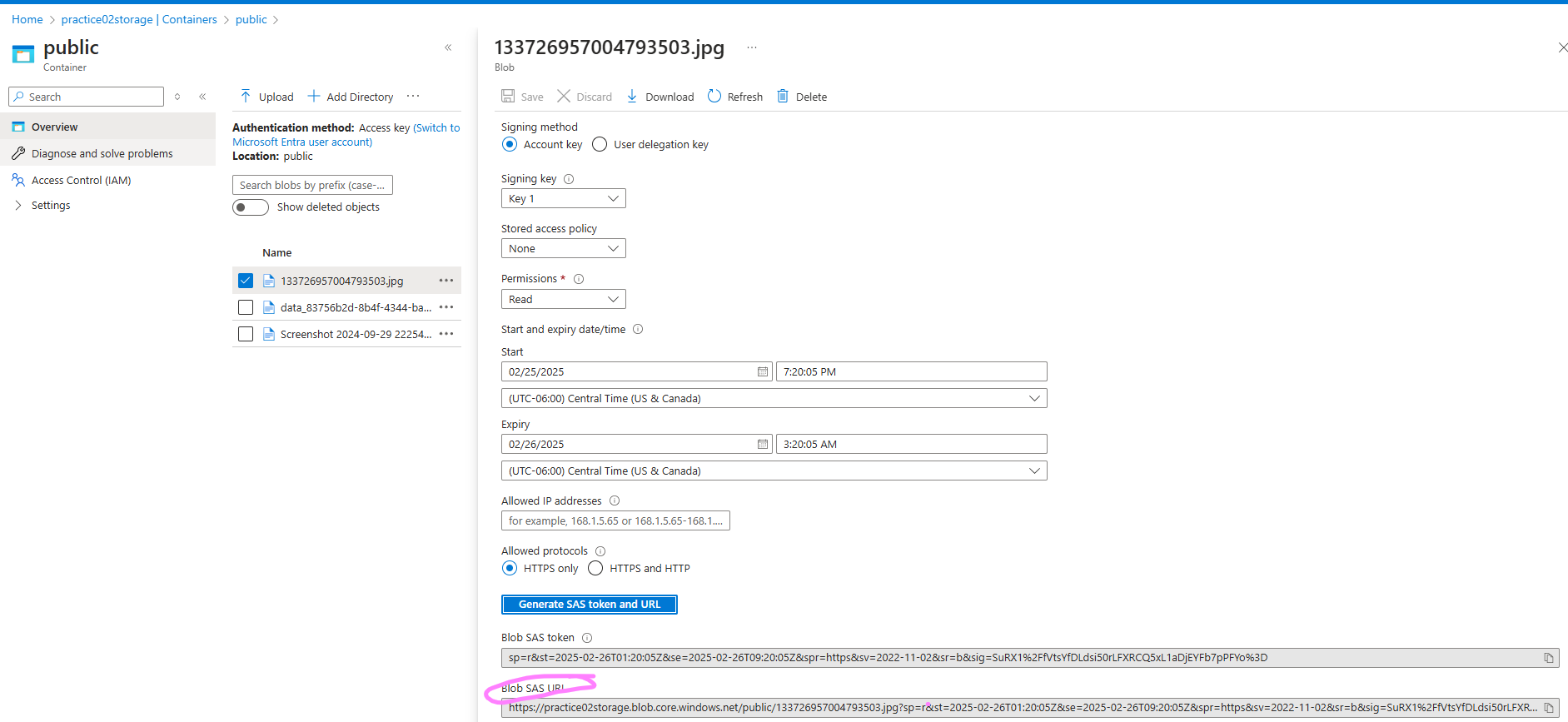
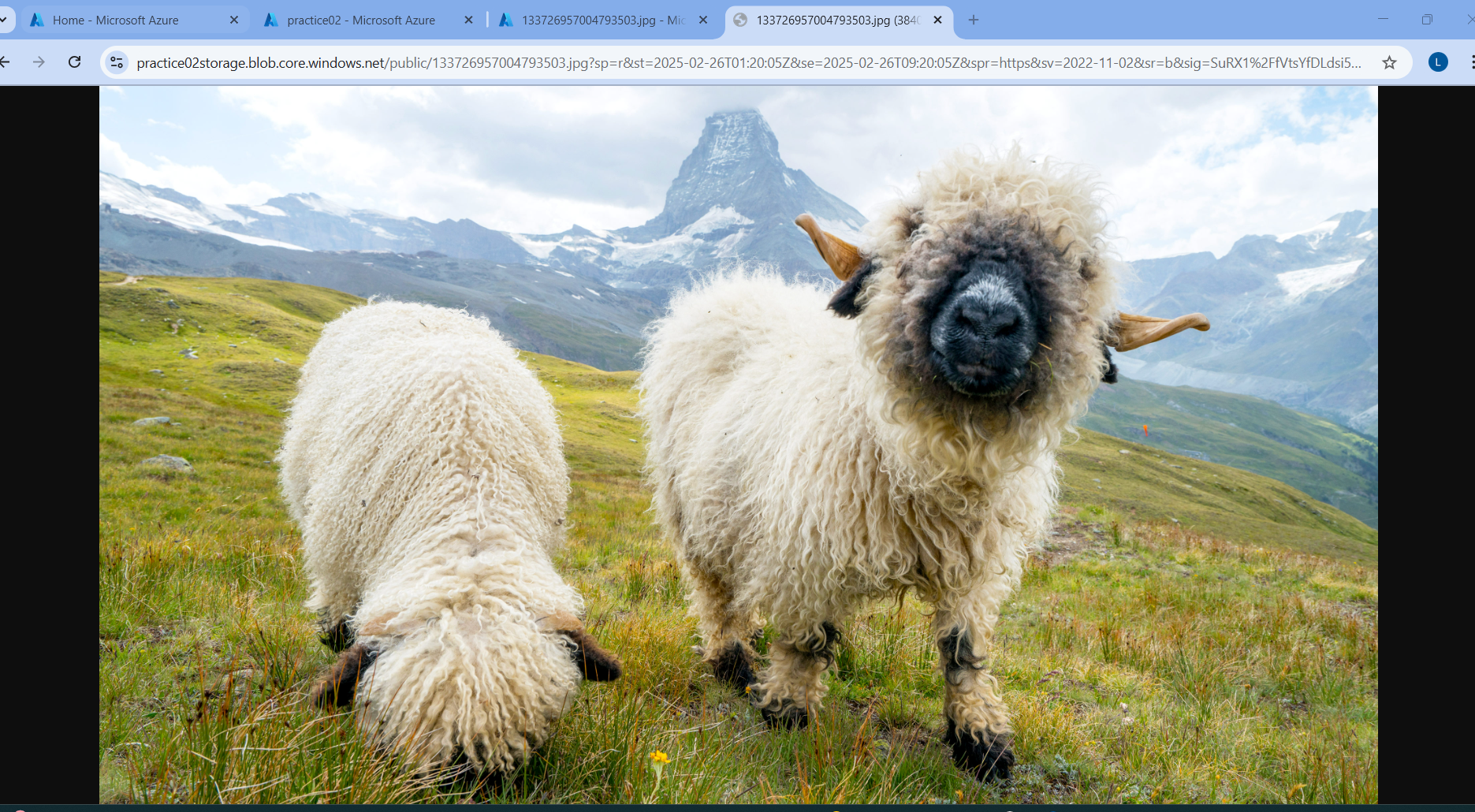
**Service level SAS:**

