Experiment: Data Engineering Pipeline

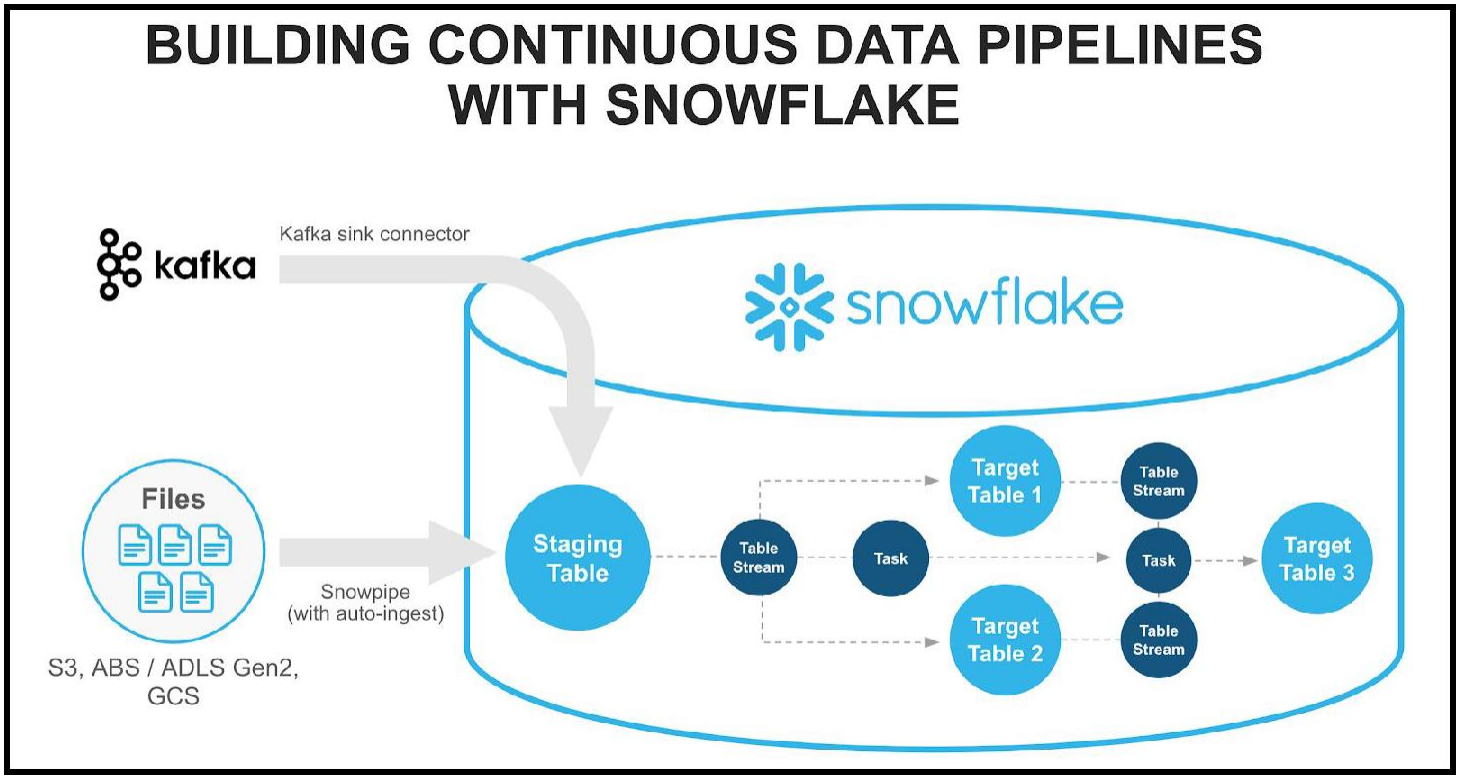
Overview

This experiment demonstrates Snowflake features specifically aligned to “Data Engineering” workload to build modern data pipelines. Data pipelines automate many of the manual steps involved in transforming and optimizing continuous data loads. Frequently, the “raw” data is firstloaded temporarily into a staging table used for interim storage and then transformed using a series of SQL statements before it is inserted into the destination reporting tables. The most efficient workflow for this process involves transforming only data that is new or modified.

Snowflake provides the following features to enable continuous data pipelines:

- Continuous data loading using Snowpipe, Snowflake Connector for Kafka, or Third-party data integration tools.
- Change data tracking using Streams.
- Recurring tasks.

The below picture illustrates how to build continuous data pipelines with Snowflake.



Snowflake Connector for Kafka makes it fast and easy to reliably publish continuous streams of records from Kafka to your Snowflake instance for storage and analysis. The Streams and Tasks features enable you to build data pipelines and turn Snowflake into a nimble data transformation engine in addition to a powerful data warehouse.

What you'll learn

This hands-on lab will demonstrate using the following features of the platform:

1. Create a Stage and configure it to automatically call **Snowpipe** using S3 event notifications. NOTE: This will require the user to create an external AWS S3 bucket in order to demonstrate.
2. **Streams** - Abstraction over staging table used by Snowpipe to allow one time processing of ingested data.
3. **Tasks** (in conjunction with Stored Procedures) - Regularly scheduled execution of transformation logic over data in the stream.

Environment

- We will use our Snowflake free 30-day trial environment.
- AWS Account provided by our session leader.

Experiment 21: Prepare Environment

21.1 Create AWS Account

21.1.1 If you were to do this lab in the future you would have to create or have access to an AWS account. Please create a new account using this link -Create AWS Account. Once logged in to your account select an AWS region closest to your Snowflake account, US-WEST-2 (Oregon), US-EAST-2 (Ohio) and US-EAST-1(N.Virginia) are good choices.

21.2 Download Snowflake SQL files to your local machine

21.2.1 SQL file **ModernDataPipelines_ExperimentDataSetup.sql** - to create database, warehouse, and data setup for the labs.

21.2.2 SQL file **ModernDataPipelines_ExperimentStoredProcedures.sql** - to create stored procedures for
(1) mimicking periodically arriving data and (2) for cleanup of files that are successfully loaded.

21.2.3 SQL file **ModernDataPipelines_Experiment.sql** - to configure/create Snowpipe, Streams, Tasks, and execute the end-to-end data pipeline.

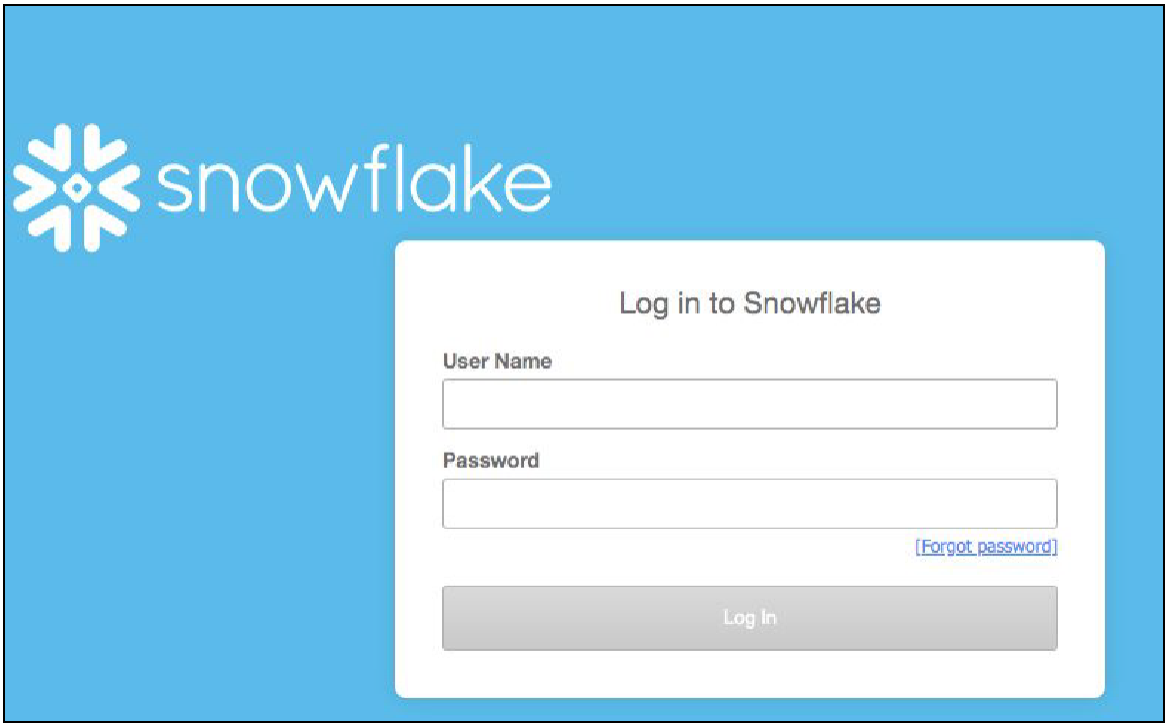
21.2.4 SQL file **ModernDataPipelines_ExperimentCleanup.sql** - to remove all the Snowflake data/objects.

21.3 Data Setup in Snowflake for the labs

About the screen captures, sample code, and environment
Screen captures in this experiment depict examples and results that may slightlyvary from what you may see when you complete the

21.3.1 Open a browser window and enter the URL of your Snowflake 30-day trial environment.

21.3.2 You should see the login screen below. Enter your unique credentials to log in.



The top menu allows you to switch between the different areas of Snowflake:

21.3.3 The **Warehouses** tab is where you set up and manage compute resources (virtual warehouses) to load or query data in Snowflake. Note a warehouse called "COMPUTE_WH (XL)" already exists in your environment.

21.3.4 If you click on the top right of the UI where your username appears, you will see that here you can do things like change your password, roles, or preferences. Snowflake has several system defined roles. You are currently in the default role of SYSADMIN and we will Switch the role to ACCOUNTADMIN for these labs.

