive redundant hardware and manage software updates. The shares are cross-platform, and you can connect to them from Windows, Linux, or macOS.
- All the resilience of the Azure platform is inherited by your file share, which makes files globally redundant. You also gain options to use the integrated snapshots feature and set up automatic backups by using Recovery Services vaults.
- All the data is encrypted in transit by using HTTPS and is stored encrypted when at rest.

Choose your data access method

Azure file shares can be used in two ways: by directly mounting these serverless Azure file shares (SMB) or by caching Azure file shares on-premises using Azure File Sync.

- **Direct mount of an Azure file share:** Since Azure Files provides SMB access, you can mount Azure file shares on-premises or in the cloud. Mounting uses the standard SMB client available in Windows, macOS, and Linux. Because Azure file shares are serverless, deploying for production scenarios doesn't require managing a file server or NAS device. Direct mounting means you don't have to apply software patches or swap out physical disks.
- **Cache Azure file share on-premises with Azure File Sync:** [Azure File Sync](#) lets you centralize your organization's file shares. Azure Files provides the flexibility, performance, and compatibility of an on-premises file server. Azure File Sync transforms an on-premises (or cloud) Windows Server into a quick cache of your Azure file share.

Choose your performance level

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

Performance level	Latency	IOPS	Bandwidth
Standard	Double-digit ms	10,000 IOPS	300-MBps
Premium	Single-digit ms	100,000 IOPS	5-GBps

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

Determine your storage tier

Azure Files offers four different tiers of storage, premium, transaction optimized, hot, and cool. These tiers allow you to tailor your shares to the performance and price requirements of your scenario.

Storage tier	Usage
Premium	File shares are backed by solid-state drives (SSDs) and provide consistent high performance and low latency. Used for the most intensive IO workloads. Suitable workloads include databases, web site hosting, and development environments. Can be used with both Server Message Block (SMB) and Network File System (NFS) protocols.

Transaction optimized	Used for transaction heavy workloads that don't need the latency offered by premium file shares. File shares are offered on the standard storage hardware backed by hard disk drives (HDDs).
Hot	Storage optimized for general purpose file sharing scenarios such as team shares. Offered on standard storage hardware backed by HDDs.
Cool	Cost-efficient storage optimized for online archive storage scenarios. Offered on storage hardware backed by HDDs.

When to use Azure files instead of Azure blobs or Azure NetApp Files

Let's take a minute to review when you should select Azure blob storage or [Azure NetApp Files](#) instead of Azure file storage.

NetApp Files is a fully managed, highly available, enterprise-grade NAS service. NetApp Files can handle the most demanding, high-performance, low-latency workloads. It enables the migration of workloads, which are deemed "non-migratable" without.

[Your decision on which technology](#) depends on the use case, protocol, and performance required.

Category	Azure Blob Storage	Azure Files	Azure NetApp Files
Use cases	<p>Blob Storage is best suited for large scale read-heavy sequential access workloads where data is ingested once and modified later.</p> <p>Blob Storage offers the lowest total cost of ownership, if there is little or no maintenance.</p> <p>Some example scenarios are: Large scale analytical data, throughput sensitive high-performance computing, backup and archive, autonomous driving, media rendering, or genomic sequencing.</p>	<p>Azure Files is a highly available service best suited for random access workloads.</p> <p>For NFS shares, Azure Files provides full POSIX file system support and can easily be used from container platforms like Azure Container Instance (ACI) and Azure Kubernetes Service (AKS) with the built-in CSI driver, in addition to VM-based platforms.</p> <p>Some example scenarios are: Shared files, databases, home directories, traditional applications, ERP, CMS, NAS migrations that don't require advanced management, and custom applications requiring scale-out file storage.</p>	<p>Fully managed file service in the cloud, powered by NetApp, with advanced management capabilities.</p> <p>NetApp Files is suited for workloads that require random access and provides broad protocol support and data protection capabilities.</p> <p>Some example scenarios are: On-premises enterprise NAS migration that requires rich management capabilities, latency sensitive workloads like SAP HANA, latency-sensitive or IOPS intensive high performance compute, or workloads that require simultaneous multi-protocol access.</p>
Available protocols	NFS 3.0 REST Data Lake Storage Gen2	NFS 4.1 (preview) (No interoperability between either protocol)	NFS 3.0 and 4.1 SMB
Performance (Per volume)	Up to 20,000 IOPS, up to 100 GiB/s throughput.	Up to 100,000 IOPS, up to 80 GiB/s throughput.	Up to 460,000 IOPS, up to 36 GiB/s throughput.

Tip: Take a few minutes to think through your company file share strategy. Will you need files shares?

Will you need file sync?

Design for Azure disk solutions

For this module, weâ€™ll concern ourselves with data disks. Data disks are used by virtual machines to store data. For example, database files, website static content, or custom application code would be stored on data disks. The number of data disks you can add depends on the virtual machine size. Each data disk has a maximum capacity of 32,767 GB.

Tip: Microsoft recommends always using managed disks. With **managed disks**, you specify the disk size, the disk type, and provision the disk. Once you provision the disk, Azure handles the rest.

Determine the type of data disk

Azure offers several types of data disks. When selecting a disk type, consider your scenario, throughput, and IOPS. The following table provides a **comparison of the four disk types**.

Detail	Ultra-disk	Premium SSD	Standard SSD	Standard HDD
Disk type	SSD	SSD	SSD	HDD
Scenario	IO-intensive workloads such as SAP HANA, top tier databases (for example, SQL, Oracle), and other transaction-heavy workloads.	Production and performance sensitive workloads	Web servers, lightly used enterprise applications and dev/test	Backup, non-critical, infrequent access
Max throughput	2,000 MB/s	900 MB/s	750 MB/s	500 MB/s
Max IOPS	160,000	20,000	6,000	2,000

- **Ultra-disk storage.** Azure Ultra Disk storage provides the best performance. Choose this option when you need the fastest storage performance in addition to high throughput, high input/output operations per second (IOPS), and low latency. Ultra-disk storage may not be available in all regions.
- **Premium SSD storage.** Azure Premium SSD-managed disks provide high throughput and IOPS with low latency. These disks offer a slightly less performance compared to Ultra Disk Storage. Premium SSD storage is available in all regions.
- **Standard SSD.** Azure Standard SSD-managed disks are a cost-effective storage option for VMs that need consistent performance at lower speeds. Standard SSD disks aren't as fast as Premium SSD disks or Ultra Disk Storage. You can attach Standard SSD disks to any VM.
- **Standard HDD.** In Azure Standard HDD-managed disks, data is stored on conventional magnetic disk drives that have moving spindles. Disks are slower and the variation in speeds is higher compared to solid-state drives (SSDs). Like Standard SSD disks, you can use Standard HDD disks for any VM.

Tip: Read more about how to [Select a disk type for Azure IaaS VMs - managed disks - Azure Virtual Machines | Microsoft Docs](#).

Improve performance with disk caching

Azure virtual machine **disk caching** is about optimizing read and write access to the virtual hard disk (VHD) files. These VHDs are attached to Azure virtual machines. Here are the recommended disk cache settings for data disks.

Disk caching setting	Recommendation
None	Use for write-only and write-heavy disks.
Read-only	Provides low read latency and high read IOPS and throughput.
Read-write	Use only if your application properly handles writing cached data to persistent disks.

Warning: Disk Caching isn't supported for disks 4 TiB and larger. When multiple disks are attached to your VM, each disk that is smaller than 4 TiB will support caching. Changing the cache setting of an Azure disk detaches and reattaches the target disk. When it's the operating system disk, the VM is restarted.

Secure your data disks with encryption

There are several encryption types available for your managed disks. Encryption types includes Azure Disk Encryption (ADE), Server-Side Encryption (SSE) and encryption at host.

- **Azure Disk Encryption** ADE encrypts the virtual machine's virtual hard disks (VHDs). If VHD is protected with ADE, the disk image will only be accessible by the virtual machine that owns the disk.
- **Server-Side Encryption** (also referred to as encryption-at-rest or Azure Storage encryption) is performed on the physical disks in the data center. If someone directly accesses the physical disk, the data will be encrypted. When the data is accessed from the disk, it's decrypted and loaded into memory.
- **Encryption at host** ensures that data stored on the VM host is encrypted at rest and flows encrypted to the Storage service. Disks with encryption at host enabled aren't encrypted with SSE. Instead, the server hosting your VM provides the encryption for your data, and that encrypted data flows into Azure Storage.

Note: To fully protect your data disks, combine encryption services.

Design for storage security

Azure Storage provides a layered security model. This model enables you to secure and control the level of access to your storage accounts. In this unit, we'll cover some best practices for storage security.



Grant limited access to Azure storage resources

The [Azure security baseline for Azure Storage baseline](#) provides a comprehensive list of ways to secure your Azure storage.

Use Shared Access Signatures

One of the most common ways is to use a [Shared Access Signature](#). A SAS provides secure delegated access to resources in your storage account. With a SAS, you have granular control over how a client can access your data. For example:

- What resources the client may access.
- What permissions they have to those resources.
- How long the SAS is valid.

Enable firewall policies and rules

- [Configure firewall rules](#) to limit access to your storage account. Requests can be limited to specific IP addresses or ranges, or to a list of subnets in an Azure virtual network. The Azure storage firewall provides access control for the public endpoint of your storage account. You can also use the firewall to block all access through the public endpoint when using private endpoints. Your storage firewall configuration also enables select trusted Azure platform services to access the storage account securely.

Restrict network access using service endpoints

Use [virtual network service endpoints](#) to provide direct connection to your Azure storage.

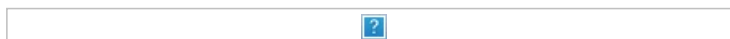


Service endpoints provide several advantages.

- Allows you to secure Azure storage accounts to your virtual networks.
- Provides optimal routing by always keeping traffic destined to Azure Storage on the Azure backbone network.
- Service Endpoints enable private IP addresses in the VNet to reach the service endpoint. A public IP address isn't needed.
- Enables on-premises networks to access resources using NAT IP addresses.

Determine when to use private endpoints

A [private endpoint](#) is a special network interface for an Azure service in your virtual network. When you create a private endpoint for your storage account, it provides secure connectivity between clients on your VNet and your storage.



Enable secure transfer

You can configure your storage account to accept requests from secure connections. This is done by setting the [Secure transfer required](#) property for the storage account. When you require secure transfer, any requests originating from non-secure connections are rejected. Microsoft recommends that you always require secure transfer for all your storage accounts.

Use customer-managed encryption keys

Data in your storage account is automatically encrypted. Azure Storage encryption offers two options for managing encryption keys at the level of the storage account:

- **Microsoft-managed keys.** By default, Microsoft manages the keys used to encrypt your storage account.
- **Customer-managed keys.** You can optionally choose to manage encryption keys for your storage account. [Customer-managed keys](#) must be stored in Azure Key Vault. When you bring-your-own-key (BYOK), you gain full control over who can use the encryption keys and who can access the encrypted data.

Introduction

Many organizations have an aging or under-engineered data platform strategy. There's been a significant trend of moving existing systems to the cloud, building new applications quickly with the cloud, and offloading some on-premises costs. You need a plan for how to move data workloads to the cloud. And you need to understand how to set up your organization for success.

Meet Tailwind Traders



Tailwind Traders is a fictitious home improvement retailer. It operates retail hardware stores across the globe and online. It currently manages an on-premises datacenter that hosts the company's retail website. The datacenter also stores all of the data and streaming video for its applications. The on-premises SQL server also provides storage for customer data, order history, and product catalogs, apart from data storage for your internal-only training portal website.

Tailwind Traders wants to effectively manage database needs by migrating it to the cloud. You are tasked with finding a cost efficient database solution that provides low latency and high availability.

The Tailwind Trader's CTO asks, "What is our storage solution for relational data?" The CTO's question is a reasonable and an area Azure Architects can help. In this module, we'll explore different storage solutions that solve different types of problems. Storage solutions will include Azure SQL Database, Azure SQL Managed Instance, SQL Server in an Azure virtual machine, SQL edge, Azure table storage and Cosmos DB. You will also learn how to design your solution with data encryption.

Learning objectives

In this module, you'll be able to:

- Design for Azure SQL Database
- Design for Azure SQL Managed Instance
- Design for SQL Server on Azure VM
- Recommend a solution for Database Scalability
- Design encryption for data at rest, data in transmission, and data in use
- Design for Azure SQL Edge
- Design for Azure tables.
- Design for Azure Cosmos DB.

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions. The module concepts are covered in:

Design a Data Storage Solution for Relational Data

- Recommend database service tier sizing
- Recommend a solution for database scalability
- Recommend a solution for encrypting data at rest, data in transmission, and data in use

Recommend a Data Storage Solution

- Recommend a solution for storing relational data

Prerequisites

- Working experience with database solutions
- Conceptual knowledge of SQL Server

Design for Azure SQL databases

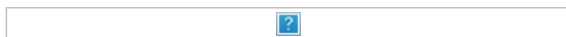
The CTO has asked you to design databases for Azure to meet all of the needs of existing structured data on premises and suggestion solutions for any new relational data workloads that Tailwinds might need. Structured data includes relational data and has a shared schema. Structured data is often stored in database tables with rows, columns, and keys. Structured data is often used for application storage like an ecommerce website.

Within the umbrella of the Azure SQL platform, there are many deployment options and choices that you need to make to meet your needs. These options give you the flexibility to get and pay for exactly what you need. Here, we'll cover some of the considerations you need to make when you choose various Azure SQL deployment options. We'll also cover some of the technical specifications for each of these options. The deployment options discussed here include SQL Server on virtual machines, Azure SQL Managed Instance, Azure SQL Database, Azure SQL Managed Instance pools, and Azure SQL Database elastic database pools.

Analyze Azure SQL deployment options

As displayed in the following graphic, Azure offers SQL Server in the following ways:

- SQL Server on Azure VMs
- Managed instances:
 - Single instances
 - Instance pool
- Databases:
 - Single database
 - Elastic pool



Azure SQL Database

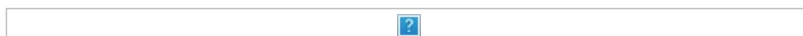
Azure SQL Database is a PaaS deployment option of Azure SQL that abstracts both the OS and the SQL Server instance. It is a highly scalable, intelligent, relational database service built for the cloud with the industry's highest availability SLA.

SQL Database is also the only deployment option that supports scenarios that require:

- Very large databases (currently up to 100TB)
- Autoscaling for unpredictable workloads (serverless)

In the following image, AccuWeather provides an example of using SQL Database.

AccuWeather has been analyzing and predicting the weather for more than 55 years. The company wanted to access Azure for its big data, machine learning, and AI capabilities. AccuWeather wants to focus on building new models and applications, not on managing databases. The company chose SQL Database to use with other services, like Azure Data Factory and Azure Machine Learning to quickly and easily deploy new internal applications to make sales and customer predictions.



What are SQL elastic pools?

When you create your Azure SQL database, you can create a SQL elastic pool. They enable you to buy a set of compute and storage resources that are shared among all the databases in the pool. Each database can use the resources they need, within the limits you set, depending on current load.

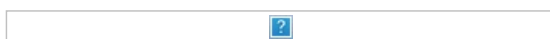
Read more about [SQL elastic pools](#).

Analyze Azure purchasing models

The next decision to be made after deciding on deployment option is the purchasing model.

Azure SQL Database has two purchasing models as shown in the following graphic:

- DTU
- vCore



- vCores stands for Virtual cores and lets you choose the number of vCores which gives you greater control over the compute and storage resources that you create and pay for.
- DTU (Database Transaction Unit) is a combined measure of compute, storage, and IO resources. DTU model is a simple, preconfigured purchase option. This is unavailable on SQL managed Instance.
- Serverless model is a compute tier for single databases in Azure SQL Database. It automatically

scales compute, based on workload demand and bills only for the amount of compute used.

The vCore-based model is recommended because it allows you to independently select compute and storage resources. The DTU-based model is a bundled measure of compute, storage, and I/O resources.

The vCore model also allows you to use Azure Hybrid Benefit for SQL Server and/or reserved capacity (pay in advance) to save money. Neither of these options is available in the DTU model.

In the table below, you can determine which purchasing model is recommended based on the requirements.

Recommendation	Requirement
DTU Model	When you need a bundled measure of compute, storage and I/O resources
vCore Model	When you need the flexibility of selecting compute and storage resources independently

Analyze Azure database service tiers

Based on performance, availability and storage needs, Azure offers three database service tiers, within the vCore module discussed in the following table.

Azure SQL Database and Azure SQL Managed Instance ensure 99.99% availability, even in the cases of infrastructure failures.

Service tiers available to Azure SQL Database and SQL Managed Instance are - General Purpose and Business Critical. Azure SQL Database also has Hyperscale service tier that is unavailable for Azure SQL Managed Instance.



In the preceding image, with **General Purposeservice tier**, the primary replica uses locally attached SSD for the tempdb. The data and log files are stored in Azure Premium Storage. The backup files are then stored in Azure Standard Storage. When failover occurs, the Azure service fabric will identify a node with spare capacity and spin up a new SQL Server instance. The database files will then be attached, recovery will be run, and gateways will be updated to point applications to the new node.

Business critical architecture is meant for mission-critical applications that need low latency and minimal downtime. Business Critical is like deploying an Always On availability group (AG) behind the scenes, the way the data and log files are stored differs from General purpose, in that they are stored on the direct attached SSD.

In the Business Critical scenario, the data and log files are all running on direct-attached SSD, which significantly reduces network latency. In this architecture group, there are three secondary replicas. If any type of failure occurs, failing over to a secondary replica is fast because the replica already exists and has the data attached to it.

Azure SQL Database Hyperscale is a fully managed service that adapts to changing requirements by rapidly scaling storage up to 100 TB. Flexible, cloud-native architecture allows storage to grow as needed and enables you to back up data almost instantaneously and restore your database in minutes—regardless of the size of the data operation.

Hyperscale provides fast database restores, rapid scale outs, and scale ups. A Hyperscale database grows as needed - and you are billed only for the capacity you use.



The image above illustrates the different types of nodes in Hyperscale architecture. The compute node is where the relational engine lives and where language, query, and transaction processing occur. Compute nodes have SSD-based caches (labeled RBPEX - Resilient Buffer Pool Extension). There are one or more secondary compute nodes that act as hot standby nodes for failover purposes, as well as act as read-only compute nodes. Page servers are systems representing a scaled-out storage engine. The job of a page server is to serve database pages out to the compute nodes on demand. Page servers also maintain covering SSD-based caches to enhance performance. The log service accepts log records from the primary compute replica, persists them in a durable cache, and forwards the log records to the rest of compute replicas (so they can update their caches) as well as the relevant page server(s), so that the data can be updated there. In this way, all data changes from the primary compute replica are propagated through the log service to all the secondary compute replicas and page servers.

The following table compares the different scenarios when the three service tiers can be selected based on your specific needs.

Recommendation	Requirement
General Purpose	When you need balanced compute and storage options for business workloads
Business Critical	When you need low latency requirements and highest resilience to failures for business applications
Hyperscale	When you need highly scalable storage and have read-scale requirements for business workloads

Design for Azure SQL Managed Instances

Azure SQL Managed Instance is a PaaS deployment option of Azure SQL. It provides an instance of SQL Server, but removes much of the overhead of managing a virtual machine. Most of the features available in SQL Server are available in SQL Managed Instance.

SQL Managed Instance is ideal for customers who want to use instance-scoped features and want to move to Azure without re-architecting their applications.

SQL Managed Instance instance-scoped features include:

- SQL Server Agent
- Service Broker
- Common language runtime (CLR)
- Database Mail
- Linked servers
- Distributed transactions (preview)
- Machine Learning Services

Let us explore another industry scenario. Komatsu is a manufacturing company that produces and sells heavy equipment for construction. The company had multiple mainframe applications for different types of data.

Komatsu wanted to consolidate these applications to get an overall view. Additionally, Komatsu wanted a way to reduce overhead.

Because the company uses a large surface area of SQL Server features, the IT department chose to move to Azure SQL Managed Instance. They were able to move about 1.5 terabytes of data smoothly and get the following benefits:

- Automatic patching and version updates
- Automated backups
- High availability
- Reduced management overhead

The following image shows Komatsu's challenge and their Azure solution consideration.



Scalability for Azure SQL Managed Instance

- SQL Managed Instance uses vCores mode and enables you to define maximum CPU cores and maximum of storage allocated to your instance. All databases within the managed instance will share the resources allocated to the instance.

Review this [comparison of SQL Database and SQL Managed Instance](#)

Design for SQL Server on Azure virtual machines

SQL Server on Azure Virtual Machines is a version of SQL Server that runs in an Azure VM. This means:

- All your SQL Server skills should directly transfer, though Azure can help automate backups and security patches.
- You have access to the full capabilities of SQL Server
- You're responsible for updating and patching the OS and SQL Server

Consider the following image. Allscripts is a leading manufacturer of healthcare software, serving physician practices, hospitals, health plans, and the pharmaceutical industry.

To transform its applications frequently and host them securely and reliably, Allscripts wanted to move to Azure quickly.

In just three weeks, the company used Azure Site Recovery to migrate dozens of acquired applications running on approximately 1,000 VMs to Azure.



If you have existing on-premises Windows Server and SQL Server licenses, you can leverage [Azure Hybrid Benefit](#).

Now that we have reviewed the different deployment options, let's compare the different requirements and the solutions recommended.

Recommendation	Requirement
SQL Virtual machines	When considering migrations and applications requiring OS level access
Managed Instances	When considering Lift and Shift migrations to the cloud
Databases	When considering modern cloud applications solution

Design a solution for database scalability

Tailwind Traders, a fictitious home improvement retailer, looking to migrate to cloud had to choose a low latency and high availability database solution for storing relational data. Based on the organization's needs, they decided to implement Azure SQL Database, which is a fully managed service with an availability SLA of 99.995% in a Business critical service tier using Availability Zones.

The next decision to make for Tailwind Traders is how to handle scalability. Specifically, the scalability of a relational database. You have to help them design a solution that is dynamically scalable to handle the incoming workload. Your solution should be able to handle an expanding number of requests over time without a negative effect on availability or performance. How will you help them in this situation?

Azure SQL Database enables you to easily change resources allocated to your databases with minimal downtime, including:

- CPU power
- Memory
- IO throughput
- Storage

You can easily scale an Azure SQL Database using the Azure portal without the worry of new hardware purchase or changing the existing infrastructure.

Dynamic Scalability for Azure SQL Database enables you to:

- Use either DTU or vCore models to define maximum amount of resources that will be assigned to each database with a single database implementation
- Use Elastic pools that allow you to purchase resources for the group and set minimum and maximum resource limits for the databases within the pool.

Types of scaling in Azure SQL Database

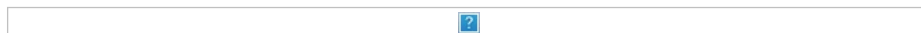
There are two types of scaling you can implement in Azure SQL Database. These are:

- **Vertical** - Vertical scaling refers to increasing or decreasing the compute size of an individual database, also known as "scaling up."
- **Horizontal** - Horizontal scaling refers to adding or removing databases in order to adjust capacity or overall performance, also called "scaling out". Horizontal scaling is managed using the [Elastic Database client library](#).

Design Vertical scaling solution

Imagine a scenario where a small business experiences rapid growth globally and needs to maintain and scale separate Azure SQL databases for each location. However, the rates of growth and database load vary significantly, so resource requirements are unpredictable. How would you manage scaling to meet the demands of this organization?

In this case, it is ideal to choose SQL Elastic pools, to scale, manage performance, and manage costs for a set of Azure SQL databases.



The above image shows SQL elastic pools and the scaling capability for the different service tiers. The databases within the pool share the allocated resources. In situations where there is a low average utilization, but infrequent, high utilization spikes you can allocate enough capacity in the pool to manage the spikes for the group. To properly configure SQL elastic pools to reduce server costs, the right purchasing model and the service tier (DTU-based model in basic, standard & premium or VCore-based model in general purpose or business critical) must be selected.

Read more about [SQL elastic pools](#).

Design Horizontal Scaling Solution

There are two types of horizontal scaling. These are:

- Read Scale-Out
- Sharding

Read Scale-Out

Imagine a scenario where an application's database is accessed for OLTP for updating the database as well as an Analytics app for read-only workload to render visualizations. What can you do to offload some of the compute capacity so that performance of the application isn't affected?

An easy choice here is to use the pre-provisioned read scale-out feature for certain service tiers. The read scale-out feature allows you to offload read-only workloads using the compute capacity of one of the read-only replicas, instead of running them on the read-write replica. This way, some read-only workloads can be isolated from the read-write workloads and will not affect their performance.

The following table shows Read Scale-out provisioning for Azure SQL DB and Azure SQL Managed Instance:

Azure SQL DB Managed Instance

For the basic, standard and general purpose tier, read scale-out feature is unavailable

For the Business Critical tier, read scale-out is auto-provisioned

Hyperscale tier is unavailable in Azure

Azure SQL DB

For the basic, standard and general purpose tier, read scale-out feature is unavailable

For the Premium and Business Critical tier, read scale-out is auto-provisioned

Read scale-put feature is available in

In the image below, in Business Critical scenario, the data and log files are all running on direct-attached SSD, which significantly reduces network latency. In this architecture group, there are three secondary replicas. If any type of failure occurs, failing over to a secondary replica is fast because the replica already exists and has the data attached to it.



The premium/business critical tier of Azure SQL DB or Azure Managed Instance has an Always On Availability Group. This group is for disaster recovery and high-availability of the application. There is a primary read-write replica and several secondary read-only replicas. The secondary replicas are provisioned with the same compute size as the primary replica. You set the connection string option to decide whether the connection is routed to the write replica or to a read-only replica.

You can disable and re-enable read scale-out on single databases and elastic pool databases in the Premium or Business Critical service tiers using the following methods:

- Azure portal
- Azure PowerShell
- REST API

Data changes made on the primary replica propagate to read-only replicas asynchronously. Within a session connected to a read-only replica, reads are always transactionally consistent. However, because data propagation latency is variable, different replicas can return data at slightly different points in time relative to the primary and each other.

Sharding

Imagine a scenario where you need to solve a database problem for an application accessing the database that has a huge amount of transaction throughput that exceeds the database capability. How will you provision the database for performance and availability?

A possible solution for the above scenario is Horizontal Scaling or horizontal partitioning by Sharding. This is a technique to distribute large amounts of identically structured data across a number of independent databases.

Reasons for Sharding include:

- If the total amount of data is too large to fit constraints of a single database
- If the transaction throughput of the overall workload exceeds capacities of an individual database
- When different customers or tenants' data needs physical isolation from each other
- Within an organization, there is a geographical separation of data for compliance reasons

Adopt suitable database scaling strategy

The following table identifies key points to remember before choosing Vertical/Horizontal scaling.

Requirement

Do you have to manage and scale multiple Azure SQL databases that have varying and unpredictable resource requirements?

Do you have different sections of the database residing in different parts of the world for compliance concerns?

Are there dependencies such as commercial BI or data integration tools where multiple databases contribute rows into a single overall result for use in Excel, Power BI, or Tableau?

Description

SQL elastic pools. Vertical scale up is a good solution for this scenario. Elastic pools solve this problem by ensuring that databases get the performance resources they need when they need it. They provide a simple resource allocation mechanism within a predictable budget. There is no per-database charge for elastic pools. You are billed for each hour a pool exists at the highest eDTU or vCores, regardless of usage or whether the pool was active for less than an hour.

Horizontal scaling by Sharding works best. Sharding enables you to split your data into several databases and scale them independently. The shard map manager is a special database that maintains global mapping information about all shards (databases) in a shard set. The metadata allows an application to connect to the correct database, based upon the value of the sharding key.

Use **Elastic database tools** and elastic query feature within it to access data spread across multiple databases. Elastic query is available on standard tier, querying can be done in T-SQL that spans multiple databases in Azure SQL Database. Cross-database queries can be executed to access remote tables, and to connect Microsoft and third-party tools (Excel, Power BI, Tableau, etc.) to query across data tiers. Using this feature, you can scale out queries to large data tiers and visualize the results in business intelligence (BI) reports.

Design a solution for database availability

To understand the availability options and capabilities in Azure SQL, you need to understand service tiers. The service tier you select will determine the underlying architecture of the database or managed instance that you deploy.

There are two purchasing models to consider: DTU and vCore. This unit will focus on the vCore service tiers and their architectures for high availability. You can equate the DTU model's Basic and Standard tiers to General Purpose, and its Premium tiers to Business Critical.

General Purpose

Databases and managed instances in the General Purpose service tier have the same availability architecture. Using the following figure as a guide, first consider the application and control ring. The application connects to the server name, which then connects to a gateway (GW) that points the application to the server to connect to, running on a VM. With General Purpose, the primary replica uses locally attached SSD for the tempdb. The data and log files are stored in Azure Premium Storage, which is locally redundant storage (multiple copies in one region). The backup files are then stored in Azure Standard Storage, which is RA-GRS by default. In other words, it's globally redundant storage (with copies in multiple regions).

All of Azure SQL is built on Azure Service Fabric, which serves as the Azure backbone. If Azure Service Fabric determines that a failover needs to occur, the failover will be similar to that of a failover cluster instance (FCI). The service fabric will identify a node with spare capacity and spin up a new SQL Server instance. The database files will then be attached, recovery will be run, and gateways will be updated to point applications to the new node. No virtual network or listener or updates are required. This capability is built in.



Business Critical

The next service tier to consider is Business Critical, which can generally achieve the highest performance and availability of all Azure SQL service tiers (General Purpose, Hyperscale, Business Critical). Business Critical is meant for mission-critical applications that need low latency and minimal downtime.



Using Business Critical is like deploying an Always On availability group (AG) behind the scenes. Unlike in the General Purpose tier, in Business Critical, the data and log files are all running on direct-attached SSD, which significantly reduces network latency. (General Purpose uses remote storage.) In this AG, there are three secondary replicas. One of them can be used as a read-only endpoint (at no additional charge). A transaction can complete a commit when at least one of the secondary replicas has hardened the change for its transaction log.

Read scale-out with one of the secondary replicas supports session-level consistency. So if the read-only session reconnects after a connection error caused by replica unavailability, it might be redirected to a replica that's not 100% up to date with the read-write replica. Likewise, if an application writes data by using a read-write session and immediately reads it by using a read-only session, the latest updates might not immediately be visible on the replica. The latency is caused by an asynchronous transaction log redo operation.

If any type of failure occurs and the service fabric determines a failover needs to occur, failing over to a secondary replica is fast because the replica already exists and has the data attached to it. In a failover, you don't need a listener. The gateway will redirect your connection to the primary even after a failover. This switch happens quickly, and then the service fabric takes care of spinning up another secondary replica.

Hyperscale

The Hyperscale service tier is currently available for Azure SQL Database, and not Azure SQL Managed Instance. This service tier has a unique architecture because it uses a tiered layer of caches and page servers to expand the ability to quickly access database pages without having to access the data file directly.



Because this architecture uses paired page servers, you can scale horizontally to put all the data in caching layers. This new architecture also allows Hyperscale to support databases as large as 100 TB. Because it uses snapshots, nearly instantaneous database backups can occur regardless of size. Database restores take minutes rather than hours or days. You can also scale up or down in constant time to accommodate your workloads.

It's interesting to note how the log service was pulled out in this architecture. The log service is used to feed the replicas and the page servers. Transactions can commit when the log service hardens to the landing zone. So the consumption of the changes by a secondary compute replica isn't required for a

commit. Unlike in other service tiers, you can determine whether you want secondary replicas. You can configure zero to four secondary replicas, which can all be used for read-scale.