It is a SINGLE SERVICE, we can give access to aa blob

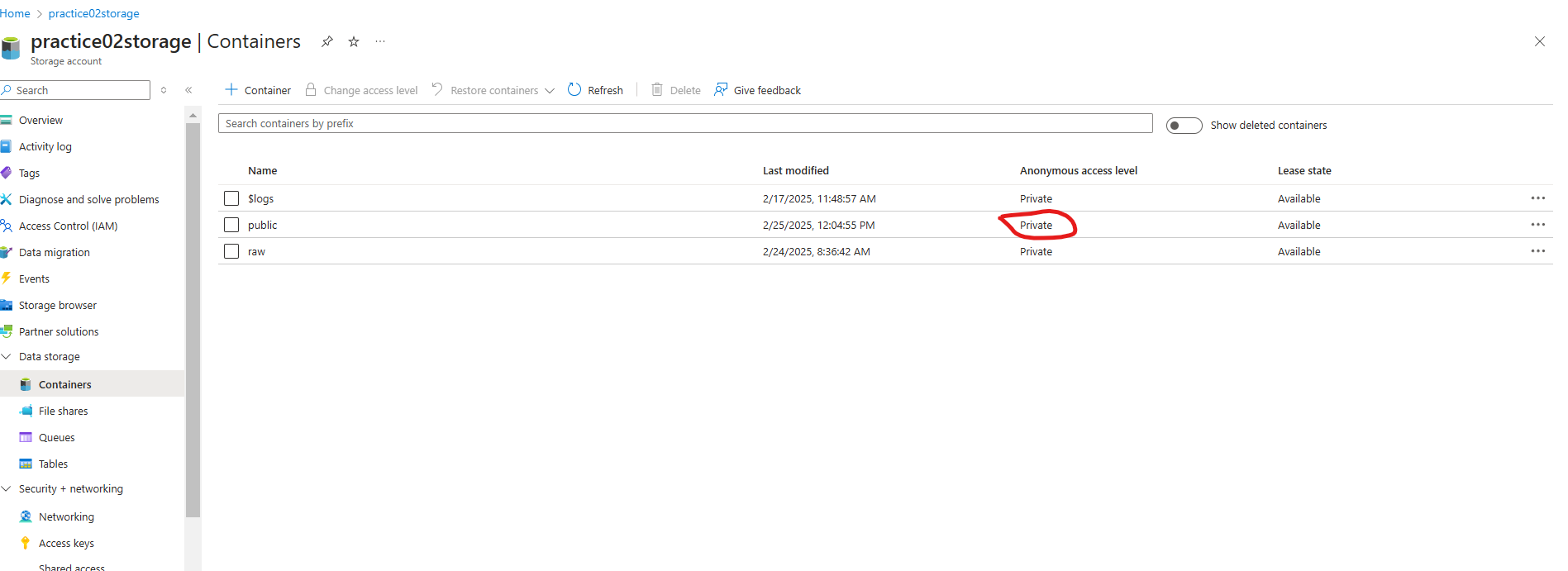
To give a Service level SAS we go to container or file inside the container to give access ; HERE I AM GIVING A FIGURE HOW WE GIVE ACCESS:



