

Microsoft Azure is a cloud computing service offered and operated by Microsoft. Use this service to host your data and applications in the cloud.

An Azure storage account is an Azure cloud service that contains all of your Azure storage data objects: blobs, file shares, queues, tables, and disks. You can host files, including images, videos, music, and binary, as well as NoSQL data and messages in the cloud using this service.

In this lab you will learn how to work with Azure storage account keys and SAS tokens.

Learning Objective

In this lab, you will learn how to do the following:

Log into the Azure CLI

- Understand Azure storage account authentication methods
- Work with Azure storage keys using CLI
- Work with Azure storage SAS tokens using CLI
- Check storage account keys and SAS tokens in the portal
- Clean up

To log into the Azure subscription using CLI:

```
az login -u $username -p $password.
```

From the CLI you can create and manage your Azure resources.

To create a new Azure storage account:

```
az storage account create -n $storageAccountName --resource-group $resource --sku Standard_LRS
```

```
az storage account create -n $storageAccountName -g $resource --sku Standard_LRS
```

To create a stamp for 60 minutes from the current time. This will be used as the expiry date of the SAS token we create in this lab:

A Shared Access Signature (SAS) token is a security mechanism used in Azure that provides delegated access to resources such as Azure Storage, Azure Event Hubs, Azure Service Bus, and others. A SAS token is a string that contains a set of query parameters that define the permissions, duration, and other properties of the token.

SAS tokens are used to grant temporary access to specific resources, without requiring the user to share their access keys or account credentials. This is useful in scenarios where you want to provide temporary access to a specific resource or set of resources, without giving full account access to the user or application.

To create a SAS token, you can use the Azure portal, Azure PowerShell, or Azure CLI. Here's an example of how to create a SAS token using Azure CLI:

Here's a breakdown of how this command works:

"date": This is the command to print or set the system date and time.

"-u": This option sets the output to be in UTC time.

"+%Y-%m-%dT%H:%MZ": This is a format string that specifies how the date and time should be formatted. In this case, it uses the ISO 8601 format, which is commonly used for date and time representations in computing.

"--date="+60 minutes": This option specifies the amount of time to add to the current date and time. In this case, it adds 60 minutes.

"\$()": This syntax runs the command enclosed in the parentheses and captures its output as a string.

"expiryDate": This is the variable that will hold the resulting string.

The resulting value of "expiryDate" will be a string that looks something like "2023-03-27T15:24Z", representing the current date and time plus 60 minutes, in UTC time.

Understand Azure Storage Account Authentication Methods

You first need to authenticate with the Azure storage account to start using the service. Storage accounts offer the following authentication options:

- Storage access key
- Shared access key (SAS)
- Impersonation
- Managed identity
- Token
- AWS programmatic access keys

A Storage Access Key is a primary or secondary key used to authenticate access to an Azure Storage account. Each Storage Account in Azure has two access keys associated with it: a primary access key and a secondary access key.

These keys are used to authenticate access to the storage account via either Azure portal, REST API, Azure CLI, or any other Azure SDK. When you create a new Storage Account in Azure, Azure generates two access keys for you automatically.

You can retrieve the Storage Access Keys for your Storage Account from the Azure Portal by following these steps:

Go to the Azure portal and navigate to your storage account.

Select the "Access keys" option from the left-hand menu.

You will see two keys listed, the primary access key and the secondary access key. You can click the "Click to copy" button next to each key to copy it to the clipboard.

It is important to keep your Storage Access Keys secure, as anyone with access to them will be able to access and modify data in your storage account. You can also regenerate the keys at any time, which will invalidate the previous keys and require you to update any applications or scripts that are using them.

In this lab you will learn how to work with Azure storage account keys (option 1), and SAS tokens (option 2).

In Azure Storage, impersonation refers to the ability to perform actions on behalf of another user or service principal. Impersonation can be useful in scenarios where you want to delegate access to a storage account without sharing the access keys or requiring the user to have their own Azure AD identity.

One way to implement impersonation in Azure Storage is through Shared Access Signatures (SAS). With SAS, you can create a token that grants temporary access to a specific resource in your storage account, with a specific set of permissions. You can then provide this token to another user or service principal, who can use it to access the resource with the specified permissions, without needing access to the storage account's access keys.

Another way to implement impersonation in Azure Storage is through Azure AD-based access control. This method involves granting permissions to a user or service principal in Azure AD, which can then be used to access the storage account. This method is more secure than SAS, as it allows you to use Azure AD's identity and access management capabilities to manage access to your storage account.

