

AWS S3 Archival Using Snowflake External Function

Data Sheet | LumenData

Problem Statement

1. S3 Files after being loaded into RAW tables Snowflake should be archived into Archive folder and the original source file should be deleted.
2. As Snowflake runs python in Sandbox, any external connections to AWS S3 or Rest API connections are not allowed. We are using External Functions provided by Snowflake.

Key Components

AWS IAM Roles

Snowflake External Functions

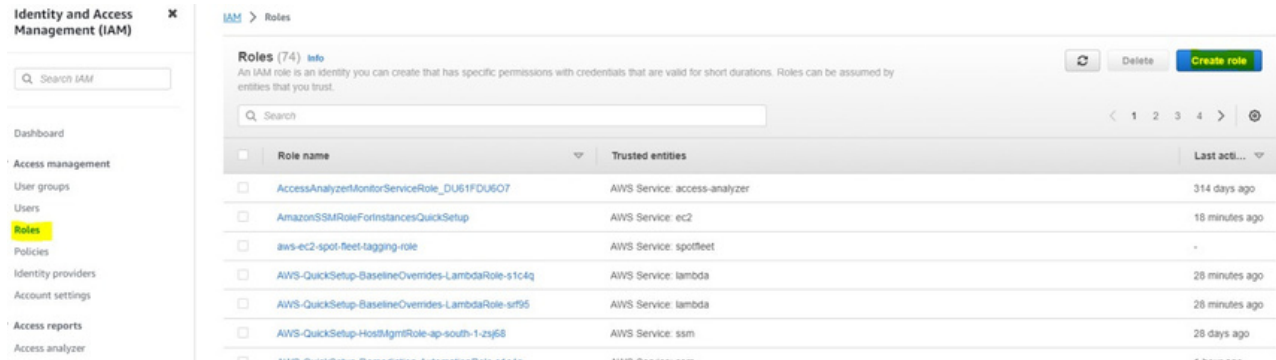
AWS Lambda

AWS API Gateway

Steps:

1. Configuring AWS Lambda Role in AWS IAM

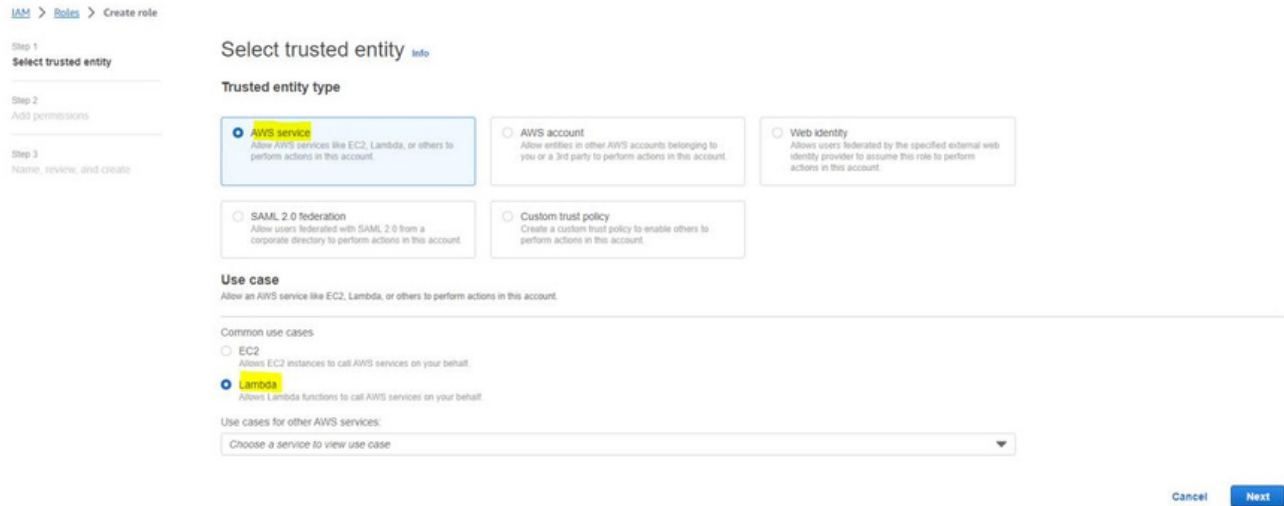
a) Navigate to IAM in AWS and Click on Create Role



The screenshot shows the AWS IAM console interface. On the left is a navigation sidebar with 'Roles' highlighted. The main content area displays a list of roles. A search bar is at the top, and a 'Create role' button is in the top right corner.

