

Configure Azure Storage security

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/configure-storage-security/1-introduction>

Introduction

Completed

Azure Storage provides a comprehensive set of security capabilities that work together to enable developers to build secure applications.

In this module, your company is storing sensitive data in Azure Storage, including personal information. The data is used internally and by external application developers. You're responsible for ensuring the data is secure for all users. You're tasked with providing configuration solutions to grant secure access to the information.

Learning objectives

In this module, you learn how to:

- Configure a shared access signature, including the uniform resource identifier (URI) and SAS parameters.
- Configure Azure Storage encryption.
- Implement customer-managed keys.
- Recommend opportunities to improve Azure Storage security.

Skills measured

The content in the module helps you prepare for [Exam AZ-104: Microsoft Azure Administrator](#).

Prerequisites

- Familiarity with navigating the Azure portal.

2. Review Azure Storage security strategies

<https://learn.microsoft.com/en-us/training/modules/configure-storage-security/2-review-strategies>

Review Azure Storage security strategies

Completed

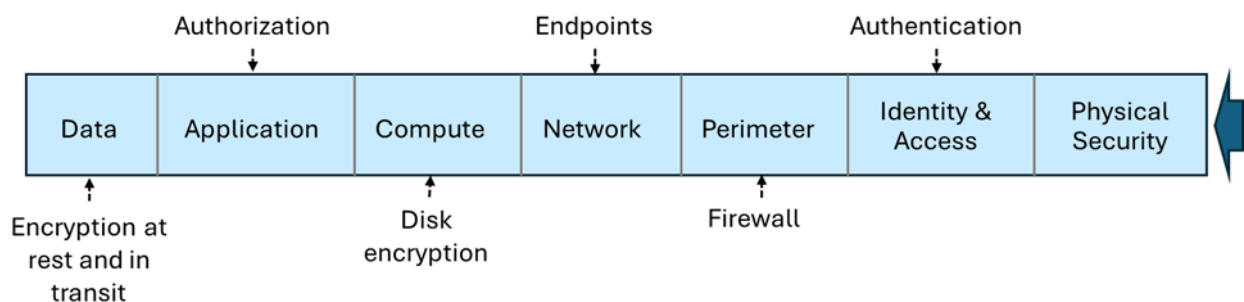
Administrators use different strategies to ensure their data is secure. Common approaches include encryption, authentication, authorization, and user access control with credentials, file permissions, and private signatures. Azure Storage offers a suite of security capabilities based on common strategies to help you secure your data.

Note

The video refers to Active Directory, which is now branded as Microsoft Entra ID.

Things to know about Azure Storage security strategies

Let's look at some characteristics of Azure Storage security. As you go through this module, consider defense in depth. How can you apply storage security features to this concept?



- **Encryption at rest.** Storage Service Encryption (SSE) with a 256-bit Advanced Encryption Standard (AES) cipher encrypts all data written to Azure Storage. When you read data from

Azure Storage, Azure Storage decrypts the data before returning it. This process incurs no extra charges and doesn't degrade performance. Encryption at rest includes encrypting virtual hard disks (VHDs) with Azure Disk Encryption. This encryption uses BitLocker for Windows images, and uses dm-crypt for Linux.

- **Encryption in transit.** You can configure your storage account to only accept requests from secure connections by setting the **Secure transfer required** property for the storage account. Existing accounts should explicitly disallow TLS 1.0 and 1.1, which are deprecated.
- **Encryption models.** Azure supports various encryption models, including server-side encryption that uses service-managed keys, customer-managed keys in Key Vault, or customer-managed keys on customer-controlled hardware. With client-side encryption, you can manage and store keys on-premises or in another secure location.
- **Authorize requests.** For optimal security, Microsoft recommends using Microsoft Entra ID with managed identities to authorize requests against blob, queue, and table data, whenever possible. Authorization with Microsoft Entra ID and managed identities provides superior security and ease of use over Shared Key authorization.
- **RBAC.** RBAC ensures that resources in your storage account are accessible only when you want them to be, and to only those users or applications whom you grant access. Assign RBAC roles scoped to an Azure storage account.
- **Storage analytics.** Azure Storage Analytics performs logging for a storage account. You can use this data to trace requests, analyze usage trends, and diagnose issues with your storage account.