To use Azure AD-based access control, you need to configure Azure AD integration for your storage account, create a service principal for the user or application that needs access, and grant the necessary permissions to the service principal. You can then use the Azure Storage SDKs or REST APIs to authenticate and perform actions on behalf of the service principal.

Note that impersonation carries some security risks, as it allows another user or service principal to perform actions on your behalf. You should always ensure that you trust the user or service principal you are impersonating, and that you have appropriate access controls and auditing in place to monitor their actions.

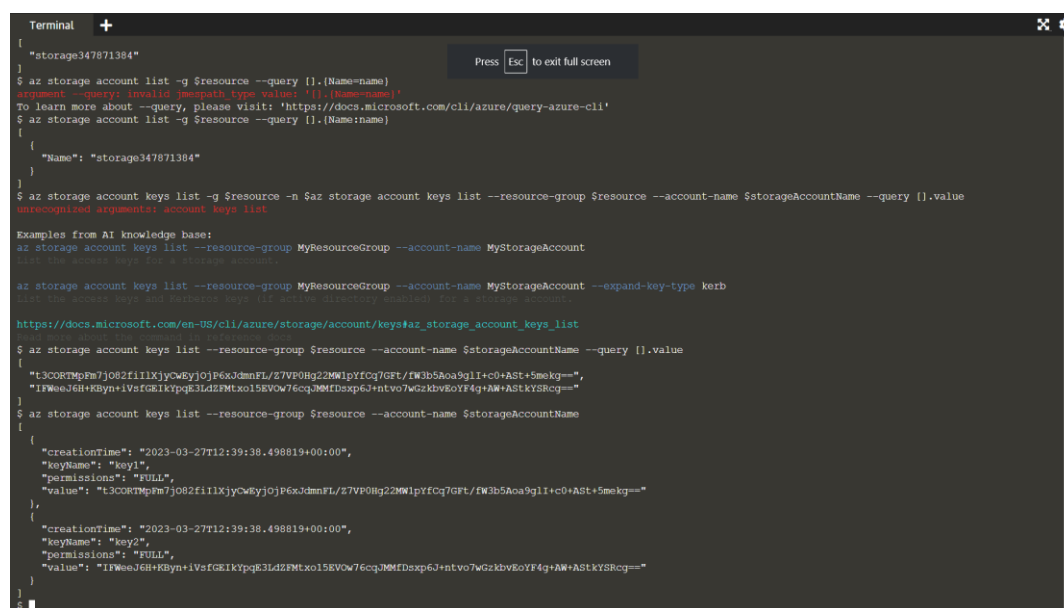
List storage account created :

```
]
$ az storage account list -g $resource --query [].name
[
  "storage347871384"
]
$ az storage account list -g $resource --query [].{Name=name}
argument --query: invalid jmespath_type value: '[].{Name=name}'
To learn more about --query, please visit: 'https://docs.microsoft.com/cli/azure/query-azure-cli'
$ az storage account list -g $resource --query [].{Name=name}
[
  {
    "Name": "storage347871384"
  }
]
$
```

Now, the stage is set to start working with Azure storage keys and SAS tokens.

Work with Storage Keys using Azure CLI

First, you can use the following command to list your storage account keys. These are master keys and whoever holds them will have full access to the storage account. You should use and share them with care.



```
Terminal
[
  "storage347871384"
]
$ az storage account list -g $resource --query [].{Name=name}
argument --query: invalid jmespath_type value: '[].{Name=name}'
To learn more about --query, please visit: 'https://docs.microsoft.com/cli/azure/query-azure-cli'
$ az storage account list -g $resource --query [].{Name=name}
[
  {
    "Name": "storage347871384"
  }
]
$ az storage account keys list -g $resource -n $az storage account keys list --resource-group $resource --account-name $storageAccountName --query [].value
unrecognized arguments: account keys list

Examples from AI knowledge base:
az storage account keys list --resource-group MyResourceGroup --account-name MyStorageAccount
List the access keys for a storage account.

az storage account keys list --resource-group MyResourceGroup --account-name MyStorageAccount --expand-key-type kerb
https://docs.microsoft.com/en-US/cli/azure/storage/account/keys#az_storage_account_keys_list
$ az storage account keys list --resource-group $resource --account-name $storageAccountName --query [].value
[
  {
    "keyName": "key1",
    "value": "t3CORTMph7j082f11XjyCwEjyOjP6xJdmFL/Z7VP0Hg22MW1pYfCq7GfT/fW3b5Aoa9g1I+c0+AST+5mekg=="
  },
  {
    "keyName": "key2",
    "value": "IFWeeJ6H+KByn+ivsfGEIkYpqE3Ld2FMTx015EVOW76cqJMMFdxp6J+ntvo7WgZkbvBoYF4g+AW+ASTkYSRcg=="
  }
]
$
```

Confirm that in the command output you can see two keys. Both keys offer the same permission level to the key holder.