Role name	Trusted entities	Last accessed
AccessAnalyzerMonitorServiceRole_DU61FDU607	AWS Service: access-analyzer	314 days ago
AmazonSSMRoleForInstancesQuickSetup	AWS Service: ec2	16 minutes ago
aws-ec2-spot-fleet-tagging-role	AWS Service: spotfleet	-
AWS-QuickSetup-BaselineOvmides-LambdaRole-s1c4q	AWS Service: lambda	28 minutes ago
AWS-QuickSetup-BaselineOvmides-LambdaRole-sr95	AWS Service: lambda	28 minutes ago
AWS-QuickSetup-HostMgmtRole-ap-south-1-z3j68	AWS Service: ssm	28 days ago

b) Select AWS service as Lambda



The screenshot shows the 'Create role' wizard in the AWS IAM console, specifically Step 1: 'Select trusted entity'. The 'AWS service' option is selected under 'Trusted entity type'. Under 'Use case', 'Lambda' is selected.

Trusted entity type

- AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity
Allow users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

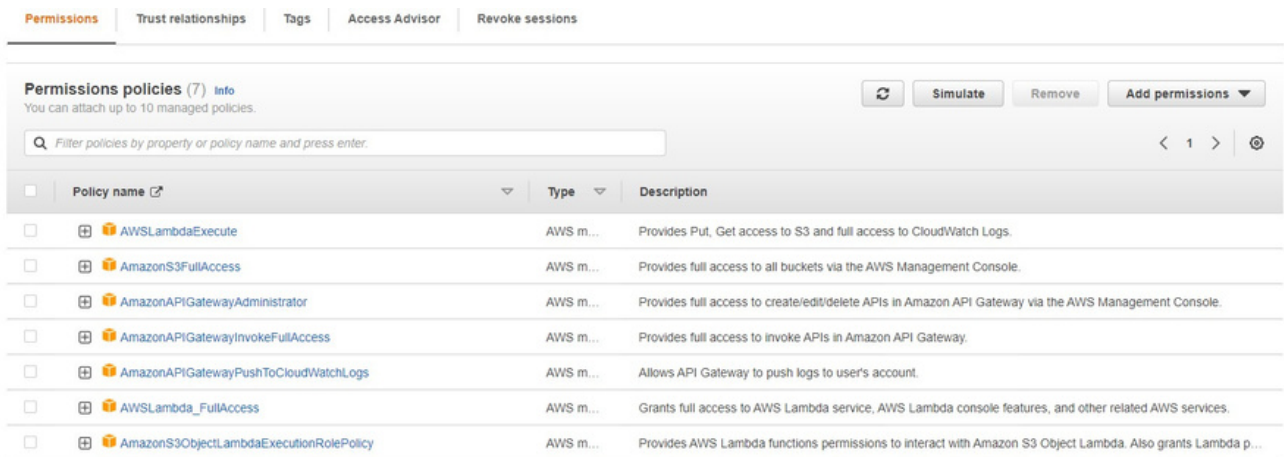
Common use cases

- EC2
Allow EC2 instances to call AWS services on your behalf.
- Lambda**
Allow Lambda functions to call AWS services on your behalf.

Use cases for other AWS services:
Choose a service to view use case

Buttons: Cancel, Next

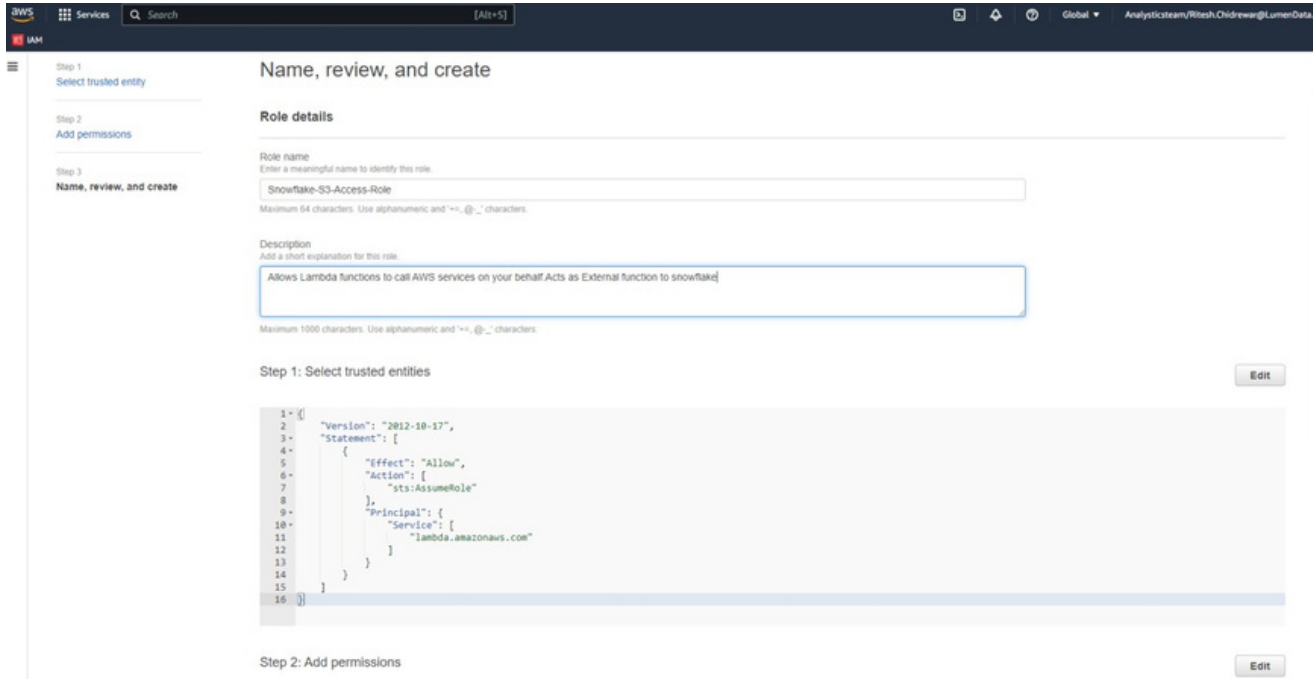
c) Attach the below policies to **Snowflake-S3-Access-Role**



