Level: Advanced

Microsoft Azure Exam AZ-104 Certification

← Back to the **Course**

Azure Storage Access Management - **Practice Mode**

Completed on Sat, 11 Oct 2025

1st	2/5	40.00%	FAIL
Attempt	Marks Obtained	Your Score	Result

Share this Report in Social Media Share

Download Report

Domain wise Quiz Performance Report

No.	Domain	Total Question	Correct	Incorrect	Unattempted	Marked for Review
1	Implement and manage storage	5	2	3	0	0
Total	All Domains	5	2	3	0	0

Review the Answers

Filter By

Question 1

Incorrect

Domain: Implement and manage storage

You are responsible for securing access to a critical Azure Storage account used for processing regulatory data. The account must allow access from specific public IPs for auditing purposes, enable trusted Microsoft services for operations, and ensure that all other access, including from unknown networks, is denied. Additionally, only predefined Virtual Network subnets should interact with the storage account. Match the required configurations with their technical implementation steps by dragging and dropping the options into the corresponding answer area

Your Answers

A. Set AllowBlobPublicAccess to False

Deny all traffic not explicitly allowed

B. Enable "Trusted Microsoft Services" in the firewall

Configure access for trusted IPs

C. Define rules under "Networking > Firewall"

Enable exceptions for specific Azure services

D. Add subnet rules to the Virtual Network

Restrict access to predefined VNets

Correct Answers

A. Set AllowBlobPublicAccess to False

Deny all traffic not explicitly allowed

B. Enable "Trusted Microsoft Services" in the firewall

Enable exceptions for specific Azure services

C. Define rules under "Networking > Firewall"

Configure access for trusted IPs

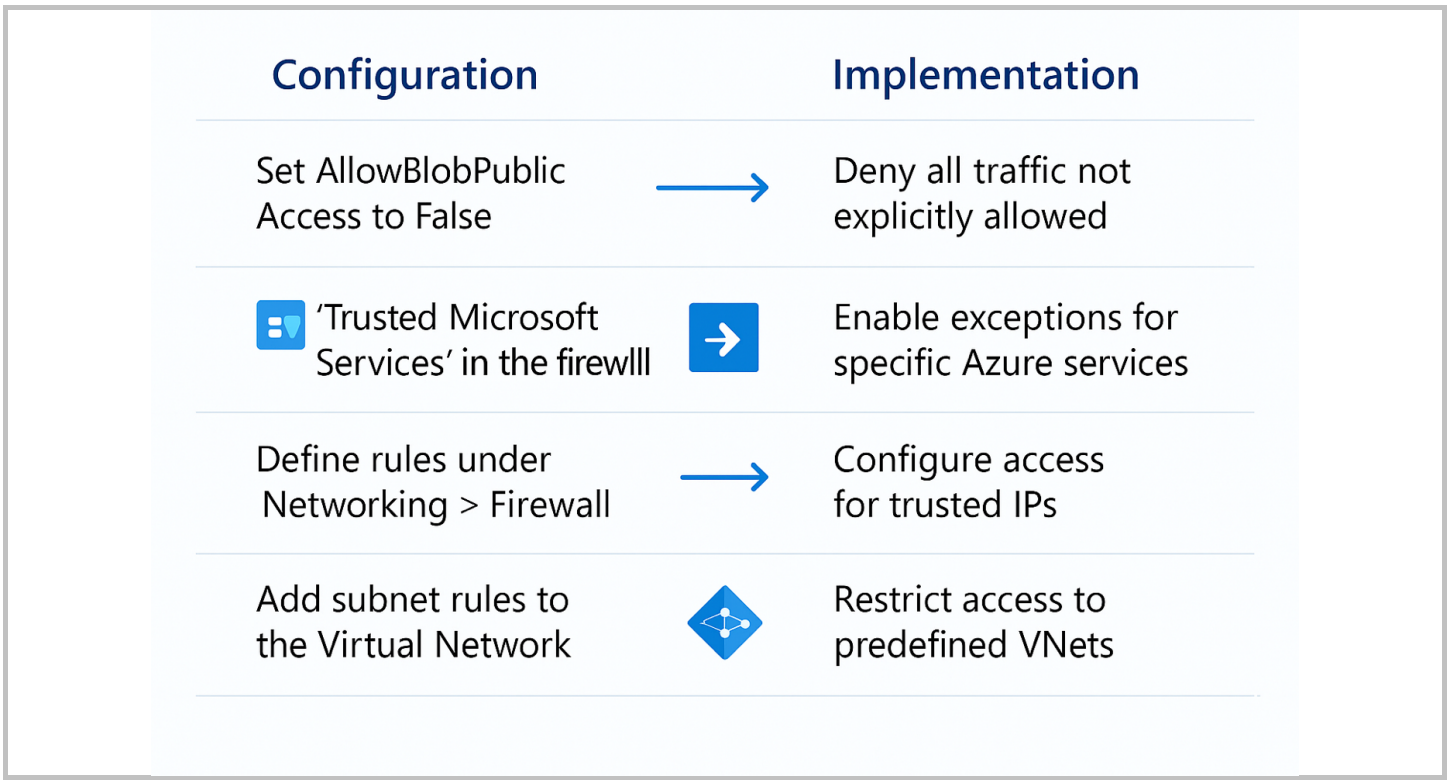
D. Add subnet rules to the Virtual Network

