**Accessing of Azure Storage Account**

In Azure Storage, access methods define how users, applications, or systems interact with the data stored in a **Storage Account**.

Azure provides several ways to access and manage storage data securely and efficiently.

Some of them are:

1. Azure portal
2. Shared access signature (SAS)
3. Access Key’s

B, C, & D are Secure Transfer Methods

1. Role Based Access Control (Azure AD).
2. Azure storage Explorer
3. Azure CLI
4. PowerShell.
5. Rest API
6. SDK’s
7. **Azure Portal:**

* **Description**:
  + Provides a graphical interface to manage and interact with storage accounts.
* **Capabilities**:
  + Upload, download, and delete blobs/files.
  + Manage containers, tables, queues, and file shares.
  + Configure settings like firewalls, private endpoints, and lifecycle management.
* **Use Case**: Ideal for manual tasks and basic management.

**Secure Transfer Methods**:

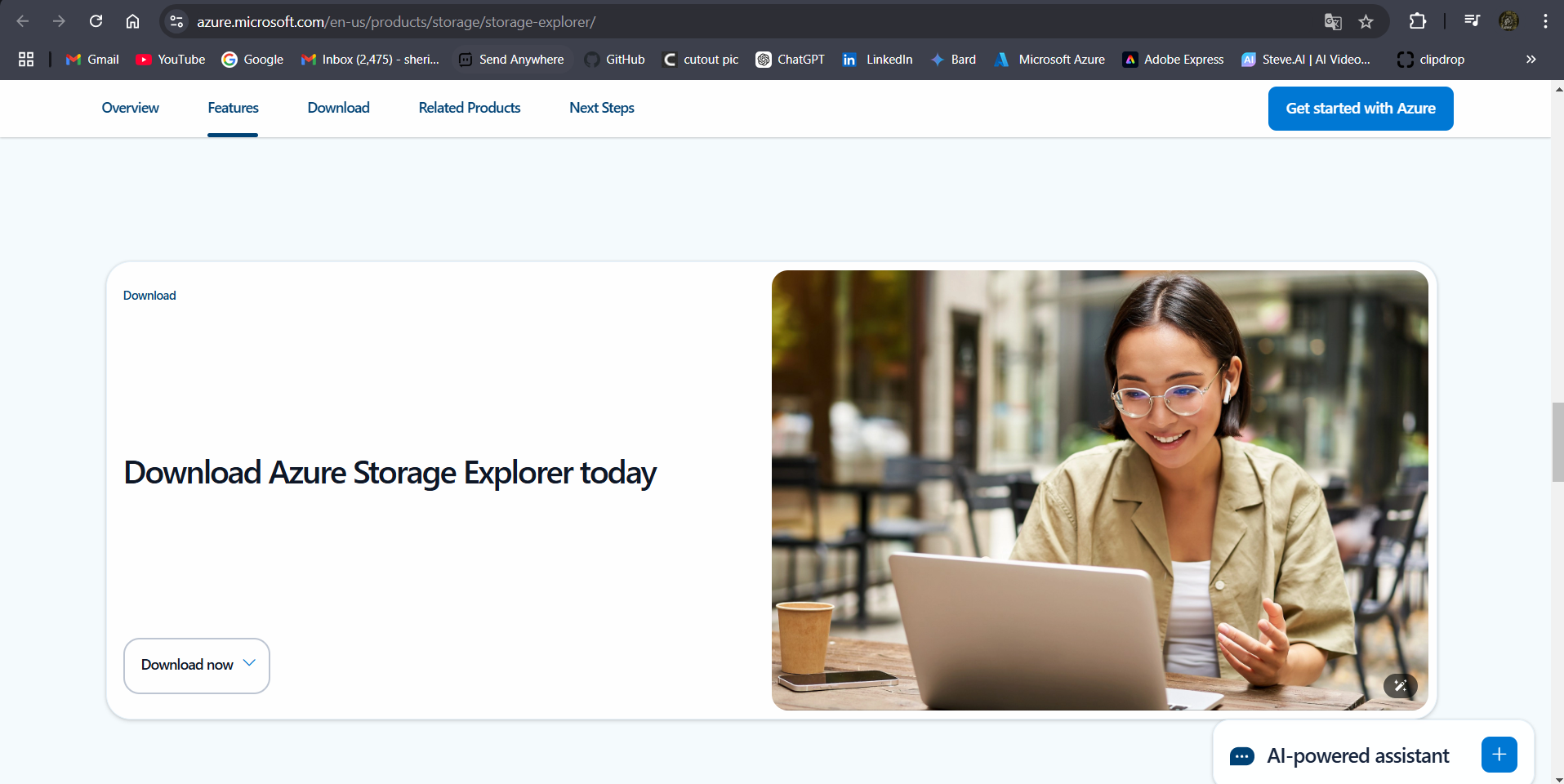
1. **Shared Access Signatures (SAS)**:
   1. Grant temporary, restricted access to specific resources in the storage account. That means it is mainly used to provide or grant a restricted access to a specific storage resource like blob, container, file, Tables and Queues.
   2. Useful for giving applications or users access without exposing access keys.
2. **Access Keys**:
   1. Two access keys are provided for full control of the storage account. That means a pair of primary and secondary keys that enables us to grant full administrative access to our entire storage account.
   2. It is mainly used to for administrator tasks such as creating, managing, and configuring of storage account.
   3. Provides administrator Level access to the user over the storage account.
3. **Azure Active Directory (Azure AD)**:
   1. Allows role-based access control (RBAC) for managing permissions securely.
   2. Use Azure AD over access keys for better security and manageability.
   3. By using RBAC we can grant read/write/owner access to the user for storage Account.
4. **Azure Storage Explorer:**