The screenshot shows the 'Permissions policies' page for the 'Snowflake-S3-Access-Role' in the AWS IAM console. The 'Permissions' tab is active, showing a list of 7 policies. A search bar is at the top, and buttons for 'Simulate', 'Remove', and 'Add permissions' are visible.

Policy name	Type	Description
AWSLambdaExecute	AWS m...	Provides Put, Get access to S3 and full access to CloudWatch Logs.
AmazonS3FullAccess	AWS m...	Provides full access to all buckets via the AWS Management Console.
AmazonAPIGatewayAdministrator	AWS m...	Provides full access to create/edit/delete APIs in Amazon API Gateway via the AWS Management Console.
AmazonAPIGatewayInvokeFullAccess	AWS m...	Provides full access to invoke APIs in Amazon API Gateway.
AmazonAPIGatewayPushToCloudWatchLogs	AWS m...	Allows API Gateway to push logs to user's account.
AWSLambda_FullAccess	AWS m...	Grants full access to AWS Lambda service, AWS Lambda console features, and other related AWS services.
AmazonS3ObjectLambdaExecutionRolePolicy	AWS m...	Provides AWS Lambda functions permissions to interact with Amazon S3 Object Lambda. Also grants Lambda p...

d) Provide the Role name and Description (Optional) and click on Create Role.

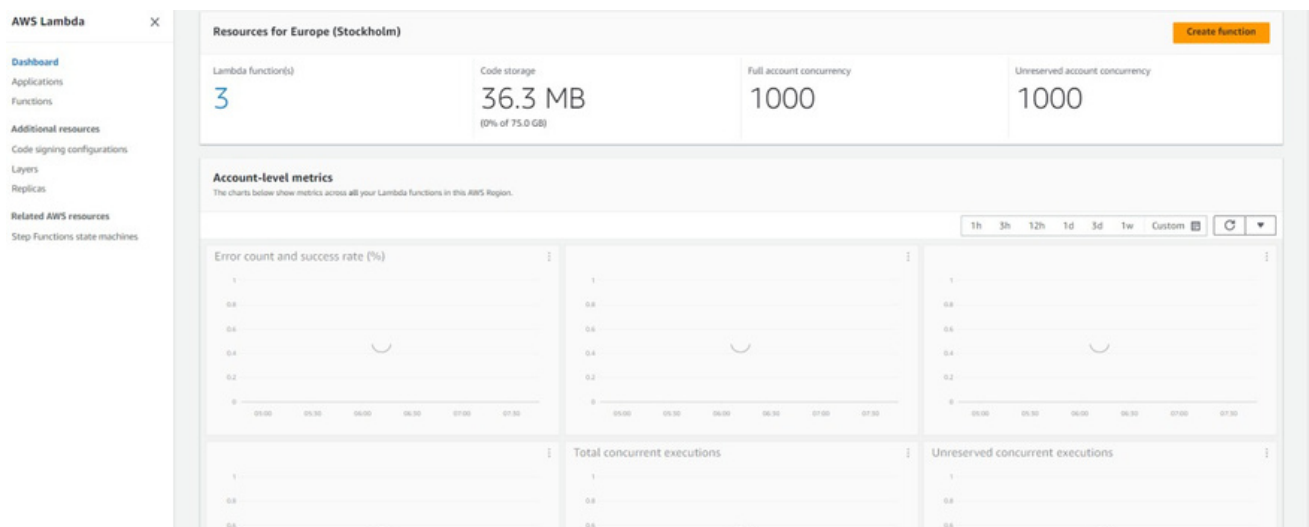


e) Make sure to note the Role Names and ARNs in notepad throughout the process.