Restrict access to predefined VNets

Explanation:

Correct Answers: 1-B, 2-A, 3-C and 4-D

Configuration	Implementation
Set AllowBlobPublicAccess to False	Deny all traffic not explicitly allowed
Enable “Trusted Microsoft Services” in the firewall	Enable exceptions for specific Azure services
Define rules under “Networking > Firewall”	Configure access for trusted IPs
Add subnet rules to the Virtual Network	Restrict access to predefined VNets



Set AllowBlobPublicAccess to False - Deny all traffic not explicitly allowed: Setting AllowBlobPublicAccess to False ensures that no blob containers in the storage account are accessible to anonymous users, even if public access is inadvertently configured for a container. This is a critical setting to enforce a default-deny policy, which restricts all unauthorized traffic unless explicitly allowed. This configuration aligns with best practices for regulatory compliance and secures data against accidental exposure to the internet. It is particularly important for scenarios where sensitive data must be tightly controlled. Administrators can configure this setting using the Azure portal, PowerShell, or Azure CLI.

Enable "Trusted Microsoft Services" in the firewall - Enable exceptions for specific Azure services: Enabling trusted Microsoft services in the storage account firewall settings ensures that Azure services like Azure Backup, Azure Monitor, and Azure DevOps can securely interact with the storage account without being blocked by the firewall rules. This configuration is essential for operational scenarios where certain Azure services require uninterrupted access to the storage account for backups, logging, or other essential workflows. By enabling this option, you simplify connectivity while maintaining a secure boundary, as only trusted services under Microsoft's control are granted access. This setting is configured in the Azure portal under the "Firewall and virtual networks" section of the storage account.

Define rules under "Networking > Firewall" - Configure access for trusted IPs: Defining rules under the "Networking > Firewall" section allows you to specify public IP addresses or IP ranges (in CIDR notation) that are permitted to access the storage account. This configuration is crucial for scenarios where access must be granted to on-premises systems, remote administrators, or specific services with known IPs for tasks such as auditing or management. By explicitly defining trusted IP rules, you ensure that only authorized external sources can interact with the storage account while all other traffic remains blocked. This granular control is key to maintaining a secure and compliant environment.

Add subnet rules to Virtual Network - Restrict access to predefined VNets: Adding subnet rules to a Virtual Network restricts access to the storage account so that only resources within the specified VNets can communicate with it. This setup ensures that internal traffic between Azure resources remains secure and isolated, preventing any direct exposure to the public internet. For example, a database or application hosted within an authorized subnet can access the storage account while all other networks are blocked. This configuration is particularly important for scenarios involving sensitive or regulated workloads that require strict data access policies.

Administrators can define these rules in the Azure portal by linking the storage account to the appropriate VNet and subnets.

References:

https://learn.microsoft.com/en-us/azure/storage/blobs/anonymous-read-access-configure?tabs=portal#disable-anonymous-public-access-to-all-blob-data

https://learn.microsoft.com/en-us/azure/storage/common/storage-network-security?tabs=azure-portal#trusted-microsoft-services

https://learn.microsoft.com/en-us/azure/storage/common/storage-network-security?tabs=azure-portal#grant-access-from-an-internet-ip-range

https://learn.microsoft.com/en-us/azure/storage/common/storage-network-security?tabs=azure-portal#grant-access-to-trusted-azure-virtual-networks