As in the other service tiers, an automatic failover will happen if service fabric determines it needs to. But the recovery time will depend on the existence of secondary replicas. For example, if you don't have replicas and a failover occurs, the scenario will be similar to that of the General Purpose service tier: the service fabric first needs to find spare capacity. If you have one or more replicas, recovery is faster and more closely aligns to that of the Business Critical service tier.

Business Critical maintains the highest performance and availability for workloads with small log writes that need low latency. But the Hyperscale service tier allows you to get a higher log throughput in terms of MB/second, provides for the largest database sizes, and provides up to four secondary replicas for higher levels of read scale. So you'll need to consider your workload when you choose between the two.

Geo-replication and auto-failover groups

After you choose a service tier (and consider Availability Zones as applicable), you can consider some other options for getting read-scale or the ability to fail over to another region: geo-replication and auto-failover groups. In SQL Server on-premises, configuring either of these options is something that would take a lot of planning, coordination, and time.

The cloud, and Azure SQL specifically, have made this process easier. For both geo-replication and auto-failover groups, you can get configured with a few clicks in the Azure portal or a few commands in the PowerShell/Azure CLI.

Design security for data

You've been hired as a Senior Database Administrator help ensure the security of the Azure SQL database environment. Your company is an online sporting goods retailer that stores customer data like phone numbers, addresses, and credit cards in Azure SQL database hosted on the cloud. What solution would you implement to prevent unauthorized data access by someone that is working to support this cloud hosted database?

Classifying stored data by sensitivity and business impact helps organizations determine the risks associated with the data. In this unit, we will learn the different data states and encryption modes in detail.

The three basic tenets of good information protection are:

- Data discovery
- Classification
- Protection

Large organizations, governments, and military entities have been using data classification to manage their data's integrity. The result of data classification is metadata that enables us to label the data as:

- Public
- Confidential
- Restricted

After the data is classified, you can implement data protection measures for highly-classified data. In this section, we explore data encryption for structured data.

Data exists in one of three basic states - data-at-rest, data-in-motion, data-in-process. This table shows data states and possible encryption methods.

DATA STATE	ENCRYPTION METHOD
Data-at-rest	TDE, Always Encrypted
Data-in-motion	SSL/TSL, Always Encrypted
Data-in-process	Dynamic data masking

Defense in depth is a strategy that employs a layered approach to slow the advance of an attack aimed at acquiring unauthorized access to information.

Protect data-at-rest

Data encryption at rest is a mandatory step toward data privacy, compliance, and data sovereignty. Encryption helps mitigate risks related to unauthorized data access. Data-at-rest needs to be protected from unauthorized or offline access to raw files or backups to an unsecured server or making a copy of all the database and transaction log files to another unsecured server.

Transparent data encryption

TDE protects Azure SQL Database, Azure SQL Managed Instance, and Azure Synapse Analytics against the threat of malicious offline activity by encrypting data at rest. It performs real-time encryption and decryption of the database, associated backups, and transaction log files at rest without requiring changes to the application. TDE is enabled by default to all newly deployed Azure SQL Databases. With Azure SQL Managed Instance, databases created after February 2019 have TDE enabled.

- TDE performs encryption and decryption of the data at the page level.
- The data is encrypted as the data is written to the data page on disk and decrypted when the data page is read into memory.
- The end result is that all data pages on disk are encrypted.
- Database backups will also be encrypted because a backup operation just copies the data pages from the database file to the backup device. No decryption is done during the backup operation.
- TDE encrypts the storage of an entire database by using a symmetric key called the Database Encryption Key (DEK).
- Service-managed TDE - where the DEK is protected by a built-in server certificate.
- Customer-managed TDE - the TDE Protector that encrypts the DEK is supplied by customer and stored in a customer-owned and managed in their key management system

To use TDE with databases in an Always On Availability Group, the certificate used to encrypt the database must be backed up and restored to the other servers within the Availability Group that will be hosting copies of the database.

Azure's Azure Key Vault service for TDE

Azure Key Vault is a tool for storing and accessing sensitive data like passwords, certificates or keys. It is a centralized storage solution for application secrets and also helps monitor how and when the keys/certificates are being accessed. Azure Key vault easily integrates with other Azure services. Key Vault data has its own RBAC policies, separate from the Azure subscription so users need explicit access to be granted.

In the scenario discussed at the beginning of the unit, you are implementing a solution to protect data-at- rest from unauthorized data access by someone that is working to support this cloud hosted database. **Customer-managed TDE - Bring Your Own Key** implementation of TDE is a good solution for the above scenario. The TDE Protector that encrypts the Database Encryption Key (DEK) is a customer-managed asymmetric key, which is stored in a customer-owned and managed Azure Key Vault (Azure's cloud-based external key management system) and never leaves the key vault.

You can read more here [Azure SQL TDE with customer-managed key](#)

Protect data-in-transit

SQL Database, SQL Managed Instance, and Azure Synapse Analytics enforce encryption (SSL-Secure Sockets Layer/TLS - Transport Layer Security) at all times for all connections. This ensures all data is encrypted “in transit” between the client and server. **Transport Layer Security (TLS)** is used by all drivers that Microsoft supplies or supports for connecting to databases in Azure SQL Database or Azure SQL Managed Instance. In some circumstances, you might want to isolate the entire communication channel between your on-premises and cloud infrastructures by using a VPN.

In the table are scenarios of when to use VPN Gateway, TLS and HTTPS

SCENARIO	SOLUTION
Secure access from multiple workstations located on-premises to an Azure virtual network	Use site-to-site VPN
Secure access from an individual workstation located on-premises to an Azure virtual network	Use point-to-site VPN
Move large data sets over a dedicated high-speed wide-area network (WAN) link	Use Azure ExpressRoute
Interact with Azure Storage through the Azure portal	All transactions occur via HTTPS. You can also use Storage REST API over HTTPS to interact with Azure Storage and Azure SQL Database.

Protect data-in-use

Consider the scenario where you have customer service representatives accessing the database containing customer Phone Number and email address. You want to still enable them to verify the users calling in so except the last four digits of the customer's phone number, the rest needs to be encrypted and not revealed to the service representative. How would you implement a solution for this case?

Dynamic Data Masking

Dynamic data masking is a policy-based security feature that hides the sensitive data in the result set of a query over designated database fields, while the data in the database is not changed. It helps prevent unauthorized access to sensitive data by enabling customers to designate how much of the sensitive data to reveal with minimal impact on the application layer.

- Dynamic data masking automatically discovers potentially sensitive data in Azure SQL Database and SQL Managed Instance and provides actionable recommendations to mask these fields.
- It works by obfuscating the sensitive data in the result set of a query
- Data masking policy can be set up in Azure portal only for Azure SQL DB
- Dynamic data masking can be set up using PowerShell cmdlets and REST API

In the image below, the last three digits of phone number column and rest of the characters are masked.



Data masking rules consist of the column to apply the mask to and how the data should be masked. Use the standard mask or specify your own custom masking logic.
“The following table shows the column that needs to be masked and the logic:

SCENARIO	SOLUTION
Default	Displays the default value for that data type instead
Credit card	XXXX-XXXX-XXXX-1234
Email	aXX @XXXX.com

Random Number

Generated random number between selected boundaries

Custom Text

Adds padding string between exposed prefix and exposed suffix characters

Note that Dynamic Data Masking is a presentation layer feature, and unmasked data will always be seen by administrators.

Consider a scenario where a database is queried and accessed by users of all authorization levels. A user that doesn't have administrator privileges queries a table that has some sensitive customer data such as phone number, email etc. along with purchase, and product information.

â€ŽHow will you implement a solution that suppresses the sensitive information while allowing the user to view the other fields?

Dynamic data masking for data-in-use - You can set up a dynamic data masking policy using default masking logic for email, credit cards etc., or specify custom text or random number for the field in question. You can allow other users to view the non-masked versions by adding them to the SQL users, excluded from the masking list.

Always Encrypted feature for data-at-rest and data-in-transit

Always Encrypted is a feature designed to protect sensitive data stored in specific database columns from access (for example, credit card numbers, national identification numbers, etc.). This includes database administrators or other privileged users who are authorized to access the database to perform management tasks. The data is always encrypted, which means the encrypted data is decrypted only for processing by client applications with access to the encryption key. This key can be stored either in the Windows Certificate Store or in Azure Key Vault.

How Always Encrypted works

Step by step process for Always Encrypted is explained below:

- Always Encrypted uses two types of keys: column encryption keys and column master keys.
- A column encryption key is used to encrypt data in an encrypted column. A column master key is a key-protecting key that encrypts one or more column encryption keys.
- The Database Engine only stores encrypted values of column encryption keys and the information about the location of column master keys, which are stored in external trusted key stores, such as Azure Key Vault, Windows Certificate Store
- To access data stored in an encrypted column in plaintext, an application must use an Always Encrypted enabled client driver. Encryption and decryption occurs via the client driver.
- The driver transparently collaborates with the Database Engine to obtain the encrypted value of the column encryption key for the column as well as the location of its corresponding column master key.
- The driver contacts the key store, containing the column master key, in order to decrypt the encrypted column encryption key value, and then it uses the plaintext column encryption key to encrypt the parameter.
- The driver substitutes the plaintext values of the parameters targeting encrypted columns with their encrypted values, and it sends the query to the server for processing.
- The server computes the result set, and for any encrypted columns included in the result set, the driver attaches the encryption metadata for the column, and then the driver decrypts the results and returns plaintext values to the application.

Read about Always Encrypted [best practices](#)

Always Encrypted does not work with Dynamic Data Masking. It is not possible to encrypt and mask the same column. You need to prioritize protecting data in use vs. masking the data for your app users via Dynamic Data Masking.

Consider this scenario - A customer has an on-premises client application at their business location. As part of their cloud migration of their database, they are evaluating Azure SQL database. They plan to outsource the database maintenance to third parties. They worry that the sensitive data shouldn't be accessed by the third party vendors or the cloud administrators. What will you suggest?

Always Encrypted for data-at-rest and data-in-transit - Always Encrypted ensures the separation of duties between database administrators and application administrators. By storing the Always Encrypted keys in a trusted key store hosted on-premises, they can ensure Microsoft cloud administrators have no access to sensitive data. Always Encrypted enables customers to confidently store sensitive data outside of their direct control.

Design for Azure SQL Edge

Azure SQL Edge is an optimized relational database engine geared for IoT and IoT Edge deployments. Azure SQL Edge is built on the same engine as SQL Server and Azure SQL. This means that the developers with SQL server skills can reuse their code to build edge-specific solutions on Azure SQL Edge. Azure SQL Edge provides capabilities to stream, process, and analyze relational and non-relational such as JSON, graph and time-series data.

Azure SQL Edge is available with two different editions, as described in the following table. These editions have identical feature sets, and only differ in terms of their usage rights and the amount of memory and cores they can access on the host system.

- Azure SQL Edge Developer. For development only. Each Azure SQL Edge Developer container is limited to up to 4 cores and 32 GB memory.
- Azure SQL Edge. For production. Each Azure SQL Edge container is limited to up to 8 cores and 64 GB memory.

Understand Azure SQL Edge deployment models

Azure SQL Edge supports two deployment modes.

- **Connected deployment through Azure IoT Edge.** Azure SQL Edge is available on the Azure Marketplace and can be deployed as a module for Azure IoT Edge
- **Disconnected deployment.** Disconnected deployment through Azure SQL Edge container images which can be pulled from docker hub and deployed either as a standalone docker container or on a Kubernetes cluster.

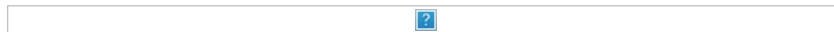
Azure SQL Edge is optimized for IoT use cases and workloads. SQL Server and SQL Database, in contrast, are built for mission-critical, data-management solutions, and line-of-business (LOB) apps.

How does Azure SQL Edge work?

Azure SQL Edge is a containerized Linux application. The startup-memory footprint is less than 500 megabytes (MB). This allows users design and build apps that run on many IoT devices. SQL Edge can:

- Capture continuous data streams in real time
- Integrate the data in a comprehensive organizational data solution

The following diagram shows SQL Edge's capability of capturing and storing streaming data.



With IoT data it is important not only to capture continuous real-time streams of increasing amounts of data but also derive valuable insights about it. Azure SQL Edge supports a built-in streaming engine to help address these needs. It has the following features

- The Streaming engine allows transformation, Windowed aggregation, Simple anomaly detection and classification of incoming data streams
- A time-series storage engine that allows storage of time-indexed data. This can be aggregated and stored in cloud for future analysis

The following diagram shows Azure SQL Edge interacts with components at the network edge including edge gateways, IoT devices, and edge servers.



Security is a primary concern when deploying IoT apps to the edge. Because Azure SQL Edge is based on SQL Server technology, one of the most secure database platforms available, it has the same security features of SQL Server Enterprise. Also the same security policies and practices are extended from cloud to the edge.

Securing Azure SQL Edge deployments involves the steps described in the following table:

1. Platform and system security. This includes the physical docker host, the operating system on the host, and the networking systems connecting the physical device to applications and clients.
2. Authentication and authorization. SQL authentication refers to the authentication of a user when connecting to Azure SQL Edge using username and password. Authorization refers to the permissions assigned to a user within a database in Azure SQL Edge
3. Database object security. "Securables" are the server, database, and objects the database contains. Encryption enhances security. Data protection with Transparent Data Encryption (TDE) enables compliance with many security regulations. Always Encrypted provides separation between users who own the data and those who manage it
4. Application security. Azure SQL Edge security best practices include writing secure client applications.

Scenario - Real time ingestion of data

Let's imagine you work for an automotive manufacturing company. You're working on an IoT app that ingests data from several IoT sensors in the vehicles your company manufactures. It's important that the data is usable all the time, regardless of whether the vehicles' apps are online or offline. Another goal is to use the data to help with product development. This means that the data must synchronize easily with cloud-based database systems built in Azure SQL.

You've been tasked to recommend a solution specifically for SQL Server and should be powerful enough to support edge compute. It should be secure enough to help meet the privacy needs of IoT applications.

Azure SQL Edge is best suited to support the above requirement due to its small footprint and is edge-

optimized for IoT devices.

When do we use Azure SQL Edge?

Azure SQL Edge is ideal for:

Requirement	SQL Edge capability
Connectivity limitations	Azure SQL Edge supports solutions that work with, or without, network connectivity.
Slow or intermittent broadband connection	Azure SQL Edge provides a powerful, local database. It negates needing to forward all data to a cloud-based database, which eliminates latency.
Data security and privacy concerns	Azure SQL Edge implements RBAC and ABAC, encryption, and data classification. This helps you secure and control access to your IoT apps' data.
Synchronization and connectivity to back-end systems	Azure SQL Edge provides ease of exchanging data with other systems like Azure SQL Database, SQL Server, and Azure Cosmos DB.
Familiarity	Azure SQL Edge shares the same codebase as SQL Server. Developers with skills in SQL Server or SQL Database can reuse their code and skills

Design for Azure Cosmos DB and tables

Azure Cosmos DB is a fully managed NoSQL database service for modern app development. It has single-digit millisecond response times and guaranteed speed at any scale.

As a fully managed service, Azure Cosmos DB takes database administration off your hands with automatic management, updates, and patching. It also handles capacity management with cost-effective serverless and automatic scaling options that respond to application needs to match capacity with demand.

Some of the features of Azure Cosmos DB are:

- Automatic and instant scalability.
- Enterprise-grade security.
- Business continuity is assured with 99.999% SLA-backed availability.
- Turnkey multiple region data distribution anywhere in the world.
- Open-source APIs and SDKs for most popular languages.
- Takes care of database administration with automatic management, updates, and patching.
- Build fast with no-ETL analytics over operational data.

Azure Cosmos DB is flexible and stores data in atom-record-sequence (ARS) format. The data is then abstracted and projected as an API.

Differences between Azure Storage tables and Azure Cosmos DB tables

Azure Table storage is a service that stores non-relational structured data (also known as structured NoSQL data) in the cloud, providing a key/attribute store with a schemaless design. Because Table storage is schemaless, it's easy to adapt your data as the needs of your application evolve.



Common uses of Table storage include:

- Storing TBs of structured data capable of serving web scale applications. For example, you can store any number of entities in a table, and a storage account may contain any number of tables, up to the capacity limit of the storage account.
- Storing datasets that don't require complex joins, foreign keys, or stored procedures. Example datasets include web applications, address books, device information.
- Quickly querying data using a **clustered index**. Access to Table storage data is fast and cost-effective for many types of applications. Table storage is typically lower in cost than traditional SQL for similar volumes of data.

Azure Cosmos DB provides the Table API for applications that are written for Azure Table storage and that need premium capabilities like high availability, scalability, and dedicated throughput.

There are some differences in behavior between Azure Storage tables and Azure Cosmos DB tables to remember if you are considering a migration. For example:

- You are charged for the capacity of an Azure Cosmos DB table as soon as it is created, even if that capacity isn't used. This charging structure is because Azure Cosmos DB uses a reserved-capacity model to ensure that clients can read data within 10 ms. In Azure Storage tables, you are only charged for used capacity, but read access is only guaranteed within 10 seconds.
- Query results from Azure Cosmos DB are not sorted in order of partition key and row key as they are from Storage tables.
- Row keys in Azure Cosmos DB are limited to 255 bytes.
- Batch operations are limited to 2 MBs.
- Cross-Origin Resource Sharing (CORS) is supported by Azure Cosmos DB.
- Table names are case-sensitive in Azure Cosmos DB. They are not case-sensitive in Storage tables.

While these differences are small, you should take care to review your apps to ensure that a migration does not cause unexpected problems.

Other benefits to moving to Cosmos DB

Applications written for Azure Table storage can migrate to the Cosmos DB Table API with few code changes. Azure Cosmos DB Table API and Azure Table storage share the same table data model and expose the same create, delete, update, and query operations through their SDKs.

If you currently use Azure Table Storage, you gain the following benefits by moving to the Azure Cosmos DB Table API:

Feature	Azure Table Storage	Azure Cosmos DB Table API
Latency	Fast, but no upper bounds on latency.	Single-digit millisecond latency for reads and writes.
Throughput	Variable throughput model.	Highly scalable with dedicated reserved throughput.
Global distribution	Single region with one optional readable secondary read region for high availability.	Turnkey global distribution from one to 30+ regions.
Indexing	Only primary index on PartitionKey and RowKey. No secondary indexes.	Automatic and complete indexing on all properties, no index management.
Query	Query execution uses index for primary key, and scans otherwise.	Queries can take advantage of automatic indexing on properties for fast query times.
Consistency	Strong within primary region	Five well-defined consistency levels to trade off availability, latency, throughput, and consistency.
Pricing	Consumption-based	Available in both consumption-based and provisioned capacity modes.
SLAs	99.99% availability	99.99% availability SLA for all single region accounts and all multi-region accounts with relaxed consistency, and 99.999% read availability on all multi-region database accounts.

What database APIs does Cosmos DB provide?

Azure Cosmos DB offers multiple database APIs, to provide native interface for a range of NoSQL databases. Using these APIs, Azure Cosmos DB helps you to use the ecosystems, tools, and skills you already have for data modeling and querying.

In this unit, we will examine some real-life scenarios where Core (SQL) API, API for MongoDB, Cassandra API and Gremlin API are applicable. The following diagram summarizes the workflow for selecting an appropriate data storage solution.



When to use Core (SQL) API

Let's imagine you work for a company that's an e-commerce retailer that sells automotive parts. The company stores a lot of product catalogs as JSON files. For semi-structured data storage, they've decided to migrate to Azure Cosmos DB. You need to choose an API for Cosmos DB that will:

- Enable fast scaling for new products.
- Supports current querying method of using SQL for JSON data.

Your requirements are:

- Use the team's existing skill set of SQL querying for JSON objects.
- Globally accessible data with guaranteed throughput.
- Support for adding new product categories quickly.

Based on the above criteria, you decide on Core (SQL) API to store the catalog for your customer facing e-commerce site. This API stores data in document format. Why choose Core (SQL) API?

- Single-digit millisecond latency for read and writes.
- Globally distributed database with automatic failover.

- Read and write availability with 99.999% SLA's.
- Leverage provisioned throughput or autoscale.
- Recommended for use cases such as storing cache data, session management repository, user & profile management, and product recommendation.

When to use MongoDB API

Let's imagine Tailwind Traders uses Mongo DB to store purchase orders for different home improvement products. The products are not limited to any structure and can have many attributes. Tailwind Traders would like to migrate to Azure Cosmos DB. You need to help database staff to choose the right API.

Your requirements are:

- Existing database uses MongoDB to process purchase orders.
- The operations team wants to migrate with few code changes and as little downtime as possible.
- The development team has spent lot of effort on building custom SDK.
- The volume of orders is increasing so scalability needs to be considered.
- There is a need to run analytics against real-time data for business intelligence (BI) use cases.

An effective choice here is to pick **MongoDB API for Azure Cosmos DB**. Azure Cosmos DB API for MongoDB implements the wire protocol for MongoDB. You can leverage your MongoDB experience and continue to use your favorite MongoDB drivers, SDKs, and tools by pointing your application to the API for MongoDB account's connection string.

The advantages of MongoDB are described in the next table.

Advantage	Description
Instantaneous scalability	The Autoscale feature allows you to scale your database up/down with zero warmup period.
Automatic and transparent sharding	The API for MongoDB manages all of the infrastructure for you. This includes sharding and the number of shards.
High Availability	99.999% availability is configurable.
Serverless deployments	This API for Azure Cosmos DB offers a serverless capacity mode. With Serverless, you are only charged per operation, and don't pay for the database when you don't use it.
Fast upgrades	All API versions are contained within one codebase, so version changes takes seconds with zero downtime.
Real time analytics at scale	This API offers the ability to run complex analytical queries for BI applications against your database data in real time. This is fast because of columnar data store and no ETL pipelines.

When to use the Cassandra API

Let's imagine you are modeling a solution for a use case involving storing of health tracker data. This also involves telemetry and sensor data. Currently this is being done in Apache Cassandra. You need to help them choose the right API for scaling their database needs.

Your requirements are:

- Your developers are currently using Cassandra Query Language (CQL), Cassandra-based tools (like cqlsh), and Cassandra client drivers
- The solution should support horizontal scaling.
- Online load balancing and cluster growth is desired.
- There should be a flexible schema.

An effective choice here is to pick **Cassandra API for Azure Cosmos DB**. Cassandra API is wire protocol compatible with the Apache Cassandra. This API stores data in column-oriented schema. Cassandra API currently only supports OLTP scenarios. The API supports CQL version 3.x.

The serverless capacity mode is now available on Azure Cosmos DB's Cassandra API.

What are the advantages of Cassandra API?

The advantages of using Cassandra API are as follows:

Feature	Description
Built-in tools	Uses native Apache Cassandra features, tools, and ecosystem with the API. The Cassandra API manages the OS, Java VM, garbage collection, read/write performance, nodes, and clusters. You don't need the node tool commands, such as repair and decommission, that are used in Apache Cassandra.
Fully managed	The Cassandra API allows you to choose single region or multiple region write configurations.
Regional writes	You can minimize latency by provisioning throughput (RUs) in the Cassandra API.
Integration	You can configure Azure Cosmos containers in autoscale provisioned throughput.

The advantage of multiple region write configurations is to maintaining strong consistency across multiple regions and avoid cross-region conflict scenarios.

When to use the Gremlin API

Let's imagine you are required to analyze and provision a new non-relational database application on Azure ecosystem for Tailwind Traders. You are looking to store social media entity relationships and be able to traverse them quickly in the database.

Your requirements are:

- Store, retrieve, and manipulate graph data and visualize it using Data Explorer.
- Process high volumes of transactions without affecting performance.
- Overcome traditional graph database limitations of flexibility and relational approaches.
- Users should have capability of working with Graph query language to ingest and query data.

An effective choice here is to pick **Azure Cosmos DB's Gremlin API** which is based on the Apache TinkerPop. Apache TinkerPop is a graph computing framework that uses the Gremlin query language. A graph is a structure that's composed of vertices and edges. Vertices represent objects and edges denote the relationships between vertices.

Use cases for this type of database are storing organizational hierarchies, online fraud detection systems, social media graphs, and IoT. Gremlin API currently only supports online transactional processing (OLTP) scenarios.

What are the advantages of Gremlin API?

- Gremlin API has a wire protocol support with the open-source Gremlin, so you can use the open-source Gremlin SDKs to build your application.
- You can store massive graphs with billions of vertices and edges. The data will be automatically distributed using graph partitioning allowing to elastically scale throughput and storage.
- You can query the graphs with millisecond latency and evolve the graph structure easily.
- Gremlin API also works with Apache Spark and GraphFrames for complex analytical graph scenarios.
- Multi-region replication provides automatic regional failover mechanism to ensure the continuity of your application.

Introduction



Imagine that you work as an Architect for Tailwind Traders, a company that specializes in hardware manufacturing with online sales. In addition to historical data, there is data streaming from real-time critical manufacturing processes, product quality control data, historical production logs, product volumes in stock etc.

Keeping in line with the cloud migration strategy your company is implementing, you must analyze and architect a cloud data solution designed to handle the ingestion, processing, and analysis of data that is too large and complex for traditional database systems. You have been asked to determine how best to combine data from multiple sources; reformat it into analytical models, and save these models for subsequent querying, reporting, and visualization.

Learning objectives

In this module, youâ€™ll be able to:

- Design a data integration solution with Azure Data Factory
- Design a data integration solution with Azure Data Lake
- Design a data integration and analytics solution with Azure Databricks
- Design a data integration and analytics solution with Azure Synapse Analytics
- Design a strategy for hot/warm/cold data path
- Design Azure Stream Analytics solution for Data Analysis

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions. The module concepts are covered in:

Design Data Storage Solutions

Design Data Integration

- Recommend a solution for data integration
- Recommend a solution for data analysis

Prerequisites

- Working experience with data integration solutions
- Conceptual knowledge of data integration solutions

Design for Azure Data Factory

A significant challenge for a fast-growing home improvement retailer like Tailwind Traders is that it generates high volume of data stored in relational, non-relational and other storage systems in both the cloud and on-premises. Management wants actionable business insights from this data as near real time as possible. Additionally, sales team wants to set up and roll out up-selling and cross-selling solutions. How will you create a large-scale data ingestion solution in the cloud? What Azure services and solutions will you adopt to help with the movement and transformation of data between various data stores and compute resources?

Azure Data Factory is a cloud-based ETL and data integration service that can help you create and schedule data-driven workflows (called pipelines) that can ingest data from disparate data stores. You can use Azure Data Factory to

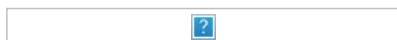
- Orchestrate data movement.
- Transform data at scale.

Data-driven workflows

There are four major steps in creating and implementing a data-driven workflow in Azure Data Factory.

1. **Connect and collect** – Data ingestion is the first step to collecting all the data from different sources into a centralized location.
2. **Transform and enrich** – Now you will use a compute service like Azure Databricks and Azure HDInsight Hadoop to transform the data
3. **Continuous integration and delivery (CI/CD) and publish** – Support for CI/CD through GitHub and Azure DevOps enables to deliver your ETL process incrementally before publishing the data to the analytics engine.
4. **Monitor** – Via the Azure portal, you can monitor the pipeline for scheduled activities and for any failures.

The following graphic shows Azure Data Factory orchestrating the ingestion of data from different data sources. Data is ingested into Storage Blob and stored in Azure Synapse Analytics. Analysis and Visualization components are also connected to Azure Data Factory. Azure data factory provides a common management interface for all of your data integration needs.

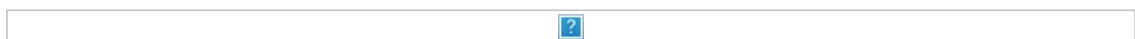


When to use Azure Data Factory

Let us evaluate Azure Data Factory against the following decision criteria:

- **Requirements for data integration** – Azure Data Factory serves two communities – Big data community and the Relational data warehousing community that uses SQL Server Integration Services (SSIS). Depending on your organization's data needs you would set up pipelines in the cloud using Azure Data Factory that can access data from both cloud and on-premises data services.
- **Coding resources** – If you prefer a graphical interface to set up pipelines, then Azure Data Factory authoring and monitoring tool is the right fit for your needs. Azure Data Factory provides a low code/no code process for working with data sources
- **Support for multiple data sources** – Azure Data Factory supports 90+ connectors to integrate with disparate data sources.
- **Serverless infrastructure** – There are advantages in using a fully managed, serverless solution for data integration - No need to maintain, configure or deploy servers, and the ability to scale with fluctuating workloads.

Components of Azure Data Factory



Azure Data factory has the following components as shown in the image above that work together to provide the platform for data movement and data integration.

- **Pipelines and activities** - A logical grouping of activities that perform a task. An activity is a single processing step in a pipeline. Azure Data Factory supports data movement, data transformation, and control activities.
- **Datasets** – These are data structures within your data stores.
- **Linked services** - Define the required connection information needed for Azure Data Factory to connect to external resources.
- **Data Flows** - Data flows allow data engineers to develop data transformation logic without

writing code. Data flow activities can be operationalized using existing Azure Data Factory scheduling, control, flow, and monitoring capabilities.

- **Integration Runtimes** – It is the bridge between the activity and linked Services objects. There are three types of Integration Runtime, including Azure, Self-hosted, and Azure-SSIS.

Let's review how ADF's components are involved in a data preparation and movement scenario for Tailwind Traders. They have many different data sources to connect to and that data needs to be ingested and transformed through stored procedures that are run on the data. Finally, the data should be pushed to analytics platform for analysis.

- In this scenario, Linked Service enables Tailwind Traders to ingest data from different sources and it stores connection strings to fire up compute services on demand.
- You can execute stored procedures for data transformation that happens through Linked Service in Azure SSIS, which is the integration runtime environment for Tailwind Traders.
- The datasets components are used by the activity object and the activity object contains the transformation logic.
- You can trigger the pipeline, which is all the activities grouped together.
- You can then use Data Factory to publish the final dataset to another linked service that is consumed by technologies such as Power BI or Machine Learning.

Design for Azure Data Lake

Suppose you are a data engineering consultant doing work for Tailwind Traders. They have multiple sources of data, from websites to Point of Sale (POS) systems, and more recently from social media sites to Internet of Things (IoT) devices. They are interested in using Azure to analyze all their business data. In this role, you will provide guidance on how Azure can enhance their existing BI systems. You will also offer advice about using Azure's storage capabilities to add value to their BI solution. Because of their needs, you plan to recommend Azure Data Lake Storage. Data Lake Storage provides a repository where you can upload and store huge amounts of unstructured data with an eye toward high-performance big data analytics.

Understand Azure Data Lake Storage

A data lake is a repository of data that is stored in its natural format, usually as blobs or files. **Azure Data Lake Storage** is a comprehensive, scalable, and cost-effective data lake solution for big data analytics built into Azure. Azure Data Lake Storage combines a file system with a storage platform to help you quickly identify insights into your data. Data Lake Storage Gen2 builds on Azure Blob storage capabilities to optimize it specifically for analytics workloads. This integration enables analytics performance, high-availability, security, and durability capabilities of Azure Storage.

The current implementation of Azure's data lake storage service is Azure Data Lake Storage Gen2.

What are the features of Azure Data Lake Storage?

