Azure Database for PostgreSQL Flexible Server - pg_azure_storage MSI improvement

Operational Guide

October 2024.

Last updated: 2024-10-31

Document version: 1.0

This document will be updated from time to time.

To access the latest copy online, use the following link:
https://aka.ms/pgazurestorage_guide

Please regard this link**,** and the contents of the document it refers to, as **confidential**.

Please **do not share** with third parties.

Contents

Introduction

Non-disclosure agreement

Please consider the information in this document as strictly confidential and subject to NDA.

The contents of this document are confidential and participation in the preview program is covered by your NDA (Non-Disclosure Agreement) with Microsoft. The details within this document are not to be shared without receiving prior approval from the Microsoft Azure Database for PostgreSQL Flexible Server team. Please reach out to your contact at Microsoft if you have further questions. Your Product Manager contact will make sure that you are covered by an NDA before the engagement begins.

Support during private preview

Private previews do not include formal support (through Microsoft Support or Azure Support help desks), do not include production workload support, and be aware that introduced functionalities are excluded from SLA guarantees. By using preview technologies, you agree with Microsoft Azure Preview terms.

Limited support for using the preview functionality, and only during the timeframe of the private preview, is provided by the Product Group developing the feature (during business hours 9AM-5PM Central European Time and Central Daylight Time). Due to Product Group's limited capacity to work with each customer individually, and due to the nature of new technology introduced, support turnaround time is not guaranteed.

In case you need support using the Azure Database for PostgreSQL Flexible Server pg_azure_storage extension with Managed System Identity based authentication, please reach out to us <pg_azure_storage Private Preview> pg_azure_storage_prp@microsoft.com.

Preview goals and expectations

The private preview will usually include the following stages:

1. Select any subscription and region of your preference to test drive this feature.
2. Create at least one instance of Azure Database for Flexible Server on which you will test the feature to report your experience with it and any potential issues observed during the try out.
3. Explore the functionality that is critical for your application and explore its availability and compatibility with the feature.
4. Commit to 30-minute weekly meetings to share your consolidated feedback with the Engineering team.

The primary goals of this private preview are:

1. Give you an early opportunity to validate version 1.5 or higher of pg_azure_storage extension, especially its new feature which supports authenticating with Azure Storage using Managed System Identity. This enables the adoption of this important feature sooner and allows direct access to the product team, to make changes to the feature before it becomes publicly released and helps shape the evolution of the feature and the way in which future features integrate with it, ensuring pg_azure_storage extension meets your needs.
2. Provide feedback to the product team from validating the feature in non-production environments but using your own workloads. Timely and actionable feedback to the pg_azure_storage team enables us to improve the extension before it moves into public preview and general availability. The team seeks positive and negative feedback on all aspects of the extension, including reliability, functional completeness, ease of use, security, and performance.

At the start of private preview, pg_azure_storage will provide full unrestricted functionality. Features and capabilities will be added gradually as the preview progresses in the upcoming weeks.

During private preview, the pg_azure_storage product team may introduce functional breaking changes scoped to only the new feature being added to the extension. That is the ability to authenticate to Azure Storage accounts using Azure Database for PostgreSQL Flexible Server System Assigned Managed Identity. This may impact your ability to use the feature or cause other impact on the selected instances during your participation in the preview.

Feature overview

pg_azure_storage is a PostgreSQL extension available in Azure Database for PostgreSQL Flexible server, which allows the user to import and export data between blob containers in the Blob service of Azure Storage accounts and Azure Database for PostgreSQL Flexible Server.

Every request made against a secured resource in the Blob service of an Azure Storage account must be authorized. Such authorization ensures that resources in your storage account are accessible only when you want them to be, and only to those users or applications to whom you grant access.

Up until recently, the extension only supported a single form of authorization, which was based on using a Shared Key as explained in Authorize with Shared Key.

Azure Storage supports using Microsoft Entra ID to authorize requests to blob data. With Microsoft Entra ID, you can use Azure role-based access control (Azure RBAC) to grant permissions to a security principal, which may be a user, group, or application service principal. The security principal is authenticated by Microsoft Entra ID to return an OAuth 2.0 token. The token can then be used to authorize a request against the Blob service.