Ask our Experts

Did you like this **Question?**

Question 2

Incorrect

Domain: Implement and manage storage

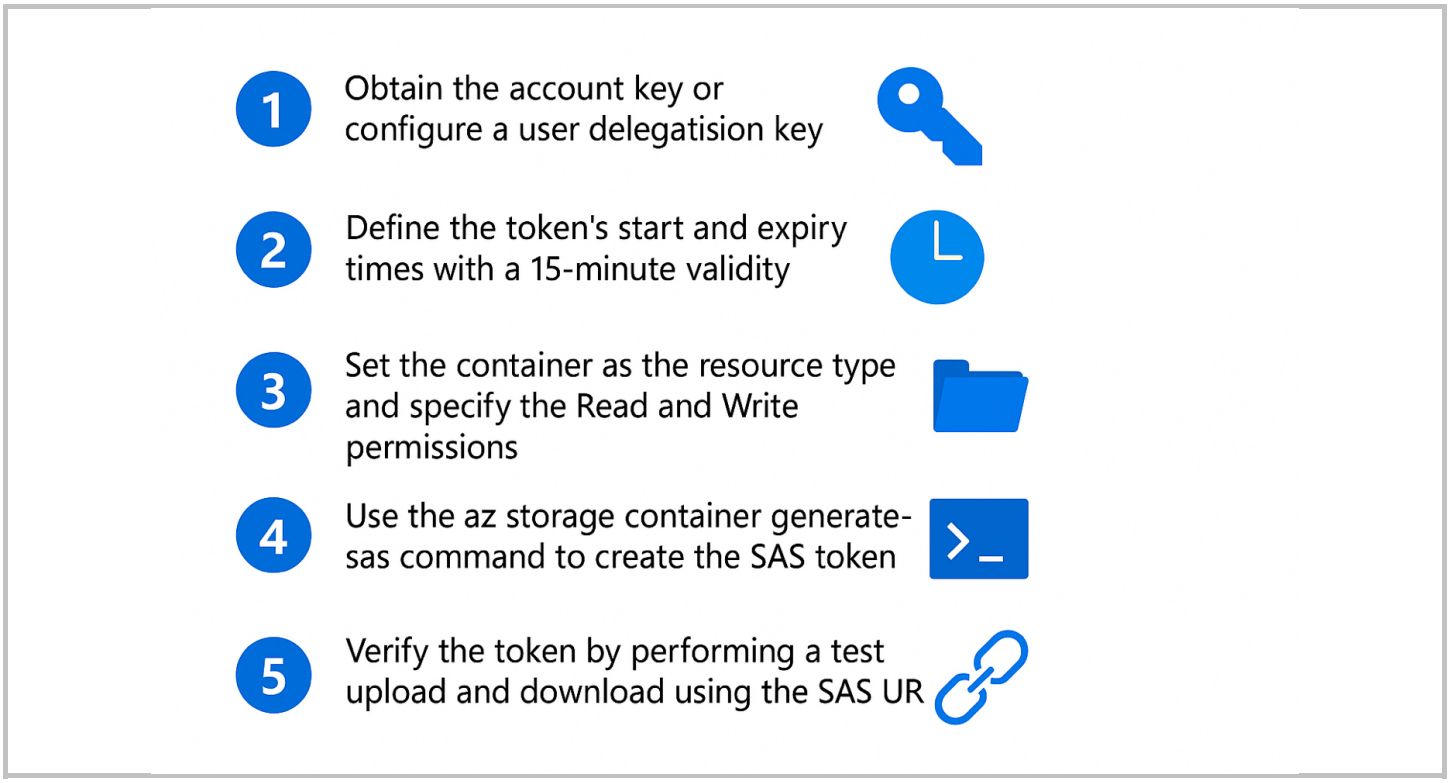
An application requires temporary access to download and upload blobs within a container in Azure Storage. To ensure security, the access must be restricted by a SAS token with a validity of 15 minutes and limited to Read and Write operations. Arrange the steps to create and validate the SAS token for this purpose.

Your Answer	Correct Answer
1. C. Set the container as the resource type and specify the Read and Write permissions	1. E. Obtain the account key or configure a user delegation key for token generation
2. E. Obtain the account key or configure a user delegation key for token generation	2. D. Define the token's start and expiry times with a 15-minute validity
3. D. Define the token's start and expiry times with a 15-minute validity	3. C. Set the container as the resource type and specify the Read and Write permissions
4. B. Use the az storage container generate-sas command to create the SAS token	4. B. Use the az storage container generate-sas command to create the SAS token
5. A. Verify the token by performing a test upload and download using the SAS URL	5. A. Verify the token by performing a test upload and download using the SAS URL

Explanation:

Correct Answers: E, D, C, B and A

1. (E) **Obtain the account key or configure a user delegation key** - Required for signing the SAS token securely.
2. (D) **Define the token's start and expiry times with a 15-minute validity** - This sets the limited access window for enhanced security.
3. (C) **Set the container as the resource type and specify Read and Write permissions** - Ensures only necessary actions are allowed on the container.
4. (B) **Use the az storage container generate-sas command to create the SAS token** - Generates the actual token using CLI and the specified parameters.
5. (A) **Verify the token by performing a test upload and download using the SAS URL** - Ensures the token works correctly with the intended permissions and validity.