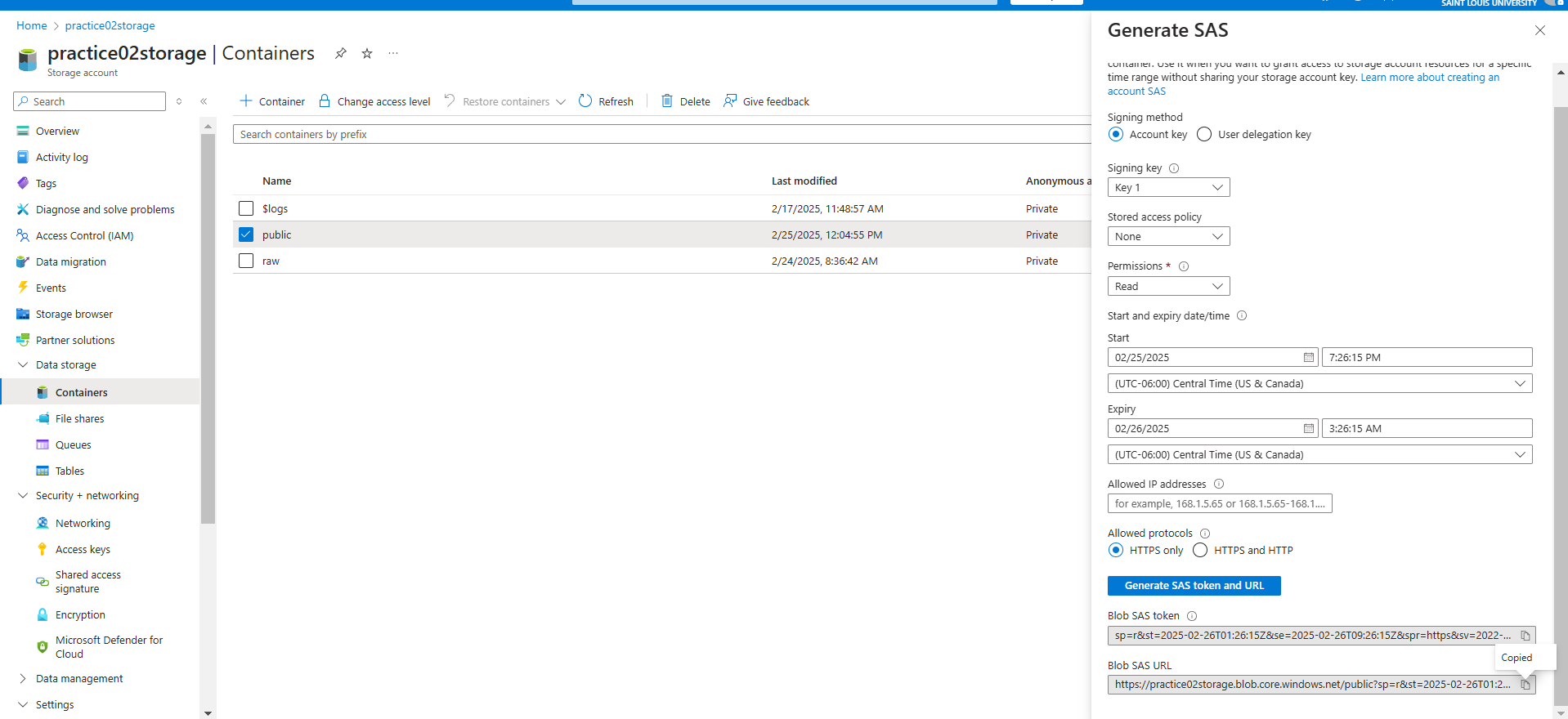
Here we can see the SAS URL generated and let us paste the URL in the private browser and see if we can access the image



We can see that we can actually access the image using that URL even though the access level is specified private ( SO SAS overrides the access level)



Now let me check what will happen at container level SAS token:



This is the container level token

I will paste that in the private browser and will see if i can access the files inside that:



It gave me the following error;

Let me paste the both URLS (SAS AT BLOB AND CONTAINER LEVELS AND EXPLAIIN)