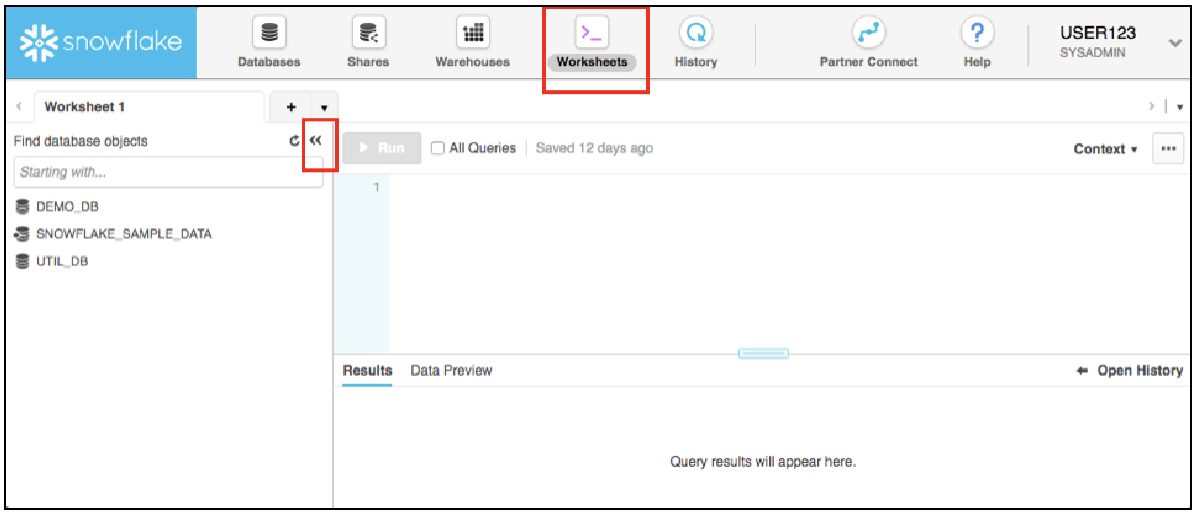
ACCOUNTADMIN
For most in these experiment you will remain in the ACCOUNTADMIN role which has privileges to create warehouses and databases and other objects in an account.

In a real-world environment, you would use different roles for the tasks in this experiment, and assign the roles to your users. More on access control in Snowflake isin towards the end of this experiment and also at https://docs.snowflake.net/manuals/user-guide/security-access-control.html

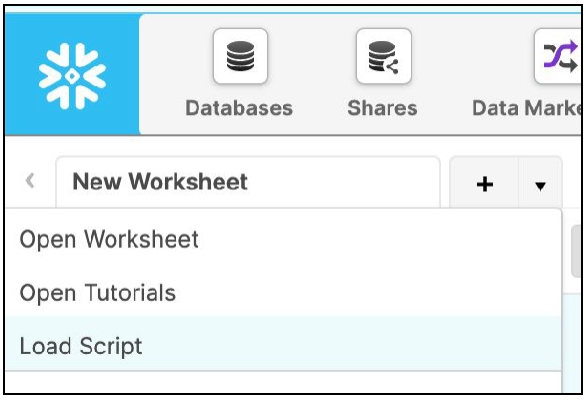
21.3.5 The **Worksheets** tab provides an interface for submitting SQL queries, performing DDL and DML operations and viewing results as your queries/operations complete. The default "Worksheet 1" appears.

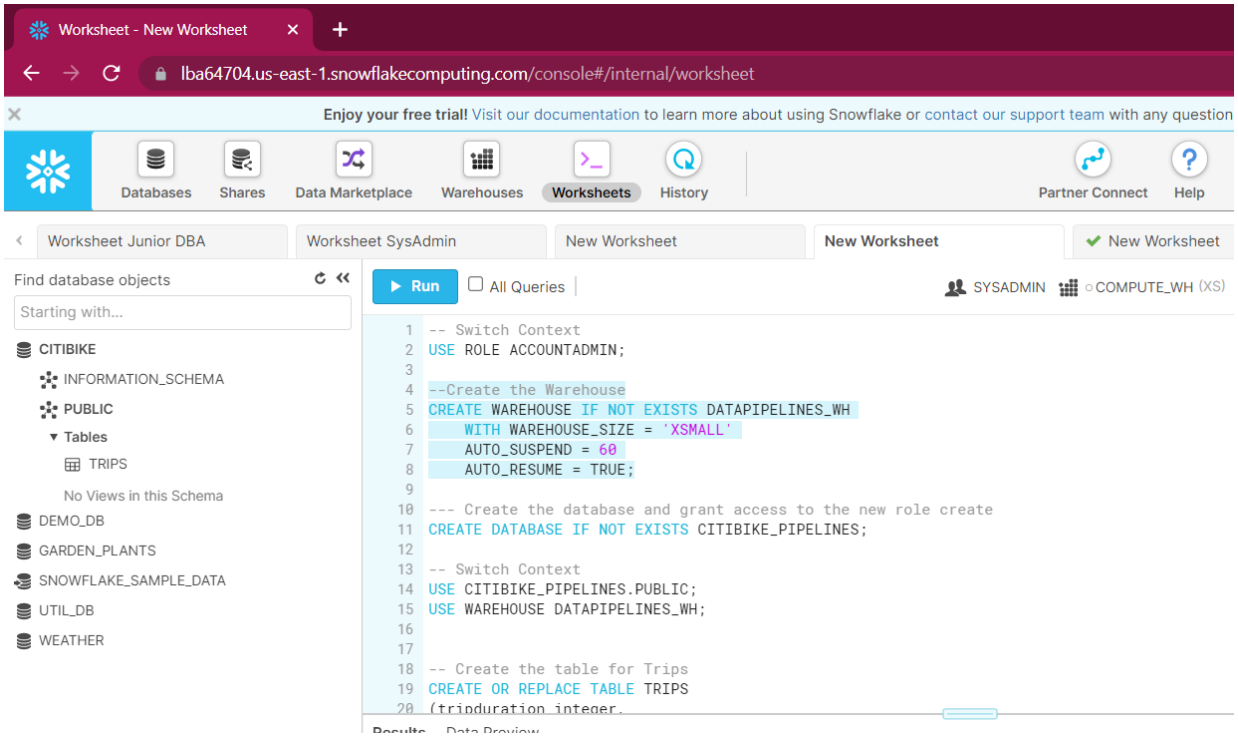
In the left pane is the database objects browser which enables users to explore all databases, schemas, tables, and views accessible by the role selected for a worksheet. The bottom pane shows results of queries and operations.

The various windows on this page can be resized by moving the small sliders on them. And if during the lab you need more room to work in the worksheet, collapse the database objects browser in the left pane. Many of the screenshots in this guide will have this database objects browser closed.



21.3.6 At the top left of the default “Worksheet 1,” just to the right of the worksheet tab, click on the small, downward facing arrow, select “Load Script”, then browse to the “**ModernDataPipelines_ExperimentDataSetup.sql**” file you downloaded in the prior section and select “Open”.





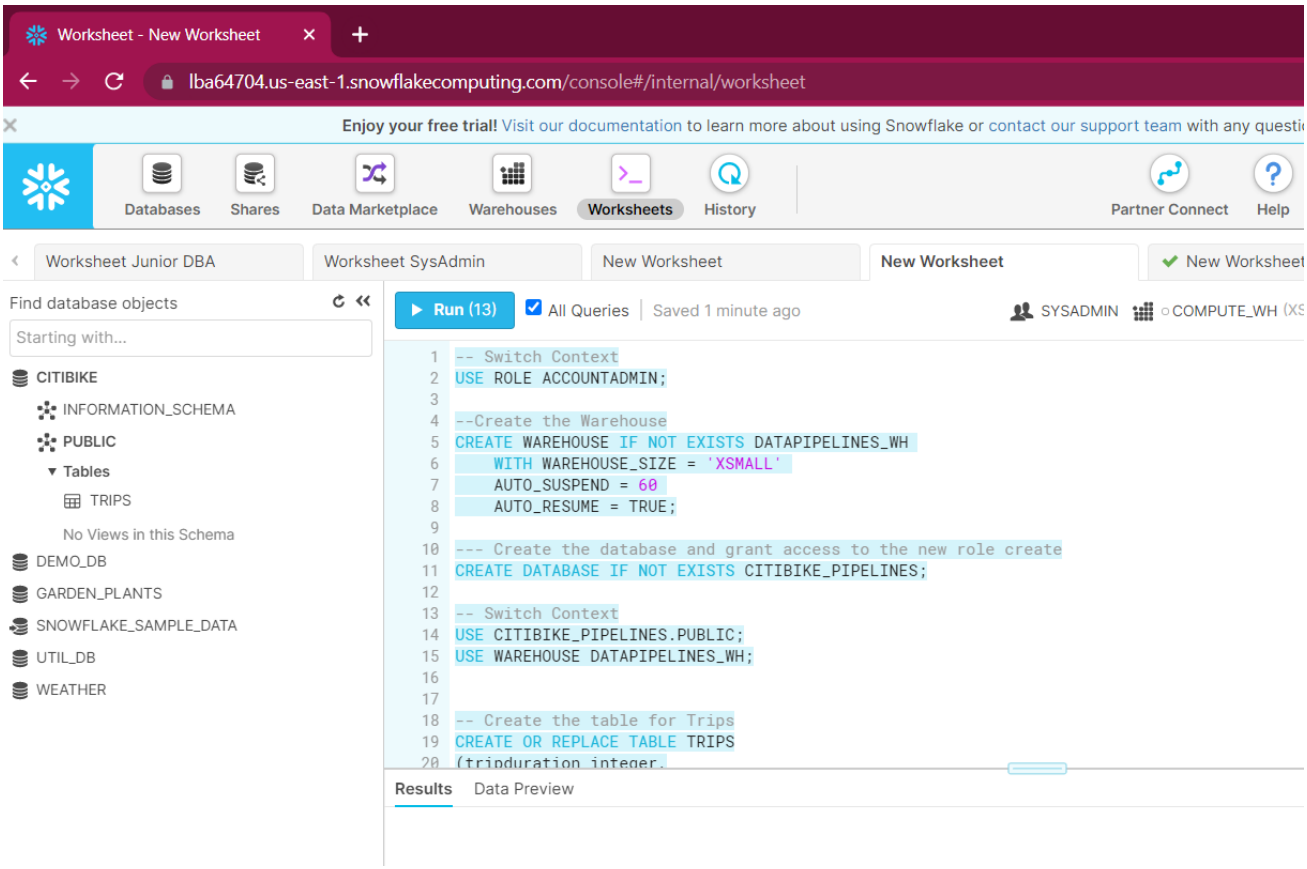
Warning - Do Not Copy/Paste SQL From This PDF to a Worksheet
Copy-pasting the SQL code from this PDF into a Snowflake worksheet will result in formatting errors and the SQL will not run correctly. Make sure to use the “Load Script” method just covered above.