This new feature in preview, introduced in the pg_azure_storage extension, brings the possibility to leverage a more flexible, more secure, and more recommended authorization mechanism for the requests issued against the Blob service of the Azure Storage account. That mechanism is explained in Authorize with Microsoft Entra ID. It leverages the Microsoft Entra ID service principal that is created when you enable System Assigned Managed Identity in your instance of Azure Database for PostgreSQL Flexible Server.

To learn more about pg_azure_storage extension in Flexible Server, you can refer to the following documentation articles:

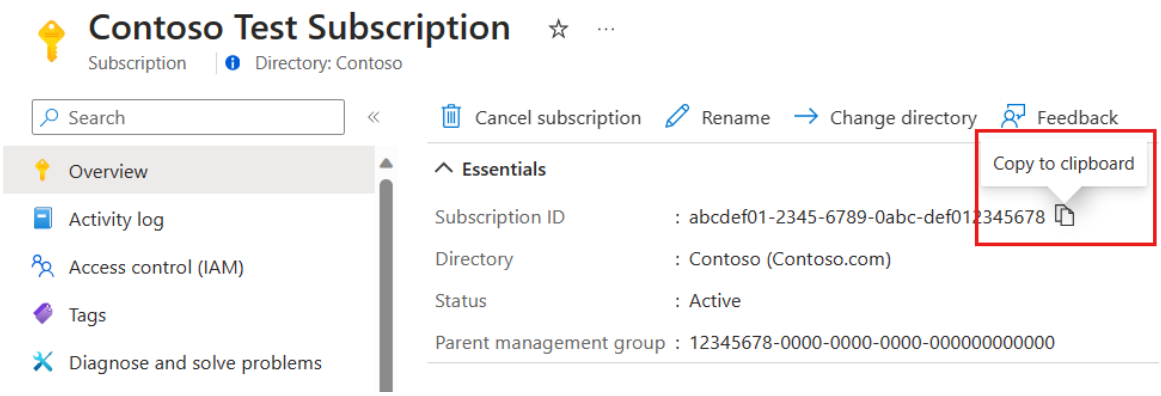
- Azure Storage extension in Azure Database for PostgreSQL
- Copy data with Azure Storage Extension on Azure Database for PostgreSQL

Getting started

Prerequisites

- Azure subscription. You will need subscription ID to start using the API (Reference documentation - Get subscription and tenant IDs in the Azure portal - Azure portal |

Microsoft Learn)



- You will have to provide us the region which you want to test

Prepare an instance of Flexible Server

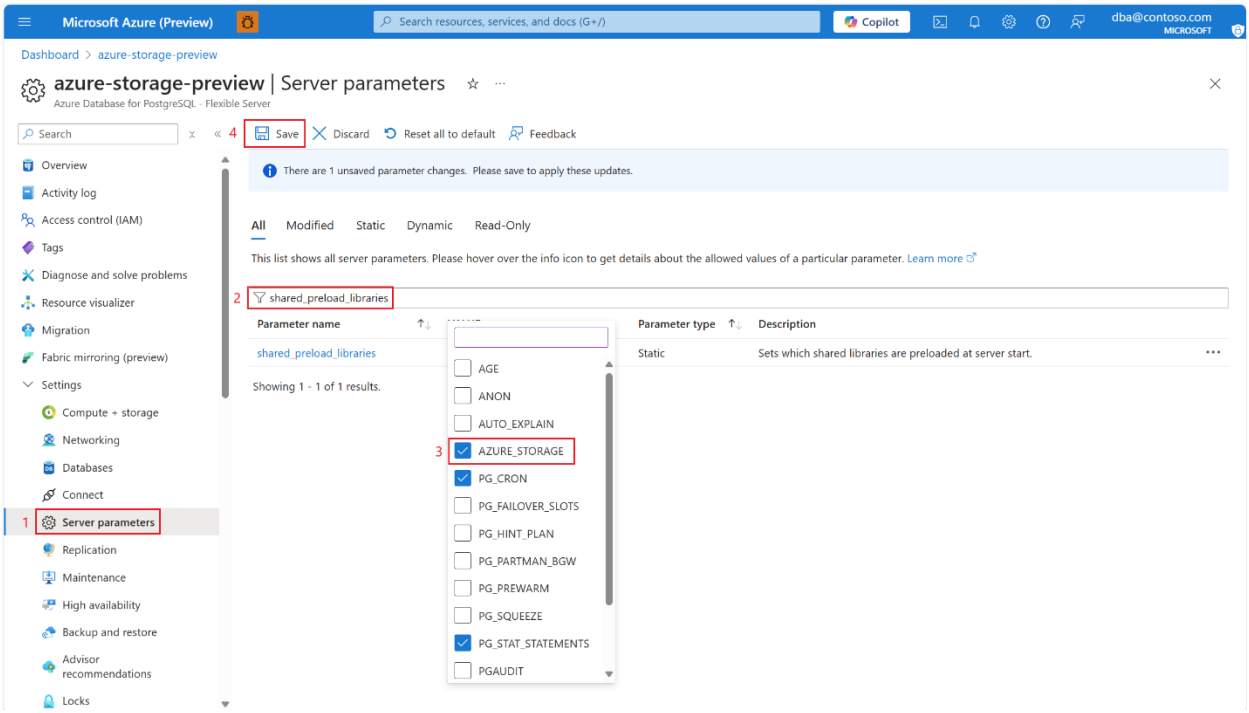
Follow the corresponding instructions in the documentation, to create an Azure Database for PostgreSQL Flexible Server instance using either of the following methods: Servers – Create API, ARM template, Azure CLI, or Azure portal.