To better understand Azure Data Lake Storage, you can examine its following characteristics:

Feature	Description
Data storage	Azure Data Lake Storage can store any type of data by using the data's native format. With support for any data format and massive data sizes, Azure Data Lake Storage can work with structured, semi-structured, and unstructured data.
Data access	Azure Data Lake Storage is primarily designed to work with Hadoop and all frameworks that use the Apache Hadoop Distributed File System (HDFS) as their data access layer.
Data costs	Azure Data Lake Storage is priced at Azure Blob Storage levels.
Data performance	Azure Data Lake Storage supports high throughput for input/output-intensive analytics and data movement.
Data security	The Azure Data Lake Storage access control model supports both Azure role-based access control (RBAC) and Portable Operating System Interface for UNIX (POSIX) access control lists (ACLs).
Data redundancy	Azure Data Lake Storage utilizes Azure Blob replication models. These models provide data redundancy in a single datacenter with locally redundant storage (LRS).
Data scalability	Azure Data Lake Storage offers massive storage and accepts numerous data types for analytics.
Data analysis	Data analysis frameworks that use HDFS as their data access layer can directly access.

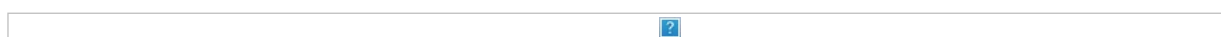
How Azure Data Lake Storage works

There are three important steps to use Azure Data Lake Storage. These are:

Ingesting data - Azure Data Lake Storage offers many different data ingestion methods.

- For ad hoc data -Use tools like AzCopy, CLI, PowerShell, Storage Explorer etc.
- For relational data-Use the Azure Data Factory service. It enables you to transfer data from any of the sources including Cosmos DB, SQL Database, Managed instances etc.
- For streaming data-Use tools such as Apache Storm on Azure HDInsight, Azure Stream Analytics etc.

Following diagram shows how ad hoc data and streaming data are bulk ingested or ad hoc ingested in Azure Data Lake Storage.



Accessing stored data

The easiest way to access your data is to use Azure Storage Explorer. Storage Explorer is a standalone application with a graphical user interface (GUI) for accessing your Azure Data Lake Storage data. You can also use PowerShell, Azure CLI, HDFS CLI or other programming language SDKs for accessing the data.

Setting access control

To control who can access the data stored in Azure Data Lake Storage, you can implement one or both of the following authorization mechanisms:

- Azure RBAC
- ACL

When to use Azure Data Lake Storage

Now we will consider whether Azure Data Lake Storage is the right choice for your organization's big data requirements.

Requirement	Description
When you need a Data warehouse on the cloud for managing large volumes of data	Azure Data Lake Storage runs on virtual hardware on the Azure platform, making storage scalable, fast, and reliable without incurring massive charges. It separates storage costs from compute costs. As your data volume grows, only your storage requirements change.
When you need to manage a diverse collection of data types such as JSON files, CSV, log files or other diverse formats	Azure Data Lake Storage enables data democratization for your organization by storing all your data formats (including raw data) in a single location. Eliminating data silos enables users to use a tool such as Azure Data Explorer to access and work with every data item in their storage account.
When you need real time data ingestion and storage	Azure Data Lake Storage can ingest real-time data directly from an instance of Apache Storm on Azure HDInsight, Azure IoT Hub, Azure Event Hubs, or Azure Stream Analytics. It also works with semi-structured data and lets you ingest all your real-time data into your storage account.

When to choose Azure Blob Storage over Azure Data Lake

Let us review some criteria that will help you decide when to pick one storage solution over the other. In the following table, the two storage solutions are compared against a set of criteria.

Criteria	Azure Data Lake	Azure Blob Storage
Data type	Good for storing large volumes of text data	Good for storing unstructured non-text based data such as photos, videos, backup etc.
Geographic redundancy	Need to set up replication of data	By default, provides geo redundant storage
Namespaces support	Supports hierarchical namespaces	Supports flat namespaces
Hadoop compatibility	Hadoop services can use data stored in Data Lake	Is not Hadoop compatible
Security	Allows for more granular access	Granular access not supported

Design for Azure Databricks

Azure Databricks is a fully managed, cloud-based Big Data and Machine Learning platform, which empowers developers to accelerate AI and innovation. Azure Databricks provides data science and engineering teams with a single platform for Big Data processing and Machine Learning. Azure Databricks's managed Apache Spark platform makes it simple to run large-scale Spark workloads.

Azure Databricks offers three environments for developing data intensive applications: Databricks SQL, Databricks Data Science & Engineering, and Databricks Machine Learning.

Environment	Description
Databricks SQL	<p>Provides an easy-to-use platform for analysts who want to run SQL queries on their data lake, create multiple visualization types to explore query results from different perspectives, and build and share dashboards.</p> <p>Provides an interactive workspace that enables collaboration between data engineers, data scientists, and machine learning engineers. For a big data pipeline, the data (raw or structured) is ingested into Azure through Azure Data Factory in batches, or streamed near real-time using Apache Kafka, Event Hub, or IoT Hub. This data lands in a data lake for long term persisted storage, in Azure Blob Storage or Azure Data Lake Storage. As part of your analytics workflow, use Azure Databricks to read data from multiple data sources and turn it into breakthrough insights using Spark.</p>
Databricks Data Science & Engineering	<p>An integrated end-to-end machine learning environment incorporating managed services for experiment tracking, model training, feature development and management, and feature and model serving.</p>
Databricks Machine Learning	

Let us analyze a situation for Tailwind Traders, a heavy machinery manufacturing organization and arrive at the best Azure Databricks environment that is applicable for them. Tailwind Traders is leveraging Azure cloud services for their big data needs – the data is both batch and streaming data. They have data engineers, data scientists and data analysts who collaborate produce quick insightful reporting for many stakeholders.

In the above scenario, you would choose Databricks Data Science and Engineering environment because:

- It provides an integrated Analytics “workspace” based on Apache Spark that allows collaboration between different users
- Using Spark components like Spark SQL and Dataframes it can handle structured data and integrates with real-time data ingestion tools like Kafka and Flume for processing streaming data
- Secure data integration capabilities built on top of Spark that enable you to unify your data without centralization and Data scientists can visualize data in a few clicks, and use familiar tools like Matplotlib, ggplot, or d3.
- The runtime abstracts out the infrastructure complexity and the need for specialized expertise to set up and configure your data infrastructure so users can use the languages they are skilled in like Python, Scala, R and explore the data
- This also integrates deeply with Azure databases and stores: Synapse Analytics, Cosmos DB, Data Lake Store, and Blob storage for Contoso's big data storage needs
- Integration with Power BI allows for quick and meaningful insights which is a requirement for Contoso.
- Databricks SQL is not the right choice since it cannot handle unstructured data
- Databricks Machine Learning is also not the right environment choice since machine learning is not their requirement right now.

How Azure Databricks works

Azure Databricks has a Control plane and a Data plane.

- The Control Plane hosts Databricks jobs, notebooks with query results, and the cluster manager.

The Control plane also has the web application, hive metastore, and security access control lists (ACLs) and user sessions. These components are managed by Microsoft in collaboration with Databricks and do not reside within your Azure subscription.

- The Data Plane contains all the Databricks runtime clusters hosted within the workspace. All data processing and storage exists within the client subscription. This means no data processing ever takes place within the Microsoft/Databricks-managed subscription.

When to use Azure Databricks

Azure Databricks is entirely based on Apache Spark and as such is a great tool for those already familiar with the open-source cluster-computing framework. As a unified analytics engine, it's designed specifically for big data processing and data scientists can take advantage of built-in core API for core languages like SQL, Java, Python, R, and Scala.

You can use Azure Databricks as a solution for the following scenarios.

- **Data science preparation of data** - Create, clone and edit clusters of complex, unstructured data, turn them into specific jobs and deliver them to data scientists and data analysts for review
- **Find relevant insights in the data** - Recommendation engines, churn analysis, and intrusion detection are common scenarios that many organizations solve across multiple industries using Azure Databricks.
- **Improve productivity across data and analytics teams** - Create a collaborative environment and shared workspaces for data engineers, analysts, and scientists to work together across the data science lifecycle with shared workspaces, saving teams precious time and resources.
- **Big data workloads** - Leverage Data Lake and engine to get the best performance and reliability for your big data workloads, and to create no-fuss multi-step data pipelines
- **Machine learning programs** - Leverage integrated end-to-end machine learning environment incorporating managed services for experiment tracking, model training, feature development and management, and feature and model serving

Design for Azure Synapse Analytics

Tailwind Traders is serving clients with stock market information. You need a combination of batch and stream processing. The up-to-the-second data might be used to help monitor real-time where an instant decision is required to make informed split-second buy or sell decisions. Historical data is equally important for a view of trends in performance. What kind of data warehouse and data integration solution you would recommend that provides access to the streams of raw data, and the cooked business information derived from this data?

With Azure Synapse Analytics, you can ingest data from external sources and then transform and aggregate this data into a format suitable for analytics processing.

In this unit we will explore the high-level architecture and component parts of Azure Synapse Analytics and how to get started using Azure Synapse Analytics for data integration.



Azure Synapse Analytics leverages a massively parallel processing (MPP) architecture. This architecture includes a control node and a pool of compute nodes.

The Control node is the brain of the architecture. It's the front end that interacts with all applications. The Compute nodes provide the computational power. The data to be processed is distributed evenly across the nodes. You submit queries in the form of Transact-SQL statements, and Azure Synapse Analytics runs them. Azure Synapse Analytics uses a technology named **PolyBase** that enables you to retrieve and query data from relational and non-relational sources. You can save the data read in as SQL tables within the Synapse Analytics service.

Components of Azure Synapse Analytics



The image shows the Azure Synapse components.

- **Synapse SQL pool:** Synapse SQL offers both serverless and dedicated resource models to work with using node-based architecture. For predictable performance and cost, you can create dedicated SQL pools, for unplanned or ad hoc workloads, you can use the always-available, serverless SQL endpoint.
- **Synapse Spark pool:** This is a cluster of servers running Apache Spark to process data. You write your data processing logic using one of the four supported languages: Python, Scala, SQL, and C# (via .NET for Apache Spark). Apache Spark for Azure Synapse integrates Apache Spark - the open source big data engine used for data preparation, data engineering, ETL, and machine learning.
- **Synapse Pipelines:** Azure Synapse Pipelines leverages the capabilities of Azure Data Factory and is the cloud-based ETL and data integration service that allows you to create data-driven workflows for orchestrating data movement and transforming data at scale. You could include activities that transform the data as it is transferred, or you might combine data from multiple sources together.
- **Synapse Link:** This component allows you to connect to Cosmos DB. You can use it to perform near real-time analytics over the operational data stored in a Cosmos DB database.
- **Synapse Studio:** This is a web-based IDE that can be used centrally to work with all capabilities of Azure Synapse Analytics. You can use Synapse Studio to create SQL and Spark pools, define and run pipelines, and configure links to external data sources.

Read about [Analytics end-to-end with Azure Synapse](#)

Analytics end-to-end with Azure Synapse

When to use Azure Synapse Analytics?

- When you have a variety of data sources use Azure Synapse Analytics for code-free ETL and data flow activities.
- When you have a need to implement Machine Learning solutions using Apache Spark, use Azure Synapse Analytics for built-in support for AzureML.
- When you have existing data stored on a data lake and need integration with the Data Lake and additional input sources, Azure Synapse Analytics provides seamless integration between the two.
- When management needs real-time analytics, you can use features like Azure Synapse Link to analyze in real-time and offer insights.

What kind of analytics can you do with Azure Synapse Analytics?

Following table shows the range of analytical types that Azure Synapse Analytics supports:

Analytics type	Description
Descriptive analytics - "What is happening?"	Azure Synapse Analytics leverages the dedicated SQL pool capability that enables you to create a persisted data warehouse to perform this type of analysis. You can also make use of the serverless SQL pool to prepare data from files stored in a data lake to create a data warehouse interactively.
Diagnostic analytics - "Why is it happening?"	You can use the serverless SQL pool capability within Azure Synapse Analytics that enables you to interactively explore data within a data lake. Serverless SQL pools can quickly enable a user to search for additional data that may help them to understand why questions.
Predictive analytics - "What is likely to happen?"	Azure Synapse Analytics uses its integrated Apache Spark engine and Azure Synapse Spark pools for predictive analytics with other services such as Azure Machine Learning Services, or Azure Databricks.
Prescriptive analytics - "What needs to be done?"	This type of analytics examines real-time or near real-time analysis of data.. Azure Synapse Analytics provides this capability through both Apache Spark, Azure Synapse Link, and by integrating streaming technologies such as Azure Stream Analytics.

When to choose Azure Data Factory over Azure Synapse Analytics

Let us examine some criteria that will help you decide when to pick Azure Data Factory solution over Azure Synapse Analytics. In the following table, the two solutions are compared against a set of criteria.

Criteria	Azure Data Factory	Azure Synapse Analytics
Data sharing	Can be shared across different data factories	No sharing of data
Solution templates	Provided with Azure Data Factory template gallery	Provided with Synapse Workspace Knowledge center
Integration Runtime cross region support	Support Cross region data flows	Does not support cross region data flows
Monitoring	Integrated with Azure Monitor	Diagnostic logs in Azure Monitor
Monitoring of Spark Jobs for Data Flow	Not supported	Supported by the Synapse Spark pools

Design a strategy for hot, warm, and cold data paths

Traditionally data was stored on premises without taking into consideration its usage and lifecycle. In the cloud, data can be stored based on access, lifecycle, and other compliance requirements. Let us learn how the concept of hot, warm, and cold data path determines how we store and compute data.

Understand Warm Path

Let us review a common scenario for IoT device data aggregation. The devices could be sending data while not really producing anything. This highlights a very common challenge when trying to extract insight out of IoT data. The data you are reviewing is not available in the data you are getting. Therefore, we need to infer utilization by combining the data we are getting with other sources of data, and applying rules to determine if whether or not the machine is producing. In addition, these rules may change from company to company since they may have different interpretations of what “producing” is.

The warm data path is about analyzing as the data flows through the system. We process this stream in near-real time, save it to the warm storage, and push it to the analytics clients.

The Azure platform provides many options for processing the events and one popular choice is Stream Analytics.

Stream Analytics can execute complex analysis at scale, for example, tumbling/sliding/hopping windows, stream aggregations, and external data source joins. For even more complex processing, performance can be extended by cascading multiple instances of Event Hubs, Stream Analytics jobs, and Azure functions.

Warm storage can be implemented with various services on the Azure platform, such as Azure SQL Database and Azure Cosmos DB.

Understand Cold Path

The warm path is where the stream processing occurs to discover patterns over time. However, there might be a need to calculate the utilization over a period in the past, with different pivots, and aggregations and to merge those results with the warm path results to present a unified view to the user.

The cold path includes the batch layer and the serving layers. The combination provides a long-term view of the system.

The cold path contains the long-term data store for the solution. It also contains the batch layer, which creates pre-calculated aggregate views to provide fast query responses over long periods. The technology options available for this layer on Azure platform is quite diverse.

Azure Storage is a good solution for the cold storage. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cold storage can be either Blobs, Data Lake Storage Gen2, or Azure Tables., or a combination of those.

To store massive amounts of unstructured data, such as JSON, or XML documents containing the unprocessed data received by the IoT application, Blob storage, Azure Files, or Azure Data Lake Storage Gen2 are the best options.

Azure Data Factory is a great solution for creating the batch views on the serving layer. It is a cloud-based managed data integration service that allows you to create data-driven workflows in the cloud for orchestrating and automating data movement and data transformation. It can process and transform the data by using services such as Azure HDInsight Hadoop, Spark, and Azure Databricks. This allows you to build machine learning models and consume them with the analytics clients.

Understand Hot Path

Hot path is typically used for processing or displaying data in real-time. It is employed for real time alerting and streaming operations are performed using this data. **A hot path** is where latency-sensitive data, results need to be ready in seconds or less and it flows for rapid consumption by analytics clients.

When to use Hot/Warm/Cold data path

Let us analyze the requirements where you will decide upon hot, warm, or cold data path.

Requirement	Description
When data requirements are known to change frequently	Use Hot data path
When processing or displaying data in real time	
When data is rarely used. The data might be stored for compliance or legal reasons	Use Cold data path
Used for data that is consumed for long term analytics and batch processing	

When you need to store or display a recent subset of data
Used for data that is consumed for small analytical and batch processing

Use Warm data path

Design Azure Stream Analytics solution for Data Analysis

Tailwind Traders, a fictitious home improvement retailer has begun the digital transformation of their applications and services to help with the growth of the company. A present scenario involves accessing, storing and doing analysis on sensor data from the GPS on their delivery trucks that are on the road delivering goods. Management wants a solution for being able to do real time analytics on the streaming data from the GPS on the trucks and make decisions in real time.

On further analysis, you find that they would like this data present in an existing Power BI visualization dashboard.

How will you help address this problem? Azure Stream Analytics can help in this scenario.

Describe Azure Stream Analytics

The process of consuming data streams, analyzing them, and deriving actionable insights is called stream processing. Azure Stream Analytics is a fully managed (PaaS offering), real-time analytics and complex event-processing engine. It offers the possibility to perform real-time analytics on multiple streams of data from sources such as IoT device data, sensors, clickstreams and social media feeds.

Azure Stream Analytics works on the following concepts:

- **Data stream**- Data streams are continuous data generated by applications, IoT devices or sensors. These data streams are analyzed, and actionable insights are extracted from it. For example, monitoring data stream from industrial and manufacturing equipment, Monitoring data of water pipelines by utility providers. Data streams help understand change over time.
- **Event processing** - Event processing refers to consumption and analysis of a continuous data stream to extract actionable insights from the events happening within that stream. For example, a car passing through a tollbooth should include temporal information, such as a timestamp, indicating when it occurred.



The image shows the Stream Analytics pipeline, and how data is ingested, analyzed, and then sent for presentation or action.

Key features of Azure Stream Analytics

Stream Analytics ingests data from Azure Event Hubs (including Azure Event Hubs from Apache Kafka), Azure IoT Hub, or Azure Blob Storage. The query, which is based on SQL query language, can be used to easily filter, sort, aggregate, and join streaming data over a period. You can also extend this SQL language with JavaScript and C# user-defined functions (UDFs). You can [understand stream processing](#) by reading more about it.

An Azure Stream Analytics job consists of an input, query, and an output. You can do the following with the job output:

- Route data to storage systems like Azure Blob storage, Azure SQL Database, Azure Data Lake Store, and Azure CosmosDB.
- You can send data to Power BI for real-time visualization.
- You can store data in Data Warehouse service like Azure Synapse Analytics to train a machine learning model based on historical data or perform batch analytics.
- You can trigger custom downstream workflows by sending the data to services like Azure Functions, Service Bus Topics or Queues.

You can get some information about [Overview of Azure Stream Analytics Cluster](#) for single-tenant deployment of Stream Analytics.

When to consider Azure Stream Analytics

Following are some common enterprise data use-cases where Azure Stream Analytics can be effectively leveraged.

Use case	Azure Stream Analytics consideration
Analyze real-time telemetry streams from IoT devices	Gathering real-time sensor data in Azure Stream Analytics from building automation systems relaying temperature, humidity, fan runtimes and making adjustments to maintain optimum building temperature while reducing costs.
Web logs/clickstream analytics	A consumer goods retailer can offer real-time product suggestions to users based on e-commerce analytics. Data sources include sensors, social media, satellite imagery, mobile devices etc. For example, extreme weather prediction of wildfires and hurricanes can help airlines with routing and send out alerts of adverse weather conditions to people.
Geospatial analytics	Monitoring high value assets such as Industrial equipment by gathering operational data in Azure Stream Analytics and maximizing the useful life of their equipment can be achieved
Remote monitoring and predictive maintenance of high value assets	

through predictive maintenance to avoid disruption of operation. For example, data gathered from electrical power transformers used by utility companies. Credit card fraudulent transactions can be detected and determined at point of sale depending on unusually large transaction or unusual location usage of the credit card. Alert triggers can be set up on data gathered in Azure Stream Analytics.

Real-time analytics on Point of Sale data

In the Tailwind Traders scenario discussed above, we can leverage Azure Stream Analytics to visualize real-time locations of the trucks through Power BI. For management decisions on analytical workloads, data can be stored in Data warehouse like Cosmos DB, or Azure Data Lake for future analysis.

Azure Stream Analytics supports processing events in CSV, JSON and Avro data formats.

Benefits of Azure Stream Analytics

- Fully Managed - It is offered as a Platform as a Service offering, so there is no overhead of provisioning any hardware or infrastructure. Azure Stream Analytics fully manages your job, so you can focus on your business logic and not on the infrastructure.
- Low cost - Billing is done by Streaming Units (SUs) consumed which represent the amount of CPU and memory resources allocated. Scaling up and down are based on business needs which can also bring cost down. No maintenance costs are involved.
- Run in cloud or Intelligent edge - It can run in the cloud, for large-scale analytics, or run on IoT Edge or Azure Stack for ultra-low latency analytics.
- Performance - Stream Analytics offers reliable performance guarantees. It supports higher performance by partitioning, allowing complex queries to be parallelized and executed on multiple streaming nodes. Stream Analytics can process millions of events every second and it can deliver results with ultra-low latencies.
- Security - In terms of security, Azure Stream Analytics encrypts all incoming and outgoing communications and supports TLS 1.2. Built-in checkpoints are also encrypted. Stream Analytics does not store the incoming data since all processing is done in-memory.

Introduction

The cloud is changing how applications are designed and secured. Instead of monoliths, applications are divided into smaller, decentralized services.

These services communicate through APIs or by using asynchronous messaging or events. The services scale horizontally, adding new instances as demand requires.

These design changes bring new challenges. Application states are distributed, and operations are done in parallel and asynchronously. Applications must:

- Communicate with each other effectively.
- Be able to be deployed rapidly.
- Be resilient when failures occur.
- Be able to integrate with other systems seamlessly.

Azure lets you create applications composed of various components:

- Website front ends
- Back-end services
- Triggered functions

Azure includes various communication strategies to let these various components pass data to each other.

After completing this module, you™ll be able to evaluate and design an effective application architecture. This architecture provides the best Azure solutions for exchanging messages. It also helps automate deployment solutions for your applications and manage configurations. Azure also enables integration with APIs and provides appropriate caching.

Learning objectives

After completing this module, you™ll be able to:

- Describe message and event scenarios.
- Design a messaging solution.
- Design an event hub messaging solution.
- Design an event-driven solution.
- Design an automated app deployment solution.
- Design an API integration solution.
- Design an application configuration management solution.
- Design a caching solution.

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions. The module concepts are covered in:

Design Infrastructure solutions

- Design an Application Architecture.

Prerequisites

- Working experience with developing cloud applications.
- Conceptual knowledge of messaging, events, code deployments, configurations, API management, and app caching.

Introduction

The cloud is changing how applications are designed and secured. Instead of monoliths, applications are divided into smaller, decentralized services.

These services communicate through APIs or by using asynchronous messaging or events. The services scale horizontally, adding new instances as demand requires.

These design changes bring new challenges. Application states are distributed, and operations are done in parallel and asynchronously. Applications must:

- Communicate with each other effectively.
- Be able to be deployed rapidly.
- Be resilient when failures occur.
- Be able to integrate with other systems seamlessly.

Azure lets you create applications composed of various components:

- Website front ends
- Back-end services
- Triggered functions

Azure includes various communication strategies to let these various components pass data to each other.

After completing this module, you™ll be able to evaluate and design an effective application architecture. This architecture provides the best Azure solutions for exchanging messages. It also helps automate deployment solutions for your applications and manage configurations. Azure also enables integration with APIs and provides appropriate caching.

Learning objectives

After completing this module, you™ll be able to:

- Describe message and event scenarios.
- Design a messaging solution.
- Design an event hub messaging solution.
- Design an event-driven solution.
- Design an automated app deployment solution.
- Design an API integration solution.
- Design an application configuration management solution.
- Design a caching solution.

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions. The module concepts are covered in:

Design Infrastructure solutions

- Design an Application Architecture.

Prerequisites

- Working experience with developing cloud applications.
- Conceptual knowledge of messaging, events, code deployments, configurations, API management, and app caching.

Describe message and event scenarios

Imagine you're designing the architecture for a distributed music-sharing application. You want to ensure that the application is as reliable and scalable as possible. You want to use Azure technologies to build a robust communication infrastructure.

But before you can choose the Azure technology, you must understand how each individual component communicates with the other components of the application. For each communication, you can choose a different Azure technology.

Select messages or events for your application

The first thing you must understand is whether an app sends messages or events. Knowing the difference helps you choose the appropriate Azure service.

What is a message?

Messages have the following characteristics.

- Contains raw data, produced by one component, that will be consumed by another component.
- Contains the data itself, not just a reference to that data.

The sending component expects the message content to be processed in a certain way by the destination component. The integrity of the overall system may depend on both sender and receiver doing a specific job.

Example of a message

Let's suppose a user uploads a new song by using your mobile music-sharing app. The mobile app must send that song to the web API that runs in Azure. The song file must be sent, not just an alert that indicates that a new song has been added. The mobile app expects that the web API stores the new song in the database and makes it available to other users.

What is an event?

Events are lighter weight than messages and are most often used for broadcast communications. There are two components involved with events:

- Publishers, which send the event.
- Subscribers, which receive events.

With events, receiving components generally decide in which communications they're interested and subscribe to those events. The subscription is managed by an intermediary. The intermediary can be provided by services such as Azure Event Grid or Azure Event Hubs. When publishers send an event, the intermediary routes that event to interested. This pattern is known as a publish-subscribe architecture and is the most used.

Events have the following characteristics:

- Is a lightweight notification that indicates something occurred?
- May be sent to multiple receivers or to none.
- Is often intended to "fan out" or have many subscribers for each publisher.
- Publisher has no expectation about the action a receiving component takes.
- Is a discrete unit and unrelated to other events?
- Might be part of a related and ordered series.

When should you choose messages or events?

A single application is likely to use events for some purposes and messages for others. The following table describes when to use which:

Event or message	When to use
Event	More likely to be used for broadcasts and are often ephemeral. Ephemeral means the communication might not be handled by any receiver if none is currently subscribing.
Message	More likely to be used where the distributed application requires a guarantee that the communication will be processed.

For each communication, consider the following question: Does the sending component expect the

communication to be processed in a particular way by the destination component?

- If yes, choose to use a message.
- If no, you might be able to use events.

Design a messaging solution

In this unit, you'll learn how to choose the best architecture for a message-based delivery system. Let's imagine you're planning the music-sharing application. You want to:

- Ensure that music files are uploaded to the web API reliably from the mobile app.
- Deliver the details about new songs directly to the app. For example, when an artist adds new music to their collection.

This scenario is a perfect use of a message-based system. Azure offers two message-based solutions Queue Storage and Azure Service Bus (queues and topics).

What is Azure Queue Storage?

Azure Queue storage is a service that uses Azure Storage to store large numbers of messages. These messages can be securely accessed from anywhere in the world using a simple REST-based interface. Queues can contain millions of messages. Azure Queue storage is limited only by the capacity of the storage account that owns it. Queues generally provide increased reliability, guaranteed message delivery, and transactional support.



What is Azure Service Bus?

Microsoft Azure Service Bus is a fully managed enterprise message broker with message queues and publish-subscribe topics. Service Bus is used to decouple applications and services from each other, providing the following benefits:

- Load-balancing work across competing workers.
- Safely routing and transferring data and control across service and application boundaries.
- Coordinating transactional work that requires a high degree of reliability.

What are Azure Service Bus Queues?

Azure Service Bus queues is a message broker system built on top of a dedicated messaging infrastructure. Like Azure queues, Service Bus holds messages until the target is ready to receive them.



Azure Service Bus is intended for enterprise applications. For example, an application that uses communication protocols and different data contracts.

What is an Azure Service Bus publish-subscribe topic?

Azure Service Bus topics are like queues but can have multiple subscribers. When a message is sent to a topic, multiple components can be triggered to perform a task.



For example, suppose a user is listening to a song using a music-sharing application. The mobile app might send a message to the *Listened* topic. That topic could have a subscription for *UpdateUserListenHistory*, and a different subscription for *UpdateArtistsFanList*. Each subscription receives its own copy of the message.

Which messaging service should I choose?

Each messaging product has a slightly different feature set. This which means you can choose one or the other or use both. It depends on the problem you're solving.

Use Azure Queue storage if you need/have:

- A simple queue to organize messages.
- An audit trail of all messages that pass through the queue.
- Queue to exceed 80 GB in size.
- To track progress for processing a message inside of the queue.

Use Azure Service Bus queues if you need/have:

- An At-Most-Once delivery guarantee.
- A first-in-first-out guarantee.
- To group messages into transactions.
- To receive messages without polling the queue.

- To provide a role-based access model to the queues.
- To handle messages larger than 64 KB but less than 256 KB.
- Queue size will not grow larger than 80 GB..
- To publish and consume batches of messages.

Use Azure Service Bus topics if you need/have:

- Multiple receivers to handle each message.
- Multiple destinations for a single message but need queue-like behavior.

Design an Event Hub messaging solution

There are certain applications that produce a massive number of events from almost as many sources. We often refer to this as Big Data. Big Data can require extensive infrastructure.

Let's imagine you work for Contoso Aircraft Engines. The engines your employer manufactures have hundreds of sensors. Before an aircraft can be flown, its engines are connected to a test harness and put through their paces. Additionally, cached in-flight data is streamed when the aircraft is connected to ground equipment.

In this situation, you might choose an event hubs-based messaging solution. Event hubs can receive and process millions of events per second. Data sent to an event hub can be transformed in real-time and stored for later analysis.

How Azure Event Hub works

Azure Event Hubs is a fully managed, real time data ingestion. Event hubs supports real time data ingestion and microservices batching on the same stream. Here are some common scenarios for Event Hubs.

- Anomaly detection (fraud/outliers) and live dashboarding
- Analytics pipelines, such as clickstreams, and archiving data
- Transaction processing with real-time analysis

This diagram shows how event hubs could be used in the aircraft engine application.

- Event hub captures streaming data from the testing equipment.
- The data is stored in Azure blob storage.
- Azure Stream Analytics identifies patterns in the sensor data.
- Power BI is used to make decisions on manufacturing improvements.



Considerations for Event Hub

When selecting Event Hub, consider the following:

- **Expect language and framework integration.** You can send and receive events in many different languages. Messages can also be received from Event Hubs using Apache Storm.
- **Choose a tier and throughput.** Scaling of Event Hubs is controlled by how many throughput units or processing units you **purchase**. Other performance aspects depend on the pricing tier chosen, with basic, standard, premium, and dedicated pricing tiers being available. A single throughput unit equates to:
 - Ingress: Up to 1 MB per second or 1000 events per second (whichever comes first).
 - Egress: Up to 2 MB per second or 4096 events per second.
- **Remember Event Hubs uses a pull model.** The pull model used by Event Hubs differentiates it from some other messaging services, such as Azure Service Bus Queues. The pull model means that Event Hubs simply holds the message in its cache and allows it to be read. When a message is read from Event Hubs, it isn't deleted but is left to be read, as needed, by more consumers.
- **Account for data failures.** There's no built-in mechanism to handle messages that aren't processed as you expect them. For example, imagine your consumer malfunctions because of data format. Event Hubs will not detect this and delete the message once its time-to-live has expired.
- **Process the data stream.** Events received by Event Hubs are added to the end of its data stream. This data stream orders events by the time they are received, and consumers can seek along this stream using time offsets.

Design and Event Grid solution

Event driven architecture enables you to connect to the core application without needing to modify the existing code. When an event occurs, you can react with your own code to these events. Event-driven applications use the send and forget principle. The event gets sent toward the next system, which can be another service, an event hub, a stream, or a message broker.

