Accessing Data Files on Cloud Object Storage

Hands-On Workshop Instructions

Stephen Foerster
Data Engineering and Decisioning Lead, SAS Education

Authenticate to our GelDM Azure Tenant

1. Launch Google Chrome from the desktop and select **Microsoft Azure Portal** from the Bookmarks bar or navigate to **https://portal.azure.com/**.

2. Click Sign in and enter **geldmui@gelenable.sas.com** when asked to pick an account.

3. Use the password found at https://gelweb.race.sas.com/scripts/gelenable/users/geldmui@gelenable.sas.com.txt.

4. When asked for a token, use the one you found at https://geltotp.gelenable.sas.com/user/geldmui@gelenable.sas.com. The token changes every 25 seconds. Take the latest one and enter it before it expires.

5. If asked to stay signed in, answer yes.

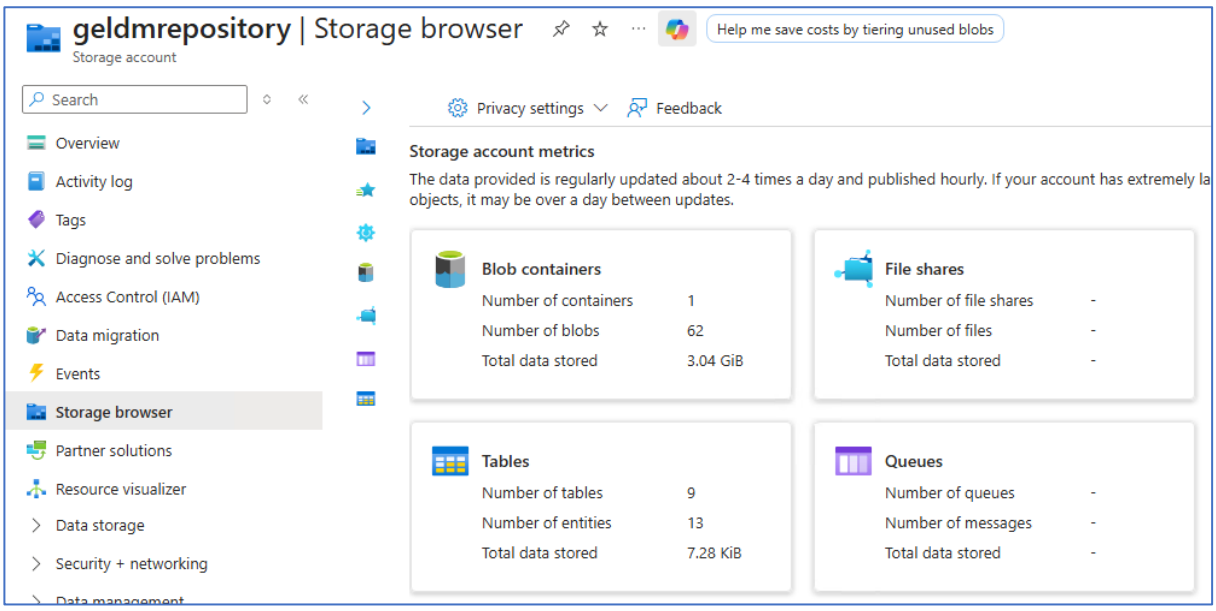
Explore Azure Object Storage

6. In the Azure portal, search "**geldmrepository**" in the ***Search resources, services, and docs (G+/)*** box and click it when it appears under Resources.

   Alternatively, navigate directly using this link:
   https://portal.azure.com/#@gelenable.sas.com/resource/subscriptions/5483d6c1-65f0-400d-9910-a7a448614167/resourceGroups/GEL_Storage_Accounts/providers/Microsoft.Storage/storageAccounts/geldmrepository/storagebrowser

7. If not already there, click **Storage browser** in the left sidebar.

8. You should see this:



9. Click the **Blob containers** tile and click **data** and then **yellow_taxi** on the following screens.

10. Explore the geldmrepository data container. Note there are Json files, parquet files, iceberg files, deltalake files, and csv files. Note that each file has its own URL.