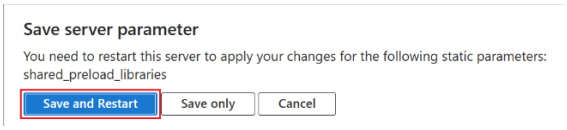
As an alternative, use an existing instance.

Include azure_storage in shared_preload_libraries

Make sure you include azure_storage in the list of shared_preload_libraries. You can do so in any of the following ways:

Using Azure portal:





Using Azure CLI

(because this parameter is static, it requires a restart of the server for the change to take effect)

```
az postgres flexible-server parameter set --resource-group <flexible_server_resource_group> --server-name <flexible_server_name> --name shared_preload_libraries --source user-override --value azure_storage,$(az postgres flexible-server parameter show --resource-group <flexible_server_resource_group> --server-name <flexible_server_name> --name shared_preload_libraries --query value --output tsv)
```

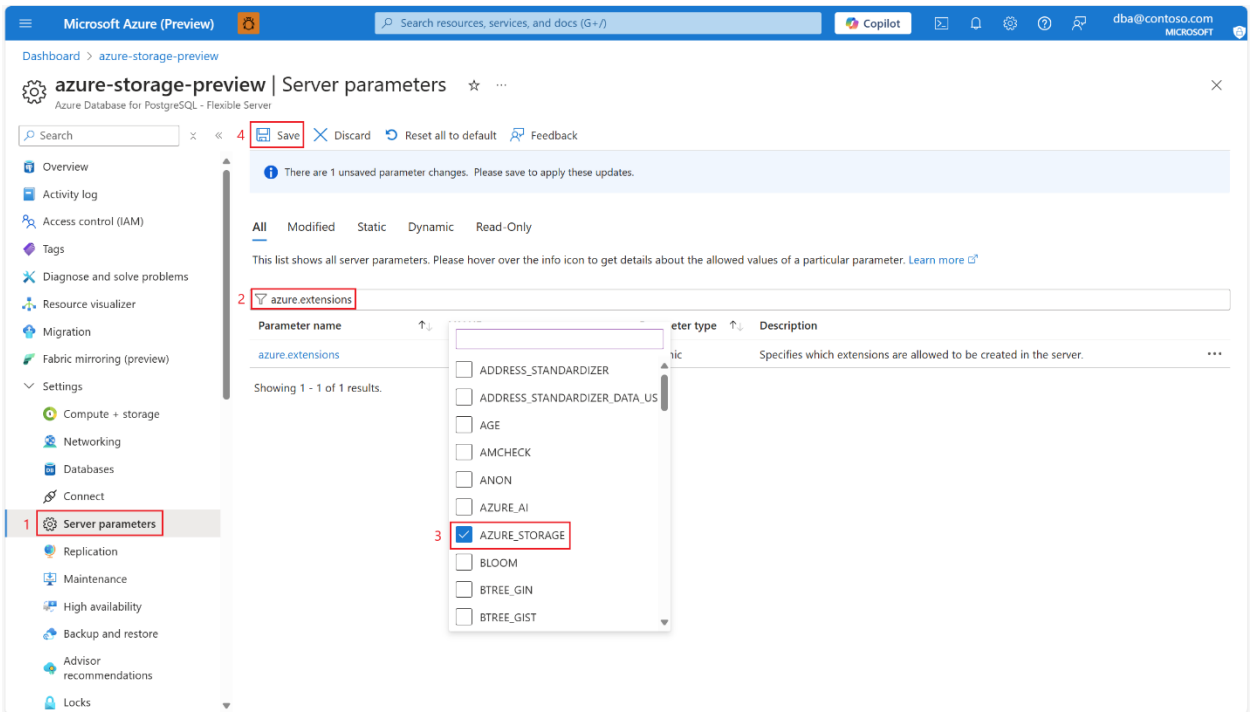
```
az postgres flexible-server restart --resource-group <flexible_server_resource_group> --name <flexible_server_name>
```