<https://practice02storage.blob.core.windows.net/public/133726957004793503.jpg?sp=r&st=2025-02-26T01:20:05Z&se=2025-02-26T09:20:05Z&spr=https&sv=2022-11-02&sr=b&sig=SuRX1%2FfVtsYfDLdsi50rLFXRCQ5xL1aDjEYFb7pPFYo%3D>

<https://practice02storage.blob.core.windows.net/public/133726957004793503.jpg===>> THIS IS GENERAL LINK

&sv=2022-11-02 ===🡺

?sp=r

&st=2025-02-26T01:20:05Z

&se=2025-02-26T09:20:05Z

&spr=https

&sr=b

&sig=SuRX1%2FfVtsYfDLdsi50rLFXRCQ5xL1aDjEYFb7pPFYo%3D

| **Parameter** | **Value** | **Explanation** |
| --- | --- | --- |

|  |  |  |
| --- | --- | --- |
| **Storage Account Name** | practice02storage | This is your Azure Storage account name. |

|  |  |  |
| --- | --- | --- |
| **Container Name** | public | This is the container inside your storage account where blobs (files) are stored. |

|  |  |  |
| --- | --- | --- |
| **sp (Permissions)** | r | r = Read permission (can only read blobs inside the container). |

|  |  |  |
| --- | --- | --- |
| **st (Start Time)** | 2025-02-26T01:26:15Z | This SAS token is valid starting from **Feb 26, 2025, 01:26:15 UTC**. |

|  |  |  |
| --- | --- | --- |
| **se (Expiry Time)** | 2025-02-26T09:26:15Z | This SAS token **expires** on **Feb 26, 2025, 09:26:15 UTC**. |

|  |  |  |
| --- | --- | --- |
| **spr (Protocols)** | https | This token can only be used over a **secure HTTPS connection**. |

|  |  |  |
| --- | --- | --- |
| **sv (Storage Version)** | 2022-11-02 | The API version of Azure Storage being used. |

|  |  |  |
| --- | --- | --- |
| **sr (Resource Type)** | c | c = Container-level SAS (applies to everything inside the container). |

|  |  |  |
| --- | --- | --- |
| **sig (Signature)** | hgqT/Sz/Ubr0LEW9m8LKpweJvRDpRek4CUJFVIznLW0= | This is the **HMAC-SHA256 signature**, used to validate the token. |

Signature is formed by all the component before that ; even if we change something in date or something in URL to access ; we will get error signature will be generated based on previous components.

<https://practice02storage.blob.core.windows.net/public?sp=r&st=2025-02-26T01:26:15Z&se=2025-02-26T09:26:15Z&spr=https&sv=2022-11-02&sr=c&sig=hgqT%2FSz%2FUbr0LEW9m8LKpweJvRDpRek4CUJFVIznLW0%3D>

So as when I tried to access the link it is giving nothing so let me try to access the link with including the image of BLOB

Now I am changing the link by adding the name of the file inside the container named “public”

<https://practice02storage.blob.core.windows.net/public?sp=r&st=2025-02-26T01:26:15Z&se=2025-02-26T09:26:15Z&spr=https&sv=2022-11-02&sr=c&sig=hgqT%2FSz%2FUbr0LEW9m8LKpweJvRDpRek4CUJFVIznLW0%3D>

into

<https://practice02storage.blob.core.windows.net/public/133726957004793503.jpg?sp=r&st=2025-02-26T01:26:15Z&se=2025-02-26T09:26:15Z&spr=https&sv=2022-11-02&sr=c&sig=hgqT%2FSz%2FUbr0LEW9m8LKpweJvRDpRek4CUJFVIznLW0%3D>

By accessing the above link through a private browser I am getting the following :

Which means I can access the files inside the container with CONTAINER SAS=== IMPORTANT

**SHARED ACCESS POLICY:**

A Shared Access Policy (SAP) is a way to define and manage SAS (Shared Access Signature) tokens at the container or service level. Instead of creating ad-hoc SAS tokens, you can define a policy with specific permissions and expiration rules and then generate SAS tokens based on that policy.

**Why Use a Shared Access Policy?**

1. **Centralized Control** – You can define access policies once and apply them to multiple SAS tokens.
2. **Easy Revocation** – If you need to **revoke access**, you can simply delete or modify the policy.
3. **Consistency** – Ensures all SAS tokens follow the same security policies.