On older or locked-down browsers, this “load script” step may not work as the browser will prevent you from opening the .sql file. If this is the case, open the .sql file with a text editor and then copy/paste all the text from the .sql file to the “Worksheets”

Worksheets vs the UI
Much of the configurations in this lab will be executed via this pre-written SQL in the Worksheets in order to save time. These configurations could also be done via the UI in a less technical manner but would take more time.

21.3.7 All of the SQL commands for the lab data setup will now appear on the new worksheet. ***Click the checkbox before "All Queries"*** inside the worksheet and execute by clicking on Run.

21.3.8 Ensure the database, tables and data is loaded by reviewing the results.



21.3.9 At this point the data setup for the labs is complete.

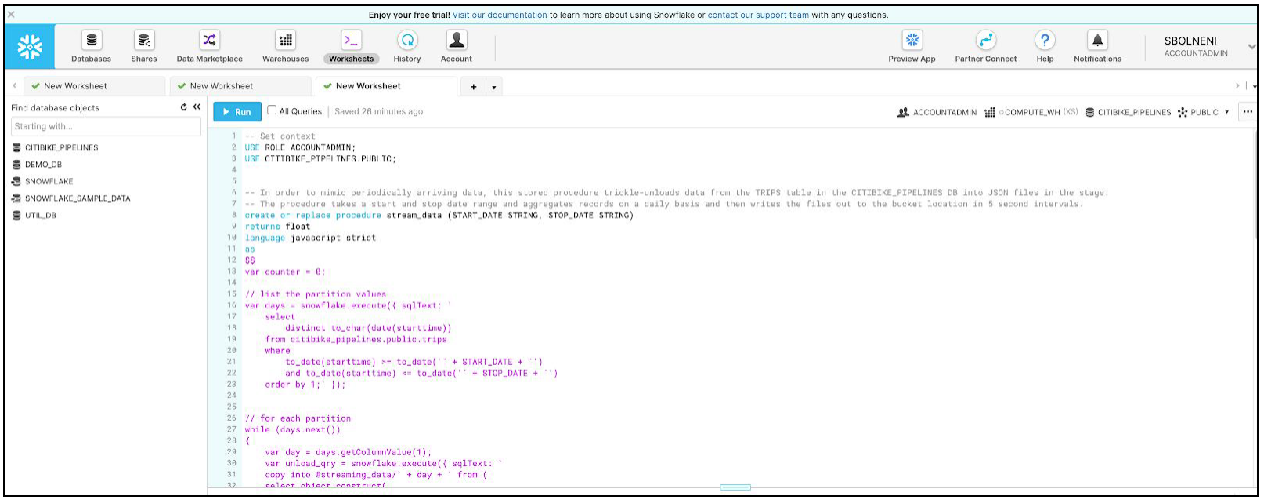
21.4 Stored Procedures for Data Generator & Cleaning Up Stage Files after Successful Load

21.4.1 In this section we will create two new stored procedures for data generation and cleaning up of stage files after successful load.

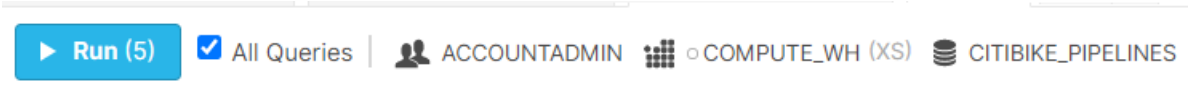
21.4.2 In order to mimic periodically arriving data, stored procedure ‘STREAM_DATA’ trickle-unloads data from the TRIPS table in the CITIBIKE_PIPELINES database into JSON files in the external stage which we will create in the next step. The procedure takes a start and stop date range and aggregates records on a daily basis and then writes the files out to the bucket location in 5 second intervals. When it is run, it will produce a steady flow of data arriving in your defined STAGE object.

21.4.3 In this lab, data being written to the External STAGE by the STREAM_DATA stored procedure will be automatically loaded into a table via Snowpipe. The ‘PURGE_FILES’ stored procedure will be used to clean up files that are successfully loaded. It will compare records entries in the information_schema.copy_history view with the file names still present in the stage. Files that were found to have been loaded are safely deleted by the procedure.

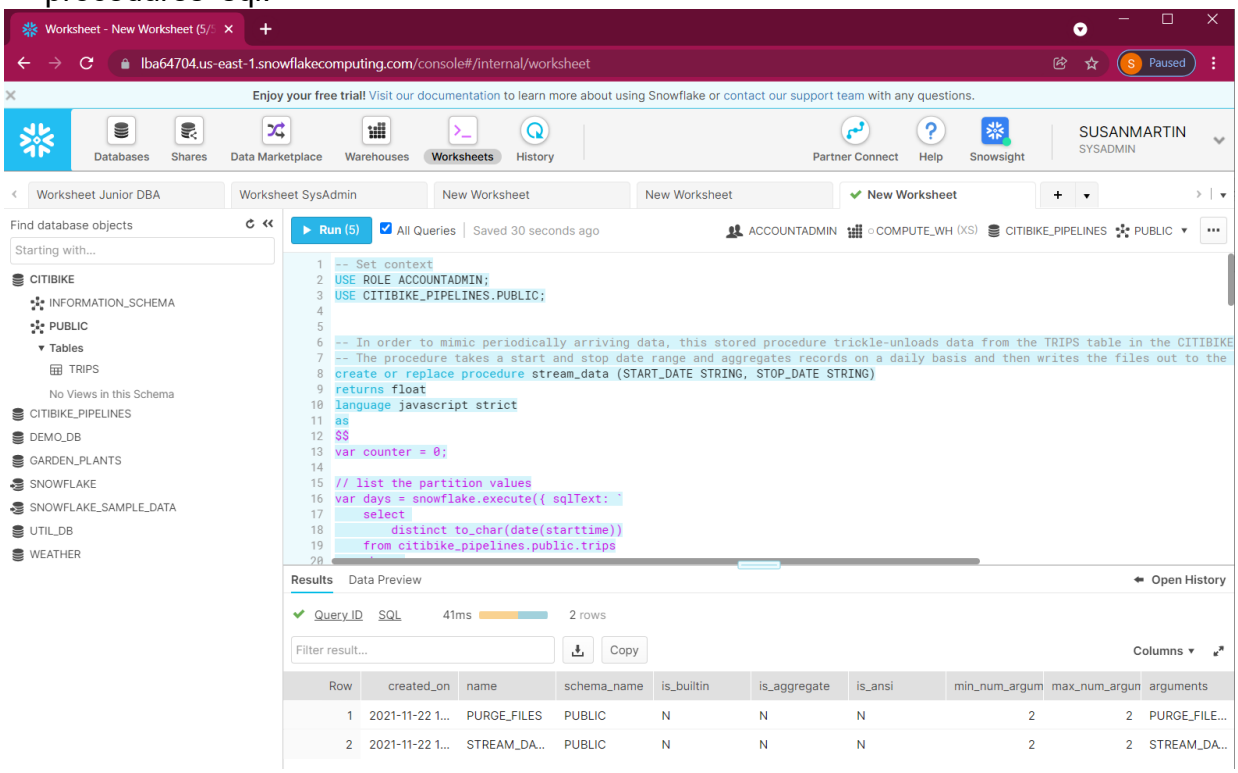
21.4.4 Similar to step 21.4.10 in the worksheets tab, click on the small, downward facing arrow,select “Load Script”, then browse to the “**ModernDataPipelines_ExperimentStoredProcedures.sql**” file you downloaded in section 21.3 and select “Open”.



21.4.5 All of the SQL commands for setting the context and creating the stored procedures - ‘STREAM_DATA’, ‘PURGE_FILES’ will now appear on the new worksheet. ***Click the checkbox before “All Queries”*** inside the worksheet and execute by clicking on Run.



21.4.6 Ensure the stored procedures are created successfully by reviewing the results of ‘show procedures’ sql.



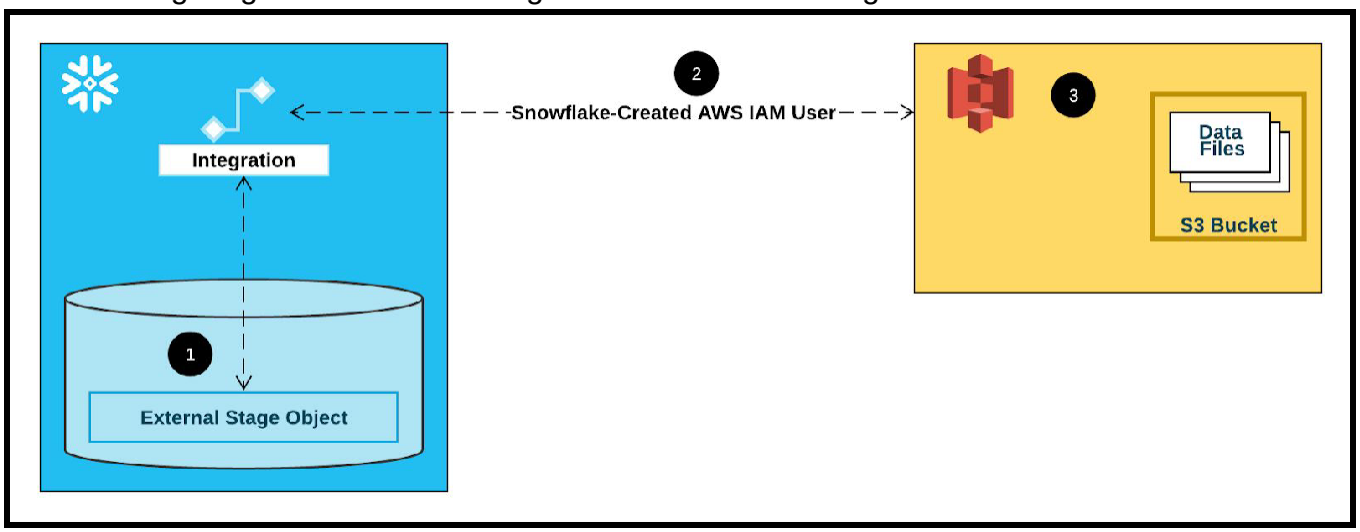
21.4.7 At this point the creation of stored procedures is complete.