* **Description**:
  + A desktop application that allows users to manage storage accounts.
* **Capabilities**:
  + Upload, download, and manage blobs, files, tables, and queues.
  + Copy/move data between storage accounts or on-premises systems.
  + Supports connection via Azure Active Directory (Azure AD) or access keys.
* **Use Case**: Suitable for developers or administrators needing a user-friendly tool for bulk operations.

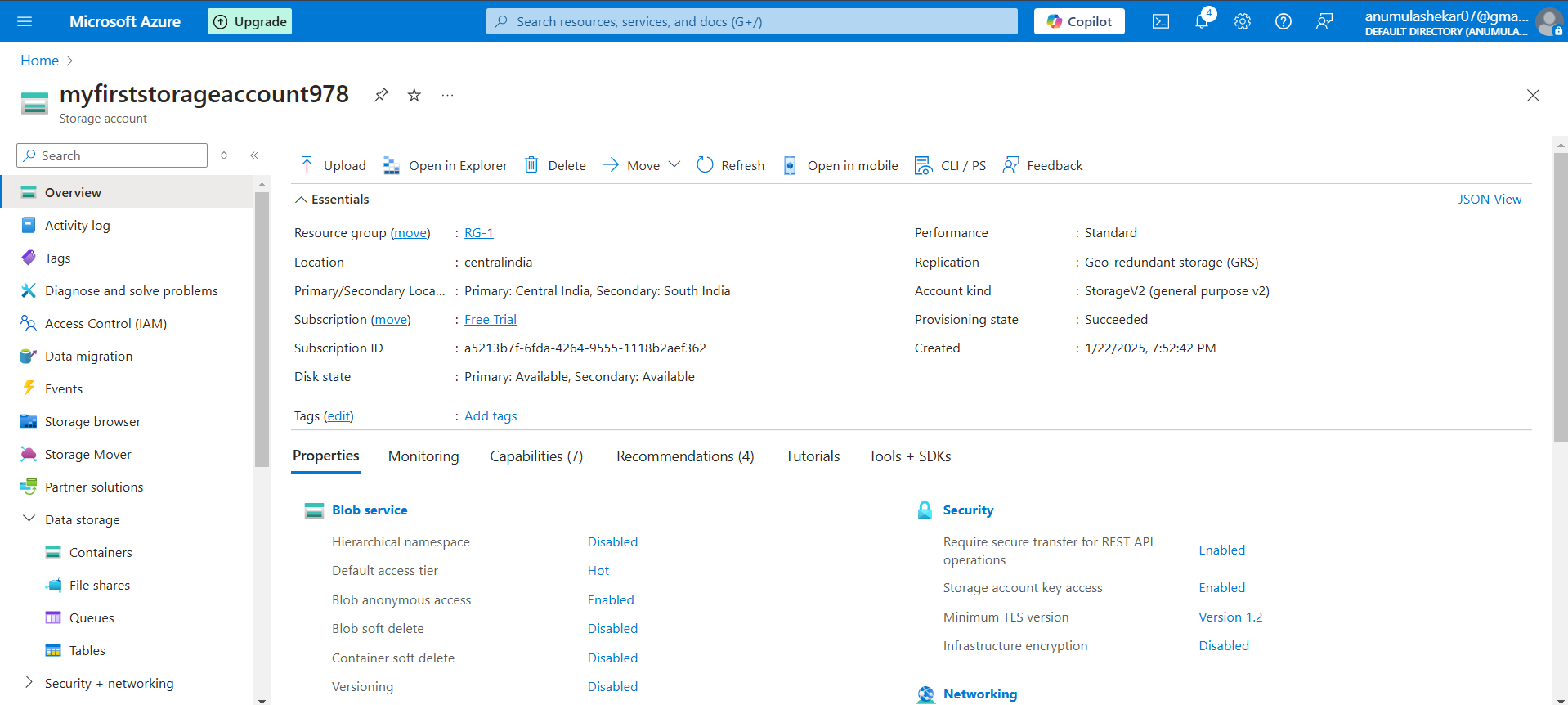
**Now let’s perform a task that connect of azure storage account to the Azure storage Explorer using Access Key’s and SAS.**