Explore the SAS Studio Interface

11. Launch Google Chrome from the desktop and select **SAS Studio** from the Bookmarks bar.

12. Enter **student** in the **User ID** field and **Metadata0** in the **Password** field. Click **Sign in**. Select **Yes** when prompted to opt in to all assumable groups.

13. Click the **applications menu** icon  in the upper left corner. The applications menu shows all the available applications. Depending on your environment, these applications might differ. Select **Develop Code and Flows** to open SAS Studio. We use SAS Studio to write and submit code in this workshop.

14. The Start Page tab is open by default, and it provides convenient links to start writing a program or use other point-and-click tools, including importing and querying data. Click **Program in SAS** to open a new tab with the Program Editor. Each item that you open in SAS Studio has its own tab.

Establish Azure Connection

15. Select **SAS Server** on the left and expand **Files** > **Home** > **Courses** > **CLOBJ**. Double-click **CLOBJ.sas**.

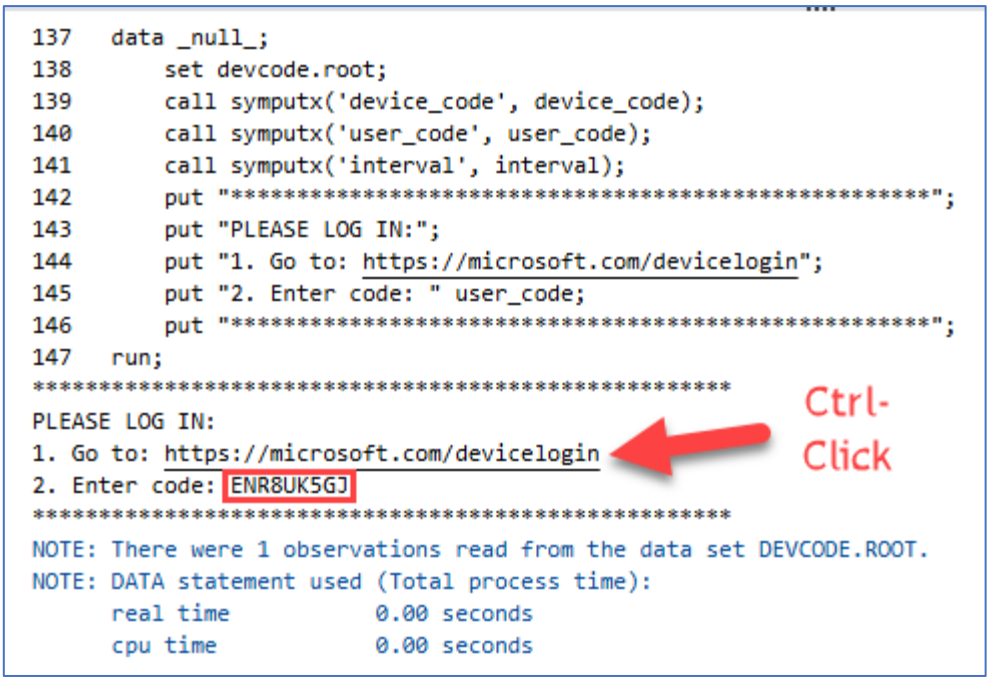
16. Execute the code under the **/* ADLS Connection – Part 1 */** commend header.

```
/* Retrieve the Geldmrepository Resource IDs */

filename getinfo url "https://gelgitlab.race.sas.com/GEL/utilities/edge/-
/raw/main/scripts/sas/collect_adls_info.sas" ;
%include getinfo / nosource2 ;
options azuretenantid="&TENANT_ID" ;

%let requestbody =
"client_id=&APP_ID.%str(&)scope=https://geldmrepository.blob.core.windows.net/
.default offline_access" ;

*** Request Device Code and User Code ***;
filename devcode temp;
proc http
    method="POST"
url="https://login.microsoftonline.com/&TENANT_ID./oauth2/v2.0/devicecode"
    ct="application/x-www-form-urlencoded"
    in=&requestbody
    out=devcode;
quit;

*** Parse the response (JSON) ***;
```

```
libname devcode json fileref=devcode;
data _null_;
    set devcode.root;
    call symputx('device_code', device_code);
    call symputx('user_code', user_code);
    call symputx('interval', interval);
    put "**************************************************";
    put "PLEASE LOG IN:";
    put "1. Go to: https://microsoft.com/devicelogin";
    put "2. Enter code: " user_code;
    put "**************************************************";
run;
```