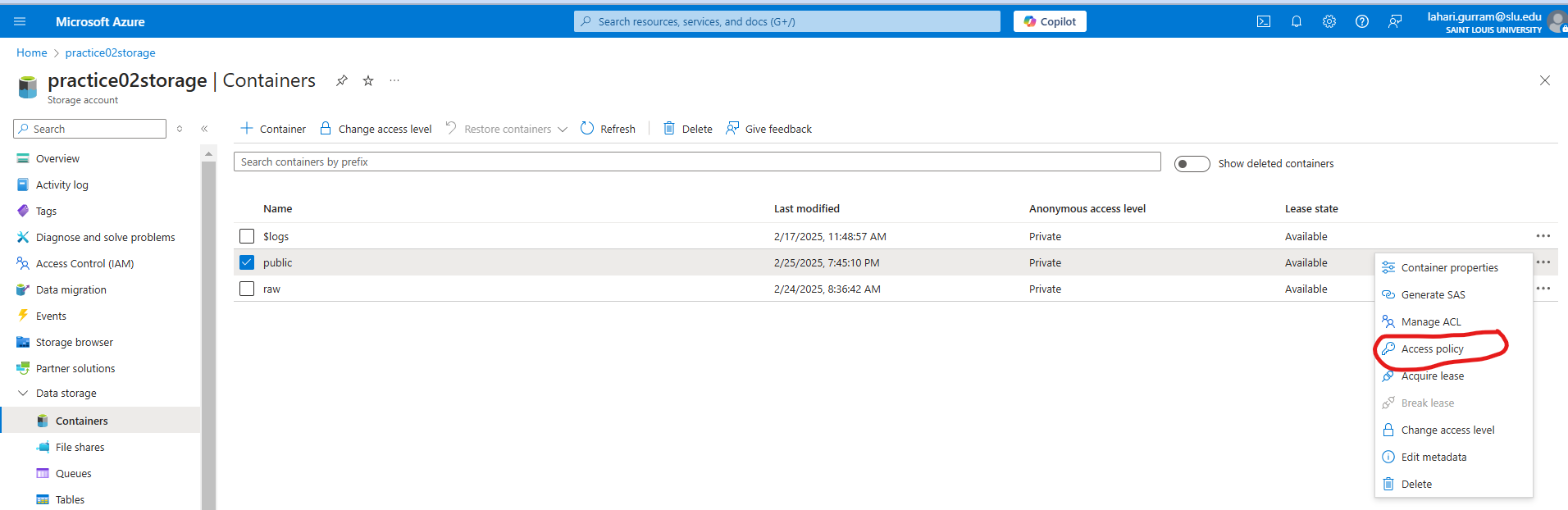
**🔹 Where Can You Define a Shared Access Policy?**

You can create SAPs at two levels:

1. **Container Level** – Controls access to blobs inside a container.
2. **Queue/Table/File Share Level** – Controls access for those storage services.

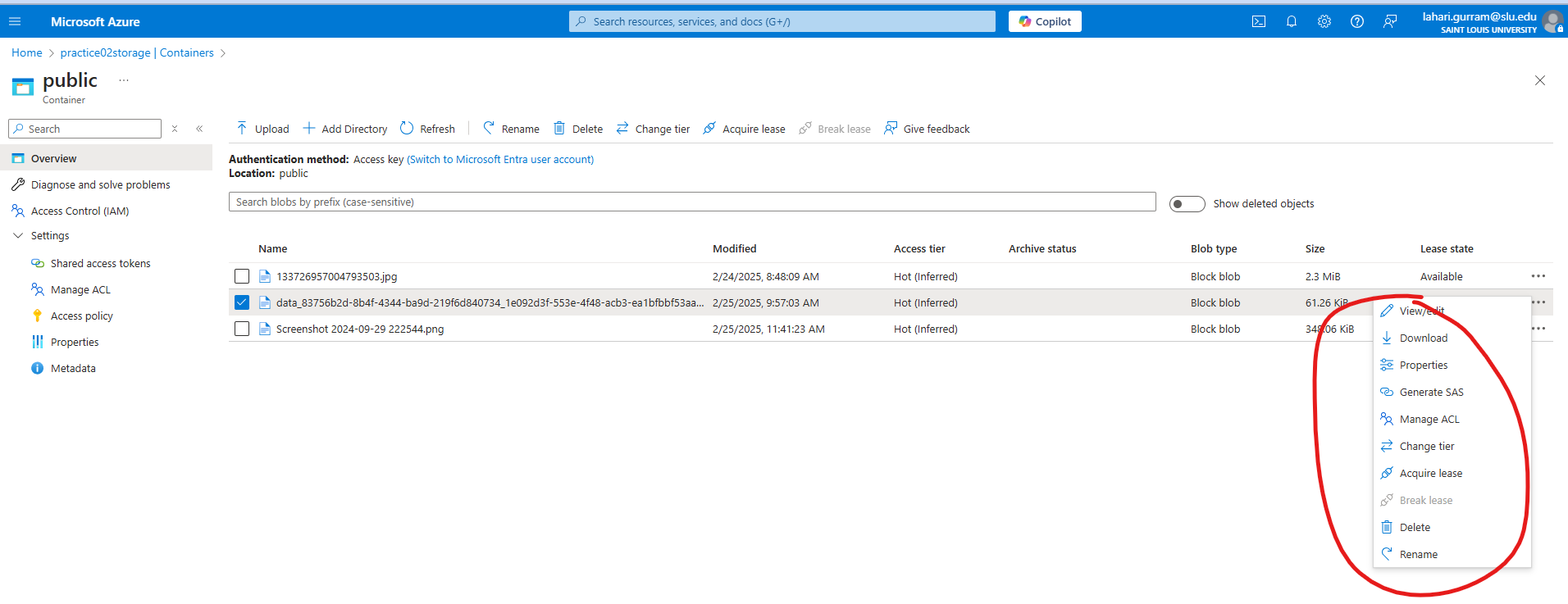
SAP is **not applicable** to individual blobs/files; it only applies to a **container, queue, table, or file share**. In the following figures at container level we can observe that there is an option for access policy but at BLOB levele there is no chance for creating an access policy and