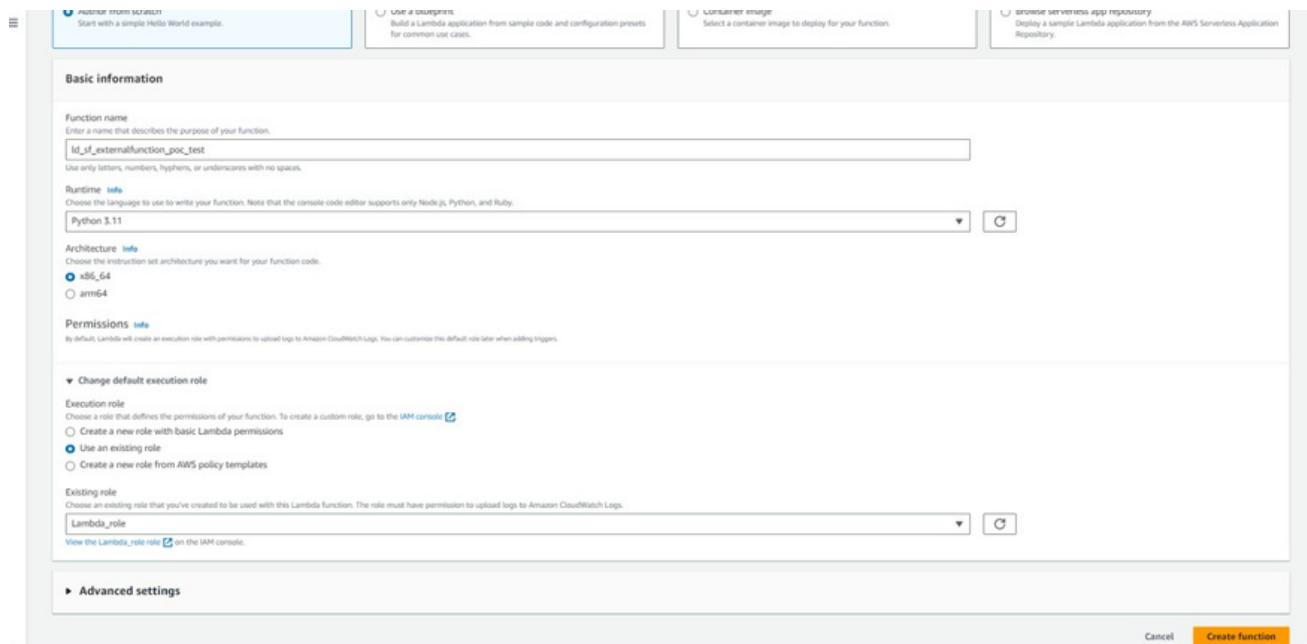
Lambda Service Role: **Snowflake-S3-Access-Role**

2. Configuring AWS Lambda Function

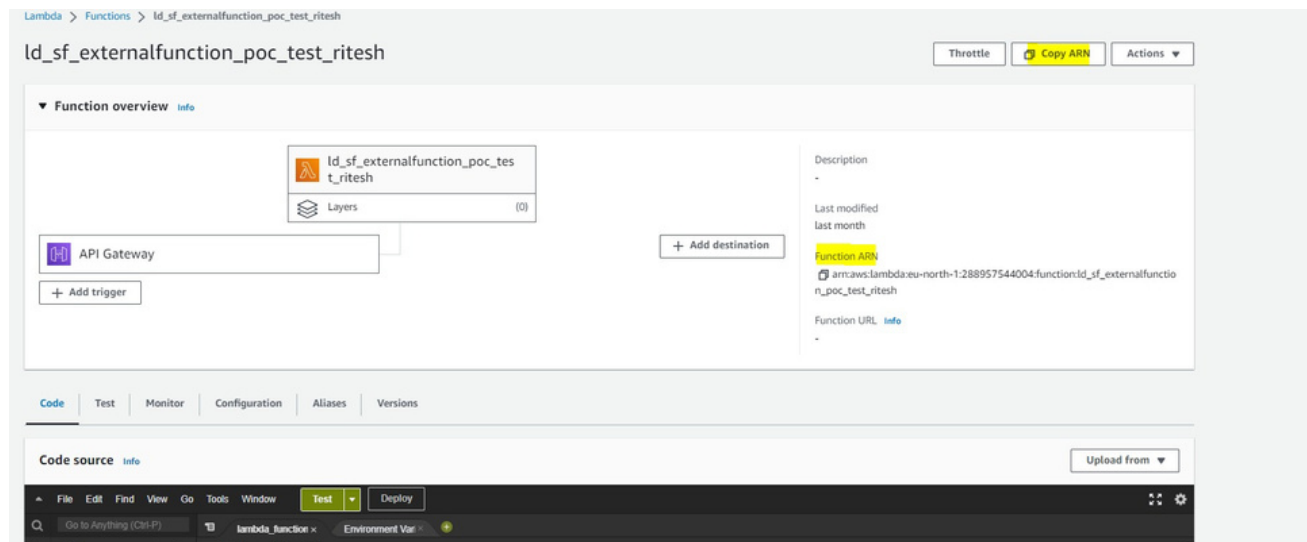
a) Navigate to AWS Lambda Service and Create a new function