Let’s consider that music-sharing application again, this time with a Web API that runs in Azure. When a user uploads a new song, you must notify all the mobile apps installed on user devices around the world that are interested in that genre of music. In this requirement. Azure Event Grid is a perfect solution for this sort of requirement.

- The publisher of the sound file doesn't need to know about any of the subscribers interested in the shared music.
- We want to have a one-to-many relationship where we can have multiple subscribers. Subscribers who can optionally decide whether they’re interested in this new song.

What is Azure Event Grid?

Azure Event Grid is a fully managed event routing service running on top of Azure Service Fabric. Event Grid distributes events from sources such as Azure blob storage accounts and Azure media services. These events are distributed to handlers such as Azure Functions and Webhooks. Event Grid exists to make it easier to build event-based and serverless applications on Azure.

- Aggregates all your events and provides routing from any source to any destination.
- Is a service that manages the routing and delivery of events from many sources and subscribers. This process eliminates the need for polling and results in minimized cost and latency.

The following illustration displays an Azure Event Grid positioned between multiple event sources and multiple event handlers. The event sources send events to the Event Grid and the Event Grid forwards relevant events to the subscribers. Event Grid use topics to decide which events to send to which handlers. Events sources tag each event with one or more topics, and event handlers subscribe to the topics they’re interested in.



Event Grid sends an event to indicate something has happened or changed. However, the actual object that was changed isn’t part of the event data. Instead, a URL or identifier is often passed to reference the changed object.

Comparison of services

Let’s take a few minutes to review the message and event solutions we have covered.

Service	Purpose	Type	When to use
Event Grid	Reactive programming	Event distribution (discrete)	React to status changes
Event Hubs	Big data pipeline	Event streaming (series)	Telemetry and distributed data streaming
Service Bus	High-value enterprise messaging	Message	Order processing and financial transactions

Use the services together

In some cases, you use the services side by side to fulfill distinct roles. For example, an e-commerce site can use Service Bus to process the order, Event Hubs to capture site telemetry, and Event Grid to respond to events like an item was shipped. In other cases, you link them together to form an event and data pipeline. You use Event Grid to respond to events in the other services. The following image shows the workflow for streaming the data.



Design a caching solution

Caching is a common technique that aims to improve the performance and scalability of a system. It does this by temporarily copying frequently accessed data to fast storage that's located close to the application. If this fast data storage is located closer to the application than the original source, then caching can significantly improve response times for client applications by serving data more quickly.

Caching is most effective when a client instance repeatedly reads the same data, especially if all the following conditions apply to the original data store:

- It remains relatively static.
- It's slow compared to the speed of the cache.
- It's subject to a high level of contention.
- It's far away when network latency can cause access to be slow.

In this unit, you'll learn how to use Azure Cache for Redis to manage caching requirements.

Recommend a caching solution for applications

Suppose you work at a Tailwind Traders that has launched a new game ,which will have real time leaderboards for gamers to check their scores. The leaderboard would show the user's rank in the game in real time while getting updated as soon as the events in the game changes. This requires in-memory fast read and writes for which you have been asked to design a caching solution.

What is Azure Cache for Redis?

Azure Cache for Redis provides an in-memory data store based on the Redis software. Redis improves the performance and scalability of an application that uses backend data stores heavily. It's able to process large volumes of application requests by keeping frequently accessed data in the server memory, which can be written to and read from quickly. Redis brings a critical low-latency and high-throughput data storage solution to modern applications.

Azure Cache for Redis offers both:

- The Redis open source (OSS Redis)
- A commercial product from Redis Labs (Redis Enterprise) as a managed service.

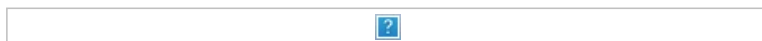
Azure Cache for Redis provides secure and dedicated Redis server instances and full Redis API compatibility. The service is operated by Microsoft, hosted on Azure, and usable by any application within or outside of Azure.

You can use Azure Cache for Redis for several purposes, including as a:

- Distributed data or content cache
- A session store
- A message broker

Tip: You can deploy Azure Cache for Redis as a standalone. Alternatively, you can deploy it with other Azure database services, such as Azure SQL or Cosmos DB.

The following diagram shows how cache works in applications.



When to use Azure Cache for Redis?

Azure Cache for Redis improves application performance by supporting common application architecture patterns. Some of the most common include the following patterns.

Audience

Data cache

Azure API Management benefits

Databases are often too large to load directly into a cache. It's common to use the cache-aside pattern to load data into the cache only as needed. When the system makes changes to the data, the system can also update the cache, which is then distributed to other clients. Additionally, the system can set an expiration on data, or use an eviction policy to trigger data updates into the cache.

Many web pages are generated from templates that use static content such as

Content cache

headers, footers, banners. These static items shouldn't change often. Using an in-memory cache provides quick access to static content compared to backend datastores. This pattern reduces processing time and server load, allowing web servers to be more responsive. It can allow you to reduce the number of servers needed to handle loads. Azure Cache for Redis provides the Redis Output Cache Provider to support this pattern with ASP.NET.

This pattern is commonly used with shopping carts and other user history data that a web application might associate with user cookies. Storing too much in a cookie can have a negative effect on performance as the cookie size grows and is passed and validated with every request. A typical solution uses the cookie as a key to query the data in a database. Using an in-memory cache, like Azure Cache for Redis, to associate information with a user, is much faster than interacting with a full relational database.

Session store

Applications often add tasks to a queue when the operations associated with the request take time to execute. Longer running operations are queued to be processed in sequence, often by another server. This method of deferring work is called task queuing. Azure Cache for Redis provides a distributed queue to enable this pattern in your application.

Job and message queuing

Applications sometimes require a series of commands against a backend data-store to execute as a single atomic operation. All commands must succeed, or all must be rolled back to the initial state. Azure Cache for Redis supports executing a batch of commands as a single transaction.

Distributed transactions

Design an API integration solution

Publishing an API is a great way to increase market share, generate revenue, and foster innovation. However, maintaining even one API brings significant challenges, such as onboarding users, managing revisions, and implementing security.

How do you reduce the complexity inherent in having numerous APIs and their management? You need an API Management that acts as a front door for all your APIs. API Management provides tools for implementing security, managing revisions, and performing analytics.

In this unit, you'll learn about Azure API Management, and determine whether it's the correct solution to help reduce your API complexity.

Select an API management solution

Suppose you work at a Tailwind Traders that has acquired a food-delivery platform. The customers use a mobile app or a website to browse the menus of multiple restaurants. Customers then place an order for the food they want. Your new company delivers the food. The backbone of your platform is a large collection of APIs. Some of the APIs that you publish are used by:

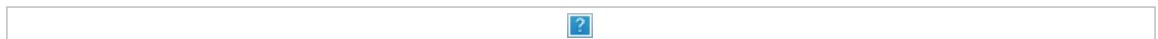
- Your mobile app
- Your web app
- Your partner restaurants
- The IoT devices on your delivery vehicles
- Your in-house development teams
- Your employees, such as business analysts

Each published API resides on a different server, has its own process for onboarding users, and has its own policies for security, revisions, analytics, and more. You've been tasked to find a way to reduce this complexity.

Let's learn how Azure API Management can standardize, centralize, and help secure all the aspects of publishing and maintaining APIs across the full API lifecycle.

What is Azure API Management?

Azure API Management is a cloud service platform that lets you publish, secure, maintain, and analyze all your company's APIs. The following diagram shows Azure API Management that acts as a "front door" for all an organization's APIs, which are then routed to the server where the API is deployed.



Note: Azure API Management does not host your actual APIs; your APIs remain where they were originally deployed. Instead, Azure API Management acts as a kind of façade or "front door" for your APIs. In this way, Azure API Management is said to decouple your APIs. This lets you set API policies and other management options in Azure, while leaving your deployed backend APIs untouched.

When to use Azure API Management?

Here are the criteria we'll use to help you decide whether Azure API Management is a suitable choice for managing and publishing your organization's inventory of APIs.

- Number of APIs
- Rate of API changes
- API administration load

When you have numerous deployed APIs that you revise frequently and that require significant administrative overhead, Azure API Management can help you administer and publish them. Azure API Management might not be the correct choice for use cases that typically involve small, static, or simple API deployments. Let's review the decision criteria in more detail.

Criteria

Number of APIs

Analysis

The key consideration when you're evaluating Azure API Management is the number of APIs that you manage. The more APIs you've deployed, the greater the need for deployment standardization, and centralization of API control.

The next consideration is the rate at which your organization implements API revisions and versions. The faster you

Rate of API changes	create API revisions and publish new API versions, the greater the need for a robust, and flexible versioning control system.
API administration load	The last consideration is how much policy overhead you apply to your APIs. This includes usage quotas, call rate limits, request transformations, and request validation. The more configurations and options your APIs require, the greater the need for standardized, and centralized policy implementations.

Consider Azure API Management

There are many reasons to choose Azure API Management. Using the food delivery scenario above as an example, let's investigate API lifecycle management with respect to standardizing APIs, centralizing API management, and enhancing API security. The following table describes these features.

Feature	Description
Standardize all disparate APIs	Considering disparate APIs in the new company such as mobile, partner restaurants, and IoT devices. It's imperative that you use an API management solution. This solution will standardize API specs, help in creating documentation, and standardize the base URL for ease of use. API Management can provide consistent analytics across multiple APIs and ensure compliance across all APIs.
Centralize API operations	By bringing multiple APIs under a single administrative umbrella, Azure API Management enhances the centralization of all API operations. Without an API management service, each API is on its own in terms of administration, deployment, and developer access. A centralized model results in less duplicated effort and increases efficiency in the food delivery scenario.
Secure APIs	Azure API Management was designed with API security in mind. It manages permissions, access, protects the API from malicious usage and helps in achieving all corporate and government-related compliance.

Design an automated app deployment solution

With the move to the cloud, many teams have adopted agile development methods. These teams must iterate quickly and repeatedly deploy their solutions to the cloud. Teams must be assured their infrastructure is in a reliable state. Application code must be managed through a unified process.

To meet these challenges, you can automate deployments and use the practice of **infrastructure as code**. In this unit, you’ll learn how to evaluate different Azure solutions for deployment and automation for your applications. These solutions include Azure Resource Manager templates, and Azure Automation.

What are Azure Resource Manager templates?

Azure Resource Manager templates are files that define the infrastructure and configuration for your deployment. When you write a template, you take a declarative approach to your resource provisioning. These templates describe each resource in the deployment, but they don't describe how to deploy the resources.

There are many benefits to using templates for your resource provisioning. These benefits are described in the following table:

Function	Template benefit
Repeatable results	Templates are idempotent. Idempotent means you can repeatedly deploy the same template and get the same result.
Orchestration	When a template deployment is submitted to Azure Resource Manager, the resources in the template are deployed in parallel. This process allows deployments to finish faster.
Preview	The WhatIf parameter, available in PowerShell and Azure CLI, allows you to preview changes to your environment before template deployment. This parameter will detail any creations, modification, and deletions that will be made by your template.
Testing and Validation	Templates submitted to Resource Manager are validated before the deployment process. This validation alerts you to any errors in your template before resource provisioning.
Modularity	You can break up your templates into smaller components and link them together at deployment.
CI/CD integration	Your templates can be integrated into multiple CI/CD tools, like Azure DevOps and GitHub Actions.
Extensibility	With deployment scripts, you can run Bash or PowerShell scripts from within your templates. Through extensibility, you can use a single template to deploy a complete solution.

Note: Two types of templates are available for use today: JSON templates and Bicep templates. JavaScript Object Notation (JSON) is an open-standard file format that multiple languages can use. Bicep is a new domain-specific language that was recently developed for authoring templates by using an easier syntax. You can use either template format for your templates and resource deployments.

What are Bicep templates?

Bicep is an Azure Resource Manager template language that's used to declaratively deploy Azure resources. Bicep is a domain-specific language, which means that it's designed for a specific scenario or domain. Bicep is used to create Azure Resource Manager templates.



There are many reasons to choose Bicep as the main tool set for your infrastructure as code deployments. These benefits are described in the following table.

Feature	Description
Azure-native	Bicep is native to the Azure ecosystem. When new Azure resources are released or updated, Bicep will support those

	features on day one.
Azure integration	Templates, both JSON and Bicep, are fully integrated within the Azure platform. With Resource Manager-based deployments, you can monitor the progress of your deployment in the Azure portal.
Azure support	Bicep is a fully supported product with Microsoft Support.
No state management	Bicep deployments compare the current state of your Azure resources with the state that you define in the template. Azure automatically keeps track of this state for you.
Easy transition from JSON	If you're already using JSON templates as your declarative template language, it isn't difficult to transition to Bicep. You can use the Bicep CLI to decompile any template into a Bicep template.

What is Azure Automation?

Azure Automation delivers a cloud-based automation and configuration service that supports consistent management across your Azure and non-Azure environments. Automation gives you complete control of process automation, configuration management, and update management.

Process	Description
Process Automation	Enables you to automate frequent, time-consuming, and error-prone cloud management tasks. This service helps you focus on work that adds business value. By reducing errors and boosting efficiency, it also helps to lower your operational costs. The service allows you to author runbooks graphically, in PowerShell, or using Python.
Configuration Management	Enables access to two features: Change Tracking and Inventory and Azure Automation State Configuration. The service supports change tracking across services, daemons, software, registry, and files in your environment to help you diagnose unwanted changes and raise alerts.
Update management	Includes the Update Management feature for Windows and Linux systems across hybrid environments. The feature allows you to create scheduled deployments that orchestrate the installation of updates within a defined maintenance window.

Design a configuration management solution

Traditionally, shipping a new application feature requires a complete redeployment of the application itself. Testing or deployment of a feature often requires multiple versions of the application. Each deployment may require difference configurations, credentials, changing settings or parameters for testing.

Configuration management is a modern software-development practice that decouples configuration from code deployment and enables quick changes to feature availability on demand. Decoupling configuration as a service enables systems to dynamically administer the deployment lifecycle.

In this unit, youâ€™ll learn about Azure Configuration Management solutions that can help you address deployment issues.

What is Azure App Configuration?

Azure App Configuration provides a service to centrally manage application settings and feature flags. Use App Configuration to store all the settings for your application and secure their accesses in one place.

The following two diagrams show how Azure App Configuration works in Development and Productions environments:

Development



Production



What are the benefits of App Configuration?

App Configuration offers the following benefits:

- A fully managed service that can be set up in minutes.
- Flexible key representations and mappings.
- Tagging with labels.
- Point-in-time replay of settings.
- Dedicated UI for feature flag management.
- Comparison of two sets of configurations on custom-defined dimensions.
- Enhanced security through Azure-managed identities.
- Encryption of sensitive information at rest and in transit.
- Native integration with popular frameworks.

Introduction



Tailwind Traders is a fictitious home improvement retailer. It operates retail hardware stores across the globe and online. The Tailwind Trader™'s CTO asks, "What is our identity solution?" At first, this seems like a simple question. But managing and protecting your corporate identities requires planning and careful design. In this module, we'll answer the questions:

- What identity providers does Azure offer?
- What identity protections are available?

Learning objectives

In this module, you'll learn how to:

- Design for identity and access management.
- Design for Azure Active Directory.
- Design for Azure AD B2B.
- Design for Azure AD B2C.
- Design for conditional access.
- Design for identity protection.
- Design for access reviews.
- Design for service principals for applications.
- Design for Azure Key Vault.

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions. The module concepts are covered in:

Design authentication and authorization solutions.

- Recommend an identity solution.
- Recommend an access control solution for identities.
- Recommend an authorization solution.

Design Identities and Access for Applications

- Recommend a solution that securely stores passwords and secrets
- Recommend solutions to allow applications to access Azure resources
- Recommend a solution for integrating applications into Azure AD
- Recommend a user consent solution for applications

Prerequisites

- Working experience creating, assigning, and securing corporate identities.
- Conceptual knowledge of identity assignment solutions, role-based access control, and identity protection methods.

Design for identity and access management

Azure Architects design Identity and access management (IAM) solutions. These solutions must work for all your users, apps, and devices. There are four basic guidelines for a strong IAM solution.



- **Unified identity management.** Manage all your identities and access to all your apps in a central location, whether they’re in the cloud or on-premises, to improve visibility and control.
- **Seamless user experience.** Provide an easy, fast sign in experience to keep your users productive, reduce time managing passwords, and increase end-user productivity.
- **Secure adaptive access.** Protect access to resources and data using strong authentication and risk-based adaptive access policies without compromising user experience.
- **Simplified identity governance.** Control access to apps and data for all users and admins. Automated identity governance to ensure only authorized users have access.

Let’s first focus on the identity solution. There are three basic choices.

If you need this	Use this
Provide identity and access management for employees in a cloud or hybrid environment.	Azure Active Directory (Azure AD)
Collaborate with guest users and external business partners like suppliers and vendors.	Azure AD Business to Business (B2B)
Control how customers sign up, sign in, and manage their profiles when they use your applications.	Azure AD Business to Consumer (B2C)

Design for Azure Active Directory

Let's begin with **Azure Active Directory (Azure AD)**. Azure AD is the Azure solution for identity and access management. Azure AD is a multitenant, cloud-based directory, and identity management service. It combines core directory services, application access management, and identity protection into a single solution. Azure AD can be used in cloud or hybrid environments.

Cloud identity solution. You can use Azure AD as a cloud only solution for all your employee user accounts. Azure AD provides not only identity management but protection for those accounts. For example, role-based access control, conditional access, and access reviews. We'll cover those features, later in this module.

Hybrid identity solution. You can also use Azure AD in hybrid environments. Azure AD **extends on-premises Active Directory** to the cloud. With Azure AD Connect or Azure AD Connect cloud sync, you can bring on-premises identities into Azure AD. Once the on-premises accounts are in Azure AD they will get the benefits of easy management and identity protection.



Best practices with Azure ID identity management

- **Centralize identity management.** In a hybrid identity scenario, we recommend that you integrate your on-premises and cloud directories. Integration enables your IT team to manage accounts from one location, whenever an account is created. Integration also helps your users be more productive by providing a common identity for accessing both cloud and on-premises resources.
- **Establish a single Azure AD instance.** Consistency and a single authoritative source will increase clarity and reduce security risks from human errors and configuration complexity. Designate a single Azure AD directory as the authoritative source for corporate and organizational accounts.
- **Don't synchronize accounts to Azure AD that have high privileges in your existing Active Directory instance.** By default, Azure AD Connect filters out these high privileged accounts. This configuration mitigates the risk of adversaries pivoting from cloud to on-premises assets (which could create a major incident).
- **Turn on password hash synchronization.** **Password hash synchronization** is a feature used to sync user password hashes from an on-premises Active Directory instance to a cloud-based Azure AD instance. This sync helps to protect against leaked credentials being replayed from previous sign-ins.
- **Enable single sign-on (SSO).** SSO reduces the need for multiple passwords. Multiple passwords increase the likelihood of users reusing passwords or using weak passwords. With SSO, users provide their primary work or school account for their domain-joined devices and company resources. Their application access can be automatically provisioned (or deprovisioned) based on their organization group memberships and their status as an employee.

Organizations that don't integrate their on-premises identity with their cloud identity can have more overhead in managing accounts. This overhead increases the likelihood of mistakes and security breaches.

Design for Azure Active Directory Business to Business

Every organization needs to work with external users. **Azure AD Business to Business (B2B)** is a feature of Azure AD that enables you to securely collaborate with external partners. Your partner users are invited as guest users. You remain in control of what they have access to, and for how long.

With Azure AD B2B, the partner uses their own identity management solution. Azure AD is not required. You don't need to manage external accounts or passwords. You don't need to sync accounts or manage account lifecycles. Guest users sign in to your apps and services with their own work, school, or social identities.

With Azure AD B2B, external users can use their identities to collaborate with your organization. Their identities are managed by the partner themselves, or by another external identity provider on their behalf.



The following steps show how Azure AD B2B lets you collaborate with external partner users. The numbers in the diagram are explained after the diagram.



1. You invite external users as guest users to your directory. For example, you fill in a form with your guest user's details and a custom invitation message.
2. Guest user receives an invitation via email. The first time the link is used, the user is asked for consent. The user must accept the permissions needed by Azure AD B2B before they can gain access.
3. If you've enabled multifactor authentication (MFA), the user provides these extra details for their account. When MFA is configured, the user must enter a verification code sent to their mobile device before they're granted access.
4. Your guest user is then forwarded to the access panel page. This page presents all the applications and services you've shared with them. These applications and services can be cloud-based, or on-premises.

Best practices for Azure AD B2B

- **Designate an application owner to manage guest users.** It's a good idea to delegate guest user access to application owners. Application owners are in the best position to decide who should be given access to a particular application.
- **Use conditional access policies to intelligently grant or deny access.** Conditional access policies use factors that aren't credential-based. For example, you can make it mandatory for users to be on specific device platforms, such as Android or Windows. Another example, you can block users if they don't meet the required location criteria.
- **Enable MFA.** You can use conditional access policies to require a **MFA process**, before they can access applications. This action ensures that everyone who uses the application must pass an additional authentication challenge before accessing it.
- **Integrate with identity providers.** Azure AD supports external identity providers like Facebook, Microsoft accounts, Google, or enterprise identity providers. You can set up federation with identity providers so your external users can sign in with their existing social or enterprise accounts instead of creating a new account just for your application.
- **Create a self-service sign-up user flow.** With a self-service sign-up user flow, you can create a sign-up experience for external users who want to access your apps. As part of the sign-up flow, you can provide options for different social or enterprise identity providers, and collect information about the user. You can also Customize the onboarding experience for B2B guest users.

Design for Azure Active Directory Business to Customer

Azure AD B2C is a type of Azure AD tenant that you use to manage customer identities and their access to your applications.

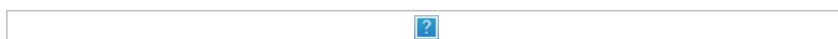
Use Azure AD B2C to:

- Securely authenticate your customers using their preferred identity providers.
- Capture sign in, preference, and conversion data for customers.
- Store custom attributes about customers to be leveraged by your applications
- Provide branded registration and sign in experiences.
- Provide separation of your organizational accounts your customer accounts.

How does Azure B2C work?

Azure AD B2C requires an Azure AD tenant. This tenant isn't the same your organization's Azure AD tenant. You use an Azure AD tenant to represent an organization. Your Azure AD B2C tenant represents the identities that are used for customer applications.

With your Azure AD B2C tenant in place, you must register your app. You use user flows to manage things like sign-ins and sign ups. Your Azure AD B2C tenant lets you create multiple types of user flows.



Best practices for Business to Customer

- **Configure user journeys by using policies.** A user journey is the path that you want people to take in your application to achieve their goal. For example, a user might want to make a new account, or update their profile. Azure AD B2C comes with preconfigured policies called **user flows**. You can reuse the same user flows across different applications. Reusing user flows creates a consistent user journey across all applications.
- **Use identity providers to let users sign in using their social identities.** There's a long **list of identity providers** and more are being added. Social providers include Amazon, Azure AD, Facebook, LinkedIn, Twitter, and Microsoft accounts.
- **Customize your user interface.** You can customize the pages in your user flow. Write your own HTML and CSS or use built-in templates called **page layout templates**.
- **Integrate with external user stores.** Azure AD B2C provides a directory that can hold 100 custom attributes per user. However, you can also integrate with external systems. For example, use Azure AD B2C for authentication, but delegate to an external customer relationship management (CRM) or customer loyalty database as the source of truth for customer data.
- **Third-party identity verification and proofing.** Use Azure AD B2C to facilitate identity verification and proofing by collecting user data, then passing it to a third-party system to perform validation, trust scoring, and approval for user account creation.

Take a few minutes to review the **WoodGrove Groceries tutorial**. WoodGrove Groceries is a live web application created by Microsoft to demonstrate several Azure AD B2C features.

With some basic knowledge on identity solutions, let's review our design choices.

Feature	Azure AD B2B	Azure AD B2C
Purpose	Collaborating with business partners from external organizations like suppliers, partners, vendors. Users appear as guest users in your directory. These users may or may not have managed IT.	Customers of your product. These users are managed in a separate Azure AD directory / tenant.
Users	Partner users acting on behalf of their company or employees of the company	Customers acting as themselves.
Profiles	Managed through access reviews, email verification, or access/deny lists.	Users manage their own profiles.
Discoverability	Partner users are discoverable and can find other users from their organization.	Customers are invisible to other users. Privacy and content are enforced.
		Consumer users with local

Identity providers supported	External users can collaborate using work accounts, school accounts, any email address, SAML and WS-Fed based identity providers, Gmail, and Facebook.	application accounts (any email address or user name), various supported social identities, and users with corporate and government-issued identities via SAML/WS-Fed based identity provider federation.
External user management	External users are managed in the same directory as employees but are typically annotated as guest users. Guest users can be managed the same way as employees, added to the same groups, and so on.	External users are managed in the Azure AD B2C directory. They're managed separately from the organization's employee and partner directory (if any).
Branding	Host/inviting organization's brand is used.	Fully customizable branding per application or organization.

Take a few minutes to decide if Azure B2B or Azure B2C would be required by your organization. Write down a few thoughts on how these options would be used.

Design for conditional access

Conditional Access is a tool that Azure Active Directory uses to allow (or deny) access to resources. During sign-in, Conditional Access examines who the user is, where the user is, and from what device the user is requesting access. Based on these signals Conditional Access can allow access, enforce multifactor authentication, or deny access.



Here are some example conditional access situations.

- Require multifactor authentication (MFA). MFA can be used to provide a secondary authentication for accessing certain apps. MFA can be selectively applied to certain users, like administrators, or just users coming from external networks.
- Require access to services only through approved client applications. For example, only allow users to access Office 365 services from a mobile device if they use approved client apps, like the Outlook mobile app.
- Require users to access applications only from managed devices. A managed device is a device that meets your standards for security and compliance.
- Block access from untrusted sources, such as access from unknown or unexpected locations.

Things to consider when using conditional access

- **Use for enabling multifactor authentication for more granular control.** Conditional Access provides a more granular multifactor authentication experience for users. For example, a user might not be challenged for second authentication factor if they're at a known location. However, they might be challenged for a second authentication factor if they're at an unexpected location.
- **Test by using report-only mode.** **Report-only mode** allows administrators to evaluate the impact of Conditional Access policies before enabling them in their environment. Report-only mode can help predict the number and names of users affected by common deployment initiatives. Use report-only mode to test blocking legacy authentication, requiring MFA, and implementing sign-in risk policies.
- **Exclude geographic areas from which you never expect a sign-in.** Azure Active Directory allows you to create named locations. Create a named location that includes all the geographic areas from which you would never expect a sign-in to occur. Then create a policy for all apps that blocks sign-in from that named location. Be sure to exempt your administrators from this policy.
- **Require managed devices.** The proliferation of supported devices to access your cloud resources helps to improve the productivity of your users. You probably don't want certain resources in your environment to be accessed by devices with an unknown protection level. For those resources, require that users can only access them using a managed device.
- **Require approved client applications.** Employees use their mobile devices for both personal and work tasks. In these scenarios, you must decide whether to manage the entire device or just the data on it. If managing only data and access, you can require only approved cloud apps. This can help to protect your corporate data.
- **Respond to potentially compromised accounts.** Three default policies can be enabled: require all users to register for MFA, require a password change for users who are high-risk, and require MFA for users with medium or high sign-in risk.
- **Block access.** Blocking access overrides all other assignments for a user and has the power to block your entire organization from signing on to your tenant. It can be used, for example, when you're migrating an app to Azure AD, but you aren't ready for anyone to sign-in yet. You can also block certain network locations from accessing your cloud apps or block apps using legacy authentication from accessing your tenant resources.
- **Block legacy authentication protocols.** Attackers exploit weaknesses in older protocols every day, particularly for password spray attacks. Configure Conditional Access to **block legacy protocols**.
- **Use the What If tool.** The **What If** tool helps you plan and troubleshoot your Conditional Access policies. The What If tool enables you to test your proposed Conditional Access policies before you implement them.

To use Conditional Access, you need an Azure AD Premium P1 or P2 license. If you have a Microsoft 365 Business Premium license, you also have access to Conditional Access features.

Design for identity protection

Identity Protection is a tool that allows organizations to accomplish three key tasks:

- **Automate the detection and remediation of identity-based risks.**
- **Investigate risks** using data in the portal.
- **Export risk detection data to other tools.**

The signals generated by and fed to Identity Protection, can be exported to other tools. For example, Conditional Access can make decisions based on your organization's policies. You could feed information to a security information and event management (SIEM) tool for further investigation.

Here's an example where a user attempts to sign in to Azure Active Directory. Azure AD calculates real-time sign in risk based on sign in properties. Identity protection then aggregates the user's risk. If the risk level meets the Identity Protection policy threshold the user may be blocked or challenged by MFA. If the user risk level is acceptable, they're granted access.



Risk policies

Risk policy detections in Azure AD Identity Protection include any identified suspicious actions related to user accounts in the directory. There are two risk policies that are evaluated: user risk and sign in risk.



User risk policies

A **user risk** represents the probability that a given identity or account is compromised. For example, the user's valid credentials have been leaked. These risks are calculated offline using Microsoft's internal and external threat intelligence sources. Here are some user risks that can be identified.

- **Leaked credentials.** Microsoft checks for leaked credentials from the dark web, paste sites, or other sources. These leaked credentials are checked against Azure AD users' current valid credentials for valid matches.
- **Azure AD threat intelligence.** This risk detection type indicates user activity that is unusual for the given user or is consistent with known attack patterns.

Microsoft's recommendation is to set the user risk policy threshold to **High**.

Sign-in risk policies

A **sign-in risk** represents the probability that a given sign-in (authentication request) isn't authorized by the identity owner. Sign-in risk can be calculated in real-time or calculated offline. Here are some sign-in risks that can be identified.

- **Anonymous IP address.** This risk detection type indicates sign-ins from an anonymous IP address. For example, a Tor browser or anonymized VPNs.
- **Atypical travel.** This risk detection type identifies two sign-ins originating from geographically distant location. Given past behavior, at least one of the locations may also be atypical for the user.
- **Malware linked IP address.** This risk detection type indicates sign-ins from IP addresses infected with malware. This malware is known to actively communicate with a bot server.
- **Password spray.** This risk detection is triggered when a password spray attack has been performed. Password spray is one of the most popular attacks. Bad actors try to defeat lockout and detection by trying many users against one password.

Microsoft's recommendation is to set the sign-in risk policy to **Medium and above** and allow self-remediation options. Self-remediation options, like password change and multifactor authentication, will have less impact than blocking users.

Design for access reviews

Over the time a user is employed by a company they may have several positions.

?

The Tailwind Traders CTO asks,

- As new employees join, how do we ensure they have the access they need to be productive?
- As users switch teams or leave the company, how do we make sure that their old access is removed?

Determine the purpose of the access review

You’re considering using **Azure Active Directory access reviews** to address the CTO’s concerns. An access review is a planned review of the access needs, rights, and history of user access. Access reviews mitigate risk by protecting, monitoring, and auditing access to critical assets.

Access reviews help ensure that the right people have the right access to the right resources. For example, access reviews could be used to review:

- User access to applications integrated with Azure AD for single sign-on (such as SaaS, line-of-business).
- Group memberships (synchronized to Azure AD, or created in Azure AD or Microsoft 365, including Microsoft Teams).
- Access Packages that group resources (groups, apps, and sites) into a single package to manage access.
- Azure AD roles and Azure Resource roles as defined in Privileged Identity Management (PIM).