BLOBS(files ) inside the container can access the parent(CONTAINER)’s shared policy:



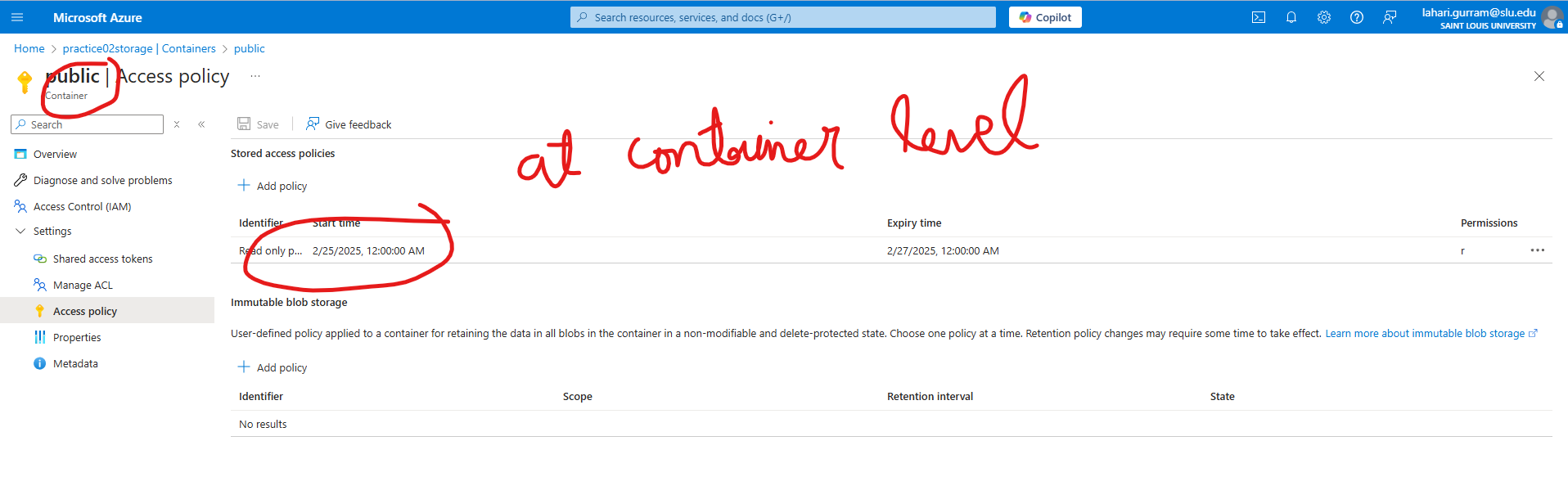
The above one is at container level

NOW THE BLOB LEVEL:

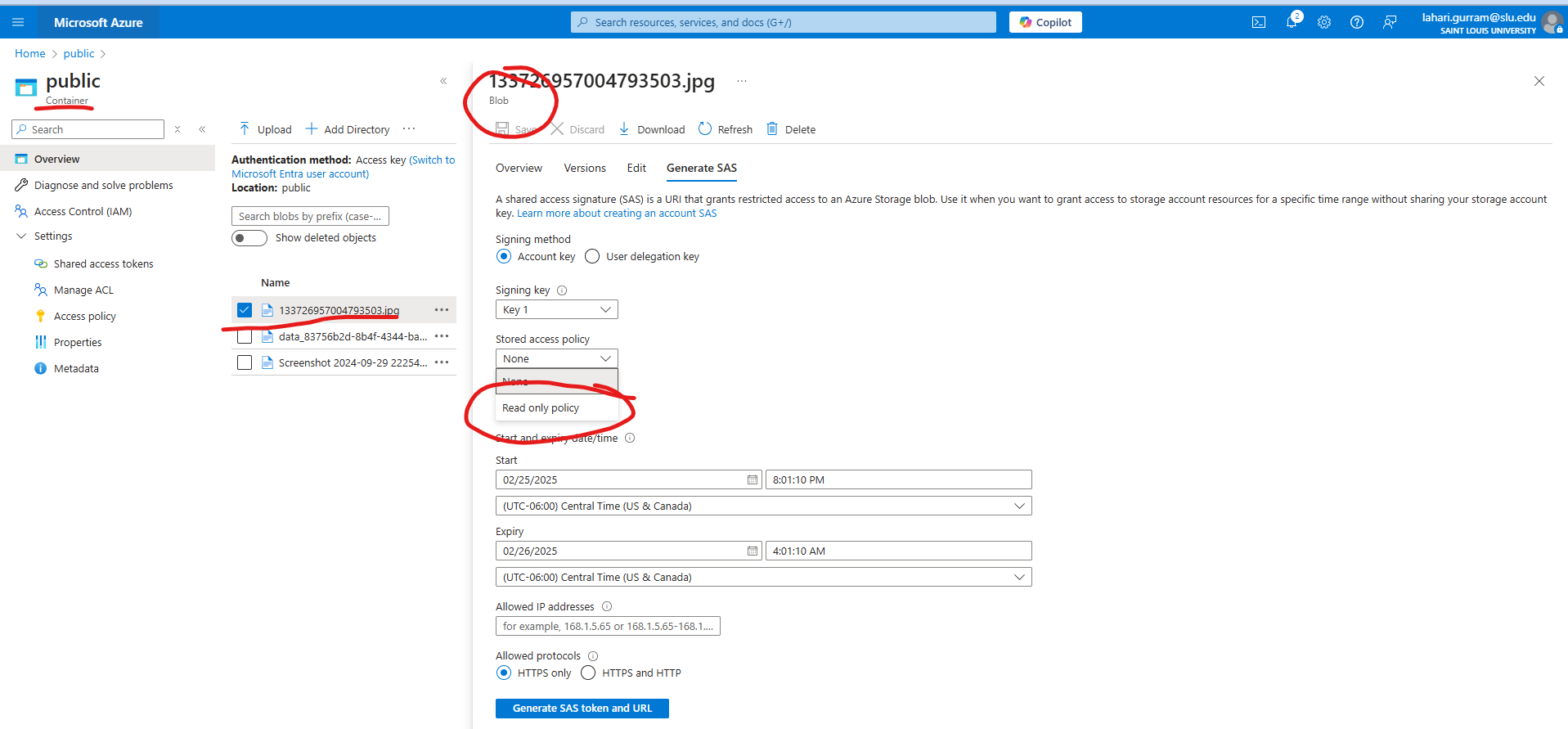


There is no access policy option here

But the access policies created at parent(container) level are accessed by childrens(files):



The access policy is created at container level;



Here the blob inherited the access policy

**USER DELEGATION KEY**

A User Delegation Key (UDK) is an advanced security feature in Azure Storage that allows you to generate SAS tokens authenticated using Azure AD credentials instead of the traditional storage account key.

**Why Use a User Delegation Key?**

Normally, SAS tokens are created using either:  
1️⃣ **Storage Account Key** (less secure, full account control).  
2️⃣ **Stored Access Policy** (limited control, only at container level).

👉 **User Delegation Key (UDK)** provides more security because:  
✅ It is **tied to an Azure AD identity**, reducing risks of key leaks.  
✅ It allows **fine-grained access control** with **RBAC roles**.  
✅ The **key expires automatically**, unlike storage account keys.

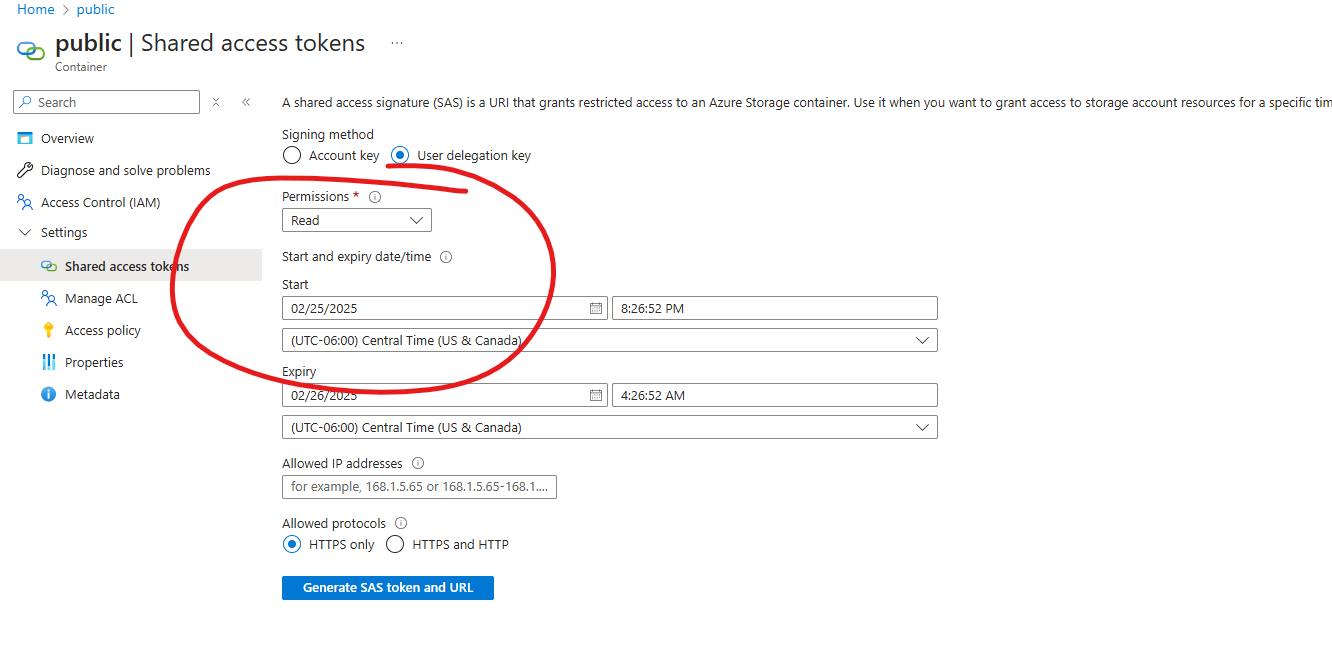
**How User Delegation Key Works (Step-by-Step)**