Tip

The Microsoft [storage cloud security benchmark](#) provides recommendations on how you can secure your cloud storage solutions.

Things to consider when using authorization security

Review the following strategies for authorizing requests to Azure Storage. Think about what security strategies would work for your Azure Storage.

Authorization strategy	Description
Microsoft Entra ID	Microsoft Entra ID is Microsoft's cloud-based identity and access management service. With Microsoft Entra ID, you can assign fine-grained access to users, groups, or applications by using role-based access control.
Shared Key	Access is authorized with an account access key. The key can be the primary or secondary access key. To enforce Entra ID authorization, disable the Shared Key at the storage account level.

Authorization strategy	Description
Shared access signatures	A SAS delegates access to a particular resource in your Azure storage account with specified permissions and for a specified time interval.
Anonymous access to containers and blobs	Anonymous public access is disabled by default on new storage accounts. Microsoft recommends keeping anonymous access disabled for accounts containing sensitive data.

3. Create shared access signatures

<https://learn.microsoft.com/en-us/training/modules/configure-storage-security/3-create-shared-access-signatures>

Create shared access signatures

Completed

A [shared access signature \(SAS\)](#) is a uniform resource identifier (URI) that grants restricted access rights to Azure Storage resources. SAS is a secure way to share your storage resources without compromising your account keys.

You can provide a SAS to clients who shouldn't have access to your storage account key. By distributing a SAS URI to these clients, you grant them access to a resource for a specified period of time. You'd typically use a SAS for a service where users read and write their data to your storage account.

- A *user delegation SAS* is secured with Microsoft Entra credentials and also by the permissions specified for the SAS. A user delegation SAS is supported for Blob Storage and Data Lake Storage,
- An *account-level SAS* to allow access to anything that a service-level SAS can allow, plus other resources and abilities. For example, you can use an account-level SAS to allow the ability to create file systems.

- A *service-level SAS* to allow access to specific resources in a storage account. You'd use this type of SAS, for example, to allow an app to retrieve a list of files in a file system, or to download a file.
- A *stored access policy* can provide another level of control when you use a service-level SAS on the server side. You can group SASs and provide other restrictions by using a stored access policy.

Recommendations for managing risks

Let's look at some recommendations that can help mitigate risks when working with a SAS.

Recommendation	Description
Always use HTTPS for creation and distribution	If a SAS is passed over HTTP and intercepted, an attacker can intercept and use the SAS. These <i>man-in-the-middle</i> attacks can compromise sensitive data or allow for data corruption by the malicious user.
Reference stored access policies where possible	Stored access policies give you the option to revoke permissions without having to regenerate the Azure storage account keys. Set the storage account key expiration date far in the future.
Set near-term expiry times for an unplanned SAS	If a SAS is compromised, you can mitigate attacks by limiting the SAS validity to a short time. This practice is important if you can't reference a stored access policy. Near-term expiration times also limit the amount of data that can be written to a blob by limiting the time available to upload to it.
Require clients automatically renew the SAS	Require your clients to renew the SAS well before the expiration date. By renewing early, you allow time for retries if the service providing the SAS is unavailable.
Plan carefully for the SAS start time	If you set the start time for a SAS to now, then due to clock skew (differences in current time according to different machines), failures might be observed intermittently for the first few minutes. In general, set the start time to at least 15 minutes in the past. Or, don't set a specific start time, which causes the SAS to be valid immediately in all cases. The same conditions generally apply to the expiry time. You might observe up to 15 minutes of clock skew in either direction on any request. For clients that use a REST API version earlier than 2012-02-12, the maximum duration for a SAS that doesn't reference a stored access policy is 1 hour. Any policies that specify a longer term fail.
Define minimum access permissions for resources	A security best practice is to provide a user with the minimum required privileges. If a user only needs read access to a single entity, then grant them read access to that single entity, and not read/write/delete access to all entities. This practice also helps lessen the damage if a SAS is compromised because the SAS has less power in the hands of an attacker.
Validate data written by using a SAS	When a client application writes data to your Azure storage account, keep in mind there can be problems with the data. If your application requires validated or authorized data, validate the data after written, but before used. This practice also protects against corrupt or malicious data being written to your account, either by a user who properly acquired the SAS, or by a user exploiting a leaked SAS.