21.5 Configuring Secure Access to Amazon S3

This section describes how to use storage integrations to allow Snowflake to read data from and write data to an Amazon S3 bucket referenced in an external (i.e. S3) stage. Integrations are named, first-class Snowflake objects that avoid the need for passing explicit cloud provider credentials such as secret keys or access tokens. Integration objects store an AWS identity and access management (IAM) user ID.

Note:
Completing the instructions in this section requires permissions in AWS to create and manage IAM policies and roles. If you are not an AWS administrator, ask your AWS administrator to perform these tasks.

The following diagram shows the integration flow for a S3 stage:

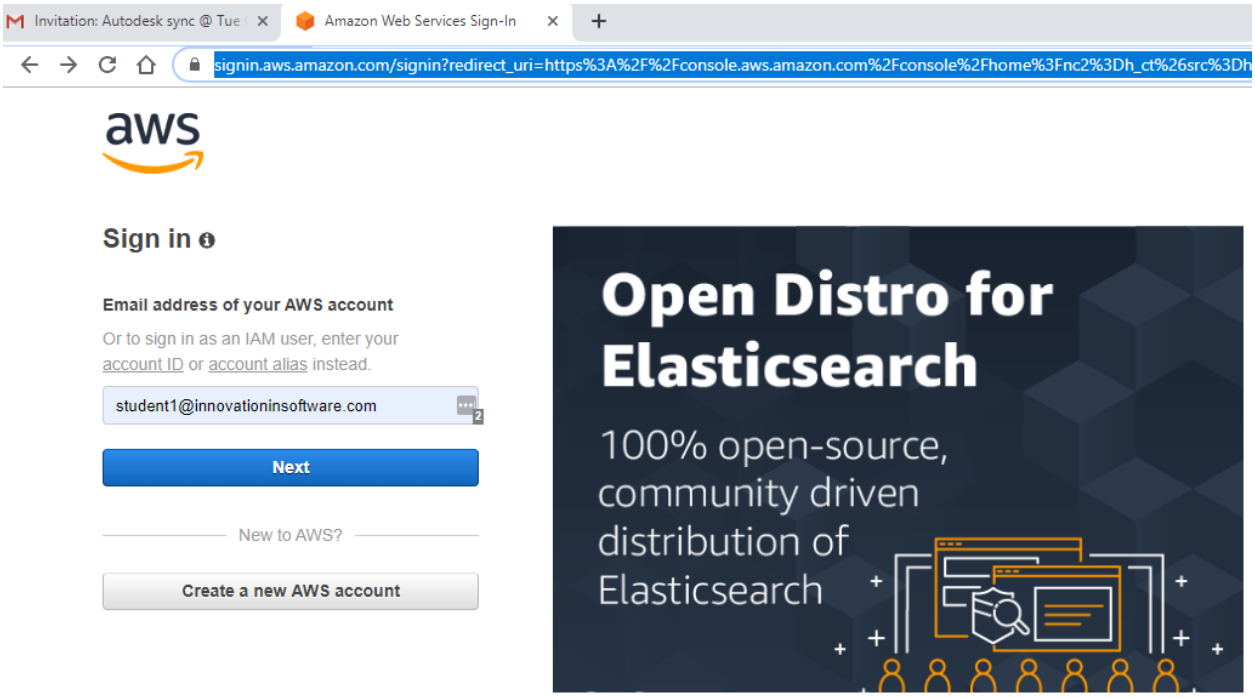


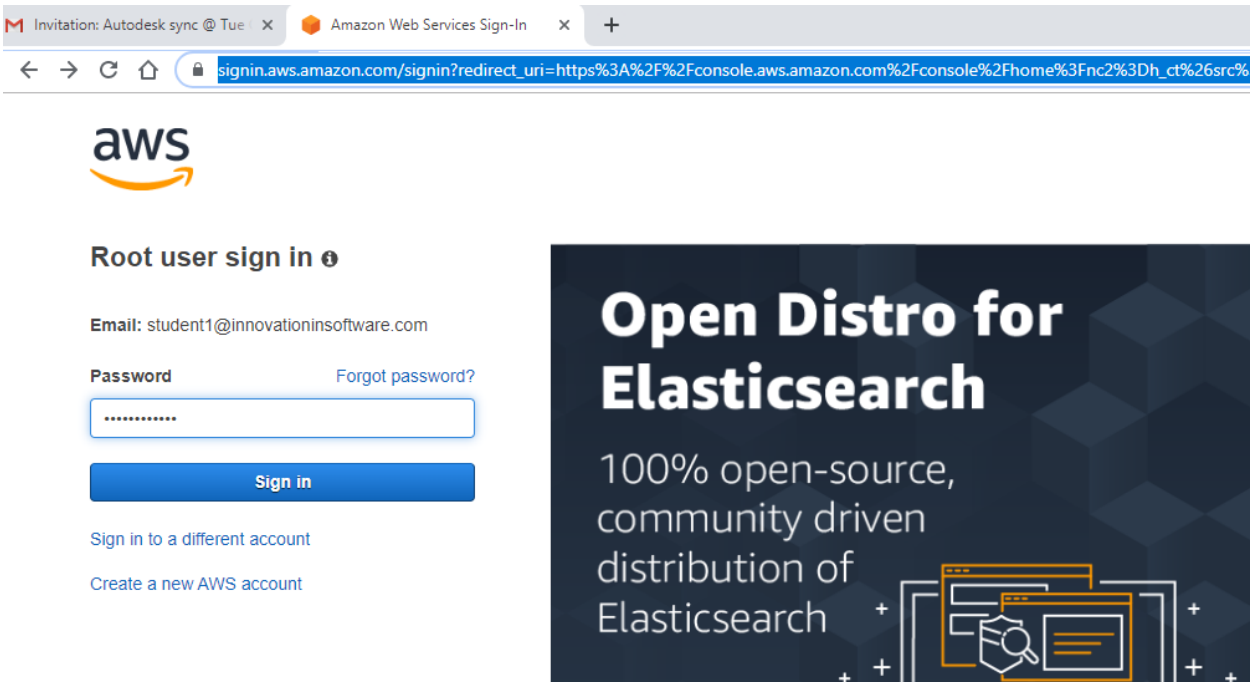
- An external (i.e. S3) stage references a storage integration object in its definition.
- Snowflake automatically associates the storage integration with a S3 IAM user created for your account. Snowflake creates a single IAM user that is referenced by all S3 storage integrations in your Snowflake account.
- An AWS administrator in your organization grants permissions to the IAM user to access the bucket referenced in the stage definition. Note that many external stage objects can reference different buckets and paths and use the same storage integration for authentication.

High level steps involved in this Section:

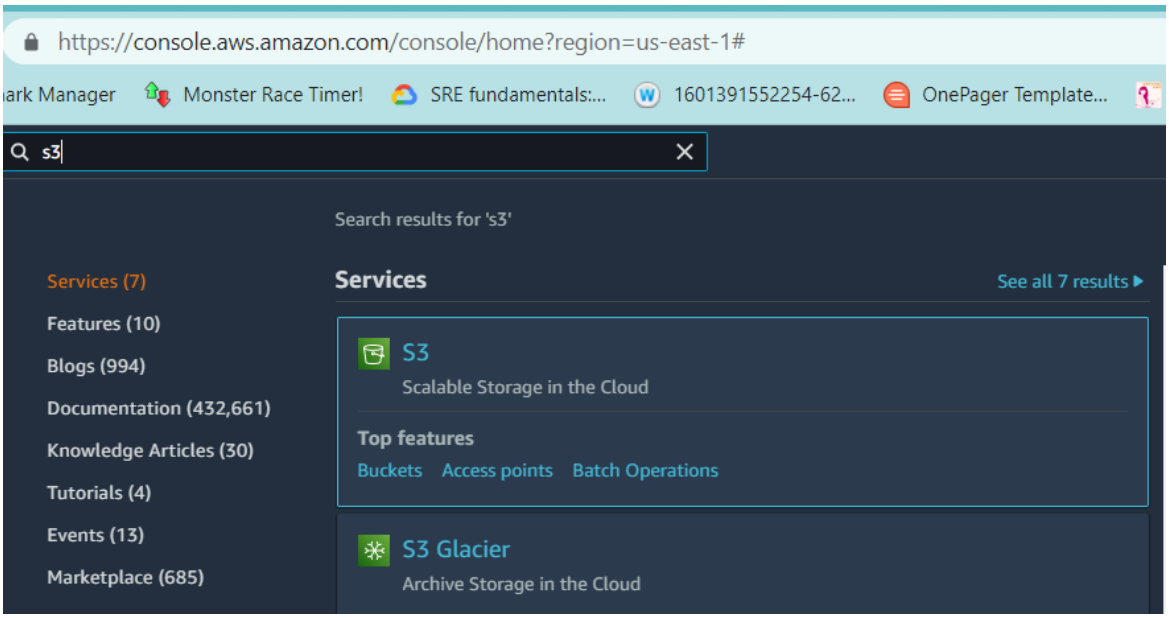
- Step 1: Configure Access Permissions for the S3 Bucket
  - AWS Access Control Requirements
  - Creating an IAM Policy
- Step 2: Create the IAM Role in AWS
- Step 3: Create a Cloud Storage Integration in Snowflake
- Step 4: Retrieve the AWS IAM User for your Snowflake Account
- Step 5: Grant the IAM User Permissions to Access Bucket Objects