Step 1 - Option E: Obtain the account key or configure a user delegation key for token generation is correct. To generate a SAS token, authentication with the Azure Storage account is required. This can be achieved either by using the account key or through Microsoft Entra ID integration to obtain a user delegation key. Using a user delegation key is recommended when enhanced security and fine-grained access control are needed, especially for scenarios involving Microsoft Entra ID. For this scenario, obtaining the account key or user delegation key is the foundational step, as it is required for signing the SAS token, ensuring that the token can enforce the defined access policies.

Step2 - Option D: Define the token's start and expiry times with a 15-minute validity is correct. After establishing the authentication credentials, the next step is to configure the SAS token's time-based validity. Defining both the start time and the expiry time ensures the token adheres to the temporary access requirements specified in the scenario. By setting a 15-minute validity period, you not only limit exposure but also align with security best practices, as shorter token lifespans reduce the risk of misuse if the token is compromised.

Step3 - Option C: Set the container as the resource type and specify the Read and Write permissions is correct. Once the token's validity is defined, the next step is to specify the resource to that the SAS token will apply, which, in this case, is the container. Configuring the permissions Read and Write ensures that the token allows only the necessary actions for downloading and uploading

blobs within the container. This step is crucial for maintaining the principle of least privilege, as it restricts access to just the operations needed for the application's functionality.

Step4 - Option B: Use the az storage container generate-sas command to create the SAS token is correct. With the permissions, resource type, and validity times defined, the next step is to generate the SAS token. The Azure CLI command az storage container generate-sas provides a streamlined way to create the token. This command uses the provided inputs resource type, permissions, time constraints, and authentication mechanism (account key or delegation key) to generate a valid SAS token. The resulting token is a URL string that incorporates all defined configurations, ready for use.

Step5 - Option A: Verify the token by performing a test upload and download using the SAS URL is correct. Validation is the final step to ensure the generated SAS token operates as intended. By appending the token to the container URL, the application can use it to perform test operations downloading and uploading blobs within the container. Successful execution confirms that the token has been correctly configured, while any failure may indicate a misconfiguration (e.g., incorrect permissions or validity period). Testing ensures the token meets security and operational requirements before deployment.

References:

https://learn.microsoft.com/en-us/rest/api/storageservices/create-user-delegation-sas

https://learn.microsoft.com/en-us/azure/storage/common/storage-sas-overview

https://learn.microsoft.com/en-us/cli/azure/storage/container?view=azure-cli-latest#az-storage-container-generate-sas

Ask our Experts

Did you like this **Question?**

Question 3

Correct

Domain: Implement and manage storage

A development team in your organization needs to store project files in an Azure Files share hosted in the storage account DevStorage1. They require access using Microsoft Entra ID credentials. The solution must enforce role-based access control (RBAC) at the storage account level and granular NTFS permissions at the file share level. Developers must be able to map the file share directly to their workstations without any manual configuration changes. Which three of the following actions should be taken to meet the requirements for access, permissions, and mapping the Azure Files share? (Select three)

Your Answer	Correct Answer
A. Enable identity-based access	A. Enable identity-based access
B. Assign SMB Contributor role	B. Assign SMB Contributor role
C. Configure NTFS permissions	C. Configure NTFS permissions