Recommendation	Description
Don't assume a SAS is always the correct choice	In some scenarios, the risks associated with a particular operation against your Azure storage account outweigh the benefits of using a SAS. For such operations, create a middle-tier service that writes to your storage account after performing business rule validation, authentication, and auditing. Also, sometimes it's easier to manage access in other ways. If you want to make all blobs in a container publicly readable, you can make the container Public, rather than providing a SAS to every client for access.

4. Identify URI and SAS parameters

<https://learn.microsoft.com/en-us/training/modules/configure-storage-security/4-identify-uri-sas-parameters>

Identify URI and SAS parameters

Completed

When you create your shared access signature (SAS), a uniform resource identifier (URI) is created by using parameters and tokens. The URI consists of your Azure Storage resource URI and the SAS token.

Things to know about URI definitions

Let's look at an example URI definition and examine the parameters. This example creates a service-level SAS that grants read and write permissions to a blob. Consider how you might configure the parameters to support your Azure Storage resources.

```
https://myaccount.blob.core.windows.net/?restype=service&comp=properties&sv=2015-04
```

Parameter	Example	Description
Resource URI	<code>https://myaccount. blob .core.windows.net/ ? restype= service &comp=properties</code>	Defines the Azure Storage endpoint and other

Parameter	Example	Description
		parameters. This example defines an endpoint for Blob Storage and indicates that the SAS applies to service-level operations. When the URI is used with <code>GET</code> , the Storage properties are retrieved. When the URI is used with <code>SET</code> , the Storage properties are configured.
Storage version	<code>sv =2015-04-05</code>	For Azure Storage version 2012-02-12 and later, this parameter indicates the version to use. This example indicates that version 2015-04-05 (April 5, 2015) should be used.
Storage service	<code>ss =bf</code>	Specifies the Azure Storage to which the SAS applies. This example indicates that the SAS applies to Blob Storage and Azure Files.
Start time	<code>st =2015-04-29T22%3A18%3A26Z</code>	(Optional) Specifies the start time for the SAS in UTC time. This example sets the start time as April 29, 2015 22:18:26 UTC. If you want the SAS to be valid immediately, omit the start time.
Expiry time	<code>se =2015-04-30T02%3A23%3A26Z</code>	Specifies the expiration time for the SAS in UTC time. This example sets the expiry time as April 30, 2015 02:23:26 UTC.
Resource	<code>sr =b</code>	Specifies which resources are accessible via the SAS. This example specifies that the accessible resource is in Blob Storage.
Permissions	<code>sp =rw</code>	Lists the permissions to grant. This example grants access to read and write operations.

Parameter	Example	Description
IP range	<code>sip =168.1.5.60-168.1.5.70</code>	Specifies a range of IP addresses from which a request is accepted. This example defines the IP address range 168.1.5.60 through 168.1.5.70.
Protocol	<code>spr =https</code>	Specifies the protocols from which Azure Storage accepts the SAS. This example indicates that only requests by using HTTPS are accepted.
Signature	<code>sig =F%6GRVAZ5Cdj2Pw4tgU7IL STkWgn7bUkkAg8P6HESXwmf%4B</code>	Specifies that access to the resource is authenticated by using a Hash-Based Message Authentication Code (HMAC) signature. The signature is computed with a key using the SHA256 algorithm, and encoded by using Base64 encoding.

Tip

Continue your learning with the [Implement shared access signatures](#) training modules.

5. Determine Azure Storage encryption

<https://learn.microsoft.com/en-us/training/modules/configure-storage-security/5-determine-storage-service-encryption>

Determine Azure Storage encryption

Completed

Azure Storage encryption for data at rest protects your data by ensuring your organizational security and compliance commitments are met. The encryption and decryption processes happen automatically. Because your data is secured by default, you don't need to modify your code or applications.

When you create a storage account, Azure generates two 512-bit storage account access keys for that account. These keys can be used to authorize access to data in your storage account via Shared Key authorization, or via SAS tokens that are signed with the shared key.