21.5.1 Log into the AWS Management Console
https://signin.aws.amazon.com/signin?redirect_uri=https%3A%2F%2Fconsole.aws.amazon.com%2Fconsole%2Fhome%3Fnc2%3Dh_ct%26src%3Dheader-signin%26state%3DhashArgs%2523%26isauthcode%3Dtrue&client_id=arn%3Aaws%3Aiam%3A%3A015428540659%3Auser%2Fhomepage&forceMobileApp=0

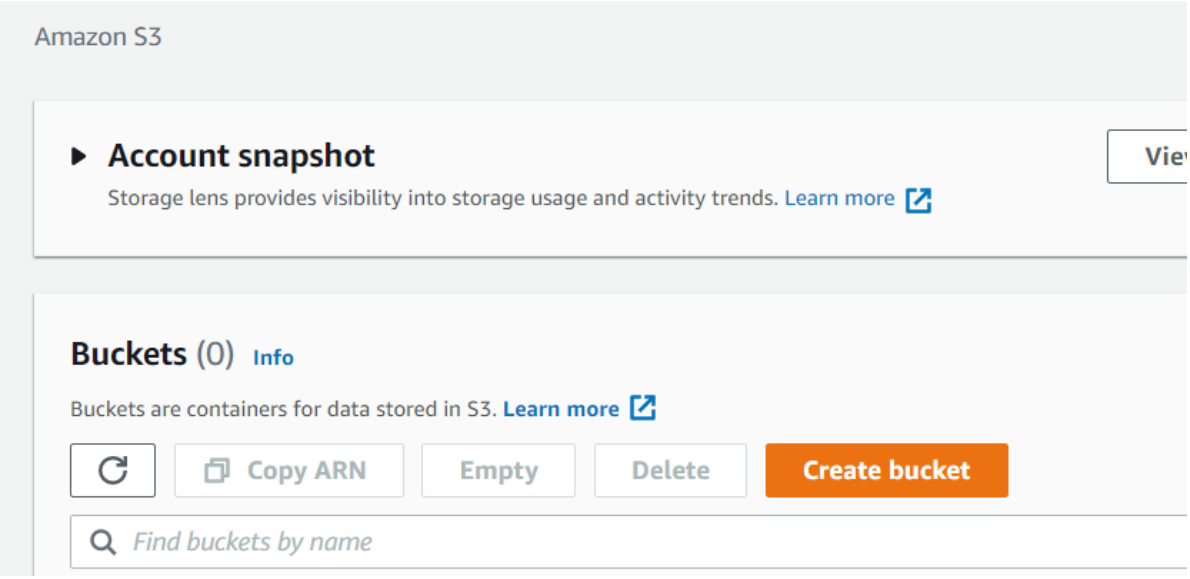




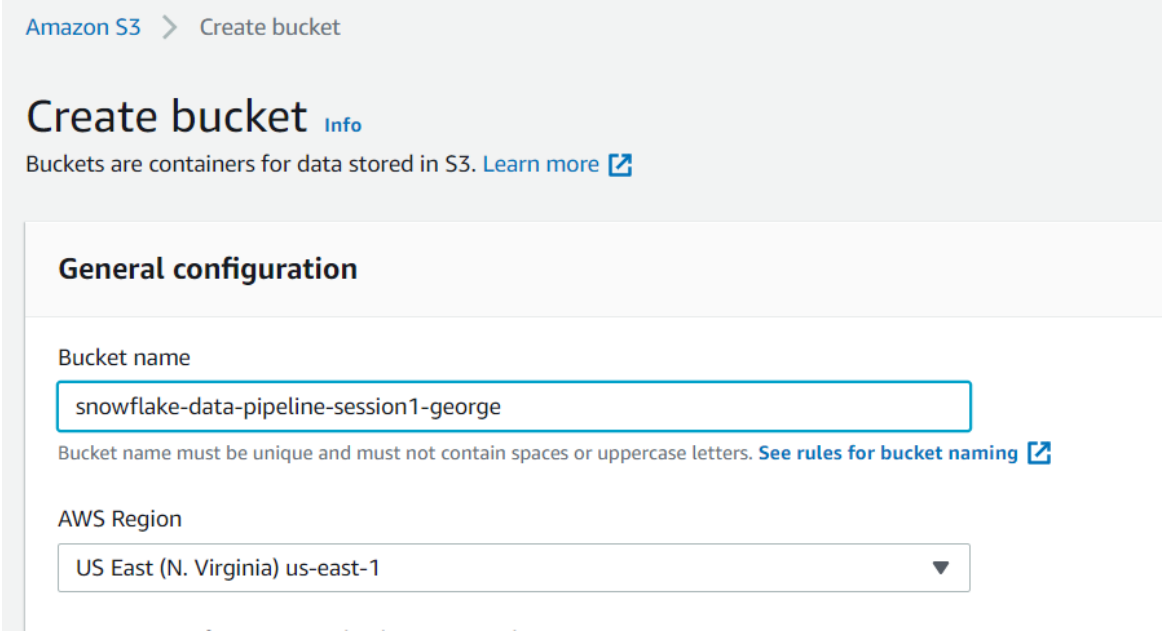
21.5.2 Search for the S3 service and select S3



21.5.3 Create an AWS S3 bucket and a folder to be used as an external stage for Snowflake. Choose **Create bucket**.

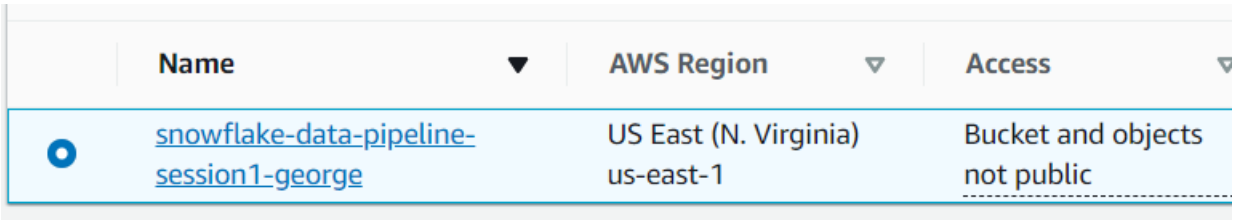


21.5.4 Make a note of the S3 bucket/folder which will be used in the below steps. Bucket uniqueness is required globally, so we'll use a simple naming like **snowflake-data-pipeline-session1-george** or similar

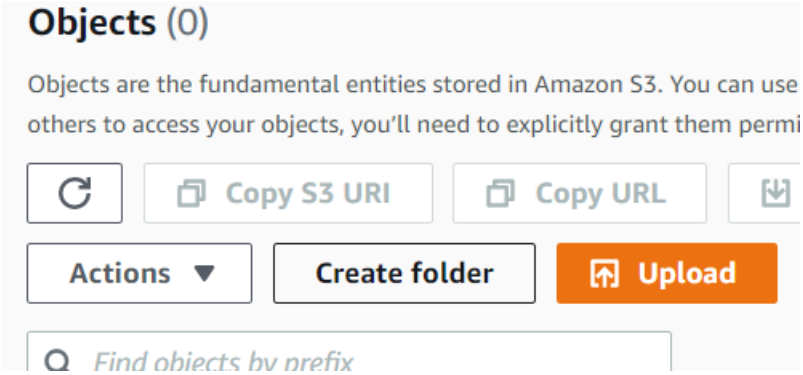


21.5.5 Scroll to the bottom, leaving the remaining options default, and choose **Create bucket**

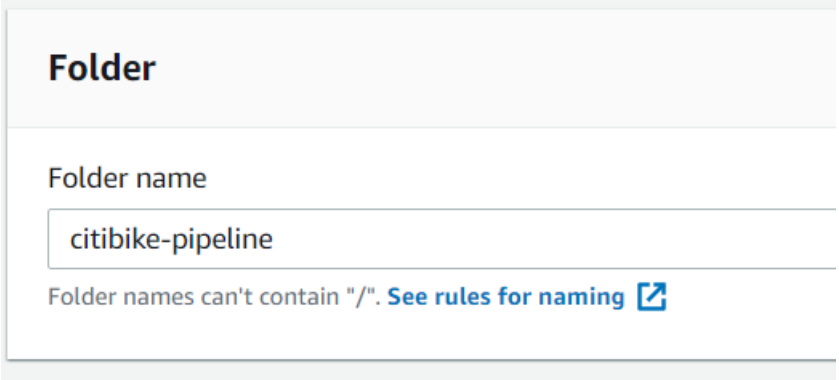
21.5.6 Select the bucket name that you created



21.5.7 Choose Create folder



21.5.8 Enter the bucket name as **citibike-pipeline**



21.5.9 Choose **Create folder**

21.5.10 Configure Access Permissions for the S3 Bucket. From the service search enter **IAM**

21.5.11 Choose **Account settings** from the left-hand navigation pane.

21.5.12 Expand the **Security Token Service Regions** list, find the AWS region corresponding to the region where your account is located, and choose **Activate** if the status is Inactive. If we chose the US-EAST-1 region it should already be "Always Active"

Security Token Service (STS)

Session Tokens from the STS endpoints

AWS recommends using regional STS endpoints to reduce latency. Session tokens from regional STS endpoints are va

Session tokens from the global STS endpoint (https://sts.amazonaws.com) are valid only in AWS Regions that are enal use session tokens from regional STS endpoints or activate the global STS endpoint to issue session tokens that are v

Endpoints	Region compatibility of session tokens	Act
Global endpoint	Valid only in AWS Regions enabled by default	Edit
Regional endpoints	Valid in all AWS Regions	