Using Configurations – Put REST API

Allow-list pg_azure_extension

Make sure you allow-list the pg_azure_extension, so that later you can run CREATE EXTENSION azure_storage on any database in which you want to put the SQL objects created by the extension in the azure_storage schema. You can do so in any of the following ways:

Using Azure portal



Using Azure CLI

```
az postgres flexible-server parameter set --resource-group <flexible_server_resource_group> --server-name <flexible_server_name> --name azure.extensions --source user-override --value azure_storage,$(az postgres flexible-server parameter show --resource-group <flexible_server_resource_group> --server-name <flexible_server_name> --name azure.extensions --query value --output tsv)
```

Using Configurations – Put REST API

Create the extension

With the extension allow-listed, using the client tool of your preference (psql, pgAdmin, etc), connect to any of the databases in your instance of Azure Database for PostgreSQL Flexible Server from which you want to use the SQL objects created by the pg_azure_storage extension, and execute the following command:

```
create extension azure_storage;
```

Confirm version of extension

Confirm that the instance of Flexible Server you've deployed, already has version 1.5 or higher of the pg_azure_storage extension, which is the one that has the functionality being evaluated in this private preview. To do so, execute the following query and make sure that the value reported in column installed_version is 1.5 or higher:

```
select * from pg_available_extensions where name =
'azure_storage';
```

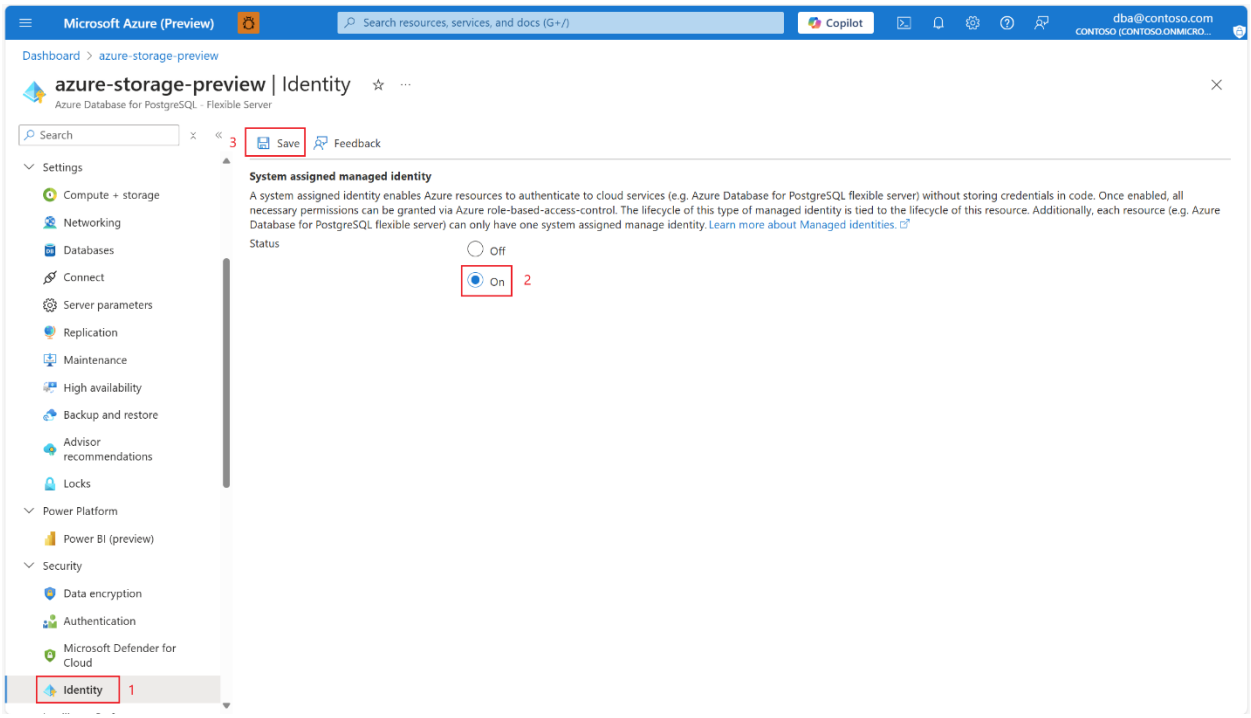
Results from executing that query should look like this:

name	default_version	installed_version	comment
azure_storage	1.5	1.5	Azure integration for PostgreSQL

Enable System Assigned Managed Identity

If the instance was created using the portal or it was created through any other means but, without having enabled a system assigned managed identity or a user assigned managed identity, it's time to enable the system assigned managed identity or to create one user assigned managed identity and associate it to the instance of Flexible Server. That can be done in any of the following ways:

Using Azure portal



Using Azure CLI

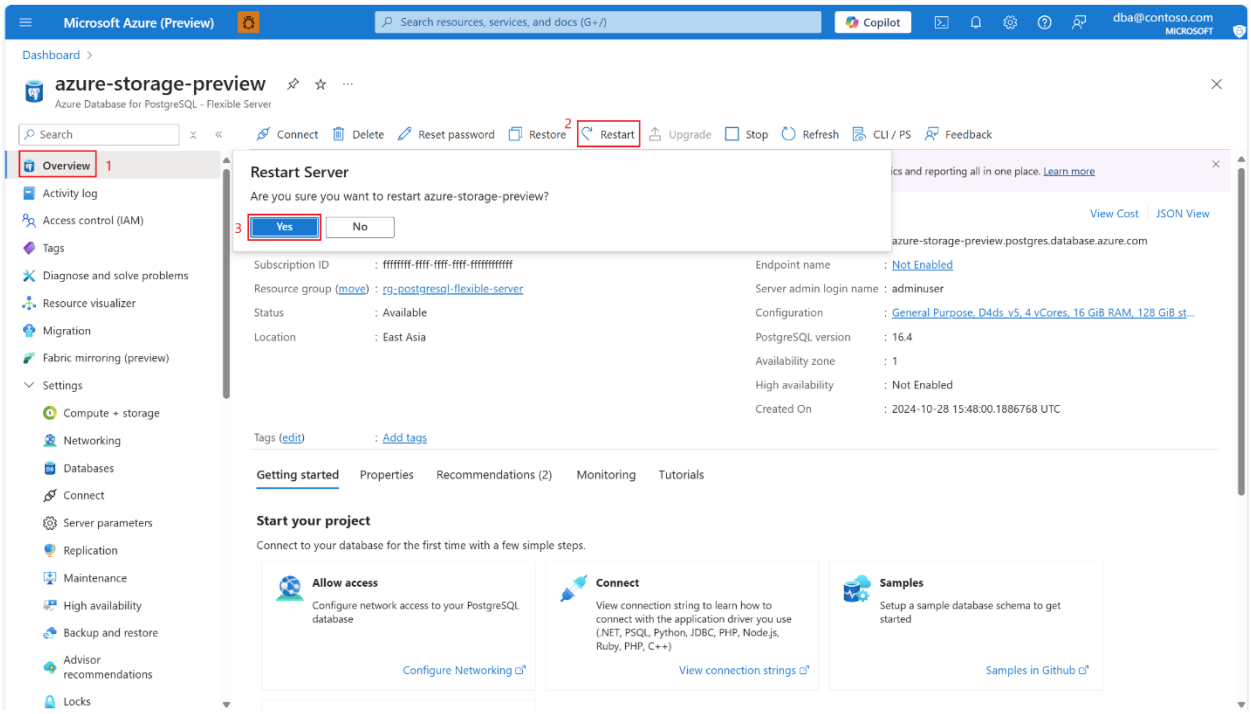
```
az rest --method patch --url
https://management.azure.com/subscriptions/<subscriptionId>/resourceGr
oups/<flexible_server_resource_group>/providers/Microsoft.DBforPostgre
SQL/flexibleServers/<flexible_server_name>?api-version=2024-08-01 --
body '{"identity":{"type":"SystemAssigned"}}'
```