**Using Access Key’s:**

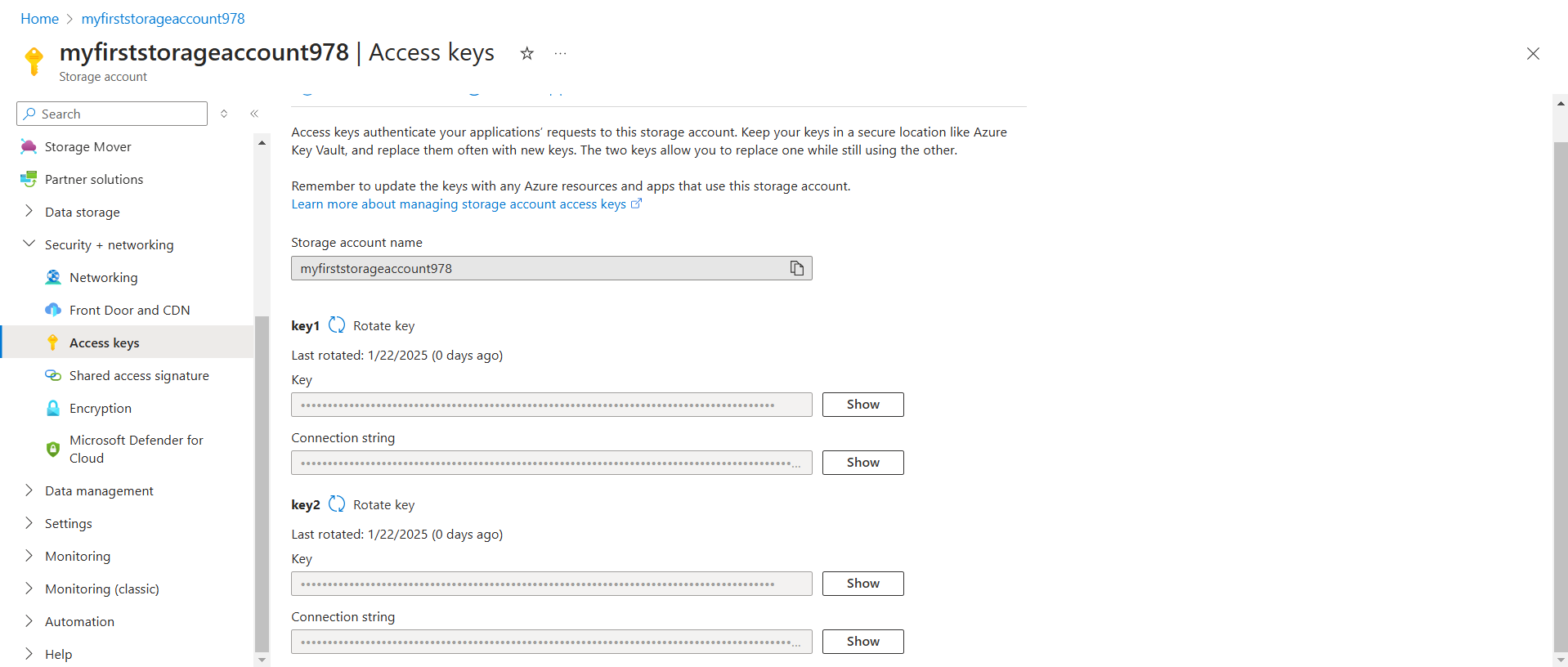
**Step1:** Install the Azure Storage explorer from any bowser.