You can also rotate or regenerate storage account keys if required. For instance, use the following command to regenerate the primary storage account key:

```

$ az storage account keys renew -g $resource -n $storageAccountName --key primary
[
  {
    "creationTime": "2023-03-27T13:06:55.403787+00:00",
    "keyName": "key1",
    "permissions": "FULL",
    "value": "gxG1IkFAUdwnUbAmp9gs7LX+hzW+5A0cKf4C7bfS5IywGOTi7WcDJrv95SdsDkVjbQk6DCmj26kU+ASTQ6ojgQ=="
  },
  {
    "creationTime": "2023-03-27T12:39:38.498819+00:00",
    "keyName": "key2",
    "permissions": "FULL",
    "value": "IFWeeJ6H+KByn+iVsfGEIkYpqE3LdZFMtxo15EVOW76cqJMMfDsxp6J+ntvo7wGzkbvEoYF4g+AW+ASTkYSRcg=="
  }
]

```

Here are the command parameters:

--resource-group: The parent resource group of the storage account
 --name: The storage account name
 --key: The key to renew/regenerate. Accepted values are primary and secondary

Now, use the list command again to store the first storage account key in a variable. We need this key in the next step.

```

$ az storage account keys list -g $resource -n $storageAccountName --query [0].value
[
  "gxG1IkFAUdwnUbAmp9gs7LX+hzW+5A0cKf4C7bfS5IywGOTi7WcDJrv95SdsDkVjbQk6DCmj26kU+ASTQ6ojgQ==",
  "IFWeeJ6H+KByn+iVsfGEIkYpqE3LdZFMtxo15EVOW76cqJMMfDsxp6J+ntvo7wGzkbvEoYF4g+AW+ASTkYSRcg=="
]
$ az storage account keys list -g $resource -n $storageAccountName --query [0].value
"gxG1IkFAUdwnUbAmp9gs7LX+hzW+5A0cKf4C7bfS5IywGOTi7WcDJrv95SdsDkVjbQk6DCmj26kU+ASTQ6ojgQ=="
$ az storage account keys list -g $resource -n $storageAccountName --query [0].value -o tsv
gxG1IkFAUdwnUbAmp9gs7LX+hzW+5A0cKf4C7bfS5IywGOTi7WcDJrv95SdsDkVjbQk6DCmj26kU+ASTQ6ojgQ==
$ storageKey=$(az storage account keys list --resource-group $resource --account-name $storageAccountName --query [0].value -o tsv)
$

```

The `az storage account keys list` command is used to retrieve the access keys for an Azure Storage account. The command returns a list of access keys for the specified storage account, along with their associated properties.

Here's what each part of the command does:

- `az storage account keys list`: This is the command to retrieve the access keys for an Azure Storage account.
- `-g $resource`: This specifies the resource group name that the storage account is located in. `$resource` is a variable that should be replaced with the actual resource group name.
- `-n $storageAccountName`: This specifies the name of the storage account that you want to retrieve the access keys for. `$storageAccountName` is a variable that should be replaced with the actual name of the storage account.
- `--query [0].value`: This uses the `--query` option to return only the value of the first access key in the list. Since the command returns a list of keys, this is necessary to extract the specific key value that you need. The query syntax `[0].value` specifies that you want to select the value of the first item in the list.
- `-o tsv`: This specifies that the output format should be tab-separated values (TSV), which is a simple format that is easy to process with scripts or command-line tools.
- When you run this command, it will return the value of the first access key for the specified storage account, in TSV format.

In Azure Storage, the master key is a cryptographic key used to protect the storage account access keys. The master key is a randomly generated 256-bit key that is unique to each storage account and is used to encrypt and decrypt the access keys.

When you create a new storage account in Azure, two access keys are generated for you automatically. These access keys are used to authenticate access to the storage account, and they are encrypted using the master key.