Determine who will conduct the reviews

Access reviews are only as good as the person doing the reviewing. Selecting good reviewers is critical to your success. The creator of the access review decides who will conduct the review. This setting can't be changed once the review is started. Reviewers are represented by three personas:

- Resource Owners, who are the business owners of the resource.
- A set of individually selected delegates, as selected by the access reviews administrator.
- End users who will each self-attest to their need for continued access.

When creating an Access Review, administrators can choose one or more reviewers. All reviewers can start and carry out a review, choosing to grant users continued access to a resource or removing them.

Create an access review plan

Before implementing your access reviews, you should plan the types of reviews relevant to your organization. To do so, you’ll need to make business decisions about what you want to review and the actions to take based on those reviews.

For example, here’s an access review plan for Microsoft Dynamics.

Component	Value
Resources to review	Access to Microsoft Dynamics
Review frequency	Monthly
Who conducts the review	Dynamics business group program managers
Notification	Email 24 hours prior to review to alias Dynamics-PMs Include encouraging custom message to reviewers to secure their buy-in
Timeline	48 hours from notification
Automatic actions	Remove access from any account that has no interactive sign-in within 90 days by removing the user from the security group dynamics-access. Perform actions if not reviewed within timeline.
Manual actions	Reviewers may perform removals approval prior to automated action if desired.
Communications	Send internal (member) users who are removed an email explaining they’re removed and how to regain access.

Design service principals for applications

Azure managed identity is a feature of Azure Active Directory (Azure AD) that you can use free of charge. This feature automatically creates identities to allow apps to authenticate with Azure resources and services.

Tailwind is planning on moving applications from on-premises servers to Azure-hosted virtual machines (VMs). Now that you host the applications on VMs in Azure, you can use managed identities.

In this unit, you'll explore the managed identity feature. You'll learn how it works and what resources you can access in Azure.

What are managed identities in Azure?

A common challenge for developers is the management of secrets and credentials used to secure communication between different components making up a solution. Managed identities eliminate the need for developers to manage credentials. Managed identities provide an identity for applications to use when connecting to resources that support Azure Active Directory (Azure AD) authentication. Applications may use the managed identity to obtain Azure AD tokens. For example, an application may use a managed identity to access resources like Azure Key Vault where developers can store credentials in a secure manner or to access storage accounts.

A managed identity combines Azure AD authentication and Azure role-based access control (RBAC).

When you use managed identities, you don't need to rotate credentials or worry about expiring certifications. Azure handles credential rotation and expiration in the background. To configure an application to use a managed identity, you use the provided token to call the service.

When to use managed identities

When you work with managed identities, you should be familiar some common terms:

- Client ID: A unique ID that's linked to the Azure AD application and service principal that was created when you provisioned the identity.
- Object ID: The service principal object of the managed identity.
- Azure Instance Metadata Service: A REST API that's enabled when Azure Resource Manager provisions a VM. The endpoint is accessible only from within the VM.

Managed identities are available in all editions of Azure AD, including the Free edition included with an Azure subscription. Using it in App Service has no extra cost and requires no configuration, and it can be enabled or disabled on an app at any time.



Resources that support system assigned managed identities allow you to:

- Enable or disable managed identities at the resource level.
- Use RBAC roles to grant permissions.
- View create, read, update, delete (CRUD) operations in Azure Activity logs.
- View sign-in activity in Azure AD sign-in logs.

There are two types of managed identities:

- **System-assigned** Some Azure services allow you to enable a managed identity directly on a service instance. When you enable a system-assigned managed identity an identity is created in Azure AD that is tied to the lifecycle of that service instance. So when the resource is deleted, Azure automatically deletes the identity for you. By design, only that Azure resource can use this identity to request tokens from Azure AD.
- **User-assigned** You may also create a managed identity as a standalone Azure resource. You can create a user-assigned managed identity and assign it to one or more instances of an Azure service. In the case of user-assigned managed identities, the identity is managed separately from the resources that use it.

When to use system assigned managed identity:

- Workloads that are contained within a single Azure resource
- Workloads for which you need independent identities.

When to use User-assigned managed identity:

- Workloads that run on multiple resources and which can share a single identity.
- Workloads that need pre-authorization to a secure resource as part of a provisioning flow.
- Workloads where resources are recycled frequently, but permissions should stay consistent.

You can add managed identities to virtual machines (VMs) in Azure. You decide to run your stock-tracking application inside a VM that has an assigned managed identity. This setup will allow the app to use an Azure key vault to authenticate without having to store a username and password in code.

Now that your company has migrated your VM from on-premises to Azure, you can remove the hard-coded authentication details from the application code. You want to use the more secure managed identity token for access to Azure resources.

Vault authentication with managed identities for Azure resources

Your application requires service passwords, connection strings, and other secret configuration values to do its job. Storing and handling secret values is risky, and every usage introduces the possibility of leakage. Azure Key Vault, in combination with managed identities for Azure resources, enables your Azure web app to access secret configuration values easily and securely without needing to store any

secrets in your source control or configuration.

Azure Key Vault uses Azure Active Directory (Azure AD) to authenticate users and apps that try to access a vault. To grant your web app access to the vault, you first need to register your app with Azure Active Directory. Registering creates an identity for the app. After the app has an identity, you can assign vault permissions to it.

Apps and users authenticate to Key Vault using an Azure AD authentication token. Getting a token from Azure AD requires a secret or certificate because anyone with a token could use the app identity to access all the secrets in the vault. To access resources that are secured by an Azure AD tenant, the entity that requires access must be represented by a security principal. This requirement is true for both users (user principal) and applications (service principal). The security principal defines the access policy and permissions for the user/application in the Azure AD tenant. This enables core features such as authentication of the user/application during sign-in, and authorization during resource access.

Select application service principals

There are three types of service principal:

- **Application** - The type of service principal is the local representation, or application instance, of a global application object in a single tenant or directory. In this case, a service principal is a concrete instance created from the application object and inherits certain properties from that application object. A service principal is created in each tenant where the application is used and references the globally unique app object. The service principal object defines what the app can do in the specific tenant, who can access the app, and what resources the app can access.

When an application is given permission to access resources in a tenant (upon registration or consent), a service principal object is created. When you register an application using the Azure portal, a service principal is created automatically. You can also create service principal objects in a tenant using Azure PowerShell, Azure CLI, Microsoft Graph, and other tools.

- **Managed identity** - This type of service principal is used to represent a managed identity. Managed identities eliminate the need for developers to manage credentials. Managed identities provide an identity for applications to use when connecting to resources that support Azure AD authentication. When a managed identity is enabled, a service principal representing that managed identity is created in your tenant. Service principals representing managed identities can be granted access and permissions, but cannot be updated or modified directly.
- **Legacy** - This type of service principal represents a legacy app, which is an app created before app registrations were introduced or an app created through legacy experiences. A legacy service principal can have credentials, service principal names, reply URLs, and other properties that an authorized user can edit, but does not have an associated app registration. The service principal can only be used in the tenant where it was created.

Relationship between application objects and service principals

The application object is the global representation of your application for use across all tenants, and the service principal is the local representation for use in a specific tenant. The application object serves as the template from which common and default properties are derived for use in creating corresponding service principal objects.

An application object has:

- A 1:1 relationship with the software application
- A 1:many relationship with its corresponding service principal object(s).

A service principal must be created in each tenant where the application is used, enabling it to establish an identity for sign-in and/or access to resources being secured by the tenant. A single-tenant application has only one service principal (in its home tenant), created and consented for use during application registration. A multi-tenant application also has a service principal created in each tenant where a user from that tenant has consented to its use.

Applications represented in Azure AD

There are two representations of applications in Azure AD:

- **Application objects** - Although there are exceptions, application objects can be considered the definition of an application.
- **Service principals** - Can be considered an instance of an application. Service principals generally reference an application object, and one application object can be referenced by multiple service principals across directories.

Application objects describe the application to Azure AD and can be considered the definition of the application, allowing the service to know how to issue tokens to the application based on its settings. The application object will only exist in its home directory, even if it's a multi-tenant application supporting service principals in other directories

Service principals are what govern an application connecting to Azure AD and can be considered the instance of the application in your directory. For any given application, it can have at most one application object (which is registered in a "home" directory) and one or more service principal objects representing instances of the application in every directory in which it acts.



- Useful when Managed Identities cannot be used
- Often used to authenticate external applications to Azure resources

Design a user consent solution for applications

The Microsoft identity platform implements the OAuth 2.0 authorization protocol. This protocol is a method that a third-party app can use to access web-hosted resources on behalf of a user. The web-

hosted resources can define a set of permissions that you can use to implement functionality in smaller chunks. Developers can leverage one of two types of permissions supported by the Microsoft identity platform depending on the app scenario.

Knowing the different types of permissions supported in Azure AD applications will enable you to design an access strategy that works for your organization. You'll also learn about the different consent framework models and how they are used to obtain permissions from users for use in custom apps.

Types of permissions

Microsoft identity platform supports two types of permissions: delegated permissions and application permissions.

- Delegated permissions are used by apps that have a signed-in user present. These permissions are provided to the application by the user so the app can perform actions on their behalf. This doesn't give permissions to the app, instead the user is simply allowing the app to act on their behalf using their permissions.
- Application permissions are used by apps that run without a signed-in user present.

Effective permissions

Effective permissions are the permissions that your app will have when making requests to the target resource. It's important to understand the difference between the delegated and application permissions that your app is granted and its effective permissions when making calls to the target resource.

For delegated permissions, the effective permissions of your app are the intersection of the delegated permissions the app has been granted and the privileges of the currently signed-in user. In other words, the app can never have more privileges than the signed-in user. Within organizations, the privileges of the signed-in user may be determined by policy or by membership in one or more administrator roles.

For example, assume your app has been granted the User.ReadWrite.All delegated permission. This permission enables your app to be used to read and update the profile of every user in an organization. If the signed-in user is a global administrator, your app can update the profile of every user in the organization. However, if the signed-in user isn't in an administrator role, your app can update only the profile of the signed-in user. It can't update the profiles of other users in the organization because the user that it has permission to act on behalf of does not have those privileges.

For application permissions, the effective permissions of your app will be the full level of privileges implied by the permission. For example, an app that has the User.ReadWrite.All application permission can update the profile of every user in the organization.

Best practices for requesting permissions

When building an app that uses Azure AD to provide sign-in and access tokens for secured endpoints, there are a few good practices you should follow.

- Only ask for the permissions required for implemented app functionality. Don't request user consent for permissions that you haven't yet implemented for your application.
- In addition, when requesting permissions for app functionality, you should request the least-privileged access. For example, if an app analyzes a user's email but takes no action on the mailbox, you shouldn't request the more permissive Mail.ReadWrite when Mail.Read will work.
- Apps should gracefully handle scenarios where the user doesn't grant consent to the app when permissions are requested. In the case where an app doesn't receive an access token with the required permissions, it should explain the situation to the user with options on how to remedy the issue.

Microsoft recommends restricting user consent to allow users to consent only for app from verified publishers, and only for permissions you select. For apps which do not meet this policy, the decision-making process will be centralized with your organization's security and identity administrator team.

After end-user consent is disabled or restricted, there are several important considerations to ensure your organization stays secure while still allowing business critical applications to be used. These steps are crucial to minimize impact on your organization's support team and IT administrators, while preventing the use of unmanaged accounts in third-party applications.

Design for Azure key vault

Storing and handling secrets, encryption keys, and certificates directly is risky, and every usage introduces the possibility of unintentional data exposure. Azure Key Vault provides a secure storage area for managing all your app secrets so you can properly encrypt your data in transit or while it's being stored.

Azure Key Vault helps solve the following problems:

- **Secrets Management** - Azure Key Vault can be used to Securely store and tightly control access to tokens, passwords, certificates, API keys, and other secrets
- **Key Management** - Azure Key Vault can also be used as a Key Management solution. Azure Key Vault makes it easy to create and control the encryption keys used to encrypt your data.
- **Certificate Management** - Azure Key Vault is also a service that lets you easily enroll, manage, and deploy public and private Transport Layer Security/Secure Sockets Layer (TLS/SSL) certificates for use with Azure and your internal connected resources.

Azure Key Vault has two service tiers: Standard, which encrypts with a software key, and a Premium tier, which includes hardware security module(HSM)-protected keys.

Why use Azure Key Vault?

- Separation of sensitive app information from other configuration and code, reducing the risk of accidental leaks.
- Restricted secret access with access policies tailored to the apps and individuals that need them.
- Centralized secret storage, allowing required changes to happen in only one place.
- Access logging and monitoring to help you understand how and when secrets are accessed.

Key Vault allows you to securely access sensitive information from within your applications:

- Keys, secrets, and certificates are protected without having to write the code yourself and you're easily able to use them from your applications.
- You allow customers to own and manage their own keys, secrets, and certificates so you can concentrate on providing the core software features. In this way, your applications will not own the responsibility or potential liability for your customers' tenant keys, secrets, and certificates.
- Your application can use keys for signing and encryption yet keeps the key management external from your application.
- You can manage credentials like passwords, access keys, and sas tokens by storing them in Key Vault as secrets.
- Manage certificates. .

Design a solution using Keys and SAS tokens

A shared access signature (SAS) provides secure delegated access to resources in your storage account. With a SAS, you have granular control over how a client can access your data. For example:

- What resources the client may access.
- What permissions they have to those resources.
- How long the SAS is valid.

When to use a shared access signature

Use a SAS to give secure access to resources in your storage account to any client who does not otherwise have permissions to those resources.

A common scenario where a SAS is useful is a service where users read and write their own data to your storage account. In a scenario where a storage account stores user data, there are two typical design patterns:

- Clients upload and download data via a front-end proxy service, which performs authentication. This front-end proxy service allows the validation of business rules. But for large amounts of data, or high-volume transactions, creating a service that can scale to match demand may be expensive or difficult.
- A lightweight service authenticates the client as needed and then generates a SAS. Once the client application receives the SAS, it can access storage account resources directly. Access permissions are defined by the SAS and for the interval allowed by the SAS. The SAS mitigates the need for routing all data through the front-end proxy service.

Many real-world services may use a hybrid of these two approaches. For example, some data might be

processed and validated via the front-end proxy. Other data is saved and/or read directly using SAS.

Introduction



Suppose you work for Tailwind Traders that is moving its systems to Azure, with a mixture of IaaS and PaaS services. In its previous environment, the organization had several instances where application performance was degraded, or systems became entirely unavailable. There was an extended delay to identify and resolve the issues. This situation affected customers' ability to access their accounts and led to poor user satisfaction.

The organization wants to design a monitoring strategy that performs full-stack monitoring across all solutions that it uses. There should also be insights and alerting into the collected data. The organization wants to quickly identify and minimize poor performance and system failures in the future. The practice of continuous monitoring must include analysis of platform metrics and logs to get visibility into the health and performance of services that are part of the architecture.

In this module, you'll explore the monitoring solutions available in Azure. You'll assess Azure Monitor and its features such as Application Insights and Azure Monitor Logs to analyze infrastructure and application performance and availability.

Learning objectives

In this module, you'll be able to:

- Design for Azure Monitor data sources
- Design for Log Analytics
- Design for Azure workbooks and Insights
- Design for Azure Data Explorer
- Monitor resources for performance efficiency

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions. The module concepts are covered in:

Design Identity, Governance, and Monitoring Solutions

- Design a Solution for Logging and Monitoring.
 - Design a log routing solution
 - Recommend an appropriate level of logging
 - Recommend a monitoring tool(s) for a solution

Prerequisites

- Working experience with monitoring and logging cloud environments
- Conceptual knowledge of monitoring and logging

Design for Azure Monitor data sources

Azure Monitor is based on a **common monitoring data platform** that includes **Logs** and **Metrics**. Collecting data into this platform allows data from multiple resources to be analyzed together using a common set of tools in Azure Monitor. Monitoring data may also be sent to other locations to support certain scenarios, and some resources may write to other locations before they can be collected into Logs or Metrics. **Azure Monitor Logs** can store various data types each with their own structure. You can also perform complex analysis on logs data using log queries, which cannot be used for analysis of metrics data. Azure Monitor Logs can support near real-time scenarios, making them useful for alerting and fast detection of issues.

This unit describes the different sources of monitoring data collected by Azure Monitor in addition to the monitoring data created by Azure resources.

Consider Tailwind Traders Azure environment. What sources of monitoring data might they want to collect?

Identify data sources and access method

?

Sources of monitoring data from Azure applications can be organized into tiers, the highest tiers being your application itself and the lower tiers being components of the Azure platform. The method of accessing data from each tier varies. The application tiers are summarized in the table below, and the sources of monitoring data in each tier are presented in the following sections. Visit **Monitoring data locations in Azure** for a description of each data location and how you can access its data.

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

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

?

Azure tenant logging solutions

Telemetry related to your Azure tenant is collected from tenant-wide services such as Azure Active Directory.

?

Azure Active Directory audit logs

Azure Active Directory reporting contains the history of sign-in activity and audit trail of changes made within a particular tenant.

Destination	Description	Reference
Azure Monitor Logs	Configure Azure AD logs to be collected in Azure Monitor to analyze them with other monitoring data.	Integrate Azure AD logs with Azure Monitor logs
Azure Storage	Export Azure AD logs to Azure Storage for archiving.	Tutorial: Archive Azure AD logs to an Azure storage account
Azure Event Hub	Stream Azure AD logs to other locations using Event Hub.	Tutorial: Stream Azure Active Directory logs to an Azure event hub.
Azure Monitor partner integrations	Specialized integrations between Azure Monitor and other non-Microsoft monitoring platforms. Useful when you are already using one of the partners.	Extend Azure with solutions from partners

Operating system (guest) logging solutions

Compute resources in Azure, in other clouds, and on-premises have a guest operating system to monitor. With the installation of one or more agents, you can gather telemetry from the guest into Azure Monitor to analyze it with the same monitoring tools as the Azure services themselves.



Design for Log Analytics

Azure Monitor stores **log** data in a Log Analytics workspace, which is an Azure resource and a container where data is collected, aggregated, and serves as an administrative boundary. While you can deploy one or more workspaces in your Azure subscription, there are several considerations you should understand in order to ensure your initial deployment is following our guidelines to provide you with a cost effective, manageable, and scalable deployment meeting your organization's needs.



Data in a workspace is organized into tables, each of which stores different kinds of data and has its own unique set of properties based on the resource generating the data. Most data sources will write to their own tables in a Log Analytics workspace.

A Log Analytics workspace provides:

- A geographic location for data storage.
- Data isolation by granting different users access rights following one of our recommended design strategies.
- Scope for configuration of settings like **pricing tier**, **retention**, and **data capping**.

Workspaces are hosted on physical clusters. By default, the system is creating and managing these clusters. Customers that ingest more than 4TB/day are expected to create their own dedicated clusters for their workspaces - it enables them better control and higher ingestion rate.

Below is an overview of the design considerations, access control overview, and an understanding of the design implementations to consider for an IT organization like Tailwind traders.

Important considerations for an access control strategy

Identifying the number of workspaces, you need is influenced by one or more of the following requirements:

- You are a global company, and you need log data stored in specific regions for data sovereignty or compliance reasons.
- You are using Azure and you want to avoid outbound data transfer charges by having a workspace in the same region as the Azure resources it manages.
- You manage multiple departments or business groups. Each group should access their data, but not the data of others. Also, there is no business requirement for a consolidated cross department or business group view.

IT organizations today are modeled following either a centralized, decentralized, or an in-between hybrid of both structures. As a result, the following workspace deployment models have been commonly used to map to one of these organizational structures:

- **Centralized:** All logs are stored in a central workspace and administered by a single team, with Azure Monitor providing differentiated access per-team. In this scenario, it is easy to manage, search across resources, and cross-correlate logs. The workspace can grow significantly depending on the amount of data collected from multiple resources in your subscription, with additional administrative overhead to maintain access control to different users. This model is known as "hub and spoke".
- **Decentralized:** Each team has their own workspace created in a resource group they own and manage, and log data is segregated per resource. In this scenario, the workspace can be kept secure and access control is consistent with resource access, but it's difficult to cross-correlate logs. Users who need a broad view of many resources cannot analyze the data in a meaningful way.
- **Hybrid:** Security audit compliance requirements further complicate this scenario because many organizations implement both deployment models in parallel. This commonly results in a complex, expensive, and hard-to-maintain configuration with gaps in logs coverage.

Centralized logging can help you uncover hidden issues that might be difficult to track down. With Log Analytics, you can query and aggregate data across logs. This cross-source correlation can help you identify issues or performance problems that might not be evident when you review logs or metrics individually. Log Analytics receives monitoring data from your Azure resources and makes it available to consumers for analysis or visualization.

When using the Log Analytics agents to collect data, you need to understand the following in order to plan your agent deployment:

- To collect data from Windows agents, you can **configure each agent to report to one or more workspaces**, even while it is reporting to a System Center Operations Manager management group. The Windows agent can report up to four workspaces.
- The Linux agent does not support multi-homing and can only report to a single workspace.

If you are using System Center Operations Manager 2012 R2 or later:

- Each Operations Manager management group can be **connected to only one workspace**.
- Linux computers reporting to a management group must be configured to report directly to a Log Analytics workspace. If your Linux computers are already reporting directly to a workspace and you want to monitor them with Operations Manager, follow these steps to **report to an Operations Manager management group**.
- You can install the Log Analytics Windows agent on the Windows computer and have it report to both Operations Manager integrated with a workspace, and a different workspace.

Access control overview

With Azure role-based access control (Azure RBAC), you can grant users and groups only the amount of access they need to work with monitoring data in a workspace. This allows you to align with your IT organization operating model using a single workspace to store collected data enabled on all your resources. For example, you grant access to your team responsible for infrastructure services hosted on Azure virtual machines (VMs), and as a result they'll have access to only the logs generated by the VMs. This is following the new resource-context log model. The basis for this model is for every log record emitted by an Azure resource, it is automatically associated with this resource. Logs are forwarded to a central workspace that respects scoping and Azure RBAC based on the resources.

The data a user has access to is determined by a combination of factors that are listed in the following table. Each is described in the table below.

Factor	Description
Access mode	Method the user uses to access the workspace. Defines the scope of the data available and the access control mode that's applied.
Access control mode	Setting on the workspace that defines whether permissions are applied at the workspace or resource level.
Permissions	Permissions applied to individual or groups of users for the workspace or resource. Defines what data the user will have access to.
Table level Azure RBAC	Optional granular permissions that apply to all users regardless of their access mode or access control mode. Defines which data types a user can access.

Recommend an access mode

The access mode refers to how a user accesses a Log Analytics workspace and defines the scope of data they can access.

Users have two options for accessing the data:

- **Workspace-context:** You can view all logs in the workspace you have permission to. Queries in this mode are scoped to all data in all tables in the workspace. This is the access mode used when logs are accessed with the workspace as the scope, such as when you select **Logs** from the **Azure Monitor** menu in the Azure portal.
- **Resource-context:** When you access the workspace for a particular resource, resource group, or subscription, such as when you select **Logs** from a resource menu in the Azure portal, you can view logs for only resources in all tables that you have access to. Queries in this mode are scoped to only data associated with that resource. This mode also enables granular Azure RBAC.

The following table summarizes and compares the access modes:

Issue	Workspace-context	Resource-context
Who is each model intended for?	Central administration. Administrators who need to configure data collection and users who need access to a wide variety of resources. Also currently required for users who need to access logs for resources outside of Azure.	Application teams. Administrators of Azure resources being monitored.
		Read access to the resource. Explore Resource permissions in Manage access using
	Permissions to the	

What does a user require to view logs?	workspace. Explore Workspace permissions in Manage access using workspace permissions .	Azure permissions. Permissions can be inherited (such as from the containing resource group) or directly assigned to the resource. Permission to the logs for the resource will be automatically assigned.
What is the scope of permissions?	Workspace. Users with access to the workspace can query all logs in the workspace from tables that they have permissions to. Explore Table access control	Azure resource. User can query logs for specific resources, resource groups, or subscription they have access to from any workspace but can't query logs for other resources.
How can user access logs?	Start logs from Azure Monitor and Log Analytics workspaces. View the logs from Azure Monitor Workbooks .	Same as workspace-context, and you start logs from the Azure resource.

Scale and ingestion volume rate limit

Azure Monitor is a high scale data service that serves thousands of customers sending petabytes of data each month at a growing pace. Workspaces are not limited in their storage space and can grow to petabytes of data. There is no need to split workspaces due to scale.

To protect and isolate Azure Monitor customers and its backend infrastructure, there is a default ingestion rate limit that is designed to protect from spikes and floods situations. The rate limit default is about **6 GB/minute** and is designed to enable normal ingestion. For more details on ingestion volume limit measurement, explore [Azure Monitor service limits](#).

Customers that ingest less than 4TB/day will usually not meet these limits. Customers that ingest higher volumes or that have spikes as part of their normal operations shall consider moving to [dedicated clusters](#) where the ingestion rate limit could be raised.

When the ingestion rate limit is activated or get to 80% of the threshold, an event is added to the Operation table in your workspace. It is recommended to monitor it and create an alert. Explore more details in [data ingestion volume rate](#).

Recommendations



This scenario covers a single workspace design in your IT organization's subscription that is not constrained by data sovereignty or regulatory compliance or needs to map to the regions your resources are deployed within. It allows your organization's security and IT admin teams the ability to leverage the improved integration with Azure access management and more secure access control.

All resources, monitoring solutions, and Insights such as Application Insights and VM insights, supporting infrastructure and applications maintained by the different teams are configured to forward their collected log data to the IT organization's centralized shared workspace. Users on each team are granted access to logs for resources they have been given access to.

Once you have deployed your workspace architecture, you can enforce this on Azure resources with [Azure Policy](#). It provides a way to define policies and ensure compliance with your Azure resources, so they send all their resource logs to a particular workspace. For example, with Azure virtual machines or virtual machine scale sets, you can use existing policies that evaluate workspace compliance and report results or customize to remediate if non-compliant.

Design for Azure Workbooks and Insights

Workbooks provide a flexible canvas for data analysis and the creation of rich visual reports within the Azure portal. They allow you to tap into multiple data sources from across Azure and combine them into unified interactive experiences. Authors of workbooks can transform this data to provide insights into the availability, performance, usage, and overall health of the underlying components. For instance, analyzing performance logs from virtual machines to identify high CPU or low memory instances and displaying the results as a grid in an interactive report.

Workbooks are currently compatible with the following data sources:

- [Logs](#)
- [Metrics](#)
- [Azure Resource Graph](#)
- [Alerts \(Preview\)](#)
- [Workload Health](#)
- [Azure Resource Health](#)
- [Azure Data Explorer](#)

But the real power of workbooks is the ability to combine data from disparate sources within a single report. This allows for the creation of composite resource views or joins across resources enabling richer data and insights that would otherwise be impossible.

Customers use workbooks in several ways—exploring the usage of an app, going through a root cause analysis, and putting together an operational playbook, for example.

How would you leverage Azure workbooks for Tailwind Traders? What recommendations would you have based on their Azure environment and business needs?

Design for Azure Insights

The reputation of your organization depends on the performance, reliability, and security of its systems. It's critical to monitor your systems closely to identify any performance problems or attacks before they can affect users. For example, if your payment system is unable to process user transactions during a high-volume holiday sales period, your customers will likely lose confidence in your business. Designing insights as a part of your overall architecture will help identify performance issues.

Insights

Insights provide a customized monitoring experience for particular applications and services. They collect and analyze both logs and metrics. Here are just a few of the insights that are provided.

Insight	Description
Application Insights	Extensible Application Performance Management (APM) service to monitor your live web application on any platform.
Container insights	Monitors the performance of container workloads deployed to either Azure Container Instances or managed Kubernetes clusters hosted on Azure Kubernetes Service (AKS).
Cosmos DB insights	Provides a view of the overall performance, failures, capacity, and operational health of all your Azure Cosmos DB resources in a unified interactive experience.
Network insights	Provides a comprehensive view of health and metrics for all your network resource. The advanced search capability helps you identify resource dependencies, enabling scenarios like identifying resource that are hosting your website, by simply searching for your website name.
Resource Group insights	Triage and diagnose any problems your individual resources encounter, while offering context as to the health and performance of the resource group as a whole.
	Provides comprehensive monitoring of

Storage insights

your Azure Storage accounts by delivering a unified view of your Azure Storage services performance, capacity, and availability.

VM insights

Monitors your Azure virtual machines (VM) and virtual machine scale sets at scale. It analyzes the performance and health of your Windows and Linux VMs, and monitors their processes and dependencies on other resources and external processes.

Key Vault insights

Provides comprehensive monitoring of your key vaults by delivering a unified view of your Key Vault requests, performance, failures, and latency.

Azure Cache for Redis insights

Provides a unified, interactive view of overall performance, failures, capacity, and operational health.

Use Application Insights to:

- Analyze and address issues and problems that affect your application's health and performance.
- Improve your application's development lifecycle.
- Measure your user experience and analyze users' behavior.

Application Insights is aimed at the development team, to help you understand how your app is performing and how it's being used. It monitors:

- **Request rates, response times, and failure rates** - Find out which pages are most popular, at what times of day, and where your users are. If your response times and failure rates go high when there are more requests, then perhaps you have a resourcing problem.
- **Dependency rates, response times, and failure rates** - Find out whether external services are slowing you down.
- **Exceptions** - Analyze the aggregated statistics, or pick specific instances and drill into the stack trace and related requests. Both server and browser exceptions are reported.
- **Page views and load performance** - reported by your users' browsers.
- **AJAX calls** from web pages - rates, response times, and failure rates.
- **User and session counts.**
- **Performance counters** from your Windows or Linux server machines, such as CPU, memory, and network usage.
- **Host diagnostics** from Docker or Azure.
- **Diagnostic trace logs** from your app - so that you can correlate trace events with requests.
- **Custom events and metrics** that you write yourself in the client or server code, to track business events such as items sold or games won.

Use Azure Monitor VM insights to:

- View the health and performance of your VMs
- Monitor your VMs at-scale across multiple subscriptions and resource groups.
- Want a topology view that shows the processes, and network connection details of your VMs and scale sets.
- Insights supports Azure virtual machines and scale sets
- Hybrid virtual machines connected with Azure Arc
- On-premises virtual machines
- Virtual machines hosted in another cloud environment

Use Azure Monitor container insights to:

- View the health and performance of your Kubernetes workloads at-scale across multiple subscriptions and resource groups.
- Want visibility into memory and processor performance metrics from controllers, nodes, and containers.

- Want view and store container logs for real time and historical analysis.
- Identify AKS containers that are running on the node and their average processor and memory utilization. This knowledge can help you identify resource bottlenecks.
- Identify processor and memory utilization of container groups and their containers hosted in Azure Container Instances.
- Identify where the container resides in a controller or a pod. This knowledge can help you view the controller's or pod's overall performance.
- Review the resource utilization of workloads running on the host that are unrelated to the standard processes that support the pod.
- Understand the behavior of the cluster under average and heaviest loads. This knowledge can help you identify capacity needs and determine the maximum load that the cluster can sustain.
- Configure alerts to proactively notify you or record it when CPU and memory utilization on nodes or containers exceed your thresholds, or when a health state change occurs in the cluster at the infrastructure or nodes health rollup.
- Integrate with **Prometheus** to view application and workload metrics it collects from nodes and Kubernetes using **queries** to create custom alerts, dashboards, and perform detailed analysis.
- Monitor container workloads **deployed to AKS Engine** on-premises and **AKS Engine on Azure Stack**.
- Monitor container workloads **deployed to Azure Red Hat OpenShift**.
- Monitor container workloads **deployed to Azure Arc enabled Kubernetes (preview)**.