Endpoints

You can enable additional endpoints from which you can request temporary credentials. Activate only endpoints you int

Region name	Endpoint	STS status
Global Endpoint	https://sts.amazonaws.com	Always active
US East (N. Virginia)	https://sts.us-east-1.amazonaws.com	Always active

21.5.13 Choose **Policies** from the left-hand navigation pane.

21.5.14 Choose **Create Policy**

21.5.15 Select the JSON tab

21.5.16 Copy and paste the contents of **s3-iam-role.json** into the policy content area. Choose Next:Tags

Visual editor | JSON

```
            "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>/*"
                    ]
                }
            }
        }
    ]
}
```

Security: 0 Errors: 0 Warnings: 0 Suggestions: 0

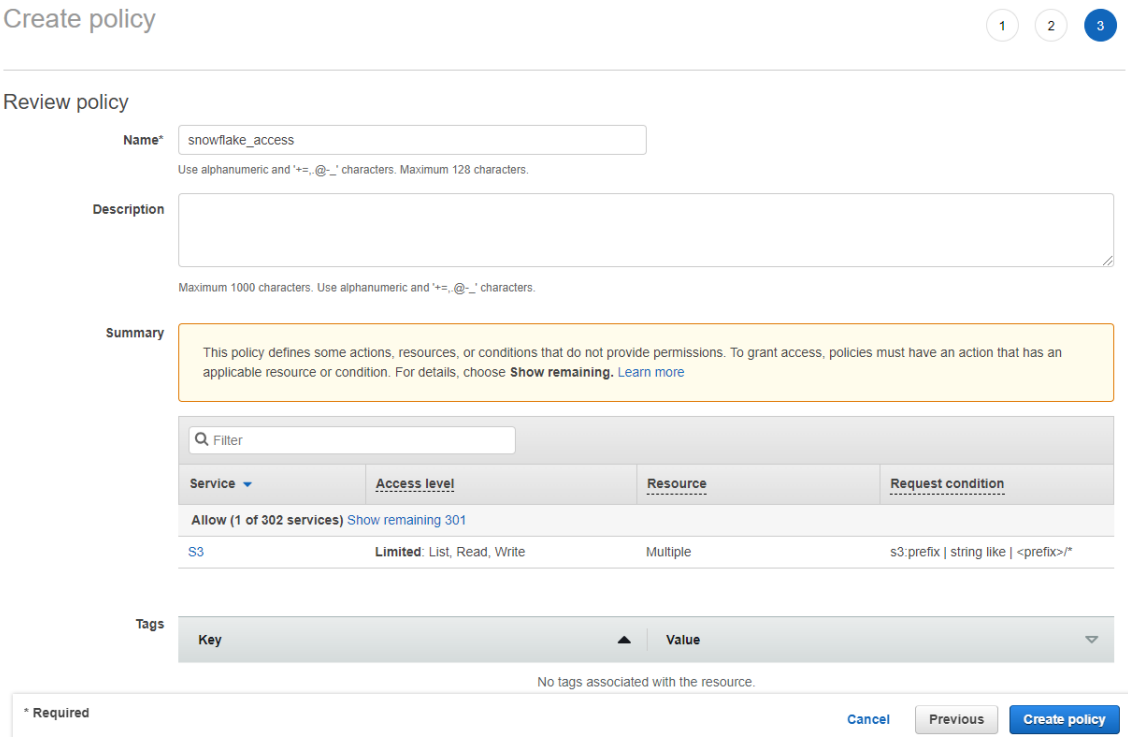
Make sure to replace bucket and prefix with your actual bucket name and folder path prefix. For the example bucket = snowflake-data-pipeline-session1-george and the prefix = citibike-pipeline

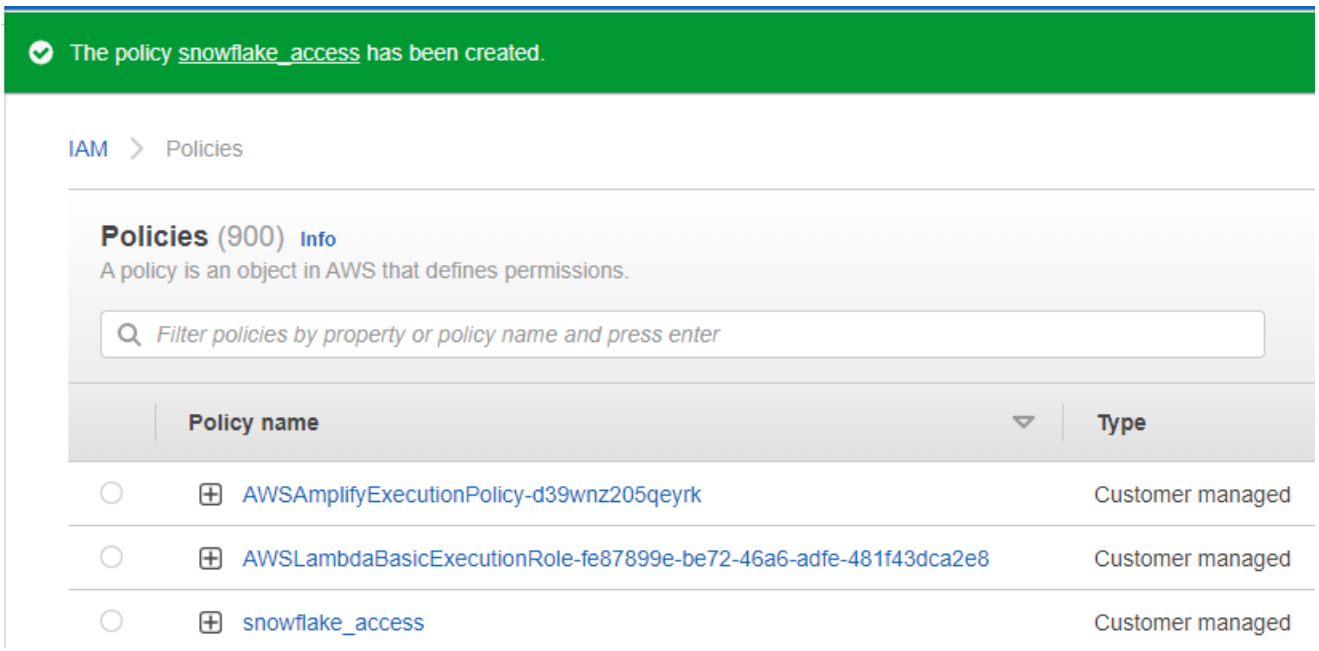
Visual editor | JSON

```
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:DeleteObject",
                "s3:DeleteObjectVersion"
            ],
            "Resource": "arn:aws:s3:::snowflake-data-pipeline-session1-george/citibike-pipeline/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::snowflake-data-pipeline-session1-george",
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "citibike-pipeline/*"
                    ]
                }
            }
        }
    ]
}
```

21.5.17 Choose **Next: Review**

21.5.18 Enter **snowflake_access** as the Policy Name and choose **Create policy**





Make sure to replace `bucket` and `prefix` with your actual bucket name and folder path prefix. For the example bucket = `snowflake-data-pipeline-session1-george` and the prefix = `citibike-pipeline`

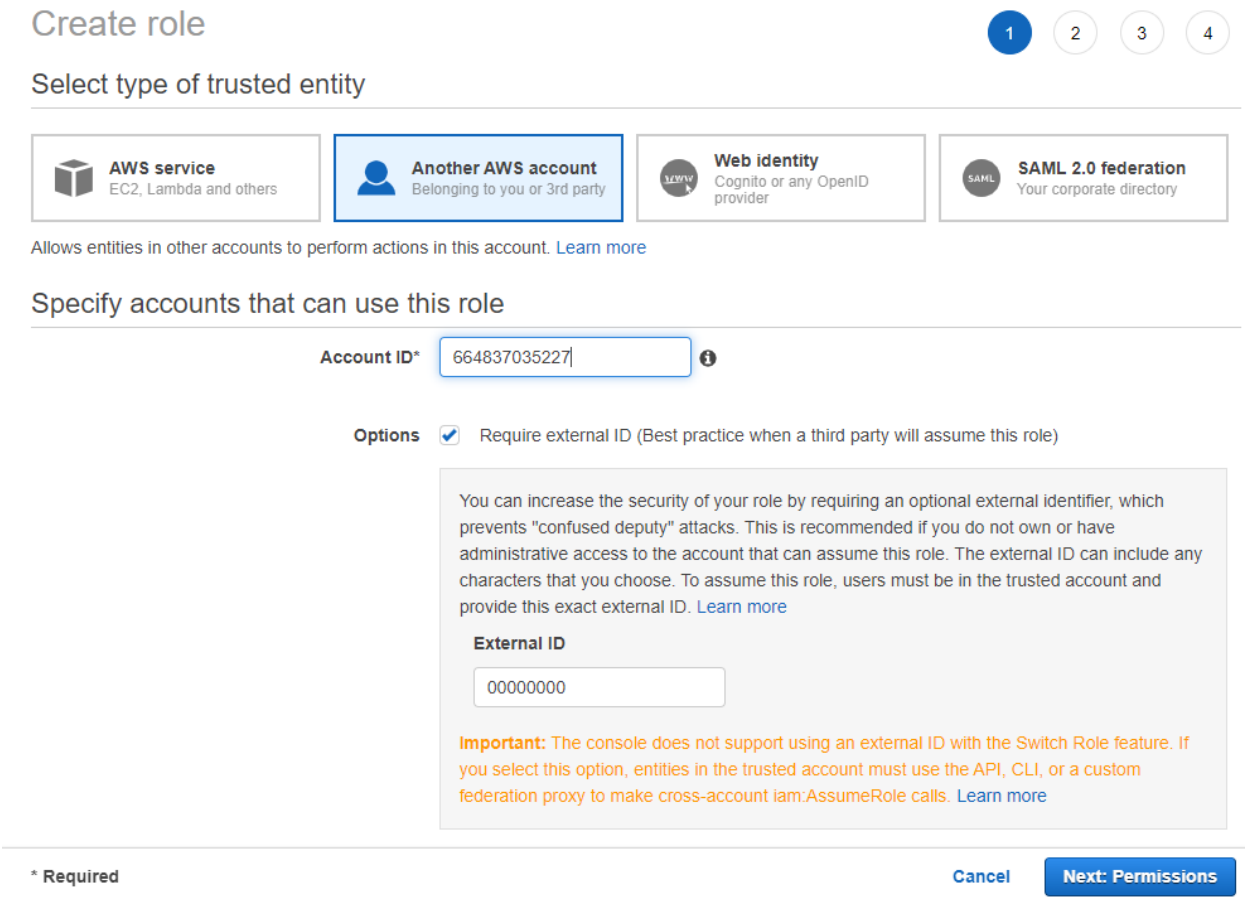
21.5.19 Choose Roles from the left-hand navigation pane.