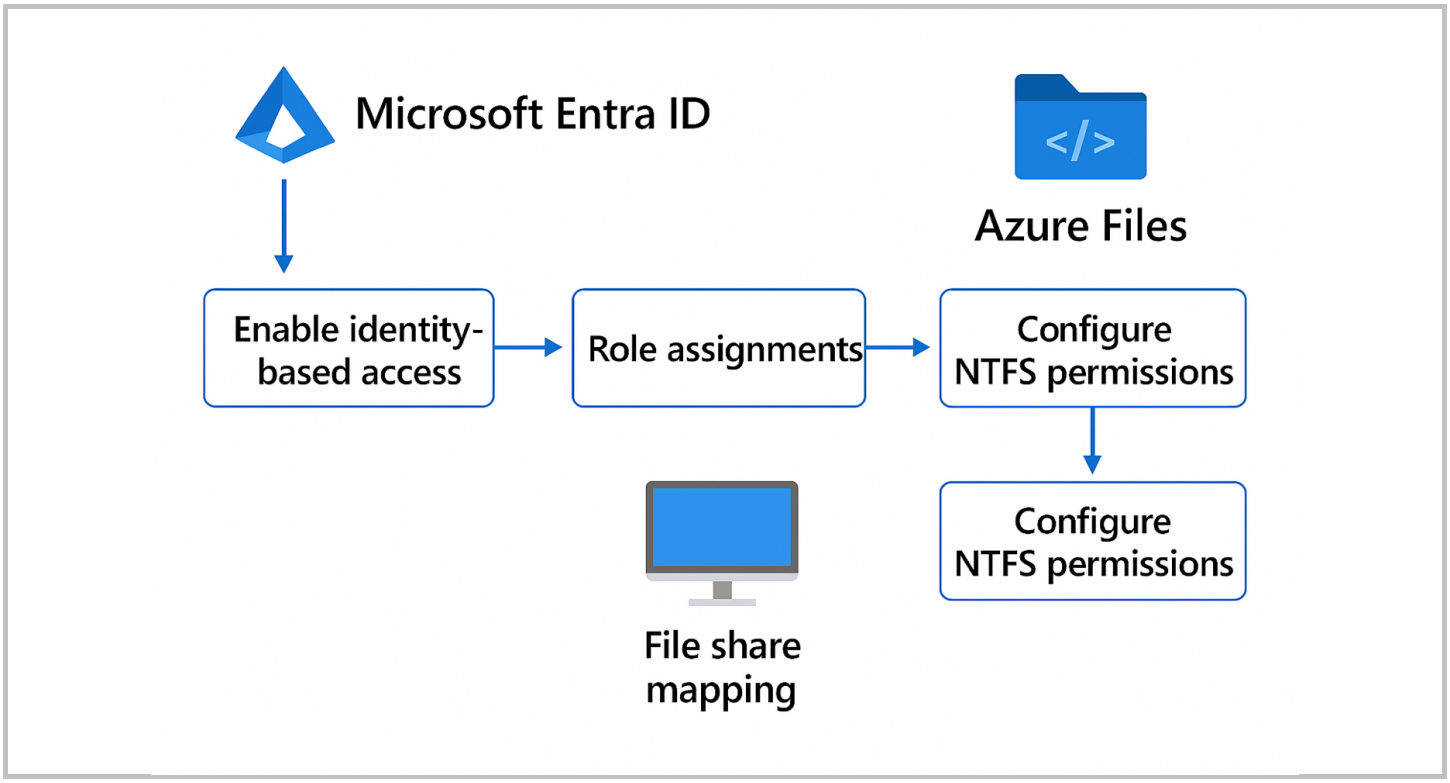
Explanation:

Correct Answers: A, B and C

Option A is correct because enabling identity-based access allows the storage account to integrate with Microsoft Entra ID for authentication. This step is critical because the developers require access to the Azure Files share using their Entra ID credentials. By enabling this feature, the storage account can enforce security policies aligned with the organization's identity management and access control mechanisms. This ensures seamless authentication and eliminates the need for manual key management or storage account keys. Without enabling this feature, the RBAC and NTFS permissions at the file level cannot be applied effectively.

Option B is correct because the SMB Contributor role is essential for granting the development team the necessary permissions to access and interact with the file share. Assigning this role enables role-based access control (RBAC) at the Azure storage account level, allowing developers to read, write, and delete files in the share. RBAC operates at the service level, which works alongside NTFS permissions to provide more granular control over file access. Without this role, users may be authenticated but will lack the required permissions to access or modify the files within the share.

Option C is correct because configuring NTFS permissions is essential for providing granular access control at the file level. While the SMB Contributor role grants access at the Azure level, NTFS permissions define what users can do at the individual file and folder level within the file share. This is a critical step to ensure that only authorized users can modify or view specific files. For example, different folders within the file share may require varying levels of access based on the developers' roles.



Option D is incorrect because while mapping a share using Entra ID credentials is part of the process for users to access the file share, it is not an action required to configure access or permissions for the Azure Files share. Mapping is a client-side operation performed by users to connect to the file share. This step assumes that identity-based access and appropriate permissions (RBAC and NTFS) are already configured.

Option E is incorrect because enabling Advanced Threat Protection (ATP) for Azure Storage is a security feature that detects and responds to anomalous activity, such as unusual access patterns or potential attacks. While ATP is a valuable security enhancement, it does not directly address the requirements for configuring access, permissions, or mapping the Azure Files share.