Microsoft recommends that you use Azure Key Vault to manage your access keys, and that you regularly rotate and regenerate your keys. Azure Key Vault supports automatic key rotation policies, allowing you to define rotation schedules (for example, every 90 days) that rotate keys automatically. You can also manually rotate your keys when needed.

Things to know about Azure Storage encryption

Examine the following characteristics of Azure Storage encryption.

- Data is automatically encrypted before written to Azure storage.
- Data is automatically decrypted when retrieved.
- Azure Storage encryption, encryption at rest, decryption, and key management are transparent to users.
- All data written to Azure Storage is encrypted through 256-bit advanced encryption standard (AES) encryption. AES is one of the strongest block ciphers available.
- Azure Storage encryption is enabled for all new and existing storage accounts and can't be disabled.

Configure Azure Storage encryption

In the Azure portal, you configure Azure Storage encryption by specifying the encryption type. You can manage the keys yourself, or you can have the keys managed by Microsoft. Consider how you might implement Azure Storage encryption for your storage security.



211795eastus | Encryption

Storage account

Encryption

Encryption scopes

Storage service encryption protects your data at rest. Azure Storage encrypts your data as it's written in our datacenters, and automatically decrypts it for you as you access it.

Please note that after enabling Storage Service Encryption, only new data will be encrypted, and any existing files in this storage account will retroactively get encrypted by a background encryption process.

[Learn more about Azure Storage encryption](#)

Encryption selection

Enable support for customer-managed keys ⓘ

Blobs and files only

Infrastructure encryption ⓘ

Disabled

Encryption type

☒ Microsoft-managed keys

☐ Customer-managed keys

- **Infrastructure encryption.** [Infrastructure encryption](#) can be enabled for the entire storage account, or for an encryption scope within an account. When infrastructure encryption is enabled for a storage account or an encryption scope, data is encrypted twice—once at the service level and once at the infrastructure level—with two different encryption algorithms and two different keys.
- **Platform-managed keys.** Platform-managed keys (PMKs) are encryption keys generated, stored, and managed entirely by Azure. Customers don't interact with PMKs. The keys used for Azure Data Encryption-at-Rest, for instance, are PMKs by default.
- **Customer-managed keys.** Customer managed keys (CMK), on the other hand, are keys read, created, deleted, updated, and/or administered by one or more customers. Keys stored in a customer-owned key vault or hardware security module (HSM) are CMKs. Bring Your Own Key (BYOK) is a CMK scenario in which a customer imports (brings) keys from an outside storage location. This topic is discussed in more detail on the next page.

6. Create customer-managed keys

Create customer-managed keys

Completed

For your Azure Storage security solution, you can use Azure Key Vault to manage your encryption keys. The Azure Key Vault APIs can be used to generate encryption keys. You can also create your own encryption keys and store them in a key vault.

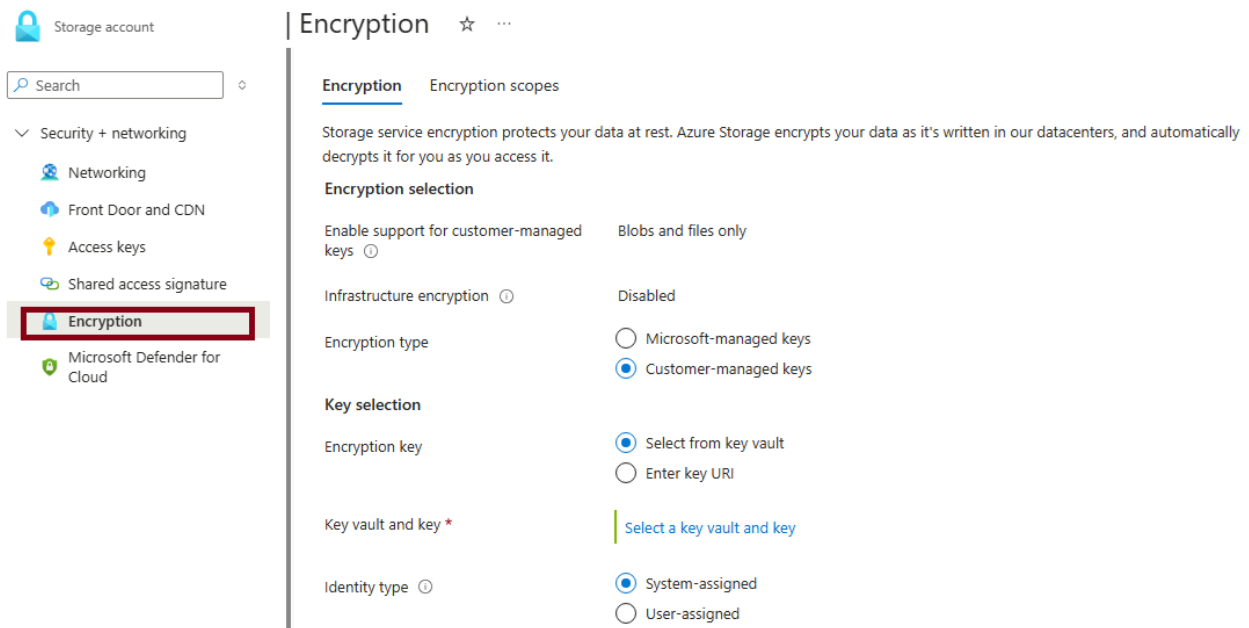
Things to know about customer-managed keys

