|  |  |
| --- | --- |
| A picture of a winding road and trees  Azure Datbricks and Azure Synapse Analytics  Load data from Azure Container to Azure Synapse Table | Abstract  This document describes how to load data present in Azure Blob Storage to Azure Synapse tables using Azure Python SDK.  Ashish Sinha  Senior Data Engineer |

Table of Contents

[**1.** **Introduction:** 2](#_Toc71416495)

[**2.** **Prerequisites:** 2](#_Toc71416496)

[**3.** **Azure Blob Storage** 3](#_Toc71416497)

[**4.** **Azure Synapse Details** 5](#_Toc71416498)

[**5.** **List and Read Files in container** 5](#_Toc71416499)

[**6.** **Load Data to Azure Synapse** 6](#_Toc71416500)

[**7.** **Functions to Copy and Delete Blob** 7](#_Toc71416501)

# **Introduction:**

The requirement is to push data from all the csv files present inside a container folder to Azure Synapse Table. The container heirachy is landing-area/ folder-name/ inbound. The file are present in Inbound folder and after being processed needs to be archived in the landing-area/ folder-name/ archive.

# **Prerequisites:**

* Azure Subscription
* Azure Storage Account
* Azure Key Vault
* Azure Databricks
* Azure Databricks Secret Scope
* Azure Synapse Analytics

1. To create your free Azure account please go to this [link](https://azure.microsoft.com/en-us/free/)
2. To see how to create your Azure resource group, Azure Storage account, Azure Databricks workspace and Azure Key-vault please follow the steps [here](https://docs.microsoft.com/en-us/azure/databricks/scenarios/store-secrets-azure-key-vault#prerequisites).

**Note**: *In the link above, we are taught how to mount your container in dbfs which is not what we are going to do here. We will be using Python SDK to connect to Azure Blob Storage and load data into Azure Synapse.*

1. To read the official documentation on how to connect to Azure Synapse Analytics using Databricks using Scala or SQL you can follow this [link](https://docs.microsoft.com/en-us/azure/databricks/data/data-sources/azure/synapse-analytics).

**Note**: *Don’t forget to Set****Allow access to Azure services****to****ON,*** *on the firewall pane of the Azure Synapse server through Azure portal.*

# **Azure Blob Storage**

*Connect to Azure Blob Storage*

***Import Libraries:***

*First of all, we will import all the required libraries to connect to azure storage blob*.

# Import Required Libraries

*from azure.storage.blob import BlobServiceClient*

*from pyspark.sql.functions import \**

*from azure.storage.blob import ContainerClient*

***Details to Access Blob Storage:***

*We will then create some variables with the required information to connect to Azure Blob Storage. This includes storage account name, container name, SAS token, wasbs path, access keys.*

*blob\_account\_name = "<your-storage-account-name>"*

*blob\_container\_name = "<your-container-name>"*

*blob\_folder\_name = "<your-folder-in-container>"*

*blob\_relative\_path = blob\_folder\_name + "/Inbound"*

*blob\_sas\_token = r”<your-storage-account-sas-token>”*

*blobwasbspath = "<your wasbs path for blob>"*

# For e.g., wasbs://<blob\_container\_name> @blob\_account\_name.blob.core.windows.net/

*blob\_account\_key = "<your\_blob\_account\_key>"*

# wasbs path for a blob and spark conf set

*wasbs\_path = 'wasbs://%s@%s.blob.core.windows.net/%s' % (blob\_container\_name, blob\_account\_name, blob\_relative\_path)*

*spark.conf.set('fs.azure.sas.%s.%s.blob.core.windows.net' % (blob\_container\_name, blob\_account\_name), blob\_sas\_token)*

*print('Remote blob path: ' + wasbs\_path)*

#connection string for blob

*connection\_string = "DefaultEndpointsProtocol=https;AccountName=%s;AccountKey=%s;EndpointSuffix=core.windows.net" % (blob\_account\_name, blob\_account\_key)*