Design for Azure Data Explorer

Azure Data Explorer is a fast and highly scalable data exploration service for log and telemetry data. It helps you handle the many data streams emitted by modern software, so you can collect, store, and analyze data. Azure Data Explorer is ideal for analyzing large volumes of diverse data from any data source, such as websites, applications, IoT devices, and more. This data is used for diagnostics, monitoring, reporting, machine learning, and additional analytics capabilities.

Below is an example of a hybrid end-to-end monitoring solution integrated with Azure Sentinel and Azure Monitor for ingesting streamed and batched logs from diverse sources, on-premises, or any cloud within an enterprise ecosystem. This could be a solution used in Tailwind Traders architecture to monitor various logs.



Combine features provided by Azure Sentinel and Azure Monitor with Azure Data Explorer to build a flexible and cost-optimized end-to-end monitoring solution. Below are some examples:

- Use Azure Monitor's native capabilities for IT asset monitoring, dashboarding, and alerting so you can ingest logs from VMs, services, and so on.
- Use Azure Data Explorer for full flexibility and control in all aspects for all types of logs in the following scenarios:
 - No out of the box features provided by Azure Sentinel and Azure Monitor SaaS solutions such as application trace logs.
 - Greater flexibility for building quick and easy near-real-time analytics dashboards, granular role-based access control, **time series analysis**, pattern recognition, **anomaly detection and forecasting**, and **machine learning**. Azure Data Explorer is also well integrated with ML services such as Databricks and Azure Machine Learning. This integration allows you to build models using other tools and services and export ML models to Azure Data Explorer for scoring data.
 - Longer data retention is required in cost effective manner.
 - Centralized repository is required for different types of logs. Azure Data Explorer, as a unified big data analytics platform, allows you to build advanced analytics scenarios.

Monitor resources for performance efficiency (optional)

Troubleshooting an application's performance requires monitoring and reliable investigation. Issues in performance can arise from database queries, connectivity between services, under-provision resources, or memory leaks in code.

Continuously monitoring services and checking the health state of current workloads is key in maintaining the overall performance of the workload. An overall monitoring strategy consider these factors:

- Scalability
- Resiliency of the infrastructure, application, and dependent services
- Application and infrastructure performance

What to consider when defining a monitoring strategy

Here is a list to consider ensuring you are monitoring your workloads with performance and scalability in mind:

- Enable and capture telemetry throughout your application to build and visualize end-to-end transaction flows for the application.
- Review metrics from Azure services such as CPU and memory utilization, bandwidth information, current storage utilization, and more.
- Use resource and platform logs to get information about what events occur and under which conditions.
- For scalability, review the metrics to determine how to provision resources dynamically and scale with demand.
- In the collected logs and metrics identify signs that might make a system or its components suddenly become unavailable.
- Use log aggregation technology to gather information across all application components.
- Store logs and key metrics of critical components for statistical evaluation and predicting trends.
- Identify antipatterns in the code.

Follow these questions to assess the workload at a deeper level.

Assessment

Are application logs and events correlated across all application components?

Are you collecting Azure Activity Logs within the log aggregation tool?

Are application and resource level logs aggregated in a single data sink, or is it possible to cross-query events at both levels?

Description

Correlate logs and events for subsequent interpretation. This will give you visibility into end-to-end transaction flows.

Collect platform metrics and logs to get visibility into the health and performance of services that are part of the architecture.

Implement a unified solution to aggregate and query application and resource level logs, such as Azure Log Analytics.

Introduction



Tailwind Traders is a fictitious home improvement retailer. It operates retail hardware stores across the globe and online. Tailwind Traders specializes in competitive pricing, fast shipping, and a large range of items. It's looking at cloud technologies to improve business operations and support growth into new markets. By moving to the cloud, the company plans to enhance its shopping experience to further differentiate itself from competitors.

As the Tailwind Traders Enterprise IT team prepares to define the strategy to migrate some of company's workloads to Azure, it must identify the required networking components and design a network infrastructure necessary to support them. Considering the global scope of its operations, Tailwind Traders will be using multiple Azure regions to host its applications. Most of these applications have dependencies on infrastructure and data services, which will also reside in Azure. Internal applications migrated to Azure must remain accessible to Tailwind Traders users. Internet-facing applications migrated to Azure must remain accessible to any external customer.

Learning objectives

In this module, you will:

- Recommend a network architecture solution based on workload requirements
- Design for on-premises connectivity to Azure Virtual Networks
- Design for Azure network connectivity services
- Design for application delivery services
- Design for application protection services

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions. The module concepts are covered in:

Design Network Solutions

- Recommend a network architecture solution based on workload requirements
- Recommend a connectivity solution that connects Azure resources to the Internet
- Recommend a connectivity solution that connects Azure resources to on-premises networks
- Optimize network performance for applications
- Recommend a solution to optimize network security
- Recommend a load balancing and routing solution

Prerequisites

- Working experience with enterprise networking.
- Conceptual knowledge of software defined networking and hybrid connectivity.

Introduction



Tailwind Traders is a fictitious home improvement retailer. It operates retail hardware stores across the globe and online. Tailwind Traders specializes in competitive pricing, fast shipping, and a large range of items. It's looking at cloud technologies to improve business operations and support growth into new markets. By moving to the cloud, the company plans to enhance its shopping experience to further differentiate itself from competitors.

As the Tailwind Traders Enterprise IT team prepares to define the strategy to migrate some of company's workloads to Azure, it must identify the required networking components and design a network infrastructure necessary to support them. Considering the global scope of its operations, Tailwind Traders will be using multiple Azure regions to host its applications. Most of these applications have dependencies on infrastructure and data services, which will also reside in Azure. Internal applications migrated to Azure must remain accessible to Tailwind Traders users. Internet-facing applications migrated to Azure must remain accessible to any external customer.

Learning objectives

In this module, you will:

- Recommend a network architecture solution based on workload requirements
- Design for on-premises connectivity to Azure Virtual Networks
- Design for Azure network connectivity services
- Design for application delivery services
- Design for application protection services

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions. The module concepts are covered in:

Design Network Solutions

- Recommend a network architecture solution based on workload requirements
- Recommend a connectivity solution that connects Azure resources to the Internet
- Recommend a connectivity solution that connects Azure resources to on-premises networks
- Optimize network performance for applications
- Recommend a solution to optimize network security
- Recommend a load balancing and routing solution

Prerequisites

- Working experience with enterprise networking.
- Conceptual knowledge of software defined networking and hybrid connectivity.

Recommend a network architecture solution based on workloads

Tailwind Traders currently runs its workloads on-premises, in its datacenter. As the Tailwind Traders Enterprise IT team prepares to define the strategy to migrate some of company's workloads to Azure, it must identify the required networking components and design a network infrastructure necessary to support them. Considering the global scope of its operations, Tailwind Traders will be using multiple Azure regions to host its applications. Most of these applications have dependencies on infrastructure and data services, which will also reside in Azure. Internal applications migrated to Azure must remain accessible to Tailwind Traders users. Internet-facing applications migrated to Azure must remain accessible to any external customer.

In this module you will learn how the networking services in Azure provide a variety of networking capabilities that can be used together or separately to meet your requirements. Take note of which services you think Tailwind Traders will need for their production environment in Azure.

- **Connectivity services:** Connect Azure resources and on-premises resources using any or a combination of these networking services in Azure - Virtual Network (VNet), Virtual WAN, ExpressRoute, VPN Gateway, Virtual network NAT Gateway, Azure DNS, Peering service, and Azure Bastion.
- **Application protection services:** Protect your applications using any or a combination of these networking services in Azure - Load Balancer, Private Link, DDoS protection, Firewall, Network Security Groups, Web Application Firewall, and Virtual Network Endpoints.
- **Application delivery services:** Deliver applications in the Azure network using any or a combination of these networking services in Azure - Content Delivery Network (CDN), Azure Front Door Service, Traffic Manager, Application Gateway, Internet Analyzer, and Load Balancer.

Gather Network Requirements

Naming

All Azure resources have a name. The name must be unique within a scope, that may vary for each resource type. For example, the name of a virtual network must be unique within a resource group, but can be duplicated within a subscription or Azure region. Defining a naming convention that you can use consistently when naming resources is helpful when managing several network resources over time.

Regions

All Azure resources are created in an Azure region and subscription. A resource can only be created in a virtual network that exists in the same region and subscription as the resource. You can however, connect virtual networks that exist in different subscriptions and regions. When deciding which region(s) to deploy resources in, consider where consumers of the resources are physically located:

- Consumers of resources typically want the lowest network latency to their resources.
- Do you have data residency, sovereignty, compliance, or resiliency requirements? If so, choosing the region that aligns to the requirements is critical.
- Do you require resiliency across Azure Availability Zones within the same Azure region for the resources you deploy? You can deploy resources, such as virtual machines (VM) to different availability zones within the same virtual network. Not all Azure regions support availability zones however.

Subscriptions

You can deploy as many virtual networks as required within each subscription, up to the limit. Some organizations have different subscriptions for different departments, for example.

Segmentation

You can create multiple virtual networks per subscription and per region. You can create multiple subnets within each virtual network. The considerations that follow help you determine how many virtual networks and subnets you require:

Virtual networks A virtual network is a virtual, isolated portion of the Azure public network. Each virtual network is dedicated to your subscription. Things to consider when deciding whether to create one virtual network, or multiple virtual networks in a subscription:

- Do any organizational security requirements exist for isolating traffic into separate virtual networks? You can choose to connect virtual networks or not. If you connect virtual networks, you can implement a network virtual appliance, such as a firewall, to control the flow of traffic between the virtual networks.
- Do any organizational requirements exist for isolating virtual networks into separate subscriptions or regions?
- A network interface enables a VM to communicate with other resources. Each network interface has one or more private IP addresses assigned to it. How many network interfaces and private IP

addresses do you require in a virtual network? There are limits to the number of network interfaces and private IP addresses that you can have within a virtual network.

- Do you want to connect the virtual network to another virtual network or on-premises network? You may choose to connect some virtual networks to each other or on-premises networks, but not others. Each virtual network that you connect to another virtual network, or on-premises network, must have a unique address space. Each virtual network has one or more public or private address ranges assigned to its address space. An address range is specified in classless internet domain routing (CIDR) format, such as 10.0.0.0/16. Learn more about address ranges for virtual networks.
- Do you have any organizational administration requirements for resources in different virtual networks? If so, you might separate resources into separate virtual network to simplify permission assignment to individuals in your organization or to assign different policies to different virtual networks.
- When you deploy some Azure service resources into a virtual network, they create their own virtual network.

Subnets A virtual network can be segmented into one or more subnets up to the limits. Things to consider when deciding whether to create one subnet, or multiple virtual networks in a subscription:

- Each subnet must have a unique address range, specified in CIDR format, within the address space of the virtual network. The address range cannot overlap with other subnets in the virtual network.
- If you plan to deploy some Azure service resources into a virtual network, they may require, or create, their own subnet, so there must be enough unallocated space for them to do so. For example, if you connect a virtual network to an on-premises network using an Azure VPN Gateway, the virtual network must have a dedicated subnet for the gateway. Learn more about gateway subnets.
- Azure routes network traffic between all subnets in a virtual network, by default. You can override Azure's default routing to prevent Azure routing between subnets, or to route traffic between subnets through a network virtual appliance, for example. If you require that traffic between resources in the same virtual network flow through a network virtual appliance (NVA), deploy the resources to different subnets. Learn more in security.
- You can limit access to Azure resources such as an Azure storage account or Azure SQL Database, to specific subnets with a virtual network service endpoint. Further, you can deny access to the resources from the internet. You may create multiple subnets, and enable a service endpoint for some subnets, but not others. Learn more about service endpoints, and the Azure resources you can enable them for.
- You can associate zero or one network security group to each subnet in a virtual network. You can associate the same, or a different, network security group to each subnet. Each network security group contains rules, which allow or deny traffic to and from sources and destinations. Learn more about network security groups.

Security

You can filter network traffic to and from resources in a virtual network using network security groups and network virtual appliances. You can control how Azure routes traffic from subnets. You can also limit who in your organization can work with resources in virtual networks.

Traffic filtering

- You can filter network traffic between resources in a virtual network using a network security group, an NVA that filters network traffic, or both. When using an NVA, you also create custom routes to route traffic from subnets to the NVA. Learn more about traffic routing.
- A network security group contains several default security rules that allow or deny traffic to or from resources. A network security group can be associated to a network interface, the subnet the network interface is in, or both. To simplify management of security rules, it's recommended that you associate a network security group to individual subnets, rather than individual network interfaces within the subnet, whenever possible.
- If different VMs within a subnet need different security rules applied to them, you can associate the network interface in the VM to one or more application security groups. A security rule can specify an application security group in its source, destination, or both. That rule then only applies to the network interfaces that are members of the application security group. Learn more about network security groups and application security groups.
- Azure creates several default security rules within each network security group. One default rule allows all traffic to flow between all resources in a virtual network. To override this behavior, use network security groups, custom routing to route traffic to an NVA, or both. It's recommended that you familiarize yourself with all of Azure's default security rules and understand how network security group rules are applied to a resource.

Traffic routing Azure creates several default routes for outbound traffic from a subnet. You can override Azure's default routing by creating a route table and associating it to a subnet. Common reasons for overriding Azure's default routing are:

- Because you want traffic between subnets to flow through an NVA. To learn more about how to configure route tables to force traffic through an NVA.
- Because you want to force all internet-bound traffic through an NVA, or on-premises, through an Azure VPN gateway. Forcing internet traffic on-premises for inspection and logging is often referred to as forced tunneling. Learn more about how to configure forced tunneling.

Best practice: Plan IP addressing

When you create virtual networks as part of your migration, it's important to plan out your virtual network IP address space.

You should assign an address space that isn't larger than a CIDR range of /16 for each virtual network. Virtual networks allow for the use of 65,536 IP addresses. Assigning a smaller prefix than /16, such as a /15, which has 131,072 addresses, will result in the excess IP addresses becoming unusable elsewhere. It's important not to waste IP addresses, even if they're in the private ranges defined by RFC 1918.

Other tips for planning are:

- The virtual network address space shouldn't overlap with on-premises network ranges.
- Overlapping addresses can cause networks that can't be connected, and routing that doesn't work properly.
- If networks overlap, you'll need to redesign the network.
- If you absolutely can't redesign the network, network address translation (NAT) can help but should be avoided or limited as much as possible.

Best practice: Implement a hub and spoke network topology

A hub and spoke network topology isolates workloads while sharing services, such as identity and security. The hub is an Azure virtual network that acts as a central point of connectivity. The spokes are virtual networks that connect to the hub virtual network by using peering. Shared services are deployed in the hub, while individual workloads are deployed as spokes.

Consider the following:

- Implementing a hub and spoke topology in Azure centralizes common services, such as connections to on-premises networks, firewalls, and isolation between virtual networks. The hub virtual network provides a central point of connectivity to on-premises networks, and a place to host services used by workloads hosted in spoke virtual networks.
- A hub and spoke configuration is typically used by larger enterprises. Smaller networks might consider a simpler design to save on costs and complexity.
- You can use spoke virtual networks to isolate workloads, with each spoke managed separately from other spokes. Each workload can include multiple tiers, and multiple subnets that are connected with Azure load balancers.
- You can implement hub and spoke virtual networks in different resource groups, and even in different subscriptions. When you peer virtual networks in different subscriptions, the subscriptions can be associated to the same, or different, Azure Active Directory (Azure AD) tenants. This allows for decentralized management of each workload, while sharing services maintained in the hub network.

Best practice: Design subnets

To provide isolation within a virtual network, you segment it into one or more subnets, and allocate a portion of the virtual network's address space to each subnet.

- You can create multiple subnets within each virtual network.
- By default, Azure routes network traffic between all subnets in a virtual network.
- Your subnet decisions are based on your technical and organizational requirements.
- You create subnets by using CIDR notation.

When you're deciding on network range for subnets, be aware that Azure retains five IP addresses from each subnet that can't be used. For example, if you create the smallest available subnet of /29 (with eight IP addresses), Azure will retain five addresses. In this case, you only have three usable addresses that can be assigned to hosts on the subnet. For most cases, use /28 as the smallest subnet.

Example:

The table shows an example of a virtual network with an address space of 10.245.16.0/20 segmented

into subnets, for a planned migration.

Subnet	CIDR	Addresses	Usage
DEV-FE-EUS2	10.245.16.0/22	1019	Front-end or web-tier VMs
DEV-APP-EUS2	10.245.20.0/22	1019	Application-tier VMs
DEV-DB-EUS2	10.245.24.0/23	507	Database VMs

Design for on-premises connectivity to Azure networks

For a successful migration, it's critical to connect on-premises corporate networks to Azure. This creates an always-on connection known as a hybrid-cloud network, where services are provided from the Azure cloud to corporate users.

This section describes services that provide connectivity between Azure resources, connectivity from an on-premises network to Azure resources, and branch to branch connectivity in Azure - Virtual Network (VNet), ExpressRoute, VPN Gateway, Virtual WAN, Virtual network NAT Gateway, Azure DNS, Azure Peering service, and Azure Bastion.

Let's compare the options for connecting an on-premises network to an Azure Virtual Network (VNet). For each option, a more detailed reference architecture is available.

VPN connection

A **VPN gateway** is a type of virtual network gateway that sends encrypted traffic between an Azure virtual network and an on-premises location. The encrypted traffic goes over the public Internet. There are different configurations available for VPN Gateway connections, such as, site-to-site, point-to-site, or VNet-to-VNet.

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

Benefits

- Simple to configure.
- Much higher bandwidth available; up to 10 Gbps depending on the VPN Gateway SKU.

Challenges

- Requires an on-premises VPN device.

Reference architecture

Hybrid network with VPN gateway



Azure ExpressRoute connection

ExpressRoute connections use a private, dedicated connection through a third-party connectivity provider. This connection is private. Traffic does not go over the internet. The private connection extends your on-premises network into Azure.

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

Benefits

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

Challenges

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

Reference architecture

- **Hybrid network with ExpressRoute**



ExpressRoute with VPN failover

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

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

Benefits

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

bandwidth network.

Challenges

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

Reference architecture

- [Hybrid network with ExpressRoute and VPN failover](#)



Hub-spoke network topology

A hub-spoke network topology is a way to isolate workloads while sharing services such as identity and security. The hub is a virtual network (VNet) in Azure that acts as a central point of connectivity to your on-premises network. The spokes are VNets that peer with the hub. Shared services are deployed in the hub, while individual workloads are deployed as spokes.

Reference architectures

- [Hub-spoke topology](#)

Hub-spoke network topology with Azure Virtual WAN

A hub-spoke architecture can be achieved two ways: a customer-managed hub infrastructure, or a Microsoft-managed hub infrastructure. In either case, spokes are connected to the hub using virtual network peering.

The hub is a virtual network in Azure that acts as a central point of connectivity to your on-premises network. The spokes are virtual networks that peer with the hub and can be used to isolate workloads. Traffic flows between the on-premises data center(s) and the hub through an ExpressRoute or VPN gateway connection. The main differentiator of this approach is the use of Azure Virtual WAN (VWAN) to replace hubs as a managed service.

Azure Virtual WAN is a networking service that provides optimized and automated branch connectivity to, and through, Azure. Azure regions serve as hubs that you can choose to connect your branches to. You can leverage the Azure backbone to also connect branches and enjoy branch-to-VNet connectivity. Azure Virtual WAN brings together many Azure cloud connectivity services such as site-to-site VPN, ExpressRoute, point-to-site user VPN into a single operational interface. Connectivity to Azure VNets is established by using virtual network connections.

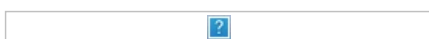
This architecture includes the benefits of standard hub-spoke network topology and introduces new benefits:

- **Less operational overhead** by replacing existing hubs with a fully managed VWAN service.
- **Cost savings** by using a managed service and removing the necessity of network virtual appliance.
- **Improved security** by introducing centrally managed secured Hubs with Azure Firewall and VWAN to minimize security risks related to misconfiguration.
- **Separation of concerns** between central IT (SecOps, InfraOps) and workloads (DevOps).

Typical uses for this architecture include cases in which:

- Connectivity among workloads requires central control and access to shared services.
- An enterprise requires central control over security aspects, such as a firewall, and requires segregated management for the workloads in each spoke.

Advantages



This diagram illustrates a few of the advantages that this architecture can provide:

- A full meshed hubs among Azure Virtual Networks
- Branch to Azure connectivity
- Branch to Branch connectivity
- Mixed use of VPN and Express Route
- Mixed use of user VPN to the site
- VNET to VNET connectivity

Design for Azure network connectivity services

Azure Virtual Network (VNet) is the fundamental building block for your private network in Azure. You can use a VNets to:

- **Communicate between Azure resources:** You can deploy VMs, and several other types of Azure resources to a virtual network, such as Azure App Service Environments, the Azure Kubernetes Service (AKS), and Azure Virtual Machine Scale Sets.
- **Communicate between each other:** You can connect virtual networks to each other, enabling resources in either virtual network to communicate with each other, using virtual network peering. The virtual networks you connect can be in the same, or different, Azure regions.
- **Communicate to the internet:** All resources in a VNet can communicate outbound to the internet, by default. You can communicate inbound to a resource by assigning a public IP address or a public Load Balancer. You can also use [Public IP addresses](#) or public [Load Balancer](#) to manage your outbound connections.
- **Communicate with on-premises networks:** You can connect your on-premises computers and networks to a virtual network using [VPN Gateway](#) or [ExpressRoute](#).

When you design a network from bottom up, you gather some basic information. This information could be number of hosts, network devices, number of subnets, routing between subnets, isolation domains such as VLANs. This information helps in sizing the network and security devices as well creating the architecture to support applications and services.

When you plan to deploy your applications and services in Azure, you will start by creating a logical boundary in Azure, which is called a virtual network. This virtual network is akin to a physical network boundary. As it is a virtual network, you don't need physical gear but still have to plan for the logical entities such as IP addresses, IP subnets, routing, and policies.

When you create a virtual network in Azure, it's pre-configured with an IP range (10.0.0.0/16). This range isn't fixed, you can define your own IP range. You can define both IPv4 and IPv6 address ranges. IP ranges defined for the virtual network are not advertised to Internet. You can create multiple subnets from your IP range. These subnets will be used to assign IP addresses to virtual network interfaces (vNICs). First four IP addresses from each subnet are reserved and can't be used for IP allocation. There is no concept of VLANs in a public cloud. However, you can create isolation within a virtual network based on your defined subnets.

You can create one large subnet encompassing all the virtual network address space or choose to create multiple subnets.

A virtual network is a virtual, isolated portion of the Azure public network. Each virtual network is dedicated to your subscription. Things to consider when deciding whether to create one virtual network, or multiple virtual networks in a subscription:

- Do any organizational security requirements exist for isolating traffic into separate virtual networks? You can choose to connect virtual networks or not. If you connect virtual networks, you can implement a network virtual appliance, such as a firewall, to control the flow of traffic between the virtual networks. For more information, explore [security](#) and [connectivity](#).
- Do any organizational requirements exist for isolating virtual networks into separate [subscriptions](#) or [regions](#)?
- A [network interface](#) enables a VM to communicate with other resources. Each network interface has one or more private IP addresses assigned to it. How many network interfaces and [private IP addresses](#) do you require in a virtual network? There are [limits](#) to the number of network interfaces and private IP addresses that you can have within a virtual network.

[Here are more questions to consider](#) when designing an Azure virtual network.

Design network segmentation

Segmentation is a model in which you take your networking footprint and create software defined perimeters using tools available in Microsoft Azure. You then set rules that govern the traffic from/to these perimeters so that you can have different security postures for various parts of your network. When you place different applications (or parts of a given application) into these perimeters, you can govern the communication between these segmented entities. If a part of your application stack is compromised, you'll be better able to contain the impact of this security breach and prevent it from laterally spreading through the rest of your network. This ability is a key principle associated with the [Zero Trust model published by Microsoft](#) that aims to bring world-class security thinking to your organization.

When you operate on Azure, you have a wide and diverse set of segmentation options available to help you be protected.

1. **Subscription:** Subscriptions are a high-level construct, which provides platform powered separation between entities. It's intended to carve out boundaries between large organizations within a company. Communication between resources in different subscriptions needs to be

explicitly provisioned.

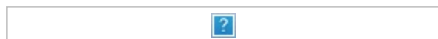
2. **Virtual Network:** Virtual networks are created within a subscription in private address spaces. The networks provide network-level containment of resources, with no traffic allowed by default between any two virtual networks. Like subscriptions, any communication between virtual networks needs to be explicitly provisioned.
3. **Network Security Groups (NSG):** NSGs are access control mechanisms for controlling traffic between resources within a virtual network. An NSG also controls traffic with external networks, such as the internet, other virtual networks, and so on. NSGs can take your segmentation strategy to a granular level by creating perimeters for a subnet, group of VMs, or even a single virtual machine.
4. **Application Security Groups (ASGs):** An ASG allows you to group a set of VMs under an application tag. Once an ASG is created and VMs are assigned to it, the ASG can be used as a source or target in the NSG to simplify management.
5. **Azure Firewall:** Azure Firewall is a cloud native stateful Firewall as a service. This firewall can be deployed in your virtual networks or in **Azure Virtual WAN** hub deployments for filtering traffic that flows between cloud resources, the Internet, and on-premise. You create rules or policies (using Azure Firewall or **Azure Firewall Manager**) specifying allow/deny traffic using layer 3 to layer 7 controls. You can also filter traffic that goes to the internet using both Azure Firewall and third parties. Direct some or all traffic through third-party security providers for advanced filtering and user protection.

The following three patterns are common when it comes to organizing your workload in Azure from a networking perspective. Each of these patterns provides a different type of isolation and connectivity. Choosing which model works best for your organization is a decision you should make based on your organization's needs. With each of these models, we describe how segmentation can be done using the above Azure Networking services.

Pattern 1: Single virtual network

In this pattern, all the components of your workload or, in some cases, your entire IT footprint is put inside a single virtual network. This pattern is possible if you're operating solely in a single region since a virtual network can't span multiple regions.

The entities you would most likely use for creating segments inside this virtual network are NSGs, potentially using ASGs to simplify administration. The image below is an example of a segmented virtual network.

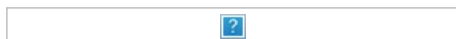


In this setup, you have Subnet1, where you placed your database workloads, and Subnet2, where you've placed your web workloads. You can put NSGs that specify that Subnet1 can talk only with Subnet2, and that Subnet2 can talk to the Internet. You can also take this concept further in the presence of many workloads. Carve out subnets that, for example, won't allow one workload to communicate to the backend of another workload.

Although we used NSGs to illustrate how subnet traffic can be governed, you can also enforce this segmentation by using a Network Virtualized Appliance from Azure Marketplace or Azure Firewall.

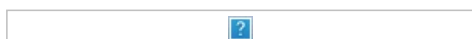
Pattern 2: Multiple virtual networks with peering in between them

This pattern is the extension of the previous pattern where you have multiple virtual networks with potential peering connections. You might opt for this pattern to group applications into separate virtual networks, or you might need presence in multiple Azure regions. You get built-in segmentation through virtual networks because you must explicitly peer a virtual network to another one for them to communicate. (Keep in mind that **virtual network peering** connectivity isn't transitive.) To further segment within a virtual network in a manner similar to pattern 1, use NSGs in the virtual networks.



Pattern 3: Multiple virtual networks in a hub & spoke model

This pattern is a more advanced virtual network organization where you choose a virtual network in a given region as the hub for all the other virtual networks in that region. The connectivity between the hub virtual network and its spoke virtual networks is achieved by using **Azure virtual network peering**. All traffic passes through the hub virtual network, and it can act as a gateway to other hubs in different regions. You set up your security posture at the hubs, so they get to segment and govern the traffic between the virtual networks in a scalable way. One benefit of this pattern is that, as your network topology grows, the security posture overhead doesn't grow (except when you expand to new regions).



The recommended Azure cloud native segmentation control is Azure Firewall. Azure Firewall works across both Virtual Networks and subscriptions to govern traffic flows using layer 3 to layer 7 controls.

You can define how your communication rules work. For example, virtual network X can't talk with virtual network Y but can talk with virtual network Z, no Internet for Virtual network X except for access to *.github.com. With Azure Firewall Manager, you can centrally manage policies across multiple Azure Firewalls and enable DevOps teams to further customize local policies.

Network capabilities	Pattern 1	Pattern 2	Pattern 3
Connectivity/Routing: how each segment communicates to each other	System routing provides default connectivity to any workload in any subnet	Same as a pattern 1	No default connectivity between spoke virtual networks. A layer 3 router, such as the Azure Firewall, in the hub virtual network is required to enable connectivity. Traffic between spoke virtual networks is denied by default. Azure Firewall configuration can enable selected traffic, such as windowsupdate.com.
Network level traffic filtering	Traffic is allowed by default. NSG can be used for filtering this pattern.	Same as a pattern 1	Azure Firewall logs to Azure Monitor all accepted/denied traffic that is sent via a hub
Centralized logging	NSG logs for the virtual network	Aggregate NSG logs across all virtual networks	Accidentally opened public endpoint in a spoke virtual network won't enable access. The return packet will be dropped via stateful firewall (asymmetric routing).
Unintended open public endpoints	DevOps can accidentally open a public endpoint via incorrect NSG rules.	Same as a pattern 1	Azure Firewall supports FQDN filtering for HTTP/S and MSSQL for outbound traffic and across virtual networks.
Application level protection	NSG provides network layer support only.	Same as a pattern 1	No default connectivity between spoke virtual networks. A layer 3 router such as the Azure Firewall in the hub virtual network is required to enable connectivity.
Connectivity/Routing: how each segment communicates to each other	System routing provides default connectivity to any workload in any subnet	Same as a pattern 1	

Virtual network NAT gateway

Virtual Network NAT (network address translation) simplifies outbound-only Internet connectivity for virtual networks. When configured on a subnet, all outbound connectivity uses your specified static public IP addresses. Outbound connectivity is possible without load balancer or public IP addresses directly attached to virtual machines.



Choose Virtual Network NAT gateway when:

- You need on-demand outbound to internet connectivity without pre-allocation
- You need one or more static public IP addresses for scale
- You need configurable idle timeout

- You need TCP reset for unrecognized connections

Routing

When you create a virtual network, Azure creates a routing table for your network. This routing table contains following types of routes.

- System routes
- Subnet default routes
- Routes from other virtual networks
- BGP routes
- Service endpoint routes
- User Defined Routes (UDR)

When you create a virtual network for the first time without defining any subnets, Azure creates routing entries in the routing table. These routes are called system routes. System routes are defined at this location. You cannot modify these routes. However, you can override systems routes by configuring UDRs.

When you create one or multiple subnets inside a virtual network, Azure creates default entries in the routing table to enable communication between these subnets within a virtual network. These routes can be modified by using static routes, which are User Defined Routes (UDR) in Azure.

When you create a virtual network peering between two virtual networks, a route is added for each address range within the address space of each virtual network a peering is created for.

If your on-premises network gateway exchanges border gateway protocol (BGP) routes with an Azure virtual network gateway, a route is added for each route propagated from the on-premises network gateway. These routes appear in the routing table as BGP routes.

The public IP addresses for certain services are added to the route table by Azure when you enable a service endpoint to the service. Service endpoints are enabled for individual subnets within a virtual network. When you enable a service endpoint, route is only added to the route table of for the subnet that belongs to this service. Azure manages the addresses in the route table automatically when the addresses change.

User-defined routes are also called Custom routes. You create UDR in Azure to override Azure's default system routes, or to add additional routes to a subnet's route table.

When you have competing entries in a routing table, Azure selects the next hop based on the longest prefix match similar to traditional routers. However, if there are multiple next hop entries with the same address prefix then Azure selects the routes in following order.