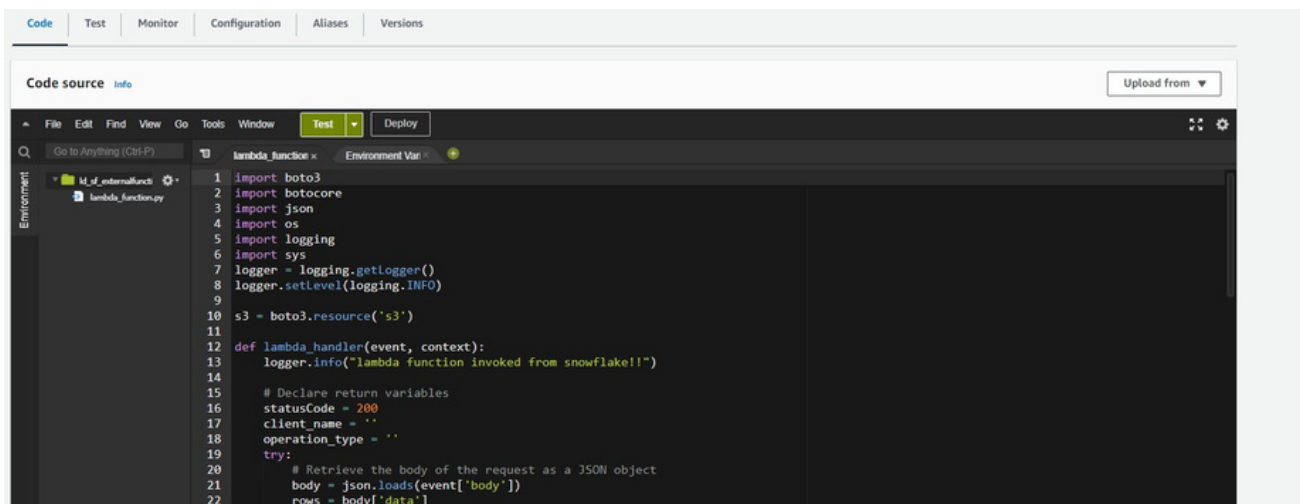
b) Provide the Function Name, Runtime as Python (Latest available), Architecture, Default Execution role (The role we created above for lambda) select from the dropdown.



c) As for us, we have already created one of the lambda functions. Please note the ARN of the Function



d) Code editor for lambda is available.



e) Snowflake Data Transfer Format

i. Snowflake transfers data to the lambda function in a specific format (JSON).

Ref link: [Snowflake-Data-Format](#)

(IMP) Please make sure you pass and retrieve the data in the above-mentioned format or else Snowflake won't be able to parse the result/data.

Ex. {

```
"body": "{\"data\": [[0, \"client-name \", \"archive\"]]}"
```

ii. (IMP) Snowflake wraps the data into JSON which has JSONArray data inside it.

Below are the two rows with respective values. First array Value i.e., 0 corresponds to 1 st row and 1 represents the second row.

At the lambda side, we have to fetch the body object and iterate through each row to access the data.