# Set up the Blob storage account access key in the notebook session conf.

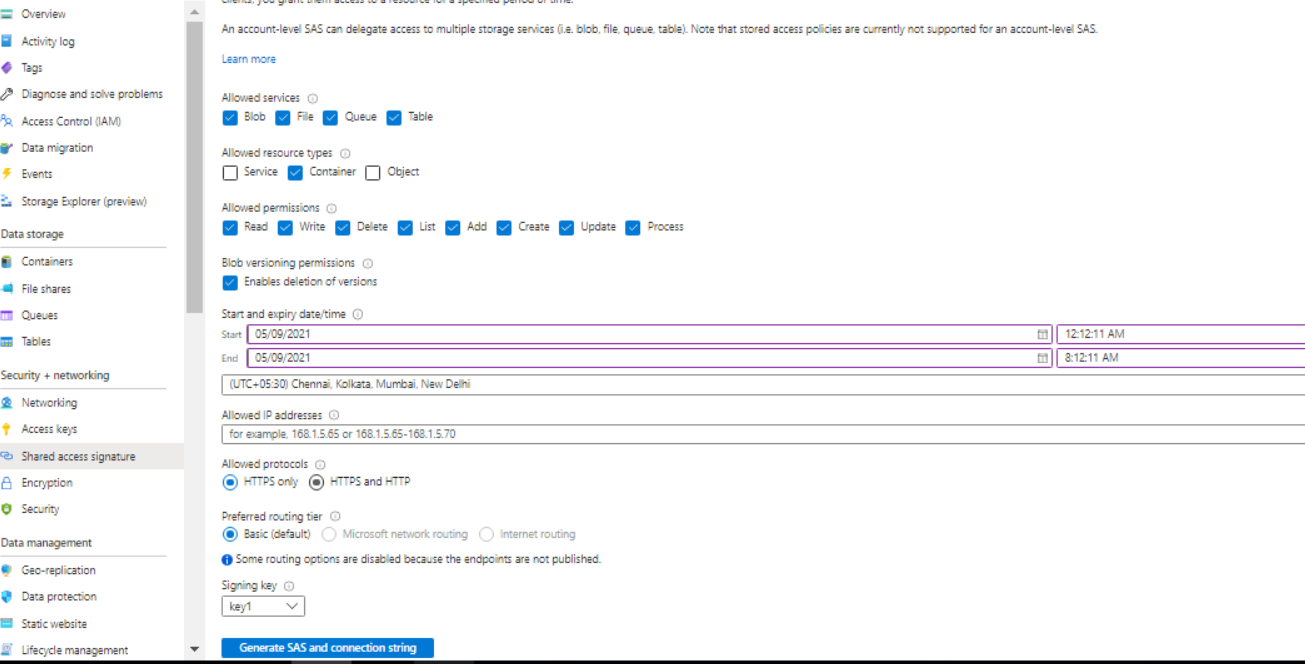
spark.conf.set(

"fs.azure.account.key. <storage-account-name>. blob.core.windows.net",

"<blob\_account\_key>")

***Steps to create Shared access signature:*** -

1. Go to your storage account
2. Go to Shared Access Signature in the Left Pane.
3. Click on “***Container***” in ***Allowed Resources Types***
4. Select the Required Permissions
5. Select the start and expiry of token
6. Click on ***Generate SAS and Connection String.***



1. After that it will generate
2. Connection String
3. SAS Token: - The query string that includes all of the information required to authenticate the SAS, as well as to specify the service, resource, and permissions available for access, and the time interval over which the signature is valid. To construct an account SAS URL, append the SAS token to the URL for a storage service, or use one of the URLs below.
4. Blob service SAS URL
5. File service SAS URL
6. Queue service SAS URL
7. Table service SAS URL

***Steps to get Access Keys: -***

1. In the same Left Pane, click on Access Keys
2. By default, there are 2 keys present. You can use either one

# **Azure Synapse Details**

***Connect to Azure Synapse Workspace***

*Now we will connect to Azure Synapse workspace. For this we would require account name, the default port and database name.*

#connection details for synapse analytics

dwServer = "<your-synapse-workspace>. sql.azuresynapse.net"

dwPort = "1433"

dwDBName = "<db name>"