1. User-defined routes (UDR)
2. BGP routes
3. System routes

Common reasons for overriding Azure's default routing are:

- Because you want traffic between subnets to flow through an NVA. To learn more about how to [configure route tables to force traffic through an NVA](#).
- Because you want to force all internet-bound traffic through an NVA, or on-premises, through an Azure VPN gateway. Forcing internet traffic on-premises for inspection and logging is often referred to as forced tunneling. Learn more about how to configure [forced tunneling](#).

System routes

- When you need traffic routed between VMs in the same virtual network or peered virtual networks
- You need communication between VMs using a VNet-to-VNet VPN
- You need site-to-site communication through ExpressRoute or a VPN gateway

User defined routes (UDRs)

- You want to enable filtering of Internet traffic via Azure Firewall or forced tunneling.
- You want traffic between subnets to flow through an NVA.
- You need to create routes to specify how packets should be routed in a virtual network.
- You need to create routes that control network traffic and specify the next hop in the traffic flow.

Design for application delivery services

This section describes networking services in Azure that help deliver applications - Content Delivery Network, Azure Front Door Service, Traffic Manager, Load Balancer, and Application Gateway.

Content Delivery Network (CDN)

Azure Content Delivery Network (CDN) offers developers a global solution for rapidly delivering high-bandwidth content to users by caching their content at strategically placed physical nodes across the world.

When to use a CDN:

- You want point-of-presence locations that are close to large clusters of users.
- You want to reduce latency - both the transmission delay and the number of router hops.
- You have networks in Microsoft, Akamai, or Verizon
- You want custom domains, file compression, caching, and geo-filtering



Azure Front Door Service

Azure Front Door Service enables you to define, manage, and monitor the global routing for your web traffic by optimizing for best performance and instant global failover for high availability. With Front Door, you can transform your global (multi-region) consumer and enterprise applications into robust, high-performance personalized modern applications, APIs, and content that reaches a global audience with Azure.



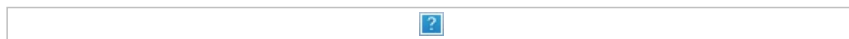
Choose front door when:

- You need to ensure that requests are sent to the lowest latency backends (low latency)
- You have primary and secondary backends (priority)
- You want to distribute traffic using weight coefficients (weighted)
- You want to ensure requests from the same end user gets sent to the same backend (affinity)
- Your traffic is HTTP(s) based and you need WAF and/or CDN integration

Traffic Manager

Azure Traffic Manager is a DNS-based traffic load balancer that enables you to distribute traffic optimally to services across global Azure regions, while providing high availability and responsiveness. Traffic Manager provides a range of traffic-routing methods to distribute traffic such as priority, weighted, performance, geographic, multi-value, or subnet.

The following diagram shows endpoint priority-based routing with Traffic Manager:



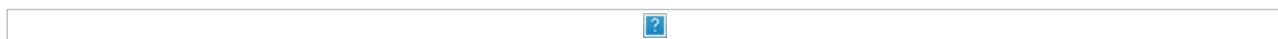
Choose Traffic Manager when:

- You need to increase application availability
- You need to improve application performance
- You need to combine hybrid applications
- You need to distribute traffic for complex deployments

Load balancer

The Azure Load Balancer provides high-performance, low-latency Layer 4 load-balancing for all UDP and TCP protocols. It manages inbound and outbound connections. You can configure public and internal load-balanced endpoints. You can define rules to map inbound connections to back-end pool destinations by using TCP and HTTP health-probing options to manage service availability.

The following picture shows an Internet-facing multi-tier application that utilizes both external and internal load balancers:



Application Gateway

Azure Application Gateway is a web traffic load balancer that enables you to manage traffic to your web applications. It is an Application Delivery Controller (ADC) as a service, offering various layer 7 load-balancing capabilities for your applications.

There are two primary methods of routing traffic, path-based routing, and multiple site routing.

Path-based routing



Use path-based routing to send requests with different URL paths to a different pool of backend servers

Multiple site routing



Use multiple-site routing for tenants with virtual machines or other resources hosting a web application

Choosing a load balancer solution

Azure provides various load-balancing services that you can use to distribute your workloads across multiple computing resources – Azure Front Door, Traffic Manager, Load Balancer, and Application Gateway.

This section describes how you can determine an appropriate load-balancing solution for your business needs.

Azure load-balancing services can be categorized along two dimensions: global versus regional, and HTTP(S) versus non-HTTP(S).

When selecting the load-balancing options, here are some factors that are considered when you select the **Help me choose** default tab in Azure load balancing:

Traffic type. Is it a web (HTTP/HTTPS) application? Is it public facing or a private application?

Global versus. regional. Do you need to load balance VMs or containers within a virtual network, or load balance scale unit/deployments across regions, or both?

Availability. What is the service [SLA](#)?

Cost. Visit [Azure pricing](#). In addition to the cost of the service itself, consider the operations cost for managing a solution built on that service.

Features and limits. What are the overall limitations of each service? Visit [Service limits](#).

The following flowchart will help you to choose a load-balancing solution for your application. The flowchart guides you through a set of key decision criteria to reach a recommendation.

Treat this flowchart as a starting point. Every application has unique requirements, so use the recommendation as a starting point. Then perform a more detailed evaluation.

If your application consists of multiple workloads, evaluate each workload separately. A complete solution may incorporate two or more load-balancing solutions.



Design for application protection services

This section describes networking services in Azure that help protect your network resources - Protect your applications using any or a combination of these networking services in Azure - DDoS protection, Private Link, Firewall, Web Application Firewall, Network Security Groups, and Virtual Network Service Endpoints.

Distributed denial of service protection

Azure DDoS Protection provides countermeasures against the most sophisticated DDoS threats. The service provides enhanced DDoS mitigation capabilities for your application and resources deployed in your virtual networks. Additionally, customers using Azure DDoS Protection have access to DDoS Rapid Response support to engage DDoS experts during an active attack.



Use DDoS protection Standard when you need:

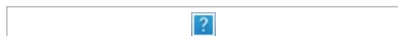
- Always-on traffic monitoring
- Adaptive tuning
- Multi-layered protection
- Mitigation scale
- Attack analytics and metrics
- Attack alerting
- DDoS rapid response team

Azure Private Link

Azure Private Link enables you to access Azure PaaS Services (for example, Azure Storage and SQL Database) and Azure hosted customer-owned/partner services over a private endpoint in your virtual network. Traffic between your virtual network and the service travels the Microsoft backbone network. Exposing your service to the public internet is no longer necessary. You can create your own private link service in your virtual network and deliver it to your customers. Private link is used to access PaaS services such as Azure Storage, Azure SQL, App Services and more as illustrated below.

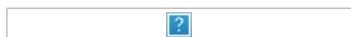
Recommend private link or private endpoints when:

- You need private connectivity to services on Azure
- You need integration with on-premises and peered networks
- You need traffic to remain on Microsoft network, with no public internet access



Azure Firewall

Azure Firewall is a managed, cloud-based network security service that protects your Azure Virtual Network resources. It's a fully stateful firewall as a service with built-in high availability and unrestricted cloud scalability. Using Azure Firewall, you can centrally create, enforce, and log application and network connectivity policies across subscriptions and virtual networks. Azure Firewall uses a static public IP address for your virtual network resources allowing outside firewalls to identify traffic originating from your virtual network. Azure Firewall provides inbound protection for non-HTTP/S protocols (for example, RDP, SSH, FTP), outbound network-level protection for all ports and protocols, and application-level protection for outbound HTTP/S.



Web Application Firewall

Azure Web Application Firewall (WAF) provides protection to your web applications from common web exploits and vulnerabilities such as SQL injection, and cross site scripting. Azure WAF provides out of box protection from OWASP top 10 vulnerabilities via managed rules. Additionally, customers can also configure custom rules, which are customer managed rules to provide additional protection based on source IP range, and request attributes such as headers, cookies, form data fields or query string parameters. Preventing such attacks in application code is challenging. It can require rigorous maintenance, patching, and monitoring at multiple layers of the application topology. A centralized web application firewall helps make security management much simpler. A WAF also gives application administrators better assurance of protection against threats and intrusions.

A WAF solution can react to a security threat faster by centrally patching a known vulnerability, instead of securing each individual web application.

WAF can be deployed with Azure Application Gateway, Azure Front Door, and Azure Content Delivery Network (CDN) service from Microsoft. WAF on Azure CDN is currently under public preview. WAF has features that are customized for each specific service.



Network security groups

You can filter network traffic to and from Azure resources in an Azure virtual network with a network security group. You can also use network virtual appliances (NVA) such as Azure Firewall or firewalls from other vendors. You can control how Azure routes traffic from subnets. You can also limit who in your organization can work with resources in virtual networks.

A network security group (NSG) contains a list of Access Control List (ACL) rules that allow or deny network traffic to subnets, NICs, or both. NSGs can be associated with either subnets or individual NICs connected to a subnet. When an NSG is associated with a subnet, the ACL rules apply to all the VMs in that subnet. In addition, traffic to an individual NIC can be restricted by associating an NSG directly to a NIC.

NSGs contain two sets of rules: inbound and outbound. The priority for a rule must be unique within each set. Each rule has properties of protocol, source and destination port ranges, address prefixes, direction of traffic, priority, and access type. All NSGs contain a set of 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.

Service endpoints

Virtual Network (VNet) service endpoints extend your virtual network private address space and the identity of your VNet to the Azure services, over a direct connection. Endpoints allow you to secure your critical Azure service resources to only your virtual networks. Traffic from your VNet to the Azure service always remains on the Microsoft Azure backbone network.



Key Benefits:

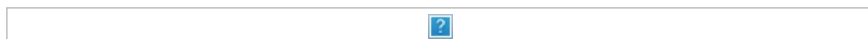
- Improved security for your Azure service resources
- Optimal routing for Azure service traffic from your virtual network
- Simple to set up with less management overhead

Azure Bastion

The Azure Bastion service is a new fully platform-managed PaaS service that you provision inside your virtual network. It provides secure and seamless RDP/SSH connectivity to your virtual machines directly in the Azure portal over TLS. When you connect via Azure Bastion, your virtual machines do not need a public IP address.

Recommend Azure Bastion when you need to:

- Secure remote connections from the Azure portal to Azure VMs
- Eliminate exposing RDP ports, SSH ports, or public IP addresses for your internal VMs



Key security features

- Traffic initiated from Azure Bastion to target virtual machines stays within the virtual network or between peered virtual networks.
- There's no need to apply NSGs to the Azure Bastion subnet, because it's hardened internally. For additional security, you can configure NSGs to allow only remote connections to the target virtual machines from the Azure Bastion host.
- Azure Bastion helps protect against port scanning. RDP ports, SSH ports, and public IP addresses aren't publicly exposed for your VMs.
- Azure Bastion helps protect against zero-day exploits. It sits at the perimeter of your virtual network. So you don't need to worry about hardening each of the virtual machines in your virtual network. The Azure platform keeps Azure Bastion up to date.
- The service integrates with native security appliances for an Azure virtual network, like Azure Firewall.
- You can use the service to monitor and manage remote connections.

Just in time (JIT) network access

With JIT, you can lock down the inbound traffic to your VMs, reducing exposure to attacks while providing easy access to connect to VMs when needed.

When you enable just-in-time VM access, you can select the ports on the VM to which inbound traffic will be blocked. Security Center ensures “deny all inbound traffic” rules exist for your selected ports in the [network security group](#) (NSG) and [Azure Firewall rules](#). These rules restrict access to your Azure VMs’ management ports and defend them from attack.

If other rules already exist for the selected ports, then those existing rules take priority over the new “deny all inbound traffic” rules. If there are no existing rules on the selected ports, then the new rules take top priority in the NSG and Azure Firewall.

When a user requests access to a VM, Security Center checks that the user has [Azure role-based access control \(Azure RBAC\)](#) permissions for that VM. If the request is approved, Security Center configures the NSGs and Azure Firewall to allow inbound traffic to the selected ports from the relevant IP address (or range), for the amount of time that was specified. After the time has expired, Security Center restores the NSGs to their previous states. Connections that are already established are not interrupted.

Introduction



You work for a Tailwind Traders, a home improvement retailer. Tailwind Traders currently manages on-premises data centers that host the company's retail website. These data centers also store all the data and streaming video for its applications.

The IT department is currently responsible for all the management tasks for its computing hardware and software. The IT team handles the procurement process to buy new hardware, installs and configures software, and deploys everything throughout the data center.

These management responsibilities create some obstacles for delivering applications to Tailwind's users and customers in a timely fashion. So, the company is shifting on-premises workloads to the cloud.

You've been tasked with selecting appropriate backup solutions for migrated workloads. You also need to select an appropriate disaster recovery option for these workloads.

Learning objectives

After completing this module, you'll be able to:

- Design for backup and recovery.
- Design for Azure Backup.
- Design for Azure blob backup and recovery.
- Design for Azure Files backup and recovery.
- Design for Azure virtual machine backup and recovery.
- Design for Azure SQL backup and recovery
- Design for Azure Site Recovery.

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions. The module concepts are covered in:

Design business continuity

- Design a Solution for Backup and Disaster Recovery

Prerequisites

- Conceptual knowledge of Business Continuity and Disaster Recovery solutions.
- Working experience with object replication, backup solution tools, and recovery options.

Design for backup and recovery

Organizations, such as Tailwind Traders, require a high degree of reliability from their mission-critical apps. To achieve the desired reliability for on-premises based apps, it's typical to purchase more computing resource, such as servers and storage. By purchasing more computing resources, organizations can build redundancy into their on-premises infrastructure.

It's also vital that any mission-critical app, and its associated data, is recoverable following a failure—ideally to the point of failure. This recoverability is often provided by backup, restore components, and procedures. For organizations with apps hosted in Azure, or organizations with hybrid app deployments, there are other considerations and options.

Reliable apps are those which are:

- Resilient to component failure.
- Highly available and can run in a healthy state with no significant downtime.

To achieve the desired resilience and high availability, you must first define your requirements.

Note: This module will use the term resiliency as the ability of a system to gracefully handle and recover from failures, both inadvertent and malicious.

Define your requirements

Defining your requirements involves:

- Identifying your business needs.
- Building your resiliency plan to address those needs.

Use the following table of considerations to provide guidance on this process.

Consideration	Description
What are your workloads and their usage?	<p>A workload is a distinct capability or task that is logically separated from other tasks, in terms of business logic and data storage requirements. Each workload probably has different requirements for availability, scalability, data consistency, and disaster recovery.</p> <p>Usage patterns can determine your requirements. Identify differences in requirements during both critical and non-critical periods. To ensure uptime, plan redundancy across several regions in case one region fails. Conversely, to minimize costs during non-critical periods, you can run your application in a single region.</p>
What are the usage patterns for your workloads?	<p>Mean time to recovery (MTTR) and mean time between failures (MTBF) are the typically used metrics. MTBF is how long a component can reasonably expect to last between outages. MTTR is the average time it takes to restore a component after a failure. Use these metrics to determine where you need to add redundancy, and to determine service-level agreements (SLAs) for customers.</p>
What are the availability metrics?	<p>The recovery time objective (RTO) is the maximum acceptable time one of your apps can be unavailable following an incident. The recovery point objective (RPO) is the maximum duration of data loss that is acceptable during a disaster. Also consider the recovery level objective (RLO). This metric determines the granularity of recovery. In other words, whether you must be able to recover a server farm, a web app, a site, or just a specific item. To determine these values, conduct a risk assessment. Ensure that you understand the cost and risk of downtime or data loss in your organization.</p>
What are the recovery metrics?	

What are the workload availability targets?

To help ensure that your app architecture meets your business requirements, define target SLAs for each workload. Account for the cost and complexity of meeting availability requirements, in addition to application dependencies.

What are your SLAs?

In Azure, the SLA describes the Microsoft commitments for uptime and connectivity. If the SLA for a particular service is 99.9 percent, you should expect the service to be available 99.9 percent of the time.

Tip: If the MTTR of any critical component in a highly available scenario exceeds the system RTO, then a failure in the system might cause an unacceptable business disruption. In other words, you can't restore the system within the defined RTO.

Define your own target SLAs for each workload in your solution by answering the preceding questions. This helps ensure that the architecture meets your business requirements. For example, if a workload requires 99.99 percent uptime, but depends on a service with a 99.9 percent SLA, that service can't be a single point of failure in the system.

After you've defined your recovery requirements, you're ready to select a suitable recovery technology.

Design for Azure Backup

The **Azure Backup** service uses Azure resources for short-term and long-term storage. Azure Backup minimizes or even eliminates the need for maintaining physical backup media. Examples of backup media are tapes, hard drives, and DVDs.

What can you do with Azure Backup?



You can use Azure Backup for these backup types:

- **On-premises.** Azure Backup can back up files, folders, and system state using the Microsoft Azure Recovery Services (MARS) agent. Alternatively, you can use Data Protection Manager (DPM) or the Microsoft Azure Backup Server (MABS) agent to protect on-premises VMs (both Hyper-V and VMware), and other on-premises workloads.
- **Azure VM.** Back up entire Windows or Linux VMs (using backup extensions), or back up files, folders, and system state using the MARS agent.
- **Azure File shares.** Back up Azure File shares to a storage account.
- **Microsoft SQL Server in Azure VMs.** Back up SQL Server databases running on Azure VMs.
- **SAP HANA databases in Azure VMs.** Back up SAP HANA databases running on Azure VMs.
- **Microsoft cloud.** Azure Backup can replace your existing on-premises or off-site backup solution with a cloud-based solution that's reliable, secure, and cost-competitive.

Azure Backup offers multiple components that you can download and deploy on the appropriate computer, server, or in the cloud. The component (or agent) that you deploy depends on what you want to protect.

Where is the data backed up?

Azure Backup organizes your backup data in a storage entity called a vault. A storage vault stores backup copies, recovery points, and backup policies.. There are two types of vaults. The primary differences in the vaults are supported data sources and supported Azure products.

Capability	Supported data sources	Supported products
Backup vault	Azure database for PostgreSQL servers Azure blobs Azure disks Azure virtual machines (VMs) SQL in an Azure VM	Azure Backup
Recovery services vault	Azure Files SAP HANA in Azure VM Azure Backup Server Azure Backup Agent Data Protection Manager	Azure Backup, Azure Site Recovery

Considerations for storage vaults

- **Think about how to organize the vaults.** If your workloads are all managed by a single subscription and single resource, then you can use a single vault. If your workloads are spread across subscriptions, then you can create multiple vaults. Use separate vaults for Azure Backup and Azure Site Recovery.
- **Use Azure policy.** If you needed consistent policy across vaults, then you can use Azure Policy to propagate backup policy across multiple vaults. A backup policy is scoped to a vault.
- **Implement regional availability.** Azure Backup requires the backup components to be in the same region as the vault.
- **Protect using Azure role-based access control (RBAC).** Protect and manage vault access by using Azure RBAC.
- **Design for Redundancy.** Specify how data in the vault is replicated for redundancy. Use Locally redundant storage (LRS) to protect against failure in a datacenter. LRS replicates data to a storage scale unit. Use Geo-redundant storage (GRS) to protect against region-wide outages. GRS replicates your data to a secondary region..

Design for Azure blob backup and recovery

Operational backup for blobs is a local backup solution. The backup data isn't transferred to the Backup vault but is stored in the source storage account itself. This is a **continuous backup** solution. You don't need to schedule any backups. All changes will be retained and restorable from a selected point in time.

Take advantage of blob soft delete and versioning

Soft delete protects an individual blob, snapshot, container, or version from accidental deletes or overwrites. Soft delete maintains the deleted data in the system for a specified retention period. During the retention period, you can restore a soft-deleted object to its state at the time it was deleted.



- **Container soft delete** can restore a container and its contents at the time of deletion. The retention period for deleted containers is between 1 and 365 days. The default retention period is seven days.
- **Blob soft delete** can restore a blob, snapshot, or version that has been deleted. Blob soft delete is useful for restoring specific files. The retention period for deleted blobs is also between 1 and 365 days.
- **Blob versioning** works to automatically maintain previous versions of a blob. When blob versioning is enabled, you can restore an earlier version of a blob. Versioning lets you recover your data if it's incorrectly modified or deleted. Blob versioning is useful if you have multiple authors editing files and need to maintain or restore their individual changes.

Tip: Container soft delete doesn't protect against the deletion of a storage account, but only against the deletion of containers in that account.

Consider point-in-time restore for block blobs

Like soft delete, **point-in-time restore for block blobs** also protects against accidental deletion or corruption. For this feature, you create a management policy for the storage account and specify a retention period. During the retention period, you can restore block blobs from the present state to a state at a previous time.

The following diagram shows how point-in-time restore works. One or more containers or blob ranges is restored to its previous state. The effect is to revert write and delete operations that happened during the retention period.



Note: Point-in-time restore enables testing scenarios that require reverting a data set to a known state before running further tests.

Prevent accidental changes by using resource locks

A **resource lock** prevents resources from being accidentally deleted or changed. Consider using resource locks to protect your data. You can set the lock level to **CanNotDelete** or **ReadOnly**.

- **CanNotDelete** means authorized people can still read and modify a resource, but they can't delete the resource without first removing the lock.
- **ReadOnly** means authorized people can read a resource, but they can't delete or change the resource. Applying this lock is like restricting all authorized users to the permissions granted by the **Reader** role in Azure RBAC.

Tip: Consider the storage protection features you've learned about. Which ones do you think would be most useful? In what scenarios will you use these features?

Design for Azure files backup and recovery



Azure Files provides the capability to take **share snapshots of file shares**. Share snapshots give you an extra level of security and help reduce the risk of data corruption or accidental deletion. You can also use them as a general backup for disaster recovery.

- Share snapshots capture the share state at that point in time.
- Snapshots can be created manually using the Azure portal, REST API, client libraries, the Azure CLI, and PowerShell.
- Snapshots can be automated using Azure Backup and backup policies.
- Snapshots are at the root level of a file share and apply to all the folders and files contained in it. Retrieval is provided at individual file level.
- Snapshots are incremental. Only the deltas between your snapshots will be stored.
- After a share snapshot is created, it can be read, copied, or deleted, but not modified.
- You cannot delete a share that has share snapshots. To delete the share you must delete all the share snapshots.

Important: Snapshots are not a replacement for cloud-side backups.

How can you automate file share backups?

It is recommended you use Azure Backup to automate and manage file share snapshots.



- Azure Backup keeps the metadata about the backup in the recovery services vault, but no data is transferred. This means a fast backup solution with built-in backup and reporting.
- When Azure Backup is enabled on the file share soft delete is also enabled.
- You can configure backups with daily/weekly/monthly/yearly retention according to your requirements.

Considerations for file share backups

- **Use instant restore.** Azure file share backup uses file share snapshots. You can select just the files you want to restore instantly.
- **Implement alerting and reporting.** You can configure alerts for backup and restore failures and use the reporting solution provided by Azure Backup. These reports provide insights on file share backups.
- **Consider self-service restore.** Backup uses server endpoint VSS snapshots. Consider giving advanced users the ability to restore files themselves.
- **Consider on-demand backups.** Azure Backup policies are limited to scheduling a backup once a day. If a user creates a file in the morning and works on it all day a nightly backup won't have the new file. For these reasons consider on-demand backups for the most critical file shares.
- **Organize file shares for backup.** If possible, consider organizing your file shares for backups. For example, public facing vs internal file shares.
- **Snapshot before code deployments.** If a bug or application error is introduced with the new deployment, you can go back to a previous version of your data on that file share. To help protect against these scenarios, you can take a share snapshot before you deploy new application code.

Design for Azure virtual machine backup and recovery

Azure Backup provides independent and isolated [Azure virtual machines backups](#). Backups are stored in a Recovery Services vault with built-in management of recovery points. Configuration and scaling are simple, backups are optimized, and you can easily restore as needed. Back up is available for Windows and Linux VMs.

How do Azure virtual machines backups work?

An Azure backup job consists of two phases. First, a virtual machine snapshot is taken. Second, the virtual machine snapshot is transferred to the Azure Recovery Services vault.



Note: A recovery point is considered created only after both steps are completed. The recovery point is used to perform a restore.

Backup policies and retention

You can define the backup frequency and retention duration for your backups. Currently, the VM backup can be triggered daily or weekly, and can be stored for multiple years.

- **Snapshot tier:** In phase 1, snapshots are stored locally for a maximum period of five days. This is referred to as the snapshot tier. Snapshot tier restores are faster (than restore from vault) because they eliminate the wait time for snapshots to copy to the vault before triggering the restore. This capability is called **Instant Restore**.
- **Vault tier:** In phase 2, snapshots are transferred to the vault for additional security and longer retention. This is referred to as **vault tier**.

Considerations for Azure virtual machine backup and recovery

- **Plan backup schedule policies.** Consider grouping VMs that require the same schedule start time, frequency, and retention settings within a single policy. Ensure the backup scheduled start time is during non-peak production application time. To distribute backup traffic, consider backing up different VMs at different times of the day and make sure the times don't overlap. For example, have policies for critical and non-critical virtual machines.
- **Plan backup retention policies.** Implement both short-term (daily) and long-term (weekly) backups. If you need to take a backup not scheduled via backup policy, then you can use an on-demand backup. For example, backup on-demand multiple times per day when scheduled backup permits only one backup per day.
- **Optimize backup policies.** As your business requirements change, make sure to review and change your backup policies. Enable monitoring and alerting features and review the results.
- **Consider Cross Region Restore (CRR).** CRR allows you to restore Azure VMs in a secondary region, which is an Azure paired region. This option lets you to conduct drills to meet audit or compliance requirements. You can also restore the VM or its disk if there's a disaster in the primary region. CRR is an opt-in feature for any recovery services vault. CRR also works for SQL databases and SAP HANA databases hosted on Azure VMs.

Design for Azure SQL backup and recovery

It's essential that you can recover your SQL database data. You should consider automated backups of your Azure SQL Database and Azure SQL Managed Instances. Database backups enable database restoration to a specified point in time and within a configured retention period.

Describe automated backups

Both SQL Database and SQL Managed Instance use SQL Server technology to create **full backups** every week, **differential backups** every 12-24 hours, and **transaction log backups** every 5 to 10 minutes. The frequency of transaction log backups is based on the compute size and the amount of database activity. When you restore a database, the service determines which full, differential, and transaction log backups need to be restored.

- **Full backups:** In a full backup, everything in the database and the transaction logs is backed up. SQL Database makes a full back up once a week.
- **Differential backups:** In a differential backup, everything that changed since the last full backup is backed up. SQL Database makes a differential backup every 12 - 24 hours.
- **Transactional backups:** In a transactional backup, the contents of the transaction logs are backed up. If the latest transaction log has failed or is corrupted, the option is to fall back to the previous transaction log backup. Transactional backups enable administrators to restore up to a specific time, which includes the moment before data was mistakenly deleted. Transaction log backups every five to 10 minutes.

Describe backup usage cases

You can use the automated backups in several ways.

- **Restore an existing database to a point in time in the past** within the retention period. This operation creates a new database on the same server as the original database but uses a different name to avoid overwriting the original database. After the restore completes, you can delete the original database.
- **Restore a deleted database to the time of deletion** or to any point in time within the retention period. The deleted database can be restored only on the same server or managed instance where the original database was created.
- **Restore a database to another geographic region.** Geo-restore allows you to recover from a geographic disaster when you cannot access your database or backups in the primary region. It creates a new database on any existing server or managed instance, in any Azure region.
- **Restore a database from a specific long-term backup** of a single database or pooled database. If the database has been configured with a long-term retention policy you can restore an old version of the database.

Long-term backup retention policies

Azure SQL Database automatic backups remain available to restore for up to 35 days. This period is enough for the purposes of day-to-day administration. But sometimes you might need to retain data for longer periods. For example, data protection regulations in your local jurisdiction might require you to keep backups for several years.

For these requirements, use the long-term retention (LTR) feature. This way, you can store Azure SQL Database backups in read-access geo-redundant storage (RA-GRS) blobs for up to 10 years. If you need access to any backup in LTR, you can restore it as a new database by using either the Azure portal or PowerShell.

Design for Azure Site Recovery

Azure Site Recovery is a service that provides BCDR features for your applications in Azure, on-premises, and in other cloud providers. Azure Site Recovery has plans that help automate your disaster recovery by enabling you to define how machines are failed over, and the order in which they're restarted after being successfully failed over. In this way, Azure Site Recovery helps to automate tasks and further reduce your recovery time objective. You also use Azure Site Recovery to periodically test failovers, and the overall effectiveness of the recovery process.



What can you do with Azure Site Recovery?

Site Recovery provides the capabilities described in the following table.

Feature	Details
Simple BCDR solution	Use Site Recovery in the Azure portal to setup and manage replication, failover, and failback from a single location.
Azure VM replication	Setup disaster recovery of your Azure VMs, and failover from a primary region to a secondary region.
On-premises VM replication	Replicate on-premises VMs and physical servers to Azure, or to a secondary on-premises datacenter.
Workload replication	Replicate any workload running on supported Azure VMs, on-premises Hyper-V and VMware VMs, and Windows or Linux physical servers.
Data resilience	Orchestrate replication without intercepting app data by using Site Recovery. When failover occurs, Azure VMs are created, based on the replicated data. When you replicate to Azure, data is stored in Azure storage, with the resilience that provides.
RTO and RPO targets	Keep RTO and RPO within defined organizational limits. Site Recovery provides continuous replication for Azure VMs and VMware VMs, and replication frequency as low as 30 seconds for Hyper-V.
Keep apps consistent over failover	By using app-consistent snapshots, you can replicate using recovery points. These snapshots capture disk data, data in memory, and all in process transactions.
Testing without disruption	You can run disaster recovery tests, without affecting ongoing replication.
Flexible failovers	You can run planned failovers for expected outages with no data loss. Run unplanned failovers with minimal data loss. And fail back to your primary site when it's available again.
Customized recovery plans	Create recovery plans so that you can customize and sequence the failover and recovery of your multi-tier apps running on multiple VMs. You can group machines together in a recovery plan. You can then, optionally, add scripts and manual actions. You can integrate recovery plans with Azure automation runbooks.
BCDR integration	You can integrate Site Recovery with other BCDR technologies. For example, use Site Recovery to protect the SQL Server backend of your corporate workloads. Because of its native support for SQL Server AlwaysOn, you can manage the failover of availability groups.
	Download, from the Azure Automation

Consider using Azure Site Recover with Azure Backup

Here we have an on-premises environment that has a Hyper-V host server for hosting virtual machines. You want to back up all the files and folders in this virtual machine to Azure. You also want to protect any workloads running on the virtual machine and keep running them even if the virtual machine fails. Azure Backup and Site Recovery can be used together as part of a single solution.



In this scenario, Azure Backup periodically backs up the files and folders on the Windows machine to Azure. This process ensures they are secure and retrievable even if the whole on-premises environment stops functioning. Separately, Site Recovery will be used to protect running workloads and keep them running. Because Site Recovery can replicate frequently, the RTO for your workloads can be reduced.

Introduction

You work for Tailwind Traders, a home improvement retailer. Tailwind Traders currently manages on-premises datacenters that host the company's retail website. These datacenters also store all the data and streaming video for its applications.



The IT department is currently responsible for all the management tasks for its computing hardware and software. The IT team handles the procurement process to buy new hardware, installs and configures software, and deploys everything throughout the datacenter.

These management responsibilities create some obstacles for delivering applications to Tailwind Trader's users and customers in a timely fashion. As a result, you've been tasked with reviewing available migration options. You must also select the appropriate options to use to support the planned migrations. The planned workloads include virtual machines (VMs), databases, and applications.