The code retrieves and executes an external SAS program which sets the Azure Tenant ID, Azure Storage Account, Client ID, and Client secret. These values will be visible in the log. It, then, uses that information to request a device code from Azure.

17. **Ctrl-Click** the https://microsoft.com/devicelogin link shown in the SAS log and enter the code also shown in the log.



18. If requested, select the **gelmdui@gelenable.sas.com** user. If asked, enter the user's password from earlier, and enter the user token from geltotp URL from earlier. The token changes every 25 seconds. Take the latest one and enter it before it expires.

19. Execute the code under the **/* ADLS Connection – Part 2 */** commend header.

```
data _null_;
    call symputx('azure_cache_path', getoption('AZUREAUTHCACHELOC'));
run;

filename authcode "&azure_cache_path/.sasadls_1001.json";;
```

```
*** Request the Device Code Authorization Token ***;
%let requestbody2 = "grant_type=urn:ietf:params:oauth:grant-
type:device_code%str(&)client_id=&APP_ID%str(&)device_code=&device_code" ;

proc http
    method="POST"
    url="https://login.microsoftonline.com/&TENANT_ID./oauth2/v2.0/token"
    ct="application/x-www-form-urlencoded"
    in=&requestbody2
    out=authcode;
quit;

libname authcode json fileref=authcode;

data _null_;
    retain success 0;
    set authcode.alldata end=eof;
    if p1 = "refresh_token" then success = 1;
    if eof then if success then do;
         put "*******************************************************";
         put "Authorization Successful!!  Please move to the next step.";
         put "*******************************************************";
      end;
      else do;
         put "*****************************************************";
         put "Please run part 2 again.";
         put "*****************************************************";
      end;
run;
```

The code requests authorization to the device code retrieved earlier from Azure and send the response to the SAS Azure Authorization Cache location.

20. IF you receive the message that the authorization process was successful, move on to the next step. If not, please run the code under **/* ADLS Connection – Part 2 */** again.

Process Azure Object Storage Files with SAS Studio

1. Deltalake via a DuckDB Libname

21. Execute the code under the **/* Deltalake on ADLS */** commend header.

```
*** Create a SAS library to the Azure Blob directory ***;
libname duck_az duckdb file_path="az://data/yellow_taxi/deltalake"
file_type=delta  azure_tenant_id="&TENANT_ID"
  azure_accountName="&AZ_STORAGE_ACCOUNT"
  azure_client_id="&CLIENT_ID"
  azure_client_secret="&CLIENT_SECRET" ;

*** Validate the Library ***;
proc datasets lib=duck_az ;  quit ;
```

```
*** Run SAS analytics directly on the Blob Deltalake file ***;
proc anova data=duck_az.yellow_tripdata plots=none;
     class payment_type;
     model total_amount = payment_type;
     title "One-Way ANOVA: Fare by Payment Type";
quit;
```

The code runs an analysis of variance analysis directly on a deltalake file housed in Azure ADLS2 blob storage. Without contention, it should take around 7 seconds and produce statistics about the correlation between payment types (cash, credit cards, ...) and taxi fares.