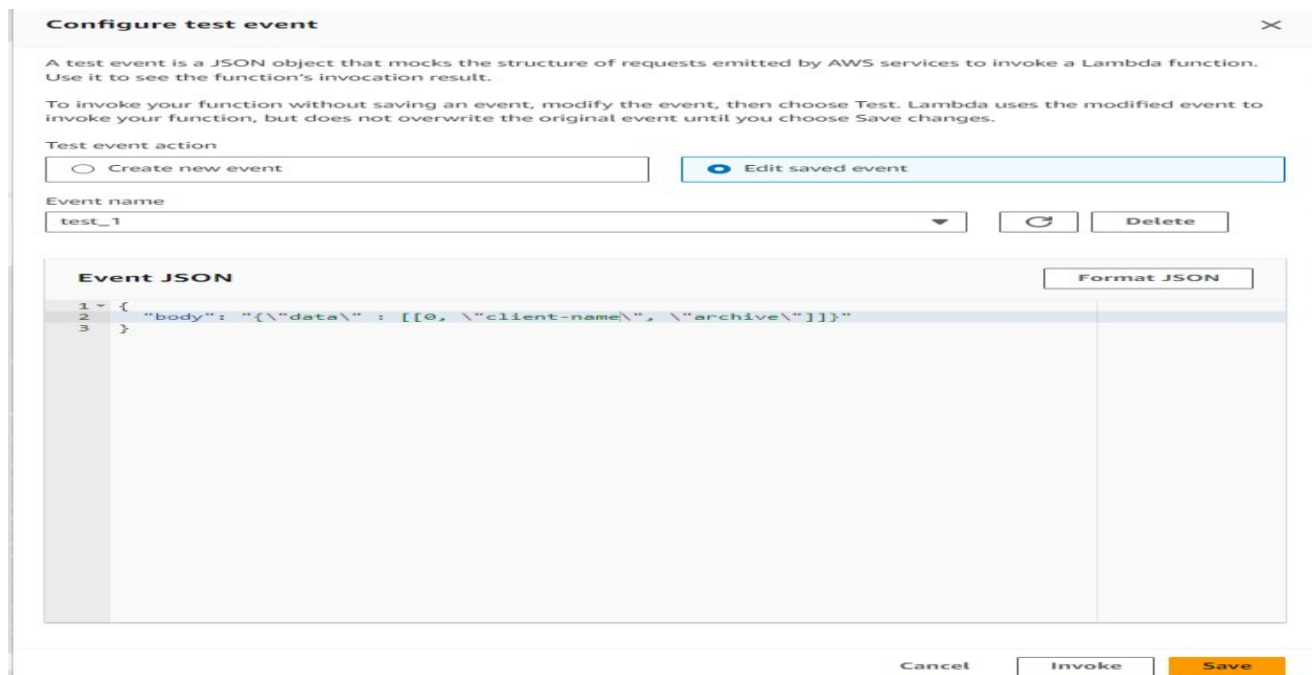
Ex: Snowflake data format- {"data": [[0,"arg 1",arg2"], [1,"arg3","arg4"]]}

This entire structure is wrapped into JSON string while transferring data through Snowflake-Api Below is the final format.

```
Ex: {  
  "body": "{\"data\": [[0,\"arg 1\",arg2\"],[1,\"arg3\", \"arg4\"]]}"  
}
```

f) Configure the test event in the lambda function.

We provided the client's name and the operation type(archive) to be performed on S3 file.



g) Sample S3 archival code

Once your code is ready, you can test it with the Test Event configured in the previous steps.

```
Code source info Upload from  
File Edit Find View Go Tools Window Test Deploy  
Go to Anything (Ctrl-P)  
Environment  
Id_sf_externalfunc  
lambda_function.py  
57 print('Error Message: {}'.format(error))  
58  
59 except botocore.exceptions.ParamValidationError as error:  
60 logger.error("Missing required parameters while calling the API.")  
61 print('Error Message: {}'.format(error))  
62  
63 else:  
64 print("error archiving the files!!Exiting..")  
65 sys.exit(1)  
66 except Exception as err:  
67 # Statuscode = 400 signifies an error  
68 statusCode = 400  
69 # Function will return the error  
70  
71  
72 print("-----3-----")  
73  
74  
75  
76  
77  
78  
7:15 Python Spaces: 4
```

Response received after successful Copy with test_event

```
Code source info Upload from  
File Edit Find View Go Tools Window Test Deploy  
Go to Anything (Ctrl-P)  
Environment  
Id_sf_externalfunc  
lambda_function.py  
Execution result x  
Status: Succeeded Max memory used: 82 MB Time: 1457.05 ms  
+ Execution results  
Test Event Name  
test_1  
Response  
{  
  "body": "{\\\"data\\\": [[0, \\\"file copied to the destination bucket successfully!\\\"]]}"  
}  
Function Logs  
[INFO] 2023-09-07T08:19:43.686Z Found credentials in environment variables.  
START RequestId: 3c35a2b1-195e-46d4-8245-cfb6d3fe465 version: $LATEST  
[INFO] 2023-09-07T08:19:43.780Z 3c35a2b1-195e-46d4-8245-cfb6d3fe465 lambda function invoked from snowflake!!
```