If you need to regenerate your storage account access keys, Azure will automatically generate a new master key and use it to encrypt the new access keys. This ensures that the previous access keys are invalidated and can no longer be used to access the storage account.

It is important to keep your storage account access keys and master key secure, as they provide complete control over your storage account and the data stored within it. You should follow best practices for securing access keys, such as limiting access to only those who need it, rotating keys regularly, and monitoring key usage for suspicious activity.

Work with Storage SAS Tokens using Azure CLI

Storage account keys offer full storage account permission. You don't have enough flexibility in scoping your permission level, so Azure storage SAS tokens allow you to create storage account keys with scoped (limited) access level (for example, read-only or blob-only keys/tokens).

You can use the `az storage account generate-sas` command to create storage account SAS tokens. Here are a few parameters you can use to scope the SAS token access level:

--start: To set start date/time on the SAS token.

--expiry: To set expiry date/time on the SAS token.

--permissions: The list of permissions the SAS grants. Allowed values: (a)dd, (c)reate, (d)elele, (f)ilter_by_tags, (i)set_immutability_policy, (l)ist, (p)rocess, (r)ead, (t)ag, (u)pdate, (w)rite, (x)delete_previous_version, and (y)permanent_delete. These values can be combined; for instance, cd grants both create and delete permissions.

--resource-types: The resource types the SAS is applicable to. Allowed values: (s)ervice, (c)ontainer, and (o)bject. You can combine the values.

--services: The storage services the SAS is applicable to. Allowed values: (b)lob, (f)ile, (q)ueue, and (t)able. You can combine the values.

For example, use the following command to create a new storage account SAS token that allows read-only access to the blob and file services and expires in 10 minutes:

```
storageSAS=$(az storage account generate-sas --permissions r --account-name $storageAccountName --services bf --resource-types co --expiry $expiryDate --account-key $storageKey -o tsv)
```

The `az storage account generate-sas` command is used to generate a Shared Access Signature (SAS) token for an Azure Storage account. The SAS token provides temporary access to specific resources in the storage account, and can be used to delegate access to others without sharing the account access keys.

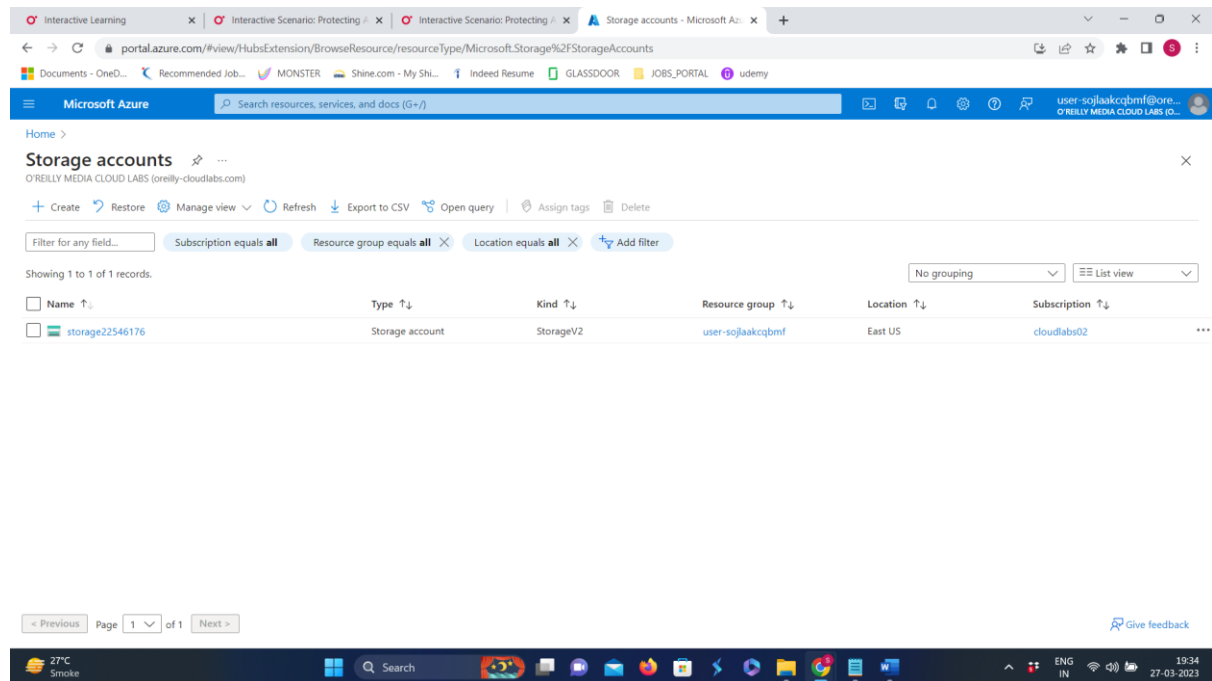
Here's what each part of the command does:

- `az storage account generate-sas`: This is the command to generate a SAS token for an Azure Storage account.
- `--permissions r`: This specifies that the SAS token should have read permissions only. You can adjust the permissions as needed for your specific use case.
- `--account-name $storageAccountName`: This specifies the name of the storage account that you want to generate the SAS token for. `$storageAccountName` is a variable that should be replaced with the actual name of the storage account.
- `--services bf`: This specifies that the SAS token should be valid for the Blob service in the storage account.
- `--resource-types co`: This specifies that the SAS token should be valid for the Blob container resource type in the storage account.
- `--expiry $expiryDate`: This specifies the expiration date and time for the SAS token. `$expiryDate` is a variable that should be replaced with the actual expiration date and time, in UTC format.
- `--account-key $storageKey`: This specifies the storage account access key to use for generating the SAS token. `$storageKey` is a variable that should be replaced with the actual storage account access key value.
- `-o tsv`: This specifies that the output format should be tab-separated values (TSV), which is a simple format that is easy to process with scripts or command-line tools.

- When you run this command, it will generate a SAS token for the specified storage account, with the specified permissions, resource types, and expiration date and time.

Check Storage Account Keys and SAS Tokens in the Azure Portal

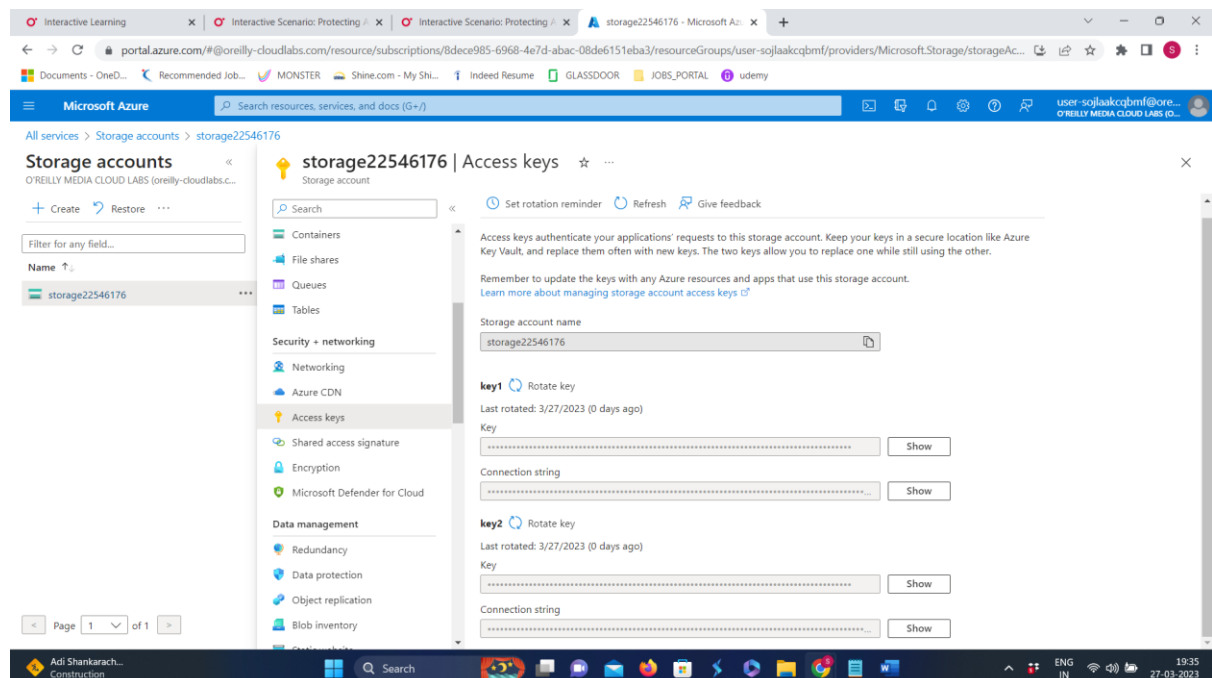
We sent a new order to the orders queue, and now we want to see it in the Azure portal.