1️⃣ **Authenticate with Azure AD** (Using an Azure AD user or service principal).  
2️⃣ **Request a User Delegation Key** from Azure Storage.  
3️⃣ **Generate a SAS token** using the **UDK instead of the Storage Account Key**.  
4️⃣ **Use the SAS token** to access blobs securely.

**🔥 Key Takeaways**

* **User Delegation SAS needs an Azure AD session**, so it won't work in a private browser.
* If you want **public access**, use a **Service SAS** instead.

There is no such thing like access policy in case of user delegation key ; all we have to do is select permissions .



So when we generate SAS through user delegation key , wec can only access that with AZURE AD account;

 **User Delegation SAS** relies on **Azure AD authentication** and the correct permissions assigned to your user account.

 If you are not logged into **Azure AD** in the browser session, the SAS URL won't work because the Azure AD token (used for signing the SAS) won't be recognized or validated.

To enable **User Delegation SAS** with Azure AD authentication, follow these steps to ensure the Azure AD login and appropriate permissions (RBAC roles) are in place. This will allow users to authenticate with their Azure AD account before accessing the resource.

**Steps to Set Up User Delegation SAS:**

1. **Ensure Azure AD Authentication is Set Up:**
   * For **User Delegation SAS** to work, the Azure Storage Account must have **Azure AD authentication** enabled. You can verify this in the Azure portal under the **Access Control (IAM)** section for the storage account.
2. **Assign Proper Roles (RBAC) to the User:**
   * Users need to have the **appropriate role** to access the Azure Storage resources.
   * The **Storage Blob Data Reader** role allows users to read blob data, and the **Storage Blob Data Contributor** role allows users to both read and write data to the blob container.

Here’s how to assign a role to a user:

* + Go to the **Azure portal** and open the **Storage Account**.
  + Navigate to **Access Control (IAM)** on the left pane.
  + Click on **Add** > **Add role assignment**.
  + In the **Role** dropdown, choose either:
    - **Storage Blob Data Reader** for read-only access.
    - **Storage Blob Data Contributor** for read/write access.
  + In the **Assign access to** section, choose **User, group, or service principal**.
  + Under **Select**, type in the user’s Azure AD account name (like user@slu.edu).
  + Click **Save** to apply the role.

1. **Generate a User Delegation SAS:**
   * Once the appropriate RBAC role is assigned to the user, you can generate a **User Delegation SAS**.
   * This SAS token can be generated using Azure PowerShell, Azure CLI, or the Azure portal. The key here is that it’s tied to the **Azure AD user’s credentials**.

**Steps to generate a User Delegation SAS** via the Azure portal:

* + Navigate to your **Storage Account** in the **Azure portal**.
  + In the left pane, select **Shared access signature** under **Security + networking**.
  + Ensure **Signing key** is set to **Azure Active Directory**.
  + Configure the SAS parameters (such as expiration date, permissions like read/write).
  + **Click Generate SAS and connection string** to generate the URL.

1. **Authenticate with Azure AD:**
   * The user accessing the SAS URL must be logged into Azure AD with the correct permissions.
   * They need to authenticate either via their browser or through an Azure AD-supported client (like an application or tool that supports Azure AD authentication).
   * The user logs in with their Azure AD credentials (e.g., user@slu.edu).
2. **Use the User Delegation SAS to Access Data:**
   * After logging in, the user can use the **SAS URL** to access the blob or file.
   * The **SAS token** will only be valid if the user has the required permissions to access the resource.

**Example:**

1. **Assign Role**: You assign the **Storage Blob Data Reader** role to a user user@slu.edu.
2. **Generate SAS Token**: You generate a **User Delegation SAS** for a blob with read access.
3. **Authenticate User**: The user user@slu.edu logs into Azure AD.
4. **Access Blob**: The user uses the **SAS URL** to access the blob, and the system verifies the Azure AD credentials and permissions before granting access.

**Key Points to Remember:**

* **User Delegation SAS** is tied to Azure AD login, and only the **users with proper RBAC roles** can access the resource.
* **Account-level SAS** does not require Azure AD authentication, but **User Delegation SAS** does.
* The SAS URL will only work **for the user who is logged into Azure AD** with the required permissions.

This allows fine-grained access control based on Azure AD users and their assigned roles.