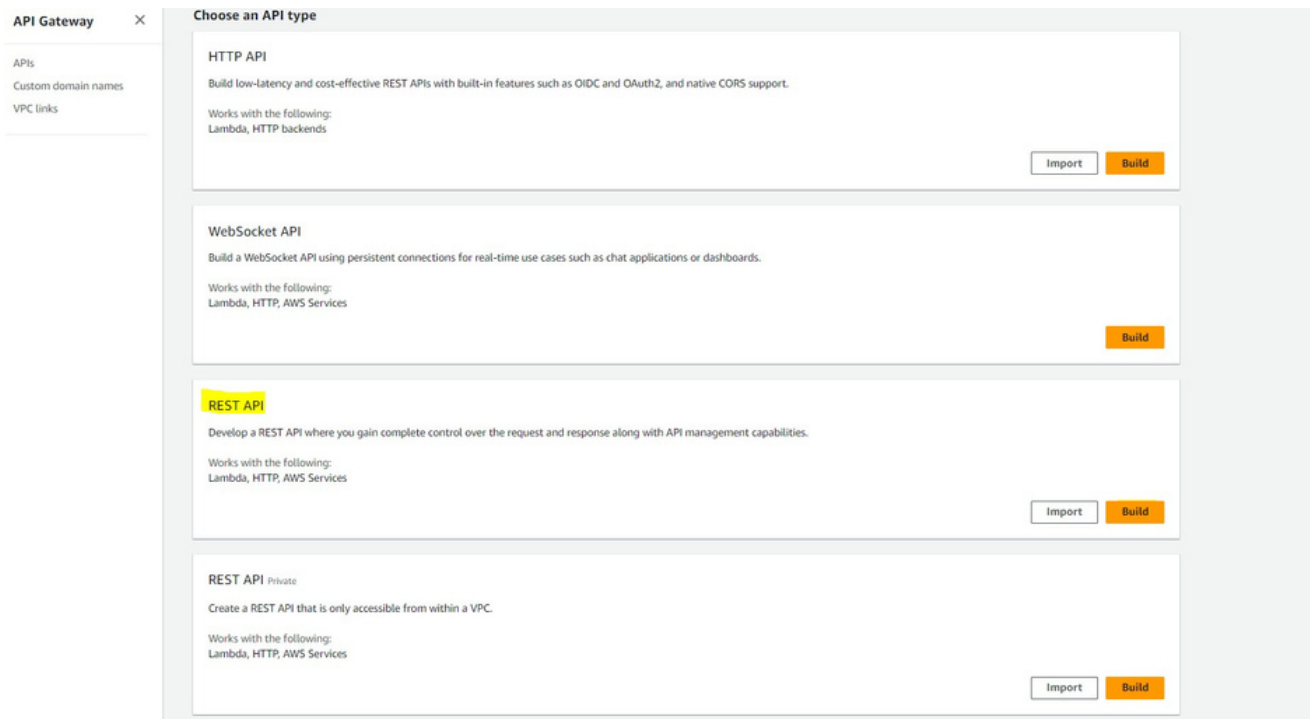
h) Now we have tested the code successfully, we can add the function name and ARN to our notepad documentation.

Lambda Service Role: Snowflake-S3-Access-Role
Lambda FunctionId_sf_externalfunction_poc
Lambda Function ARN: arn:aws:lambda:eu-north-1:288957544004:function:ld_sf_externalfunction_poc_test