References:

https://learn.microsoft.com/en-us/azure/storage/files/storage-files-identity-ad-ds-overview

https://learn.microsoft.com/en-us/azure/storage/files/storage-files-active-directory-overview#azure-rbac-for-azure-files

https://learn.microsoft.com/en-us/azure/storage/files/storage-files-identity-configure-file-level-permissions

Ask our Experts

Did you like this **Question?**

Question 4 Correct

Domain: Implement and manage storage

An organization needs to provide time-limited access to a specific blob container for an external partner to upload files.

Proposed Solution: To achieve this, the administrator creates a stored access policy with write permissions and sets the expiry time to 24 hours. The associated shared access signature (SAS) is then generated and shared with the partner. Subsequently, the stored access policy is deleted before the partner attempts to upload files. Is this proposed solution correct? (Select Yes or No)

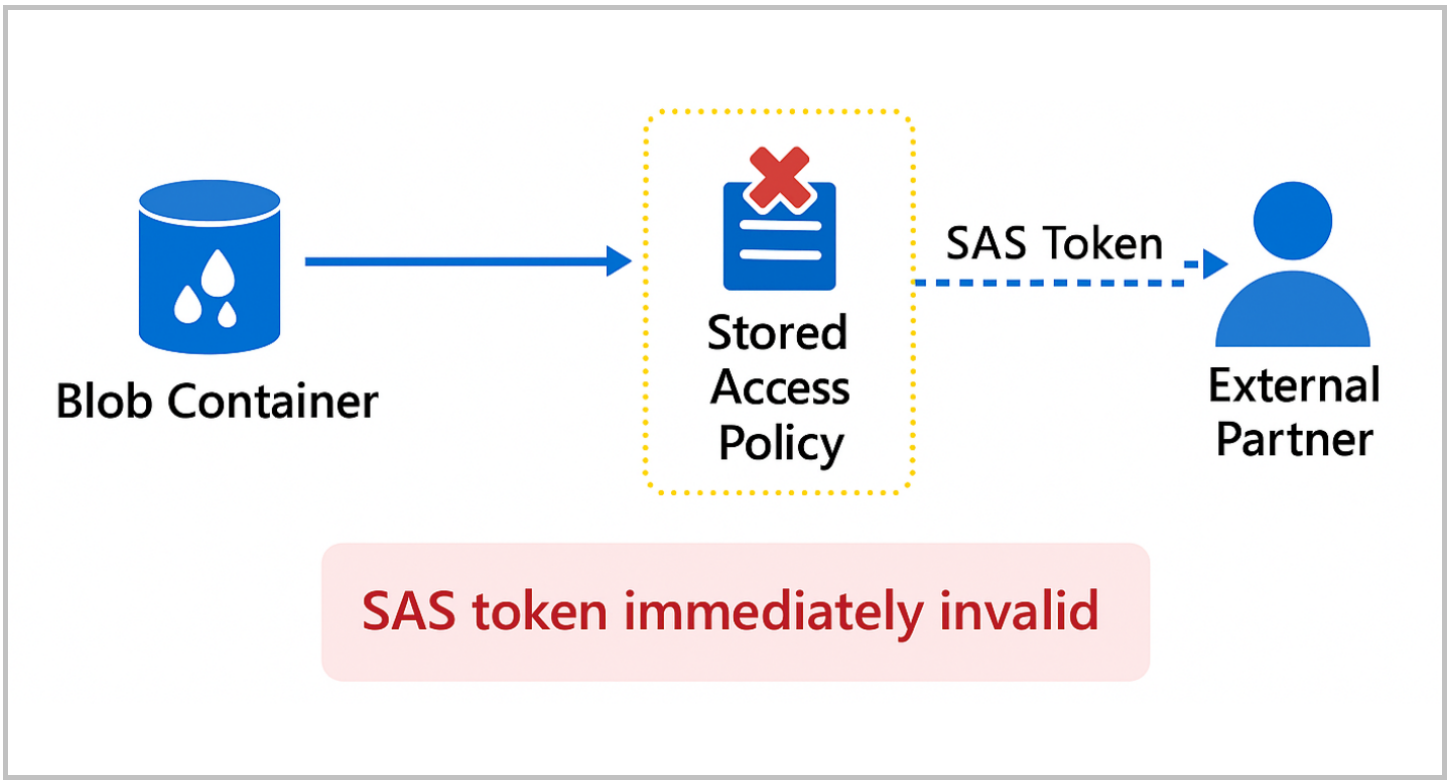
A. Yes

B. No right

Explanation:

Correct Answer: B

The proposed solution is incorrect because deleting the stored access policy immediately renders any shared access signature (SAS) tokens generated from that policy invalid. A SAS token, when tied to a stored access policy, depends on the existence and configuration of the policy to maintain its validity. Once the policy is deleted, all associated SAS tokens are no longer functional, even if their expiration time has not yet been reached. Stored access policies are a crucial feature in Azure Storage, as they enable centralized management of permissions and expiration times for SAS tokens. This allows administrators to update or revoke access permissions without needing to modify or regenerate the SAS tokens themselves. However, if the stored access policy is deleted, it revokes the underlying permissions of any SAS token associated with it. Consequently, the token holder, in this case, the external partner, will not be able to access the blob container, regardless of the token's expiration time.