Using the Servers – Update REST API

Restart the service

After having enabled a System Assigned Managed Identity to the instance of Flexible Server, you must restart the service for the extension to be able to use that identity. You can do so in any of the following ways:

Using Azure portal:



Using Azure CLI

```
az postgres flexible-server restart --resource-group <flexible_server_resource_group> --name <flexible_server_name>
```

Using the Servers – Restart REST API

Create an Azure Storage account

Create a storage account which you can use to test out the feature. For that, you can do it in the following way:

Using Azure CLI

```
az storage account create --location <location> --resource-group <storage_account_resource_group> --name <storage_account_name> --sku Standard_LRS --allow-shared-key-access false --allow-blob-public-access false
```

Create a blob container

1. Grant your user Storage Account Contributor so that it can create a blob container, and create the blob container:

```
az role assignment create --assignee $(az ad signed-in-user show --
query id --output tsv) --role "Storage Account Contributor" --scope
$(az postgres flexible-server show --resource-group
<flexible_server_resource_group> --name <flexible_server_name> --
query id --output tsv)
```

2. Create the blob container:

```
az storage container create --account-name <storage_account_name>
--name <container_name> --auth-mode login
```

Grant RBAC permissions on the Storage Account to the System Assigned Managed Identity

On the Azure Storage account, assign the Storage Blob Data Contributor to the System-Assigned Managed Identity of the Flexible Server instance:

```
az role assignment create --assignee $(az postgres flexible-server
show --resource-group <flexible_server_resource_group> --name
<flexible_server_name> --query identity.principalId --output tsv) --
role "Storage Blob Data Contributor" --scope $(az storage account show
--resource-group <storage_account_resource_group> --name
<storage_account_name> --query id --output tsv)
```

Test the new feature

Now that the extension is installed, and a storage account has been configured so that the extension can access it authenticating via the System Assigned Managed Identity of the instance of Azure Database for Flexible Server, you can start using the functionality of the extension.

For that, first create a reference to your storage account by using the new overload of the `azure_storage.add_account(account_config jsonb) RETURNS VOID` function that allows you to specify the authentication type of your preference. And let's leverage the new helper function `azure_storage.account_options_managed_identity(name text, type azure_storage.storage_type) RETURNS jsonb` that constructs the value to be passed as the `account_config` parameter to `azure_storage.add_account()`:

```
select

  azure_storage.account_add(

    azure_storage.account_options_managed_identity(
```

```
'<storage_account_name>',
'<container_name>'));
```

With that, you can proceed to use all the functionality offered by the extension as described in Azure Storage extension in Azure Database for PostgreSQL - Flexible Server and in Copy data with Azure Storage Extension on Azure Database for PostgreSQL.

Possible errors

ERROR: azure_storage: Permission is no sufficient to perform requested operation

That's the error you get when you execute any of the functions that interact with Azure Storage (azure_storage.blob_list, azure_storage.blob_get or azure_storage.blob_put), if you have not granted the adequate data plane roles or permissions to the System Assigned Managed Identity (typically a minimum of **Storage Blob Data Contributor** for azure_storage.blob_put, and a minimum of **Storage Blob Data Reader** for the other two functions).

It may also be the case that you have already granted the minimum required permissions but they are not yet in effect. It can take a few minutes until those permissions get propagated.

ERROR: azure_storage: missing storage credentials

That's the error you get when you execute any of the functions that interact with Azure Storage (azure_storage.blob_list, azure_storage.blob_get or azure_storage.blob_put), if you have not previously registered the credentials with which you want the extension to authenticate with the storage account.

ERROR: azure_storage: internal error while connecting

That's the error you get when the instance of Flexible Server doesn't have the System Assigned Managed Identity enabled.

ERROR: azure_storage: storage credentials invalid format

That's the error you get when the instance of Flexible Server has the System Assigned Managed Identity enabled, but the server has not been restarted after enabling it.

ERROR: azure_storage: current user <*user_or_role*> is not allowed to use storage account <*storage_account*>

Frequently asked questions

N/A

Changelog

Date	Changes
2024-10-31	Wrote first draft of the document