Consider the following characteristics of customer-managed keys.

- By creating your own keys (referred to as *customer-managed keys*), you have more flexibility and greater control.
- You can create, disable, audit, rotate, and define access controls for your encryption keys.
- Customer-managed keys can be used with Azure Storage encryption. You can use a new key or an existing key vault and key. The Azure storage account and the key vault must be in the same region, but they can be in different subscriptions.
- Customer-managed keys are stored in a customer-owned Azure Key Vault or Azure Key Vault Managed HSM. Managed HSM provides FIPS 140-2 Level 3 validation for organizations with the highest compliance requirements.

Configure customer-managed keys

In the Azure portal, you can configure customer-managed encryption keys. You can create your own keys, or you can have the keys managed by Microsoft. Consider how you might use Azure Key Vault to create your own customer-managed encryption keys.



- **Encryption type:** Choose how the encryption key is managed: by Microsoft or by yourself (customer).
- **Encryption key:** Specify an encryption key by entering a URI, or select a key from an existing key vault.

Tip

Expand your understanding of storage security in the [Plan and implement security for storage](#) training module.

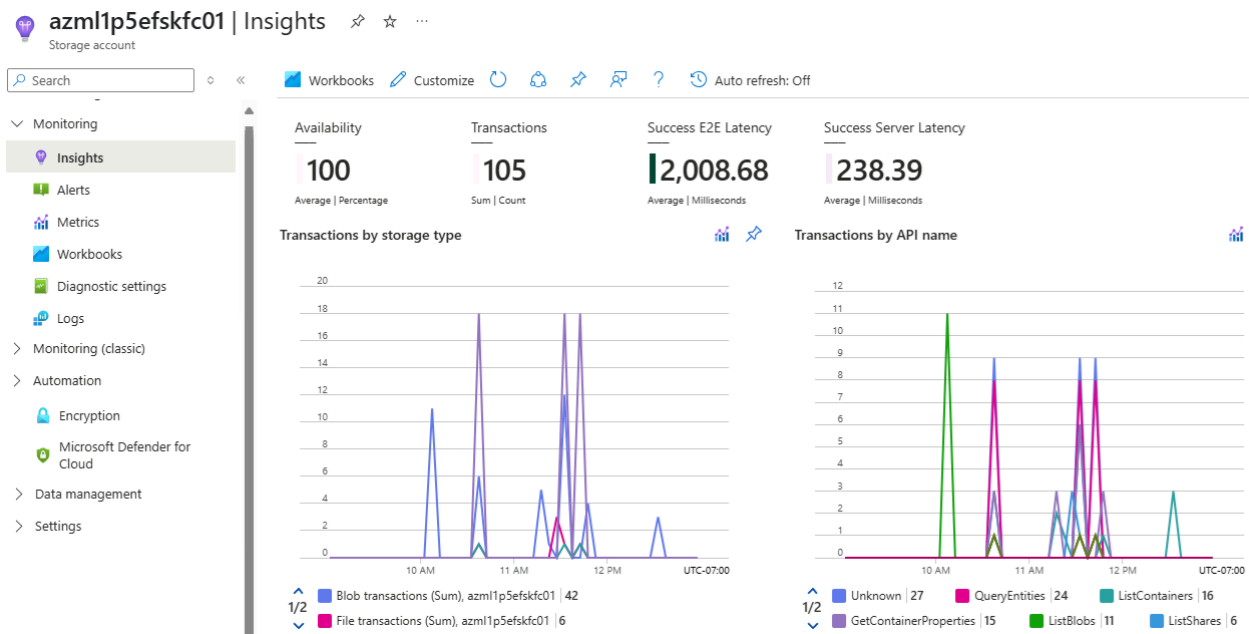
7. Apply Azure Storage security best practices