After completing this module, you'll be able to assess and select a suitable migration strategy to support migration of your on-premises workloads.

Learning objectives

After completing this module, you'll be able to:

- Evaluate migration with the Cloud Adoption Framework.
- Describe the Azure Migration Framework.
- Assess your on-premises workloads.
- Select a migration tool.
- Migrate your databases.
- Select an online storage migration tool.
- Migrate offline data.

Skills measured

The content in the module will help you prepare for Exam AZ-305: Designing Microsoft Azure Infrastructure Solutions.

Design Infrastructure

- Design Migrations

Prerequisites

- Conceptual knowledge of migrating compute, database, and storage workloads.
- Working experience with planning migrations, assessing workloads, determining migration requirements, and deploying workloads.

Evaluate migration with the Cloud Adoption Framework

The Microsoft Cloud Adoption Framework for Azure is provided to help you drive adoption of Azure in your organization. It provides recommendations, best practice guidance, documentation, and tools. The framework supports several methodologies:

- **Define strategy:** Define business justification and expected outcomes of adoption.
- **Plan:** Align actionable adoption plans to business outcomes.
- **Ready:** Prepare the cloud environment for the planned changes.
- **Migrate:** Migrate and modernize existing workloads.
- **Innovate:** Develop new cloud-native or hybrid solutions.
- **Govern:** Govern the environment and workloads.
- **Manage:** Operations management for cloud and hybrid solutions.
- **Organize:** Align the teams and roles supporting your organization's cloud adoption efforts.

Each of the methodologies is a phase within a cloud adoption lifecycle.



Tailwind Traders shouldn’t undertake cloud adoption without considerable planning. This is especially true with the migrate phase in the cloud adoption lifecycle. To prepare you for this phase, you should review the following documentation:

- **Azure migration guide overview:** Review the Azure migration guide to learn about Azure native tools and a relevant approach to migration.
- **The One Migrate approach to migrating the IT portfolio.** Review the scenarios captured in this Migrate methodology: They demonstrate the same set of consistent guidelines and processes for migrating both Microsoft and third-party technologies.
- **Azure cloud migration best practices checklist:** Review this document to learn how best to address common migration needs through the application of consistent best practices.
- **Cloud Adoption Framework migration model:** Review this document to understand mitigation. Migration can be process intensive activity. As you increase migration effort, review these process improvements to help to optimize aspects of your migration.

Understand the migration effort

The actions required to migrate Tailwind Traders' workloads will almost certainly fall into three efforts (or phases) for each workload:

- Assess
- Deploy
- Release

This section of the Cloud Adoption Framework explains how to maximize the return from each of the phases required to migrate a workload to production. The following table provides an overview of the phases of this process, as displayed in the diagram below:

Phase	Explanation
Assess	Assess your workloads to determine costs, modernization, and required deployment tools.
Deploy	After workloads are assessed, the existing functionality of those workloads is replicated (or improved) in the cloud. After workloads are replicated to the cloud, you can test, optimize, and document your migrated workloads.
Release	When satisfied, you can release these workloads to users. During this phase, ensure that you hand off the workloads to governance, operations management, and security teams for ongoing support of those workloads.



Describe the Azure Migration Framework

Before you can start migrating Tailwind Traders' on-premises workloads to Azure, you should consider creating a migration plan. This plan should help you to identify the workloads that you want to migrate. The plan should also help you identify and select the appropriate service or tools to use during the migration.

Ideally, your plan should also include details about how to optimize the migrated services. The Azure migration framework can help you work through your plan and the migration it addresses.

The Azure migration framework consists of four stages:

- Assess
- Migrate
- Optimize
- Monitor

Assess your on-premises environment

The best place to start is with an assessment of Tailwind Traders' current on-premises environment. During the assessment, you should:

- Identify apps, and their related servers, services, and data, that is within scope for migration.
- Start to involve stakeholders, such as the IT department and relevant business groups.
- Create a full inventory and a dependency map of servers, services, and apps that you're selecting for migration.
- Estimate cost savings by using the Azure Total Cost of Ownership Calculator.
- Identify appropriate tools and services you can use to perform the assessment, migration, optimization, and monitoring stages.

Strategies for migration to the cloud fall into four broad patterns: rehost, refactor, rearchitect, or rebuild. The strategy you adopt depends on your business drivers and migration goals. You might even adopt multiple patterns. For example, you might choose to rehost simple apps or apps that aren't critical to your business, but rearchitect apps that are more complex and business critical.

The following table describes the four patterns.

Pattern	Definition	When to use
Rehost	Often referred to as a lift and shift migration, this option doesn't require code changes, and allows you to migrate your existing workloads to Azure quickly. Each workload is migrated as is, without the risk and cost associated with code changes.	When you need to move workloads quickly to the cloud, when you want to move a workload without modifying it, when your apps are designed so that they can take advantage of Azure IaaS scalability after migration, and when workloads are important to your business, but you don't need immediate changes to app capabilities.
Refactor	Often referred to as repackaging, refactoring requires minimal changes to apps so they can connect to Azure platform as a service (PaaS) and use cloud offerings. For example, you could migrate existing apps to Azure App Service or Azure Kubernetes Service (AKS). Alternatively, you could refactor relational and non-relational databases into options such as Azure SQL Database Managed Instance, Azure Database for MySQL, Azure Database for PostgreSQL, and Azure	If you want to apply innovative DevOps practices provided by Azure, or if you're thinking about DevOps using a container strategy for workloads. For refactoring, you need to think about the portability of your existing code base and available development skills.

	Cosmos DB, but only if your app can easily be repackaged to work in Azure.	
Rearchitect	<p>Rearchitecting for migration focuses on modifying and extending app functionality and the code base to optimize the app architecture for cloud scalability. For example, you could break down a monolithic application into a group of microservices that work together and scale easily. Alternatively, you could rearchitect relational and nonrelational databases to a fully managed database solution such as Azure SQL Database Managed Instance, Azure Database for MySQL, Azure Database for PostgreSQL, and Azure Cosmos DB.</p>	<p>When your apps need major revisions to incorporate new capabilities, or to work effectively on a cloud platform. When you want to use existing application investments, meet scalability requirements, apply innovative DevOps practices, and minimize use of VMs.</p>
Rebuild	<p>Rebuild takes things a step further by completely rebuilding an app using Azure cloud technologies. For example, you could build green-field apps with cloud-native technologies such as Azure Functions, Azure AI, Azure SQL Database Managed Instance, and Azure Cosmos DB.</p>	<p>When you want rapid development, and existing apps have limited functionality and lifespan. When you're ready to expedite business innovation by using Azure DevOps practices. When are building new applications using cloud-native technologies, like Azure Blockchain. When you are rebuilding legacy apps as no code or low apps in the cloud.</p>

Choosing which strategy to use depends on what you're trying to accomplish.

Migrate your workloads

After you complete the assessment, you can begin the process of migrating your targeted apps and their related services and data. The migration stage typically consists of the following elements:

Deploy cloud infrastructure targets: Before you can migrate Tailwind Traders' workloads, you'll need to create the required cloud infrastructure targets. Depending on the tools you use to perform the migration, you might need to create the required Azure resources before you begin the migration. Some tools, such as Azure Migrate and Azure Database Migration Service, can create the target Azure resources for you.

Migrate workloads: It's a good idea to pilot your workload migration, and to choose a non-critical app for the pilot. This approach enables you to:

- Become familiar with tools
- Gain experience with processes and procedures
- Reduce risk when migrating large or complex workloads

Depending on the workload you plan to migrate, the steps used to perform the migration will vary.

Decommission on-premises infrastructure: After you're satisfied that your source apps and databases are migrated successfully, you must decommission those source workloads. Consider retaining the source workload backups and archived data. This data might prove useful as it provides a historical archive. You can store these backups and archives in Azure Blob storage.

Optimize the migrated workloads

During the optimization stage, you should:

- Analyze the costs of the workload migration

- Identify ways to reduce costs
- Seek to improve workload performance

To analyze costs, you can use Cost Management in the Azure portal. Select the Azure resource group in which you're interested (the one which contains the migrated workloads), and then, in the navigation pane, select **Costs analysis** in the **Cost Management** section. The following screenshot shows the cost analysis for the last billable period for a resource group called ContosoResourceGroup. The results display the costs according to service name, region, and resource, although you can customize the results to your needs.



To help to reduce costs, select the **Advisor recommendations** link in the navigation pane. Recommendations, if there are any, are displayed on the details pane.

Monitor your workloads

You can use Azure Monitor to capture health and performance information from your Azure VMs. However, you must first install a Log Analytics agent on target VMs. After you've installed the agent, you can then set up alerting and reporting.

Tip: You can install the agent on machines running either Windows or Linux.

You can set up alerts based on a range of data sources, such as:

- Specific metric values like CPU usage
- Specific text in log files
- Health metrics
- An Autoscale metric

Describe the Azure Migration Framework

Before you can start migrating Tailwind Traders' on-premises workloads to Azure, you should consider creating a migration plan. This plan should help you to identify the workloads that you want to migrate. The plan should also help you identify and select the appropriate service or tools to use during the migration.

Ideally, your plan should also include details about how to optimize the migrated services. The Azure migration framework can help you work through your plan and the migration it addresses.

The Azure migration framework consists of four stages:

- Assess
- Migrate
- Optimize
- Monitor

Assess your on-premises environment

The best place to start is with an assessment of Tailwind Traders' current on-premises environment. During the assessment, you should:

- Identify apps, and their related servers, services, and data, that is within scope for migration.
- Start to involve stakeholders, such as the IT department and relevant business groups.
- Create a full inventory and a dependency map of servers, services, and apps that you're selecting for migration.
- Estimate cost savings by using the Azure Total Cost of Ownership Calculator.
- Identify appropriate tools and services you can use to perform the assessment, migration, optimization, and monitoring stages.

Strategies for migration to the cloud fall into four broad patterns: rehost, refactor, rearchitect, or rebuild. The strategy you adopt depends on your business drivers and migration goals. You might even adopt multiple patterns. For example, you might choose to rehost simple apps or apps that aren't critical to your business, but rearchitect apps that are more complex and business critical.

The following table describes the four patterns.

Pattern	Definition	When to use
Rehost	Often referred to as a lift and shift migration, this option doesn't require code changes, and allows you to migrate your existing workloads to Azure quickly. Each workload is migrated as is, without the risk and cost associated with code changes.	When you need to move workloads quickly to the cloud, when you want to move a workload without modifying it, when your apps are designed so that they can take advantage of Azure IaaS scalability after migration, and when workloads are important to your business, but you don't need immediate changes to app capabilities.
Refactor	Often referred to as repackaging, refactoring requires minimal changes to apps so they can connect to Azure platform as a service (PaaS) and use cloud offerings. For example, you could migrate existing apps to Azure App Service or Azure Kubernetes Service (AKS). Alternatively, you could refactor relational and non-relational databases into options such as Azure SQL Database Managed Instance, Azure Database for MySQL, Azure Database for PostgreSQL, and Azure	If you want to apply innovative DevOps practices provided by Azure, or if you're thinking about DevOps using a container strategy for workloads. For refactoring, you need to think about the portability of your existing code base and available development skills.

	Cosmos DB, but only if your app can easily be repackaged to work in Azure.	
Rearchitect	<p>Rearchitecting for migration focuses on modifying and extending app functionality and the code base to optimize the app architecture for cloud scalability. For example, you could break down a monolithic application into a group of microservices that work together and scale easily. Alternatively, you could rearchitect relational and nonrelational databases to a fully managed database solution such as Azure SQL Database Managed Instance, Azure Database for MySQL, Azure Database for PostgreSQL, and Azure Cosmos DB.</p>	<p>When your apps need major revisions to incorporate new capabilities, or to work effectively on a cloud platform. When you want to use existing application investments, meet scalability requirements, apply innovative DevOps practices, and minimize use of VMs.</p>
Rebuild	<p>Rebuild takes things a step further by completely rebuilding an app using Azure cloud technologies. For example, you could build green-field apps with cloud-native technologies such as Azure Functions, Azure AI, Azure SQL Database Managed Instance, and Azure Cosmos DB.</p>	<p>When you want rapid development, and existing apps have limited functionality and lifespan. When you're ready to expedite business innovation by using Azure DevOps practices. When are building new applications using cloud-native technologies, like Azure Blockchain. When you are rebuilding legacy apps as no code or low apps in the cloud.</p>

Choosing which strategy to use depends on what you're trying to accomplish.

Migrate your workloads

After you complete the assessment, you can begin the process of migrating your targeted apps and their related services and data. The migration stage typically consists of the following elements:

Deploy cloud infrastructure targets: Before you can migrate Tailwind Traders' workloads, you'll need to create the required cloud infrastructure targets. Depending on the tools you use to perform the migration, you might need to create the required Azure resources before you begin the migration. Some tools, such as Azure Migrate and Azure Database Migration Service, can create the target Azure resources for you.

Migrate workloads: It's a good idea to pilot your workload migration, and to choose a non-critical app for the pilot. This approach enables you to:

- Become familiar with tools
- Gain experience with processes and procedures
- Reduce risk when migrating large or complex workloads

Depending on the workload you plan to migrate, the steps used to perform the migration will vary.

Decommission on-premises infrastructure: After you're satisfied that your source apps and databases are migrated successfully, you must decommission those source workloads. Consider retaining the source workload backups and archived data. This data might prove useful as it provides a historical archive. You can store these backups and archives in Azure Blob storage.

Optimize the migrated workloads

During the optimization stage, you should:

- Analyze the costs of the workload migration

- Identify ways to reduce costs
- Seek to improve workload performance

To analyze costs, you can use Cost Management in the Azure portal. Select the Azure resource group in which you're interested (the one which contains the migrated workloads), and then, in the navigation pane, select **Costs analysis** in the **Cost Management** section. The following screenshot shows the cost analysis for the last billable period for a resource group called ContosoResourceGroup. The results display the costs according to service name, region, and resource, although you can customize the results to your needs.



To help to reduce costs, select the **Advisor recommendations** link in the navigation pane. Recommendations, if there are any, are displayed on the details pane.

Monitor your workloads

You can use Azure Monitor to capture health and performance information from your Azure VMs. However, you must first install a Log Analytics agent on target VMs. After you've installed the agent, you can then set up alerting and reporting.

Tip: You can install the agent on machines running either Windows or Linux.

You can set up alerts based on a range of data sources, such as:

- Specific metric values like CPU usage
- Specific text in log files
- Health metrics
- An Autoscale metric

Compare migration tools

After you've assessed Tailwind Traders' on-premises workloads, you should begin to consider how to migrate those workloads to Azure. You can use several tools in Azure Migrate to migrate your workloads, depending on your needs.

Describe Azure Migrate

Azure Migrate is a set of features located in a centralized hub that you can use to assess and migrate different workloads to Azure. You can use Azure Migrate to perform the migration of workloads, including apps and VMs. Workloads that can be migrated to Microsoft Azure include on-premises servers, infrastructure, applications, and data.

Azure Migrate components include:

- Unified migration platform: A single portal where you can perform migration to Azure and track the migration status.
- Assessment and migration tools: Azure migration tools consist of multiple assessment and migration tools, including Azure Migrate: Server Assessment and Azure Migrate: Server Migration and other independent software vendor (ISV) tools.
- Assessment and migration of different workloads: There are several different workloads that you can migrate with Azure Migrate hub, including:
 - Servers: On-premises servers are assessed and migrated to Azure VMs.
 - Databases: On-premises databases are assessed and migrated to Azure SQL Database or to an Azure SQL Databaseâ€œmanaged instance.
 - Web applications: On-premises web applications are assessed and migrated Azure App Service by using the Azure App Service Migration Assistant.
 - Virtual desktops: On-premises virtual desktop infrastructure (VDI) is assessed and migrated to Azure Virtual Desktop.
 - Data: Large volumes of data are migrated to Azure by using Azure Data Box products.

The following table describes the different migration tools you can use, depending on the migration scenario.

Tool	Migration scenario
Azure Migrate: Server Assessment	Performs an assessment for physical servers and on-premises VMs running in Hyper-V and VMware environments as preparation for migrating to Azure.
Azure Migrate: Server Migration	Performs migration for physical servers and on-premises VMs running in Hyper-V, VMware environments, and other public cloud VMs.
Azure Migrate: Database Assessment	Performs an assessment of on-premises Microsoft SQL Server databases as preparation for migration to Azure SQL Database, an Azure SQL Databaseâ€œmanaged instance, Databaseâ€œmanaged instance, or Azure VMs running SQL Server.
Azure Migrate: Database Migration	Performs an assessment as preparation for migration to Azure VMs running SQL Server, Azure SQL Database, or Azure SQL Databaseâ€œmanaged instances.
Azure Migrate: Web App Assessment	Performs an assessment of on-premises web apps and migrates them to Azure. Uses the Azure App Service Migration Assistant to perform assessment and migration.
Azure Migrate: Data Box	Performs a move of large amounts of offline data to Azure by using Azure Data Box.

What can you do with Azure Migrate?

Azure Migrate can help with several migration scenarios. The one that you select depends on what youâ€™re trying to achieve. The six major migration scenarios are:

- Windows Server workloads

- SQL Server workloads
- Linux workloads
- Windows apps, Java apps and PHP apps
- SAP HANA
- Specialized compute

Migrate web apps to Azure

Azure Migrate uses the Azure App Service Migration Assistant to assess and migrate your web apps. The Azure App Service Migration Assistant enables you to assess and migrate your on-premises Windows ASP.NET web apps to Azure. By using this assistant, you can:

- Determine whether your app is a suitable migration candidate.
- Run readiness checks to perform a general assessment of the app's configuration settings.
- Migrate the app to the Azure App Service.

The Migration Assistant uses an agent that you install locally, and then use to perform a detailed analysis of your apps. You can then use the tool to migrate those apps to Azure. After the initial assessment of your app is complete, you're guided through the migration process using a graphical wizard-driven interface.

After moving the app to Azure, you may also consider migrating any connected databases.

Important: The Migration Assistant migrates your web application and its associated configurations, but does not migrate any backend databases connected to the app. You can use the SQL Migration Tool to complete the migration of your database.

Migrate VMs with Azure Migrate

After you've selected the appropriate server workloads, you're ready to begin the migration. There are four main technical implementation steps involved in moving a server workload to an Azure VM workload using Azure Migrate. These are:

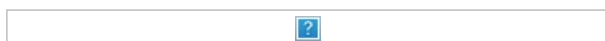
1. Prepare Azure for the Azure Migrate: Server Migration tool.
2. Prepare the on-premises VMs for migration.
3. Replicate the on-premises VMs.
4. Migrate the VMs.



Describe Azure Resource Mover

Azure Resource Mover is a tool that helps move your Azure resources between subscriptions, resource groups, and regions. The tool can be used:

- Before you migrate, to organize your resources.
- After you migrate, to optimize your resource organization.



Resource Mover provides:

- A single location for moving resources.
- Simplicity and speed in moving resources.
- A consistent interface and procedure for moving different types of Azure resources.
- A way to identify dependencies across resources that you want to move.
- Automatic clean-up of resources in the source region.
- The ability to test a move operation before you commit it.

Consider using Azure Resource Mover when, after migration to Azure, and you need to move any of your recently migrated resources across subscriptions, regions or resource groups.

Migrate your databases

Most applications use a database to store the data used by an application. It's important that you know how to migrate databases to Azure to properly support Tailwind Traders' move to the cloud.

What is the Azure Database Migration Service?

The Azure Database Migration Service is part of Azure Migrate. You can use the Database Migration Service to migrate your on-premises databases. This includes:

- Azure VMs running SQL Server
- Azure SQL Database (Database Migration Assistant)
- SQL Managed Instances
- Cosmos DB
- Azure DB for MySQL
- Azure DB for PostgreSQL

The Database Migration Service is a fully managed service. The migration service provides two different ways to migrate SQL Server databases:

- Online migration: An online migration uses a continuous synchronization of live data, allowing a cut over to the Azure replica database at any time. Online migration minimizes downtime.
- Offline migration: An offline migration requires shutting down the server at the start of the migration, which means downtime for the service.

Overview of database migration

When you begin the migration process, it's the **Data Migration Assistant** that guides you through the process. This process consists of three main elements:

1. Assess the databases you want to migrate
2. Migrate the schema: Separate the schema from the databases, and then recreate the schema in the target Azure SQL Database instances
3. Copy the databases data to the target instances and then verify the migrated databases

Prerequisites

Both online and offline migrations have the same prerequisite tasks:

- Download the Data Migration Assistant.
- Create an Azure Virtual Network instance.
- Configure the network security group.
- Configure the Windows Firewall.
- Configure credentials.
- Provision your target database in Azure.

Size the target database appropriately for the migrated workload.

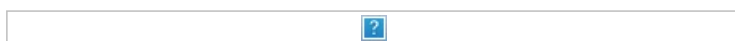
1. Assess the on-premises databases

After you've verified all the prerequisites are met, you're ready to begin the migration. This starts with the assessment of your on-premises environment.

You'll use the Data Migration Assistant to conduct the assessment. The assessment generates a report after completion, including a set of recommendations and alternative approaches that could be taken for the migration.

You'll be able to review any compatibility issues between the source and destination databases that could cause the migration to fail. Address the issues in the report, running it as many times as you need to make sure that the issues have been fixed.

The following screenshot displays a typical Data Migration Assistant report.



2. Migrate the schema by using the Data Migration Assistant

Each database has a schema that represents its entire structure. The schema defines the rules for how the data is organized and the relationships between data elements.

You migrate the schema before you migrate all the data in the database. Doing so:

- Creates an empty structure on the new Azure SQL database. This structure matches that of the on-premises source database.
- Validates connectivity before you do the full data migration. The Data Migration Assistant creates and runs a script to take the required actions.

When the script is complete, check the target server to make sure the database has been configured correctly.

3. Migrate your data with Database Migration Service

After you've conducted your assessment, and created the schema, you can migrate the data.

When these steps are complete, your schema and data have been migrated to the Azure SQL Database instance. You can then shut down and decommission your on-premises databases and servers.

Select an online storage migration solution

In addition to migrating apps, VMs, and databases, it's often necessary to migrate unstructured data. This data might be stored in several locations and be liable to frequent changes. Therefore, migration of storage containing unstructured data can be challenging.

When considering how to migrate online on-premises unstructured data, consider the following options:

- The Windows Server Storage Migration Service
- Azure File Sync

The Windows Server Storage Migration Service is part of Windows Admin Center.

Overview of the Windows Storage Migration Service

Consider using the Storage Migration Service if you have one or more servers that you want to migrate to newer hardware or VMs. By using the Storage Migration Service, you can:

- Conduct an inventory of your servers and their data.
- Rapidly transfer files, file shares, and security configuration from the source servers.
- Take over the identity of the source servers (known as cutting over). This means that users and apps don't have to change anything to access existing data.
- Manage one or multiple migrations from the Windows Admin Center interface.

Migrate data with the Storage Migration Service

Migration consists of the following three steps:

1. Inventory servers to gather information about their files and configuration.
2. Transfer data from the source to the destination servers.
3. Optionally, cut over to the new servers.

Important: The destination servers assume the source servers' former identities so that apps and users don't have to change anything.

After migration, the source servers enter a maintenance state. While in this state, the source servers still contain their original files, but are unavailable to users and apps.

Tip: Don't remove files from the source servers until you're ready to completely decommission the servers at your convenience.

As displayed in the following graphic, you can use the Storage Migration Service to migrate data stored in on-premises file servers via Azure File Sync and Azure Migrate to Azure Files and Azure hosted VMs.



Requirements

To use Storage Migration Service, you require:

- A source server, or failover cluster, containing the data you want to migrate.
- A destination server, or failover cluster, to which you want to migrate the data.
- An Orchestrator server to manage the migration.
- A PC or server running Windows Admin Center to run the Storage Migration Service user interface.

Note: There are additional requirements in terms of security, the Storage Migration Service proxy service, and required firewall port settings.

Use Azure File Sync

Azure File Sync is a feature of Azure Files. Azure Files is an Azure service that provides the functionality of an on-premises file share with the benefits of a platform as a service (PaaS) cloud service. You can use Azure Files in several common scenarios as described in the following table.

Usage	Description
Replace or supplement on-premises file servers	Virtually all companies use file servers. Azure Files can completely replace or supplement traditional on-premises file servers or Network Attached Storage (NAS) devices. With Azure file shares and AD DS authentication, you can migrate

Lift and shift (rehome)

data to Azure Files and utilize high availability and scalability while minimizing client changes.

Azure Files makes it easy to lift-and-shift applications that expect a file share to store application or user data to the cloud.

Backup and disaster recovery

You can use Azure file shares as storage for backups, or for disaster recovery to improve business continuity. You can use Azure file shares to back up your data from existing file servers while preserving configured Windows discretionary access control lists (ACLs). Data that's stored on Azure file shares isn't affected by disasters that might affect on-premises locations.

Azure File Sync

With Azure File Sync, Azure file shares can replicate to Windows Server, either on-premises or in the cloud, for performance and distributed caching of data where it's being used.

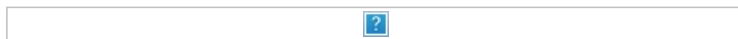
What is Azure File Sync?

Azure File Sync enables you to centralize your organization's file shares in Azure Files, while keeping the flexibility, performance, and compatibility of an on-premises file server. You can also use Azure File Sync to cache Azure file shares on Windows Server computers for fast access close to where the data is accessed. You can use any protocol that's available on Windows Server to access your data locally, including SMB, NFS, and FTPS.

In addition to using Azure Disks as back-end storage, you can utilize both Azure Files and a file server that's hosted in Azure VMs by installing Azure File Sync on a file server that's hosted on a cloud VM. If the Azure file share is in the same region as your file server, you can enable cloud tiering and set the volume of free space percentage to maximum (99%). This ensures minimal duplication of data. You also can use any applications you want with your file servers, such as applications that require NFS protocol support.

Azure File Sync terminology

If you want to understand how File Sync works, you must understand the terms that relate to it. The following diagram uses this terminology to depict how Azure File Sync works.



- The server running Windows Server in this diagram has the Azure File Sync agent and is registered with Azure File Sync.
- Next to this server are two sync groups: Accounting and Sales.
- The Accounting sync group has D:\Accounting as the server endpoint and the Sales sync group has D:\Sales as the server endpoint.
- Each sync group has a two-way interaction with the cloud endpoint, which means that the server endpoint syncs its content with the cloud endpoint content (the Azure file share is the cloud endpoint).
- Both cloud endpoints have a two-way interaction with the same Storage Sync Service.
- Azure File Sync uses the Storage Sync Service.
- Storage Sync Service has a two-way interaction with the Azure storage account, which symbolizes that the cloud endpoints (Azure file shares) are created in the Azure storage account.
- The storage account has two-way interaction with Azure Backup, which means the Azure storage account can be backed up by using Backup.

After you've configured Azure File Sync, data on the configured on-premises server endpoints is synchronized to Azure Files.

Consider using Azure File Sync when you want to migrate shared folder content to Azure. This is especially useful as a means for replacing the Distributed File System on your Windows Servers in your on-premises datacenters.

Select an offline storage migration solution

There are two choices for migrating offline data to Azure. We'll first cover Azure Import/Export. Then we'll cover Azure Data Box. A short comparison table is provided at the end.

Overview of Azure Import/Export

Azure Import/Export is an Azure service that's used to migrate large quantities of data between an on-premises location and an Azure Storage account. By using the service, you send and receive physical disks that contain your data between your on-premises location and an Azure datacenter. You ship data that's stored on your own disk drives. These disk drives can be Serial ATA (SATA) hard-disk drives (HDDs) or solid-state drives (SSDs).

When to use Azure Import/Export

The Azure Import/Export service is ideally suited to situations where you must upload or download large amounts of data, but your network backbone doesn't have sufficient capacity or reliability to support large-scale transfers. You typically use this service to:

- Migrate large amounts of data from on-premises to Azure, as a one-time task
- Back up your data on-premises in Azure Storage
- Recover large amounts of data that you previously stored in Azure Storage
- Distribute data from Azure Storage to customer sites

How Azure Import/Export works

To use Azure Import/Export, you create a job that specifies the data that you want to import or export. You then prepare the disks to use to transfer the data. For an import job, you write your data to these disks and ship them to an Azure datacenter. Microsoft uploads the data for you. For an export job, you prepare a set of blank disks and ship them to an Azure datacenter. Microsoft copies the data to these disks and ships them back to you. Here are a few other things to know.

- You can use the Import/Export service to export data from Azure Blob storage only. You can't export data that's stored in Azure Files.
- BitLocker must be enabled on the Windows system.
- You'll need an active shipping carrier account like FedEx or DHL for shipping drives to an Azure datacenter.
- If you're exporting, you'll need a set of disks that you can send to an Azure datacenter. The data center will use these disks to copy the data from Azure Storage.

Overview of Azure Data Box

Azure Data Box provides a quick, reliable, and inexpensive method for moving large volumes of data to Azure. By using Microsoft Azure Data Box, you can send terabytes of data into and out of Azure. The solution is based on a secure storage device which is shipped to your organization. Your Data Box might include disks, ruggedized server chassis, or mobile disk.

There are several products to fit different scenarios: **Data Box**, **Data Box Disk**, and **Data Box Heavy**. The configuration process is basically the same across all the products. After you receive your storage device, you can quickly set it up using the local web-based management interface. Assuming you're exporting data to Azure, copy the required data to the storage device, and then return it to Azure.

Note: You will define your encryption keys for the storage device. The entire process is tracked end-to-end by the Data Box service in the Azure portal.

When should you use Data Box?

Data Box is ideally suited to transfer data sizes larger than 40 TBs. It's especially useful in scenarios with limited internet connectivity. You could consider using Data Box in the following situations.

Scenario	Explanation
One time migration	When you want to migrate a large amount of on-premises data to Azure. For example: moving a media library from offline tapes into Azure to create an online media library; migrating your VM farm, SQL server, and applications to Azure; moving historical data to Azure for in-depth analysis and reporting using HDInsight. You perform an initial bulk transfer with Data Box and follow it with incremental

Initial bulk transfer

transfers over the network. For example: you move large volumes of historical backup to Azure. After this data is added, you continue to maintain the archive with incremental data is transfers via network to Azure storage.

Periodic uploads

When you have large volumes of data which is generated periodically. You want to move this data to Azure. For example, data generated by sensors from customer connected IoT devices.

What are the Data Box components?

The Data Box includes the following components:

Component	Description
Data Box device	A physical device that provides primary storage, manages communication with cloud storage, and helps to ensure the security and confidentiality of all data stored on the device. The Data Box device has a usable storage capacity of 80 TB.
Data Box service	An extension of the Azure portal that lets you manage a Data Box device using a web interface that you can access from different geographical locations. Use the Data Box service to perform daily administration of your Data Box device. The service tasks include how to create and manage orders, view, and manage alerts, and manage shares.
Data Box local web-based user interface	A web-based UI that is used to configure the device so that it can connect to the local network, and then register the device with the Data Box service. Use the local web UI also to shut down and restart the Data Box device, view copy logs, and contact Microsoft Support to file a service request.

How to select between Azure Import/Export and Azure Data Box

Capacity	Azure Import/Export	Azure Data Box
Form factor	Internal SATA HDDs or SDDs	Secure, tamper-proof, single hardware appliance
Microsoft manages shipping logistics	No	Yes
Integrates with partner products	No	Yes
Custom appliance	No	Yes

Tip: If you're looking to import or export more moderate volumes of data to and from Azure Blob storage, consider using other tools like AzCopy or Azure Storage Explorer.