21.5.20 Click the **Create role** button.

21.5.21 Select **Trusted Entity** Page in AWS Management Console

21.5.22 Select **Another AWS account** as the trusted entity type.

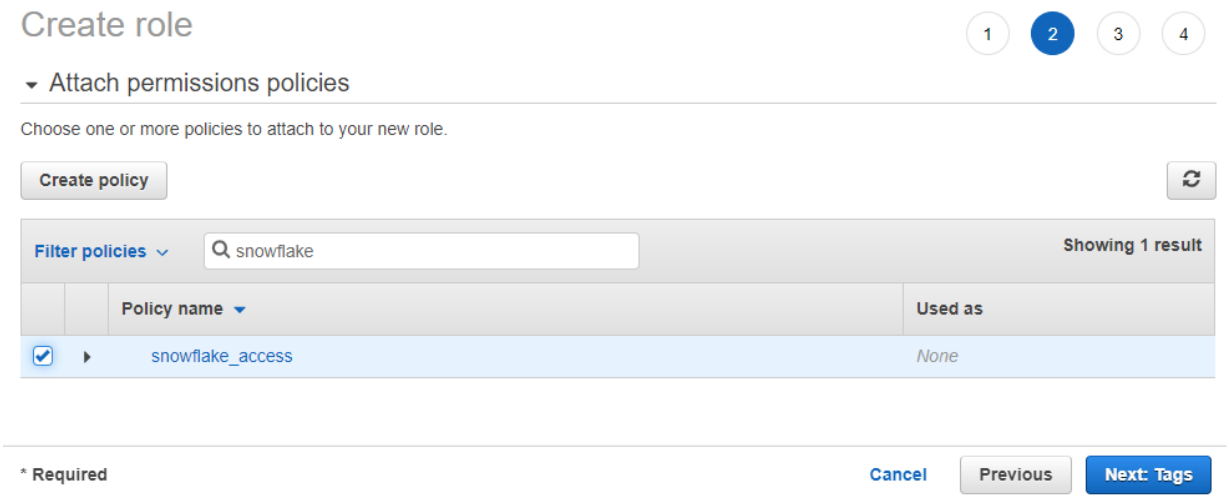
In the Account ID field, enter your own AWS account ID temporarily. Later, you will modify the trusted relationship and grant access to Snowflake.



21.5.23 Enter a dummy ID such as **0000**. Later, you will modify the trusted relationship and specify the external ID for your Snowflake stage. An external ID is required to grant access to your AWS resources (i.e. S3) to a third party (i.e. Snowflake).

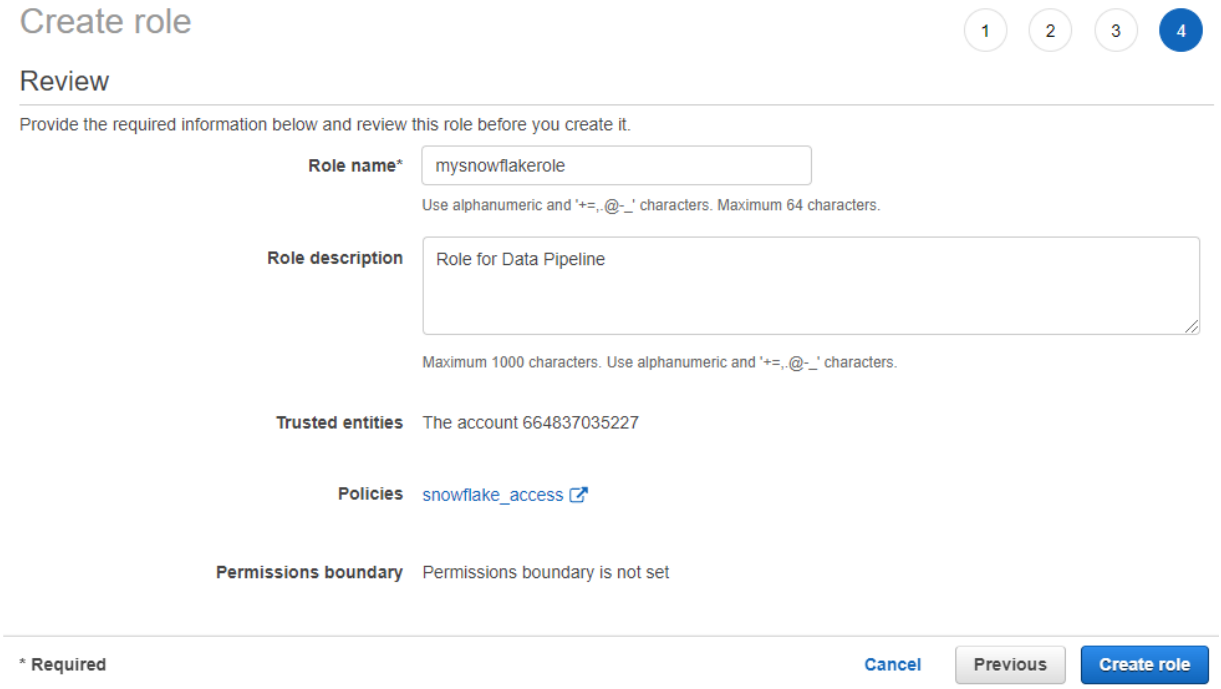
21.5.24 Click the **Next** button.

21.5.25 Locate the policy you created in Step 1: Configure Access Permissions for the S3 Bucket (in this topic), and select this policy.



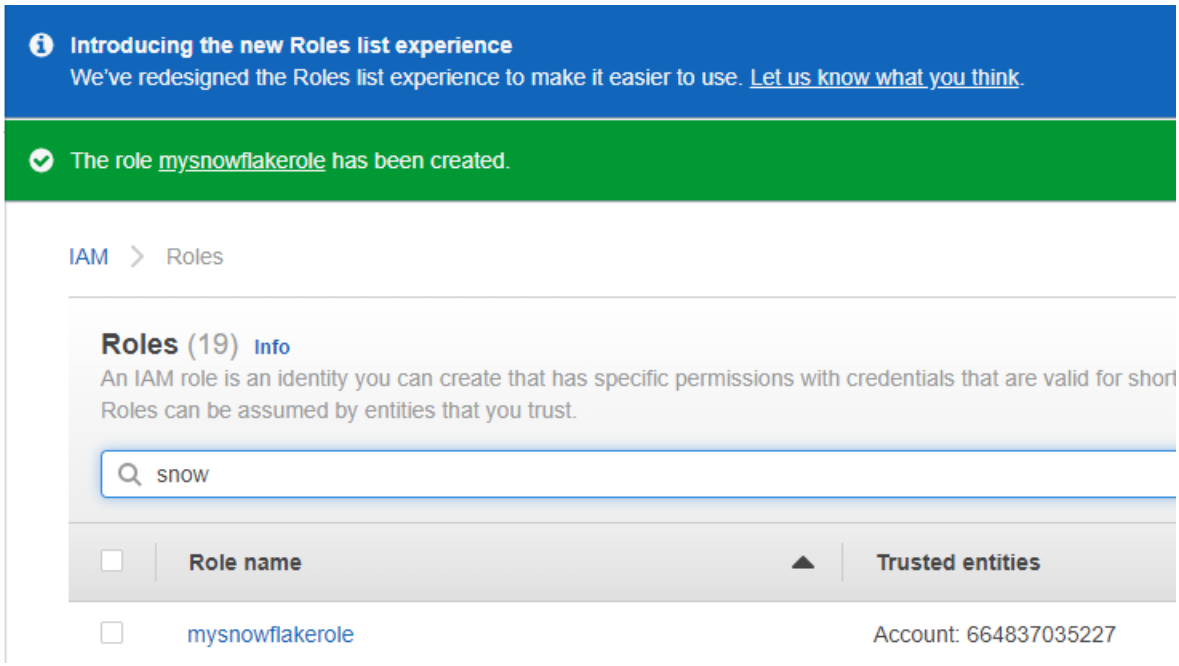
21.5.26 Click the **Next:Tags** button. Choose **Next: Review**