<https://learn.microsoft.com/en-us/training/modules/configure-storage-security/7-apply-best-practices>

Apply Azure Storage security best practices

Completed

[Storage insights](#) provides comprehensive monitoring of your Azure Storage accounts. Storage Insights delivers a unified view of your Azure Storage services performance, capacity, and availability.



What are the benefits of Storage insights?

- **Detailed Metrics and Logs.** Azure Storage Insights offers detailed metrics, logs, and diagnostic information that enhance visibility into storage operations. Insights helps in monitoring key performance indicators (KPIs) such as latency, throughput, capacity utilization, and transactions.
- **Enhanced Security and Compliance.** By using Azure Storage Insights, you can ensure enhanced security and compliance. It provides actionable insights and alerts that help in swiftly identifying and resolving security issues.
- **Role-Based Access Control (RBAC).** Azure Storage Insights integrates with Azure's security features, including role-based access control (RBAC), Microsoft Entra ID, connection strings, and access control list (ACL) permissions. RBAC ensures secure access to your data and resources.
- **Unified View.** It delivers a unified view of your Azure Storage services' performance, capacity, and availability, which is crucial for maintaining the security and efficiency of your storage accounts.

When to use Storage Insights

- **Real-Time Monitoring.** Azure Storage Insights enables real-time monitoring of storage accounts, allowing you to track usage trends, monitor performance, and set up alerts for any anomalies.
- **Security Auditing.** It aids in security auditing by providing comprehensive monitoring and detailed logs, which are essential for ensuring compliance and identifying any security issues.
- **Health Analysis and Optimization.** The tool helps in health analysis and optimization of storage accounts, ensuring security and optimal performance.

When to use Microsoft Defender for Storage

While Storage Insights provides passive monitoring and historical analysis, Microsoft Defender for Storage offers proactive threat detection for active security threats.

Key capabilities

- **Malware scanning.** Automatically scans blob uploads for malware and viruses.
- **Sensitive data threat detection.** Identifies when personally identifiable information (PII) or credentials are stored inappropriately.
- **Activity-based threat detection.** Monitors for unusual access patterns, suspicious download volumes, and hash reputation analysis.

Microsoft Defender for Storage complements Storage Insights by providing active threat detection rather than reactive monitoring and historical reporting.

8. Exercise: Manage Azure Storage

<https://learn.microsoft.com/en-us/training/modules/configure-storage-security/8-simulation-storage>

Exercise: Manage Azure Storage

Completed

Lab scenario

In this lab, you learn to create storage accounts for Azure blobs and Azure files. You learn to configure and secure blob containers. You also learn to use Storage Browser to configure and secure Azure file shares.