**Step2:** Create azure storage account.

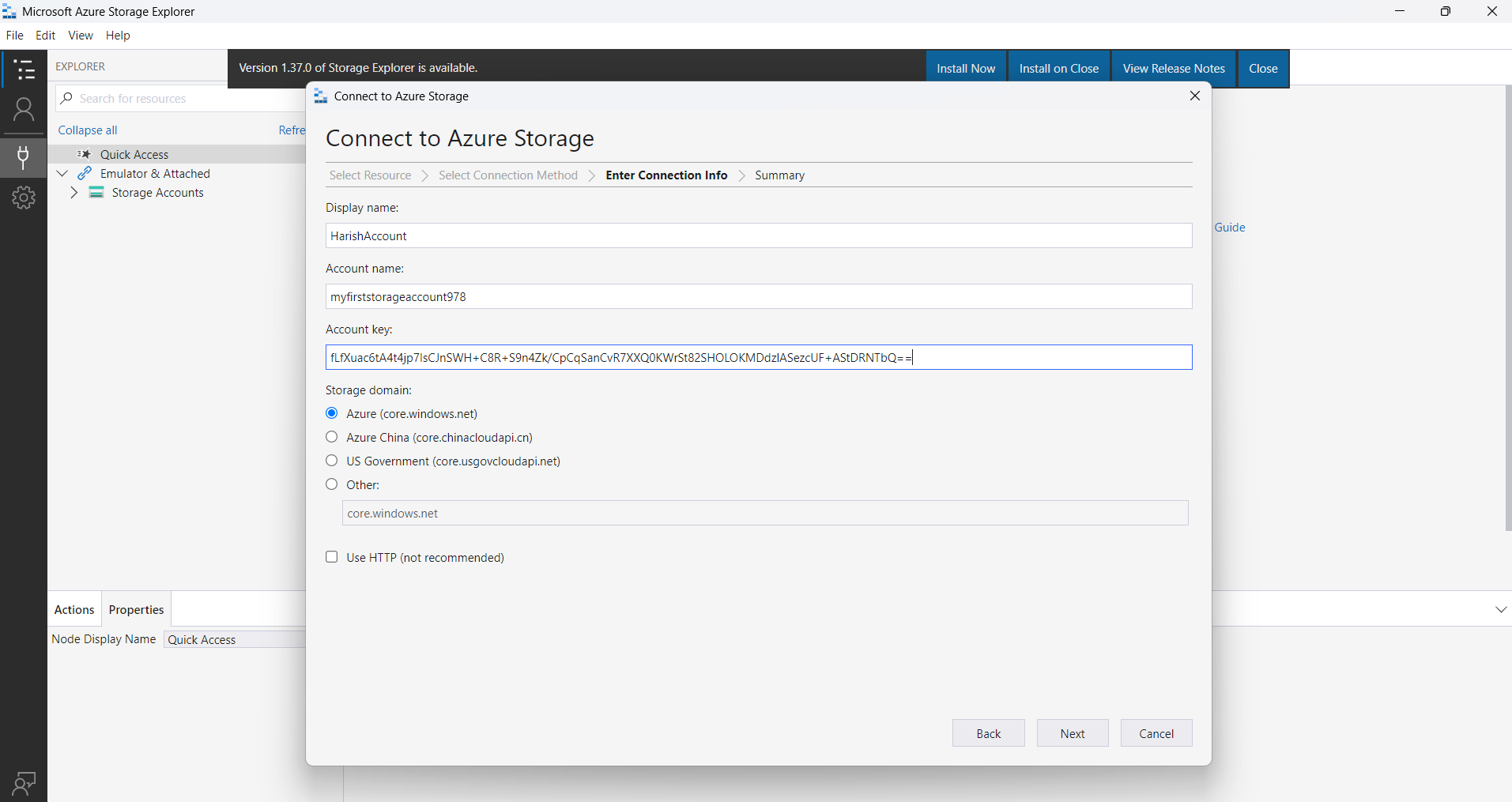


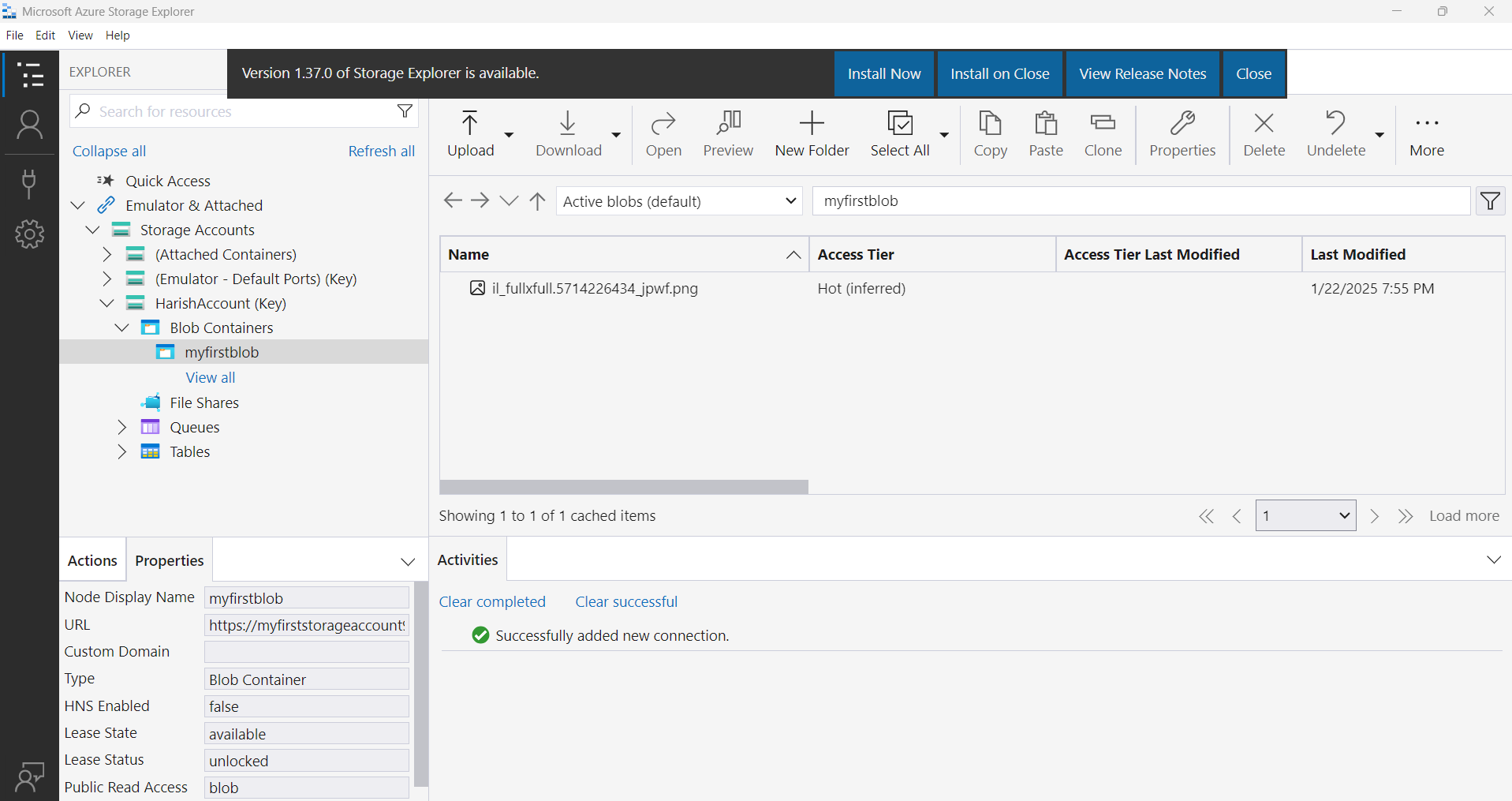
In order to connect storage account with azure storage Explorer, first copy the Access key of the storage account and paste it in the storage explorer.



Storage account provide two access keys (primary and secondary keys), it is because when one key (e.g., the **primary key**) is in use, you can regenerate the other key (e.g., the **secondary key**) without causing downtime or service disruption. After regeneration, you can update your applications or systems to use the new key and then rotate the original key.

This process is known as Key rotation, it is a best practice to minimize the risk of unauthorized access to the storage account.

