Azure Blob Storage and Databricks Connection

Step by Step Guidelines for connecting Azure Data Lake/Blob Storage with Azure Databricks.

Connect Azure Blob Storage or ADLS Gen2 to Databricks

We can connect Azure Storage to Databricks mainly in two ways:

- **Mounting the storage to DBFS (Databricks File System)**
- **Direct access without mounting**

1. Mounting Azure Blob Storage / ADLS Gen2 to DBFS

Mounting creates a folder in Databricks file system that points to Azure Storage container.

Using Storage Account Access Key

```
storage_account_name = "<your-storage-account-name>"

storage_account_access_key = "<your-storage-account-access-key>"

container_name = "<your-container-name>"

spark.conf.set(f"fs.azure.account.key.{storage_account_name}.blob.core.windows.net",
storage_account_access_key)

mount_point = f"/mnt/{container_name}"

dbutils.fs.mount(

  source = f"wasbs://{container_name}@{storage_account_name}.blob.core.windows.net",

  mount_point = mount_point,

  extra_configs = {f"fs.azure.account.key.{storage_account_name}.blob.core.windows.net":
storage_account_access_key}

)
```

Mounting ADLS Gen2 (Hierarchical Namespace enabled)

ADLS Gen2 requires slightly different configs:

```
storage_account_name = "<your-storage-account-name>"
container_name = "<your-container-name>"
storage_account_access_key = "<your-access-key>"
spark.conf.set(f"fs.azure.account.key.{storage_account_name}.dfs.core.windows.net", storage_account_access_key)
mount_point = f"/mnt/{container_name}"
dbutils.fs.mount(
  source = f"abfss://{container_name}@{storage_account_name}.dfs.core.windows.net/",
  mount_point = mount_point,
  extra_configs = {f"fs.azure.account.key.{storage_account_name}.dfs.core.windows.net": storage_account_access_key}
)
```

2. Direct Access without Mounting

We can directly read files without mounting by specifying the full path and configurations in Spark read.

```
storage_account_name = "<your-storage-account-name>"
container_name = "<your-container-name>"
storage_account_access_key = "<your-access-key>"
spark.conf.set(f"fs.azure.account.key.{storage_account_name}.dfs.core.windows.net", storage_account_access_key)
file_path = f"abfss://{container_name}@{storage_account_name}.dfs.core.windows.net/path/to/your/file.csv"
```

3. Using Service Principal Authentication

Instead of using access keys, we can authenticate using Azure AD service principals.

- Register an app in Azure AD and give it Storage Blob Data Contributor role on the storage account which needs to be connected to Azure Databricks.
- Then configure the service principal details in your spark config:

```
configs = {

  "fs.azure.account.auth.type": "OAuth",

  "fs.azure.account.oauth.provider.type":
"org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",

  "fs.azure.account.oauth2.client.id": "<application-id>",

  "fs.azure.account.oauth2.client.secret": "<authentication-key>",

  "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/<directory-
id>/oauth2/token"

}

spark.conf.set(f"fs.azure.account.auth.type.{storage_account_name}.dfs.core.windows.net",
configs["fs.azure.account.auth.type"])

spark.conf.set(f"fs.azure.account.oauth.provider.type.{storage_account_name}.dfs.core.wind
ows.net", configs["fs.azure.account.oauth.provider.type"])

spark.conf.set(f"fs.azure.account.oauth2.client.id.{storage_account_name}.dfs.core.windows.
net", configs["fs.azure.account.oauth2.client.id"])

spark.conf.set(f"fs.azure.account.oauth2.client.secret.{storage_account_name}.dfs.core.wind
ows.net", configs["fs.azure.account.oauth2.client.secret"])

spark.conf.set(f"fs.azure.account.oauth2.client.endpoint.{storage_account_name}.dfs.core.wi
ndows.net", configs["fs.azure.account.oauth2.client.endpoint"])

file_path =
f"abfss://{container_name}@{storage_account_name}.dfs.core.windows.net/path/to/file.csv"
```

Using Service Principle Authentication is more secure than the other options.

Make sure that the access keys, secret values and application IDs are not hardcoded in the databricks notebook. We can use **Secret Scope** and **Azure Vault Keys** for saving and extracting the keys to and from them via databricks notebook

Documentations:

Using **Azure Vault Keys:**
https://www.youtube.com/watch?v=ul4Gqehas0w&list=PLMWaZteqtEaKi4WAePWtCSQCfQpvBT2U1&index=25

Using **Databricks Secret Scope:**

https://www.youtube.com/watch?v=BT4TTRzEiWo&list=PLMWaZteqtEaKi4WAePWtCSQCfQpvBT2U1&index=26

Connection via **Account Key:**

https://www.youtube.com/watch?v=h_mbpd0oqOE&list=PLMWaZteqtEaKi4WAePWtCSQCfQpvBT2U1&index=28

Connection via **Service Principle Connection:**

https://www.youtube.com/watch?v=En95WQYYmbo&list=PLMWaZteqtEaKi4WAePWtCSQCfQpvBT2U1&index=30

https://docs.databricks.com/aws/en/connect/storage/azure-storage

(In this implementation, we have used Storage Account Access key to connect the Azure Data Lake Blob Storage and mounted the container in the databricks data catalog. The container has been mounted in **/mnt/dataset** path as evident in the notebooks.)