This scenario emphasizes the need for a clear understanding of how stored access policies function in Azure Storage. If there is a need to adjust or revoke permissions for specific SAS tokens, administrators should modify the stored access policy or generate new tokens rather than deleting the policy altogether. Deleting a stored access policy should be done with the understanding that it will affect all SAS tokens linked to it.

Reference:

Grant limited access to data with shared access signatures (SAS)

Ask our Experts

Did you like this **Question?**

Question 5 Incorrect

Domain: Implement and manage storage

You are working as an Azure Administrator for a large enterprise that utilizes Azure Blob Storage for storing sensitive financial data. The company requires the implementation of strict security measures for accessing this data. As part of your task, you need to manage the access keys for a storage account used by the financial application. Which of the following actions should you perform to ensure that the access keys remain secure and accessible only when necessary? (Select three options)

A. Create and use Azure Managed Identity for accessing the storage account programmatically, without the need to use access keys right

B. Configure the storage account to allow only one active access key at a time for better security management wrong

C. Enable Azure Key Vault integration and store the access keys in Key Vault, using a Key Vault-backed secret to authenticate applications right

key usage right

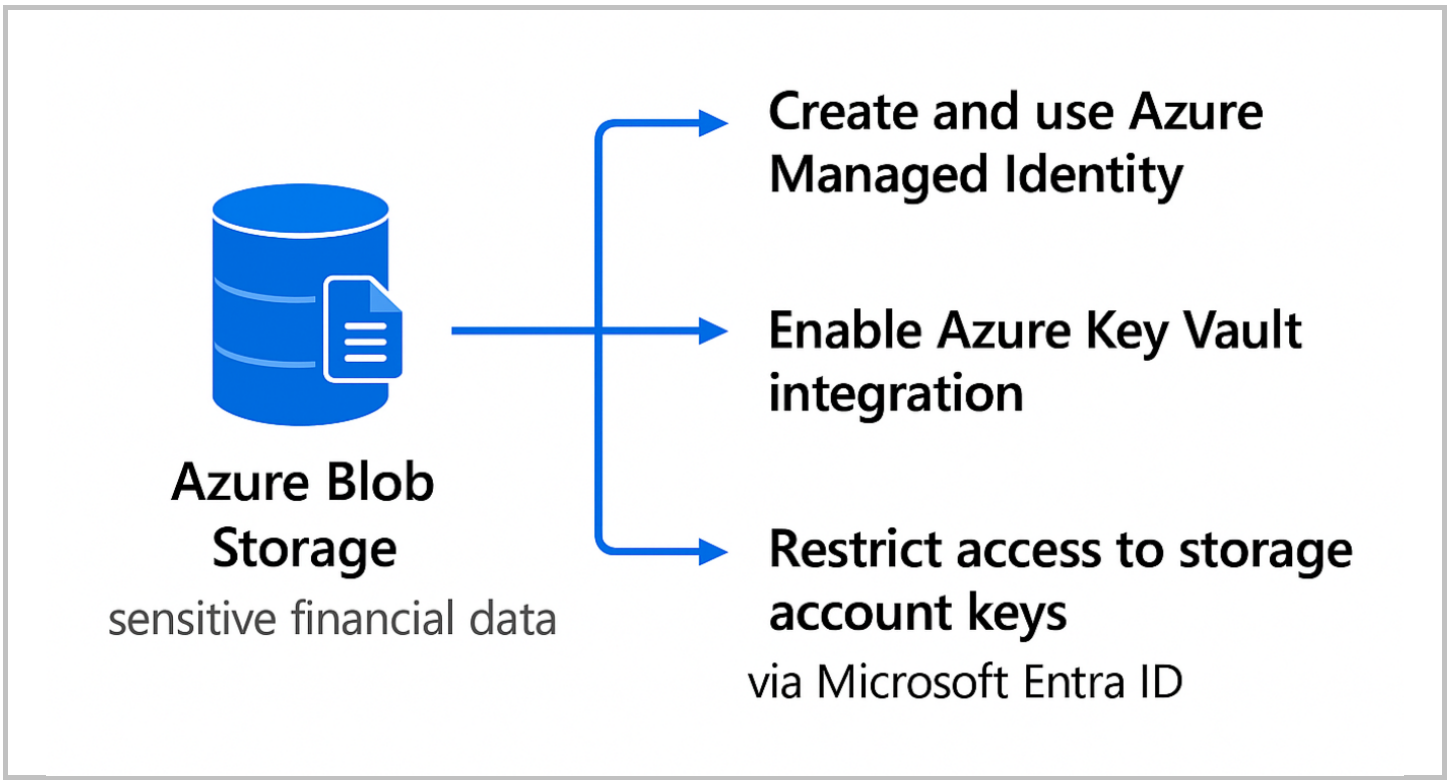
Explanation:

Correct Answers: A, C and E

Option A is correct because Azure Managed Identity is the most secure method for authenticating applications without using access keys or secrets. By enabling Managed Identity, you avoid the need to manually manage access keys, significantly reducing the security risk associated with their exposure. Managed Identity is an identity automatically assigned to resources in Azure and is used for Azure services to authenticate without using credentials. By using Managed Identity, your application can authenticate to the storage account without requiring access keys. This provides enhanced security since it eliminates the need to store keys in configuration files or hard-code them in code, reducing the risk of accidental exposure.

Option C is correct because storing access keys in Azure Key Vault provides enhanced security for managing and safeguarding keys. Azure Key Vault is specifically designed for securely storing secrets such as API keys, passwords, and access keys. When you store your storage account access keys in Key Vault, they are encrypted and managed centrally. The access is governed by Azure role-based access control (RBAC), meaning only authorized users or applications can retrieve them. Additionally, Key Vault facilitates key management by enabling the easy rotation, disabling, or regeneration of keys without needing to update your application configurations. Key Vault also offers audit capabilities, keeping track of who accessed the keys and when improving visibility and traceability. This approach reduces the risk of key exposure that might occur if they were stored directly in application code or configuration files.

Option E is correct because Microsoft Entra Authentication provides a secure method of managing access to storage accounts by replacing access keys with identity-based authentication. With this approach, applications and users are authenticated using their Microsoft Entra identities, eliminating the need for direct key usage. This enhances security by enforcing the principle of least privilege, allowing administrators to define precise access permissions for users, groups, or applications through role-based access control (RBAC). Administrators can specify roles such as read or write access, ensuring resources are accessed only by authorized entities. Additionally, identity-based authentication negates the need for key rotation, reducing the risk of key exposure. The method also supports detailed activity monitoring, offering insights into who accessed resources and when. By relying on identities, this solution prevents the exposure of access keys, ensuring that only authorized identities can interact with the storage account, thereby strengthening both security and compliance.



Option B is incorrect because while restricting the number of active keys can improve security, this option does not provide a robust solution on its own. Managing access keys manually (especially if you still use them for authentication) introduces risks related to key exposure. Storing and handling access keys, even with only one active key, still leaves the possibility of accidental exposure, unauthorized access, or mishandling. Additionally, rotating keys manually with this configuration can still cause disruptions in service if the applications do not retrieve the new keys correctly or in time. Using managed identities or Key Vault is a more secure approach that eliminates the risks associated with direct access key management.

Option D is incorrect because configuring the "Access keys" option to regenerate keys every 30 days while keeping old keys active during the transition introduces both security and operational challenges. Although the periodic regeneration of keys is intended to enhance security, this practice still relies on access keys for authentication, which is inherently less secure than using identity-based methods such as Microsoft Entra ID or Managed Identity. The 30-day interval leaves a prolonged window in which keys could potentially be compromised. Furthermore, retaining old keys during the transition period increases the risk of exposure, as both old and new keys could be accessed simultaneously, potentially by unauthorized users. This dual-key approach adds complexity to the process of securely updating applications, as administrators must ensure all applications switch to the new key within the allotted time frame. Delays or errors in this update process could result in downtime or continued reliance on insecure keys. By contrast, identity-based methods eliminate the need for key regeneration altogether, providing a more secure and streamlined solution.

References:

https://learn.microsoft.com/en-us/entra/identity/managed-identities-azure-resources/overview

https://learn.microsoft.com/en-us/azure/key-vault/general/overview

https://learn.microsoft.com/en-us/rest/api/storageservices/authorize-with-azure-active-directory

Ask our Experts

Did you like this **Question?**

Finish Review