Snowflake External Tables with S3 Integration

S3 Integration + Stages +Ext. table

Set-up:

Step 1: Configure access permissions for the S3 bucket:

Snowflake requires the following permissions on an S3 bucket and folder to be able to access files in the folder (and sub-folders):

- `s3:GetBucketLocation`
- `s3:GetObject`
- `s3:GetObjectVersion`
- `s3:ListBucket`

Creating an IAM policy:

- ☐ Log into the AWS Management Console.
- ☐ From the home dashboard, search for and select IAM.
- ☐ From the left-hand navigation pane, select Account settings.
- ☐ Under Security Token Service (STS) in the Endpoints list, find the Snowflake region where your account is located. If the STS status is inactive, move the toggle to Active.
- ☐ From the left-hand navigation pane, select Policies.
- ☐ Select Create Policy.
- ☐ For Policy editor, select JSON.

- [ ] Copy and paste the text into the policy editor (replace bucket and prefix with your actual bucket name and folder path prefix, Setting the "s3:prefix": condition to either ["*"] or ["<path>/*"] grants access to all prefixes in the specified bucket or path in the bucket) :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion"
      ],
      "Resource": "arn:aws:s3:::<bucket>/<prefix>/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::<bucket>",
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "<prefix>/*"
          ]
        }
      }
    }
  ]
}
```

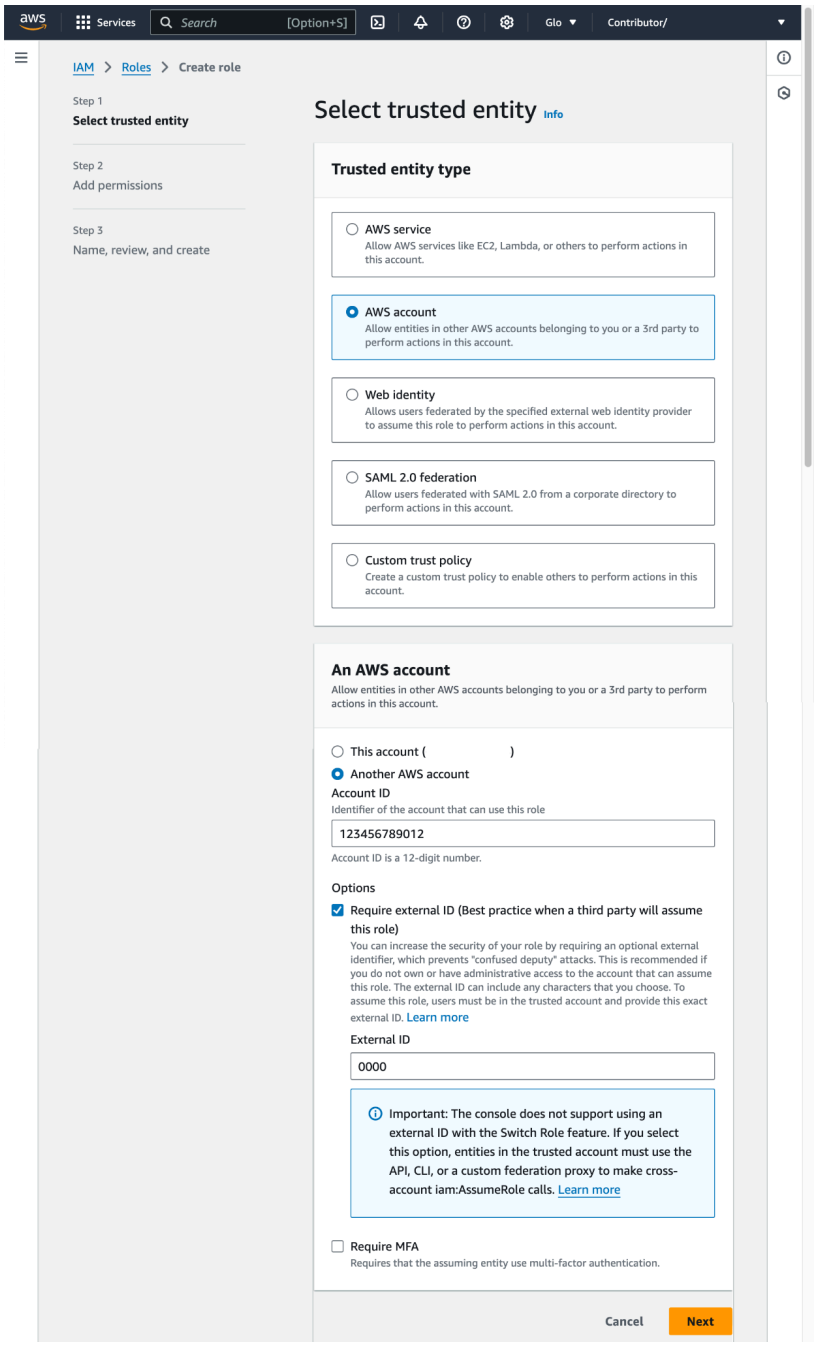
- [ ] Select Next.
- [ ] Enter a Policy name (for example, snowflake_access) and an optional Description.
- [ ] Select Create policy.

Step 2: Create the IAM role in AWS

- [ ] From the left-hand navigation pane in the Identity and Access Management (IAM) Dashboard, select Roles.
- [ ] Select Create role.

☐ Select AWS account as the trusted entity type.

☐ Select Another AWS account



☐ In the Account ID field, enter your own AWS account ID temporarily. Later, you modify the trust relationship and grant access to Snowflake.

- [ ] Select the Require external ID option. An external ID is used to grant access to your AWS resources (such as S3 buckets) to a third party like Snowflake.
- [ ] Enter a placeholder ID such as 0000. In a later step, you will modify the trust relationship for your IAM role and specify the external ID for your storage integration.
- [ ] Select Next.
- [ ] Select the policy you created in Step 1: Configure access permissions for the S3 bucket (in this topic).
- [ ] Select Next.
- [ ] Enter a name and description for the role, then select Create role.
- [ ] You have now created an IAM policy for a bucket, created an IAM role, and attached the policy to the role.
- [ ] On the role summary page, locate and record the Role ARN value. In the next step, you will create a Snowflake integration that references this role.

Step 3: Create a cloud storage integration in Snowflake

Create a storage integration using the CREATE STORAGE INTEGRATION command:

```
CREATE OR REPLACE STORAGE INTEGRATION <give integration a name>
  TYPE = EXTERNAL_STAGE
  STORAGE_PROVIDER = 'S3'
  ENABLED = TRUE
  STORAGE_AWS_ROLE_ARN = '<iam role arn>''
  STORAGE_ALLOWED_LOCATIONS = ('*');
```

Step 4: Retrieve the AWS IAM user for your Snowflake account

- [ ] To retrieve the ARN for the IAM user that was created automatically for your Snowflake account, use the DESCRIBE INTEGRATION: "*DESC INTEGRATION <integration_name>*";

- [ ] Record the values for the following properties:

"STORAGE_AWS_IAM_USER_ARN" :

The AWS IAM user created for your Snowflake account; for example, arn:aws:iam::123456789001:user/abc1-b-self1234. Snowflake provisions a single IAM user for your entire Snowflake account. All S3 storage integrations in your account use that IAM user.

"STORAGE_AWS_EXTERNAL_ID" :

The external ID that Snowflake uses to establish a trust relationship with AWS. If you didn't specify an external ID (STORAGE_AWS_EXTERNAL_ID) when you created the storage integration, Snowflake generates an ID for you to use.

Step 5: Grant the IAM user permissions to access bucket objects

- [ ] Select the role you created in Step 2: Create the IAM role in AWS (in this topic).

- [ ] Select the Trust relationships tab.

- [ ] Select Edit trust policy.

- [ ] Modify the policy document with the DESC STORAGE INTEGRATION output values you recorded in Step 4: Retrieve the AWS IAM user for your Snowflake account (in this topic):

- [ ] Policy document for IAM role

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "<snowflake_user_arn>"
```

```
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "sts:ExternalId": "<snowflake_external_id>"
      }
    }
  }
 ]
}
```

- ☐ snowflake_user_arn is the STORAGE_AWS_IAM_USER_ARN value you recorded.
- ☐ snowflake_external_id is the STORAGE_AWS_EXTERNAL_ID value you recorded.