#Retrieve sql username and password from key vault scope

dwUserName = dbutils.secrets.get(scope = "key-vault-secrets", key = "sqluser")

dwPassword = dbutils.secrets.get(scope = "key-vault-secrets", key = "sqlpassword")

#Create link to connect with the Synapse via JDBC url

tempDir = "wasbs://" + blob\_container\_name + "@" + blob\_account\_name + ".blob.core.windows.net/" + blob\_folder\_name

dwUrl = "jdbc:sqlserver://"+dwServer+":"+dwPort+";database="+dwDBName+";user="+dwUserName+";password="+dwPassword+";encrypt=true;trustServerCertificate=true;hostNameInCertificate=\*.sql.azuresynapse.net;loginTimeout=30;"

# **List and Read Files in container**

*Now we will list the files present in storage account and read it into PySpark data frame. This data frame would be used to load data into synapse table*

#Container connection to access blobs in a container

container = ContainerClient.from\_connection\_string(conn\_str=connection\_string, container\_name=blob\_container\_name)

#list of all files in a container

files = []

count = 0

blob\_list = container.list\_blobs()

for blob in blob\_list:

files.append(blob.name)

count=count+1

print("Files to be processed are: -")

print(files)

# **Load Data to Azure Synapse**

*Here we will load the data frame into table. We are using overwrite to recreate and load table if it exists. We can also use append to load data into existing table and it will append with existing records. But for this we need to make sure that the data frame structure and target table structure is same otherwise the code will fail*

*.*

#read the csv files and populate the corresponding tables

table\_names = ["I","PR","RS","TS","ASGN\_BLS","PG\_BS","TS\_BS","RESOURCES","TmSet"]

#loop to table names and process the csv files to Synapse Database

for name in table\_names:

for file in files:

if name in file:

print("File name:- {}".format(file))

df = spark.read.csv(wasbs\_path + "/"+file, header = 'true')

# Add Load Date column in the data frame as current timestamp

df = df.withColumn("LOAD\_DATE",current\_timestamp())

#Convert date columns in file to datatype datetime

datecols = [x for x in df.columns if x.endswith('Date')]

for col in datecols:

df = df.withColumn(col,to\_timestamp(df[col], 'yyyy-MM-dd HH:mm:ss'))

#write the data from the file to table

df.write \

.format("com.databricks.spark.sqldw") \

.option("url", dwUrl) \

.option("forwardSparkAzureStorageCredentials", "true") \

.option("dbTable", "dbo."+name+"\_RAW ") \

.option("tempDir", tempDir) \

.option("truncate","true") \

.option("maxStrLength", "4000" ) \

.mode("overwrite") \

print("File {} is loaded to synpase".format(file))

# **Functions to Copy and Delete Blob**

*Here we have function to copy blob from one location to another and then delete it. This basically works as Move Blob from one location to another.*

# Function to Copy blob from one folder to another

def copy\_blob(account\_name,container,folder,file,con\_str):

status = None

blob\_service\_client = BlobServiceClient.from\_connection\_string(con\_str)

source\_blob = "https://%s.blob.core.windows.net/%s/%s/Inbound/%s" % (account\_name,container,folder,file)

copied\_blob = blob\_service\_client.get\_blob\_client(container+"/"+folder+"/Archive",file)

copied\_blob.start\_copy\_from\_url(source\_blob)

for i in range(10):

props = copied\_blob.get\_blob\_properties()

status = props.copy.status

print("Copy Status: "+status)

if status == "success":

break

time.sleep(3)

if status != "success":

# if not finished after 30s, cancel the operation

props = copied\_blob.get\_blob\_properties()

print(props.copy.status)

copy\_id = props.copy.id

copied\_blob.abort\_copy(copy\_id)

props = copied\_blob.get\_blob\_properties()

print(props.copy.status)

# Function to delete blob

def delete\_blob(container\_client,path,file):

full\_path = path+"/"+file

container\_client.delete\_blob(full\_path)

print("File {} has been deleted".format(file))