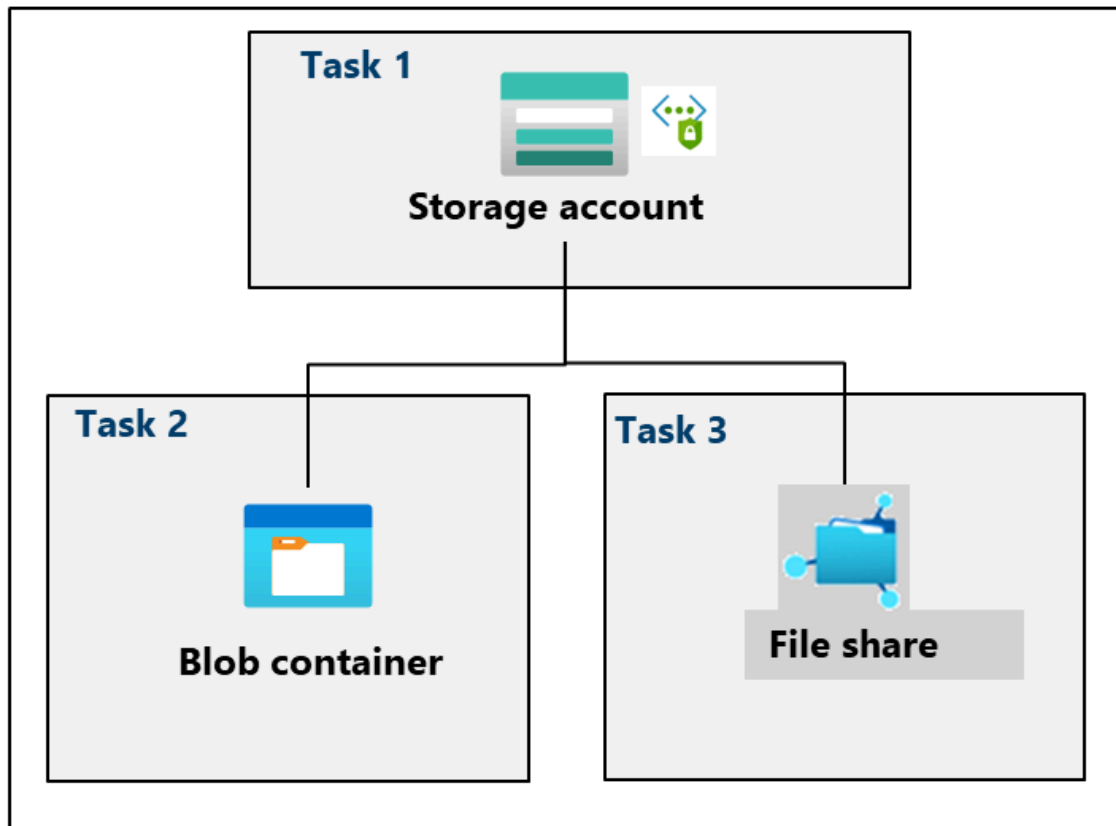
Note

This lab covers storage accounts, blobs, and files. As you go through the steps, consider the security features you learned about.

Architecture diagram



az104-07-rg7



Job skills

- Create and configure a storage account.
- Create and configure secure blob storage.
- Create and configure secure Azure file storage.

Note

Estimated timing: 50 minutes. To complete this exercise, you need an [Azure subscription](#).

Launch the exercise, and follow the instructions. When finished, be sure to return to this page so you can continue learning.

[Launch Exercise](#)

9. Module assessment

Module assessment

Completed

10. Summary and resources

<https://learn.microsoft.com/en-us/training/modules/configure-storage-security/10-summary-resources>

Summary and resources

Completed

Azure Administrators must be familiar with how to configure storage security.

In this module, you examined several options for securing Azure Storage. You discovered how to configure shared access signatures (SAS), including the uniform resource identifier (URI) and SAS parameters. You reviewed how to implement customer-managed keys and define stored access policies to configure Azure Storage encryption. You explored opportunities for improving your Azure Storage security solution.

Learn more with Copilot

Copilot can assist you in configuring Azure infrastructure solutions. Copilot can compare, recommend, explain, and research products and services where you need more information. Open a Microsoft Edge browser and choose Copilot (top right) or navigate to copilot.microsoft.com. Take a few minutes to try these prompts and extend your learning with Copilot.

- What are the different ways to secure Azure storage? Provide usage case examples.
- How do I configure an Azure Shared Access Signature?

Learn more with documentation

- Grant [limited access to Azure Storage resources with shared access signatures](#).
- Read about [Azure Storage encryption for data at rest](#).

- Create a [SAS](#) for your Azure storage account.
- Create a [service-level SAS](#).
- Construct a [user delegation SAS](#).
- Use [customer-managed keys](#) for Azure Storage encryption.

Learn more with self-paced training

- Secure your [Azure storage account](#).
- Implement [Azure Storage security](#).