2. Partitioned Parquet via a DuckDB Libname

22. Execute the code under the **/* Parquet on ADLS*/** commend header.

```
*** Create a SAS library to the Azure Blob directory ***;
libname duck_pq duckdb file_path="az://data/yellow_taxi/parquet"
file_type=parquet
  azure_tenant_id="&TENANT_ID"
  azure_accountName="&AZ_STORAGE_ACCOUNT"
  azure_client_id="&CLIENT_ID"
  azure_client_secret="&CLIENT_SECRET"
  directories_as_data=yes;

*** Validate the Library ***;
proc datasets lib=duck_pq ; quit ;

*** Run SAS analytics directly on the parquet file(s) ***;
proc tabulate data=duck_pq.yellow_tripdata_all_partitioned;
    class RatecodeID;
    var tip_amount;
    table RatecodeID, tip_amount*Mean;
run;
```

The code creates a crosstab report directly on parquet files housed in Azure ADLS2 blob storage. Note that the input is a partitioned parquet file made up of several files in several subdirectories.

3. CAS on an Iceberg via a DuckDB Libname

23. Execute the code under the **/* CAS on Iceberg from ADLS */** commend header.

```
*** Create a SAS library to the Azure Blob directory ***;
libname duck_ic duckdb
file_path="az://data/yellow_taxi/iceberg/yellow_taxi_warehouse/yt"
file_type=iceberg
  azure_tenant_id="&TENANT_ID"
```

```
  azure_accountName="&AZ_STORAGE_ACCOUNT"
  azure_client_id="&CLIENT_ID"
  azure_client_secret="&CLIENT_SECRET" ;

*** Validate the Library ***;
proc datasets lib=duck_ic ; quit ;

cas casauto sessopts=(metrics=true);

*** Load the Iceberg file to CAS ***;
proc casutil;
    droptable casdata="tripdata" incaslib="casuser" quiet; run;

    load data=duck_ic.yellow_tripdata casout="tripdata"
    outcaslib="casuser" promote;
quit;

*** Run CAS analytics on the loaded file ***;
proc cas;
simple.summary  /
 table={name="tripdata" caslib="casuser" groupBy="mnth"
        computedVars={{name="mnth"}}
        computedVarsProgram='mnth=month(datepart(pickup_datetime));'}
 inputs={"fare_amount","tip_amount", "trip_distance"}
 subSet={"MEAN", "MIN", "MAX"};
quit;
```

The code loads the iceberg data to CAS and creates a summary report.

4. SAS Dataset via FCOPY Macro

24. Execute the code under the **/* SAS7BDAT on ADLS*/** commend header.

```
*** Create a SAS library on the SAS Compute Server ***;
libname clobj "/home/sas/";

*** Copy the dataset from blob storage to the local library ***;
*** Parms:  Dataset name, adls path, adls fs, target libname ***;
%include "/home/sas/ReadSAS7BDATonAzureBlob.sas";

%ReadSAS7BDATonAzureBlob(prdsal3,clobj,data,clobj);

*** Process using SAS advanced file access capabilities ***;
data sample;
 start=ceil(20*ranuni(-1));
 drop start;
     do i=start to nobs by 20;
       set clobj.prdsal3 nobs=nobs point=i;
       output;
     end;
```

```
 stop;
run;
```

The code transfers a SAS dataset from ADLS2 blob storage to compute server and creates a 1/20 random sample using the POINT dataset option to pick every 20th record after starting at a random record.

5. CAS on a SAS Dataset via FCOPY Macro

25. Execute the code under the **/* CAS on SAS7BDAT from ADLS */** commend header.

```
*** Create a CAS session ***;
cas casauto sessopts=(metrics=true);

*** Copy the dataset from blob storage to the compute server ***;
*** Parms:  Dataset name, adls path, adls fs, target caslib ***;
%include "/home/sas/LoadCASSAS7BDATonAzureBlob.sas";

%LoadCASSAS7BDATonAzureBlob(prdsal3,clobj,data,casuser);

*** Run CAS analytics on the loaded file ***;
proc cas;
    simple.summary /
        table={caslib="casuser", name="prdsal3",
               groupBy={"country", "state"} },
        inputs={"actual", "predict"},
        subSet={"MEAN", "SUM", "MIN", "MAX", "N"};
quit;
```