Step 6: Create an external stage

```
CREATE OR REPLACE STAGE s3StageForSOD
  STORAGE_INTEGRATION = s3IntegrationForSOD
  URL = 's3://niinebucket/folder_for_sod/'
  FILE_FORMAT = (TYPE = 'CSV' FIELD_DELIMITER = '\t' SKIP_HEADER = 1);
```

Replace ‘URL’ with desired s3 folder, ‘FILE_FORMAT’ can also be changed according to CSV type but the file should be of CSV only as Snowflake allows only CSV external tables.

- ☐ Example file format:

```
create or replace file format CsvFormatForSodCSV
    type = 'csv'
    compression = 'auto'
    field_delimiter = ','
    record_delimiter = '\n'
    skip_header = 1
    field_optionally_enclosed_by = '\042'
    null_if = ('\\N');
```

Step 7: Create an external table

Below is an example format for external table that casts all columns as VARCHAR for avoiding value datatype errors:

```
CREATE OR REPLACE EXTERNAL TABLE S3_DATA_NIINE.PUBLIC.NewSodExtTable (
Order_No VARCHAR(16777216) AS (value:c1::VARCHAR),
  ESM_Erp_Id VARCHAR(16777216) AS (value:c2::VARCHAR),
  User_Designation VARCHAR(16777216) AS (value:c3::VARCHAR),
  User_Position_Level VARCHAR(16777216) AS (value:c4::VARCHAR),
  User VARCHAR(16777216) AS (value:c5::VARCHAR),
  SuperStockist VARCHAR(16777216) AS (value:c6::VARCHAR),
  SuperStockist_ErpId VARCHAR(16777216) AS (value:c7::VARCHAR),
  Distributor_Erp_Id VARCHAR(16777216) AS (value:c8::VARCHAR),
  Beat_Erp_Id VARCHAR(16777216) AS (value:c9::VARCHAR),
  Beat VARCHAR(16777216) AS (value:c10::VARCHAR),
  Outlet_Erp_Id VARCHAR(16777216) AS (value:c11::VARCHAR),
  Outlet_Name VARCHAR(16777216) AS (value:c12::VARCHAR),
  Order_Date VARCHAR(16777216) AS (value:c13::VARCHAR),
  Price VARCHAR(16777216) AS (value:c14::VARCHAR),
  Sale_Value VARCHAR(16777216) AS (value:c15::VARCHAR),
  Call_Start_Time VARCHAR(16777216) AS (value:c16::VARCHAR),
  Call_End_Time VARCHAR(16777216) AS (value:c17::VARCHAR),
  Product_ErpId VARCHAR(16777216) AS (value:c18::VARCHAR),
  Product VARCHAR(16777216) AS (value:c19::VARCHAR),
  Qty_StdUnit VARCHAR(16777216) AS (value:c20::VARCHAR),
  Gross_Value VARCHAR(16777216) AS (value:c21::VARCHAR)
)
WITH LOCATION = @s3StageForSOD
FILE_FORMAT = 'CsvFormatForSodCSV'
PATTERN = '.*Secondary_Order_Dump_New_india.*\\.csv$'
AUTO_REFRESH = TRUE;
```

NOTE: Column names should not have spaces in them, if they are present then use underscores (‘_’) or enclose the column name in double quotes, for eg: “*Gross Value*” or “*Gross_Value*”

– By Agastya Singh