The screenshot shows the Azure Portal interface for 'Storage accounts'. The top navigation bar includes the Microsoft Azure logo and a search bar. The main content area shows a list of storage accounts. A table with the following columns is displayed: Name, Type, Kind, Resource group, Location, and Subscription. The table contains one entry: 'storage22546176' (Type: Storage account, Kind: StorageV2, Resource group: user-sojlaakcqbmf, Location: East US, Subscription: cloudlabs02). The page also includes filters for 'Subscription equals all', 'Resource group equals all', and 'Location equals all'. The bottom of the page shows a taskbar with various application icons and system information.

Under Security + networking, click on Access keys.

Storage account is created .

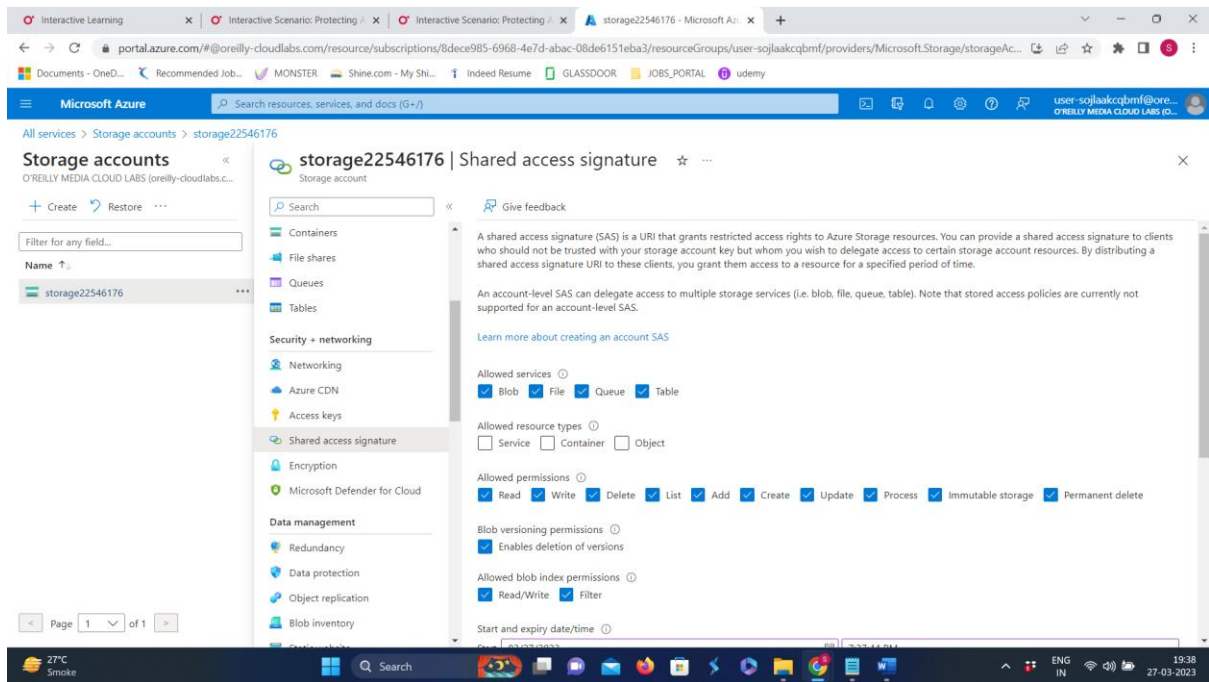


The screenshot shows the 'Access keys' page for the storage account 'storage22546176'. The page displays two keys, 'key1' and 'key2', each with a 'Rotate key' button, a 'Last rotated' date, and a 'Show' button for the key value. The page also includes a 'Connection string' field and a 'Show' button. The left sidebar shows the 'Security + networking' section expanded, with 'Access keys' selected. The top navigation bar includes the Microsoft Azure logo and a search bar. The bottom of the page shows a taskbar with various application icons and system information.

Access keys also present .

Under Security + networking, click on Shared access signature.

Observe the different options you have to limit the SAS token permissions.



Clean Up

Use the following command to clean up the storage accounts from your allocated resource group. Deleting a storage account will delete all the child blob containers, queues, file shares, and tables:

```
$ az storage account delete -g $resource -n $storageAccountName
```

Use the following command to list all storage accounts in your allocated resource group:

```
Are you sure you want to perform this operation?
$ az storage account list -g $resource
[]
$
```