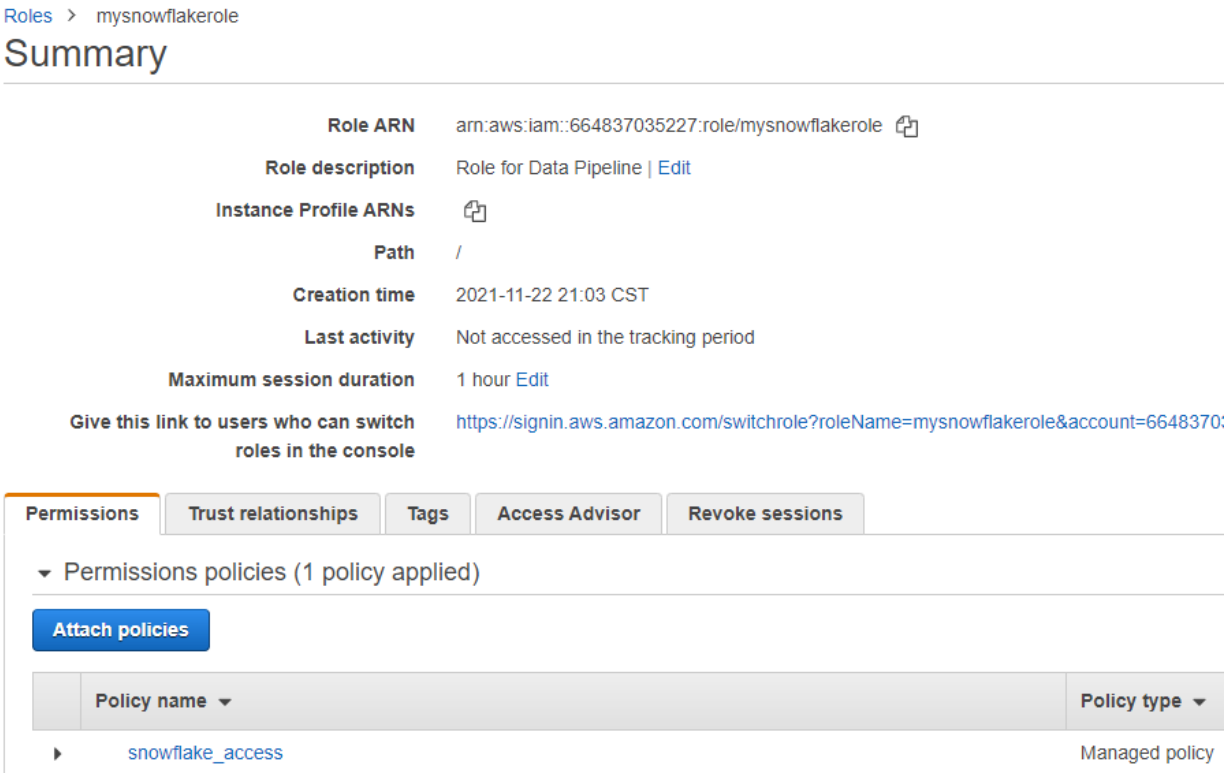
21.5.27 Review Page in AWS Management Console



21.5.28 Enter a name **mysnowflakerole** and description for the role, and click the **Create role** button.

You have now created an IAM policy for a bucket, created an IAM role, and attached the policy to the role.





Record the Role ARN value located on the role summary page. In the next step, you will create a Snowflake integration that references this role.

For example: **arn:aws:iam::664837035227:role/mysnowflakerole**

Make a note of the **ARN** for the IAM Role. You will need this in the next step.

21.5.29 Create a Cloud Storage Integration in Snowflake. **[SNOWFLAKE]**

Similar to step 21.4.10 in the worksheets tab, click on the small, downward facing arrow,select “Load Script”, then browse to the “**ModernDataPipelines_Experiment.sql**” file you downloaded in section 21.3 and select “Open”.

Before we start using SQL in Worksheets we will turn on Code Highlight by clicking on the 3 dots on the top right hand corner of the worksheet, and then clicking on **Turn on Code Highlight**. This will make it easier to identify the SQL that will be executed.

Set the context for the worksheet by executing the below sql.

```
-- Set the context for the worksheet.

USE ROLE ACCOUNTADMIN;
USE CITIBIKE_PIPELINES.PUBLIC;
USE WAREHOUSE DATAPIPELINES_WH;
```

Run the queries by placing your cursor anywhere in the command and clicking the blue "Run" button at the top of the page or by hitting Ctrl/Cmd+Enter on your keyboard

Warning
From here on in the labs, never check the "All Queries" box at the top of the worksheet. We want to run SQL queries one at a time in a specific order; not all at once.

Create Cloud Storage Integration in Snowflake using the below sql. Use the ARN of the IAM role created in step 1.6.3

```
-- Create Cloud Storage Integration.
create or replace storage integration
citibike_snowpipetype = external_stage
storage_provider
= s3enabled =
true
storage_aws_role_arn =
'arn:aws:iam::664837035227:role/mysnowflakerole'
storage_allowed_locations = ('s3://snowflake-
data-pipeline-session1-george/citibike-
pipeline');
```

Note the Storage Integration name created in Snowflake. We need the name in the following sections.

21.5.30 Retrieve the AWS IAM User for your Snowflake Account. **[SNOWFLAKE]**
Follow the instructions outlined in the Snowflake documentation using the link below.
https://docs.snowflake.com/en/user-guide/data-load-s3-config-storage-integration.html#step-4-retrieve-the-aws-iam-user-for-your-snowflake-account

```
desc integration citibike_snowpipe;
```

```
desc integration citibike_snowpipe;
```

s Data Preview

ery ID SQL 126ms 8 rows

result... Copy

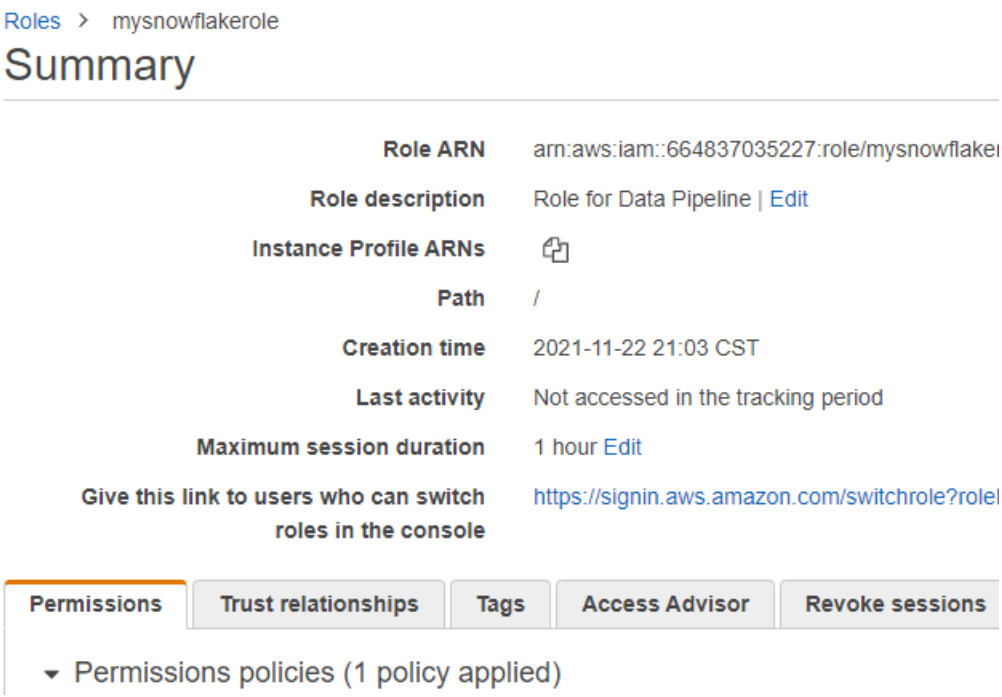
Row	property	property_type ↑	property_value
4	STORAGE_BLOCKED_LOCATIONS	List	
2	STORAGE_PROVIDER	String	S3
5	STORAGE_AWS_IAM_USER_ARN	String	arn:aws:iam::941093095223:user/92aj-s-v2sc2810
6	STORAGE_AWS_ROLE_ARN	String	arn:aws:iam::664837035227:role/mysnowflakerole
7	STORAGE_AWS_EXTERNAL_ID	String	LBA64704_SFCRole=3_G31c29cYbWdAYi71EDegGbpEM

Record the following values from the output of the above sql statement:
STORAGE_AWS_IAM_USER_ARN = arn:aws:iam::941093095223:user/92aj-s-v2sc2810
STORAGE_AWS_EXTERNAL_ID =
LBA64704_SFCRole=3_G31c29cYbWdAYi71EDegGbpEMvM=

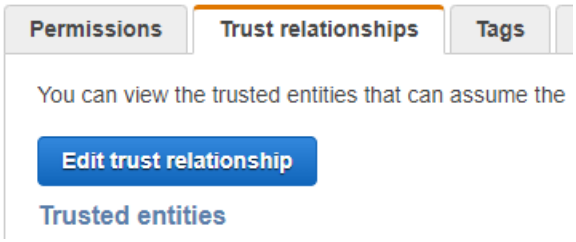
21.5.31 Grant the IAM User Permissions to Access Bucket Objects **[AWS]**

Follow the instructions outlined in the Snowflake documentation using the link below.
https://docs.snowflake.com/en/user-guide/data-load-s3-config-storage-integration.html#step-5-grant-the-iam-user-permissions-to-access-bucket-objects
Complete only the instructions for Step5 in the documentation link. After completing step 5, come back to this guide.

Edit the role that we created **mysnowflakerole**. Choose **Trust relationships**



21.5.32 Choose **Edit trust relationship**



21.5.33 Paste the updated policy replacing the JSON

Edit Trust Relationship

You can customize trust relationships by editing the following access control policy document.

Policy Document

```
1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Sid": "",
6        "Effect": "Allow",
7        "Principal": {
8          "AWS": "arn:aws:iam::941093095223:user/92aj-s-v2sc2810"
9        },
10       "Action": "sts:AssumeRole",
11       "Condition": {
12         "StringEquals": {
13           "sts:ExternalId": "LBA64704_SFCRole*"
14         }
15       }
16     }
17   ]
18 }
```

21.5.34 Choose **Update Trust Policy**

We just completed configuring a Snowflake storage integration object to delegate authentication responsibility for cloud storage to a Snowflake identity and access management (IAM) entity.

21.6 Creating an External Stage for Snowpipe

21.6.1 Create the streaming_data EXTERNAL STAGE object where the data will land. In the worksheet, execute the below sql statement to create the External Stage object.

```
-- Creating an External Stage.
-- Replace the s3 bucketname and the folder name as highlighted
below.

create or replace stage
  streaming_data url =
  's3://<s3bucketname>/<folder>/'
  storage_integration =
  citibike_snowpipe
  file_format=(type=json);
```

21.7 Finalizing Setup

21.7.1 Pause to take stock for what has been built so far. An external stage pointing to an S3 bucket was created.

```
show stages like '%STREAMING%';
```