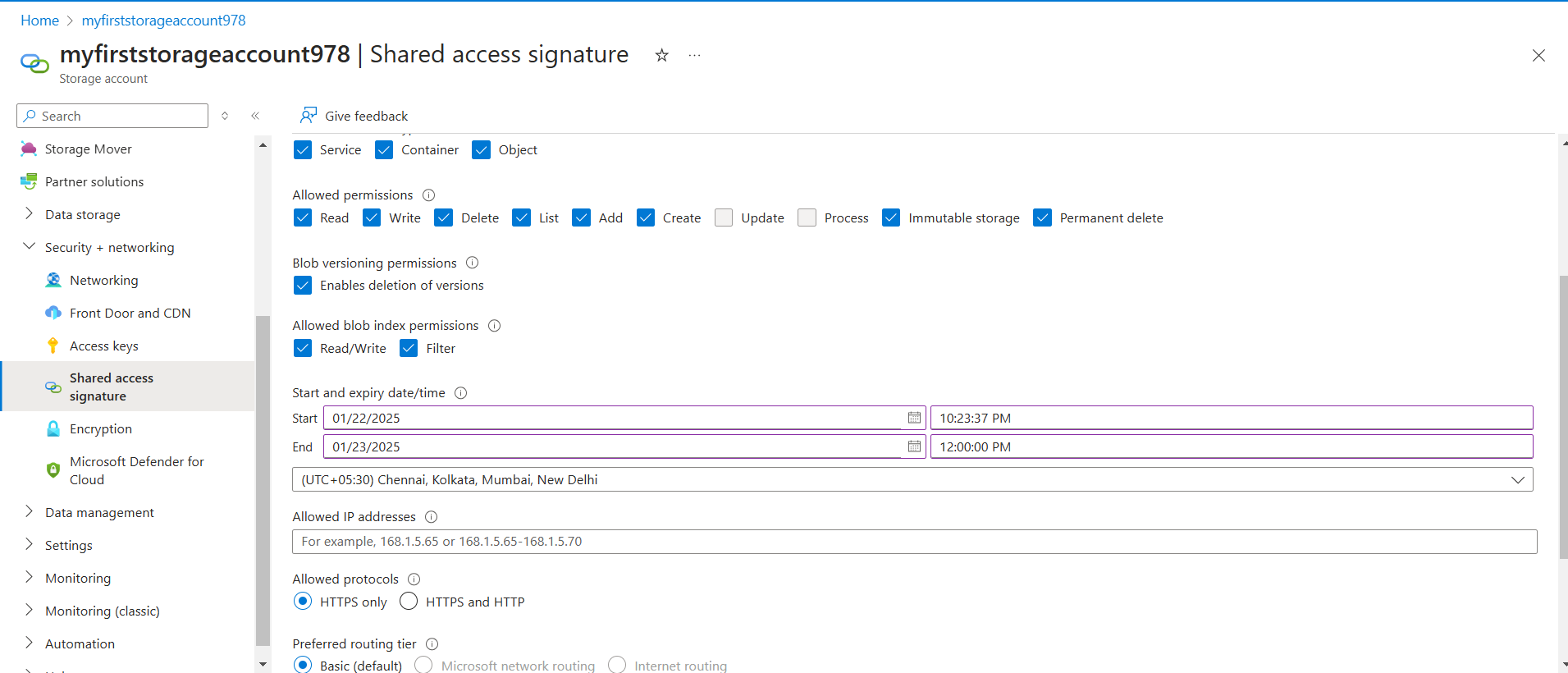




**Fig:** Azure storage account (myfiststorageaccount978) is connect to the storage explorer.

**Note:** We can upload date from local machine into the storage explorer and this uploaded date is replicated in the cloud as well.

**Using SAS keys:**

Step3: Connect the blob storage to the storage explorer by selecting the blob service SAS URL.

 **Granular Access Control**:

* SAS provides fine-grained control over permissions.
* You can define specific permissions such as **read**, **write**, **delete**, or **list**.

 **Temporary Access**:

* SAS tokens are time-bound, meaning you can specify a start and expiry time.
* Example: Grant access for only 24 hours.

 **Resource-Level Scope**:

* SAS can be scoped to:
  + **Blob-level**: Grants access to a specific blob.
  + **Container-level**: Grants access to all blobs within a container.
  + **Table**, **Queue**, or **File Share** resources.

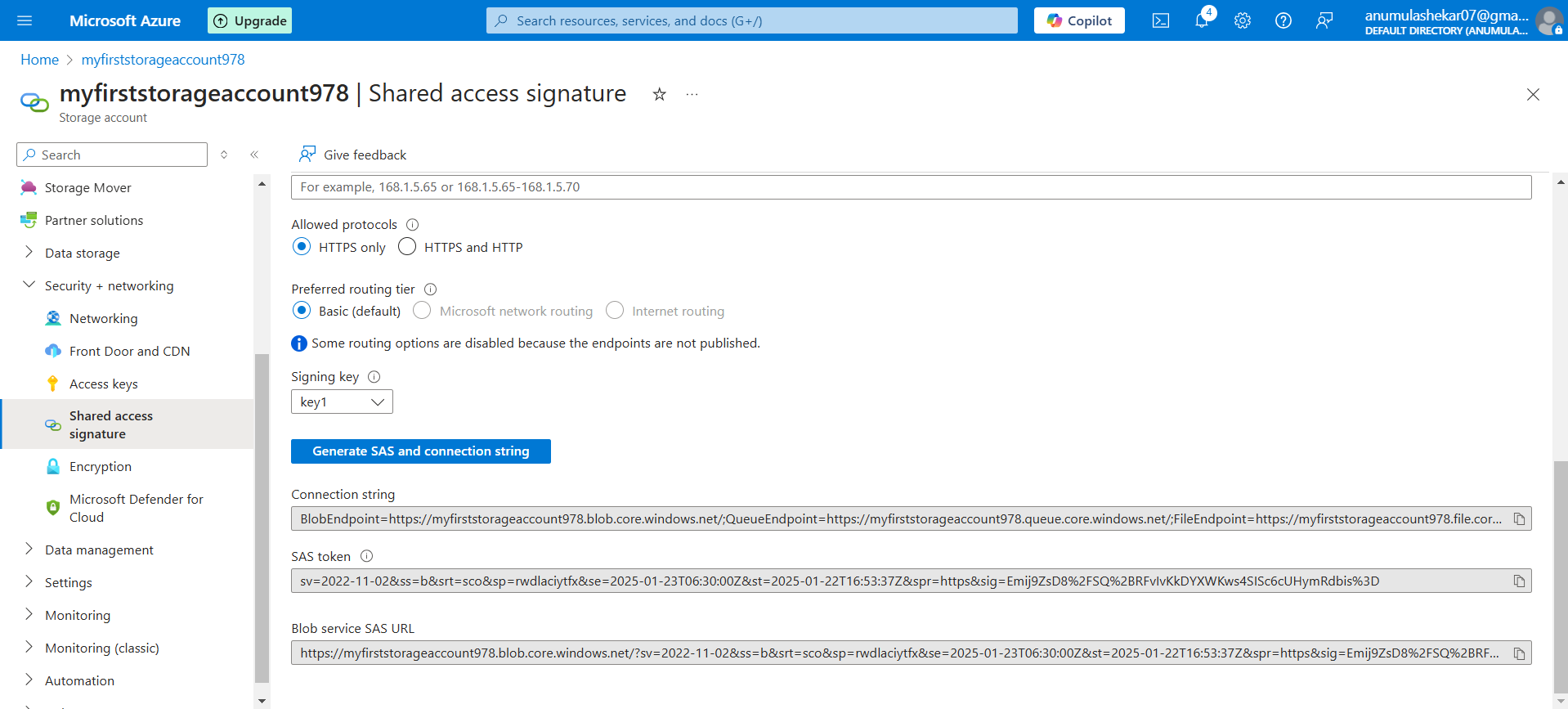
 **IP Address Restriction**:

* SAS keys can restrict access to specific IP ranges, enhancing security.
* Example: Allow access only from a specific corporate network.

 **Protocol Enforcement**:

* Specify whether the SAS should work over:
  + **HTTPS only** (recommended for security).
  + Both **HTTP and HTTPS**.

 **Ease of Revocation**:

* Revoke SAS tokens by regenerating storage account keys (for account-level SAS) or using stored access policies (for service-level SAS).

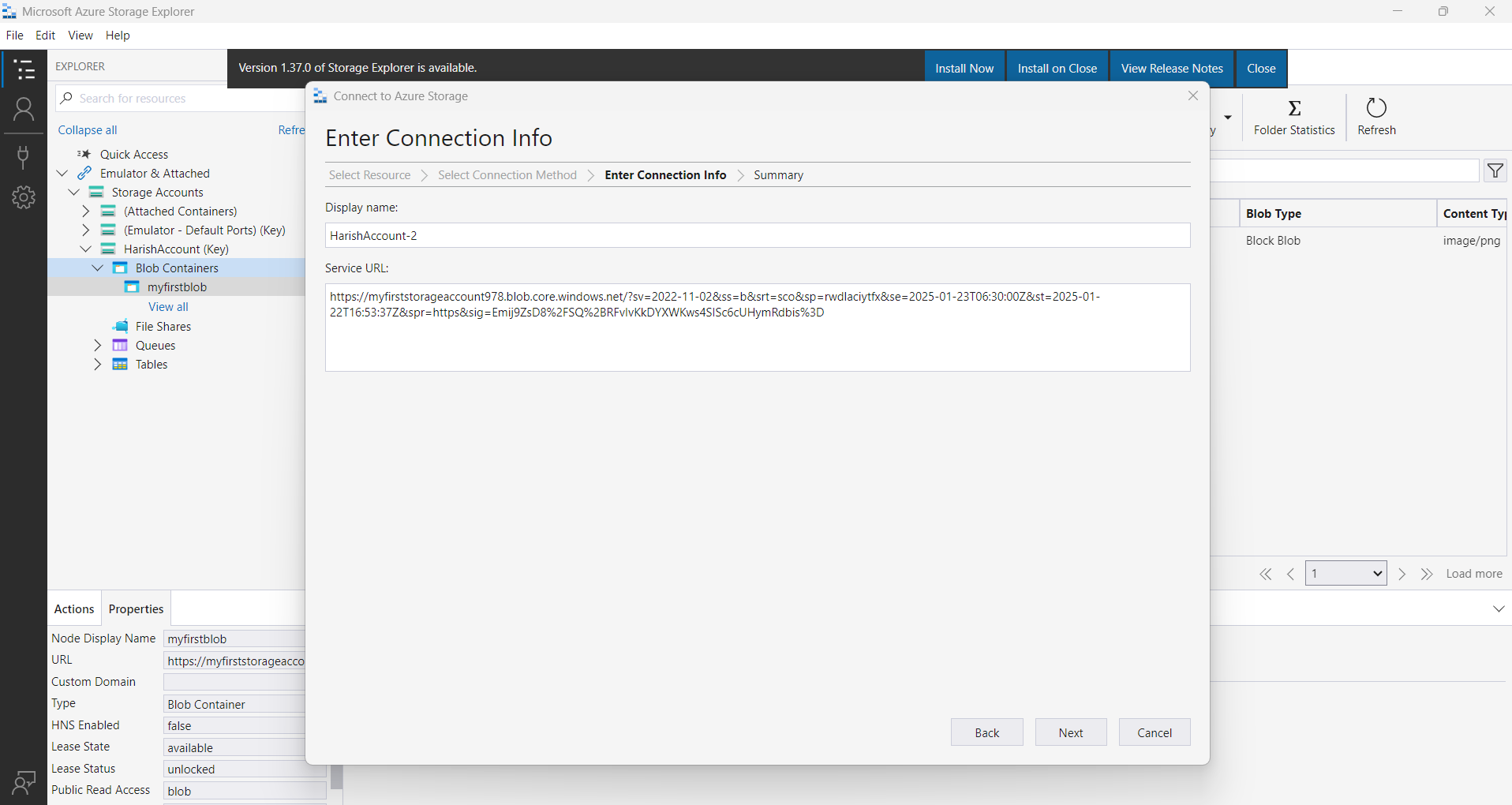
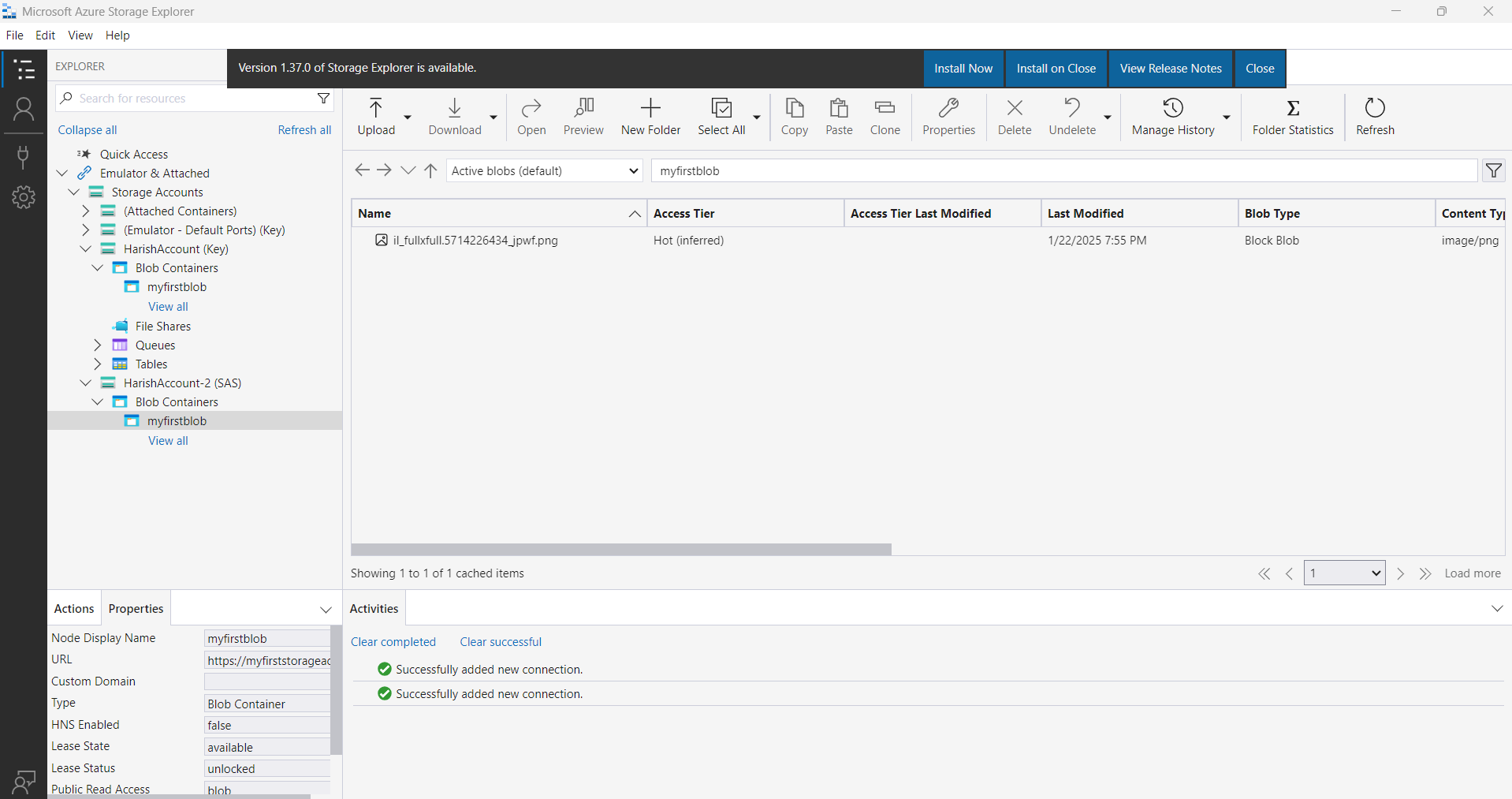


Fig: connect blob storage using SAS key.



From above figure by using SAS key only single blob storage is connect.