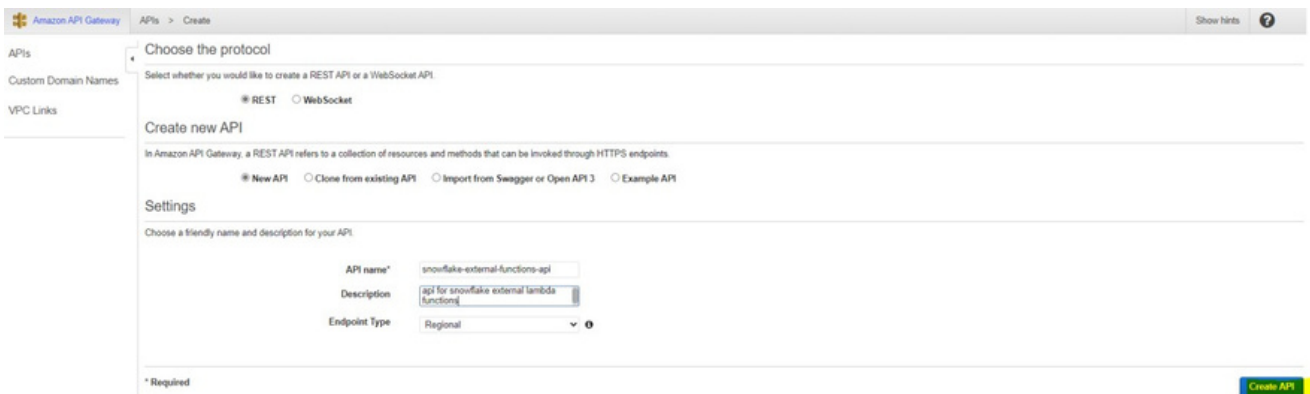
3. Configuring AWS API Gateway

a) Create the API Gateway to expose the Lambda function created.

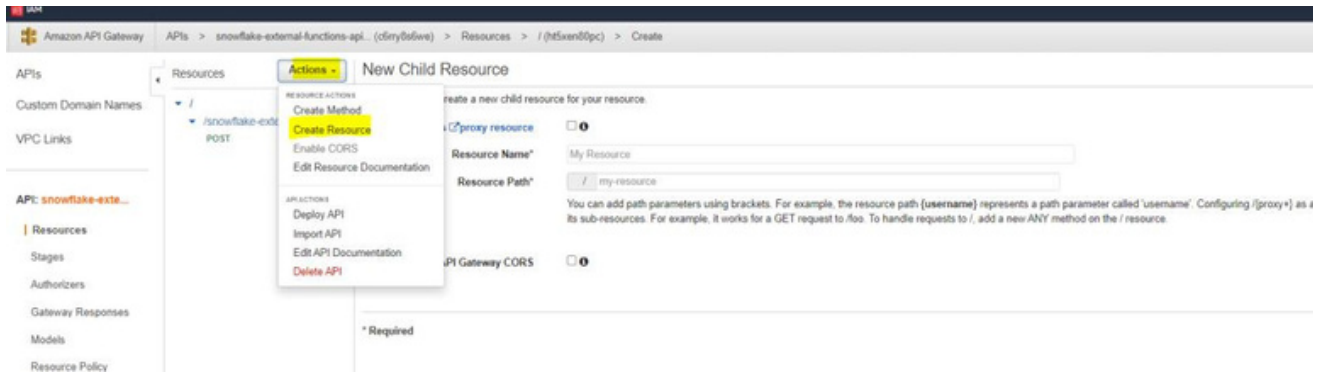
i. Select REST API and click on build.



ii. Provide the API name and Description. Click on Create API.

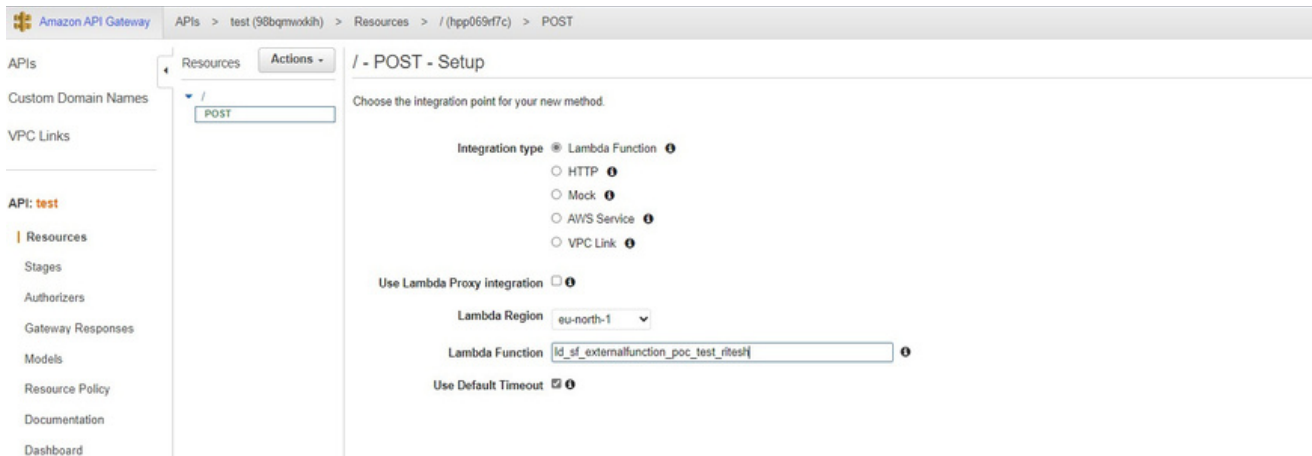


iii. Create a Resource and Method. Provide the resource name and Click on Create Resource.

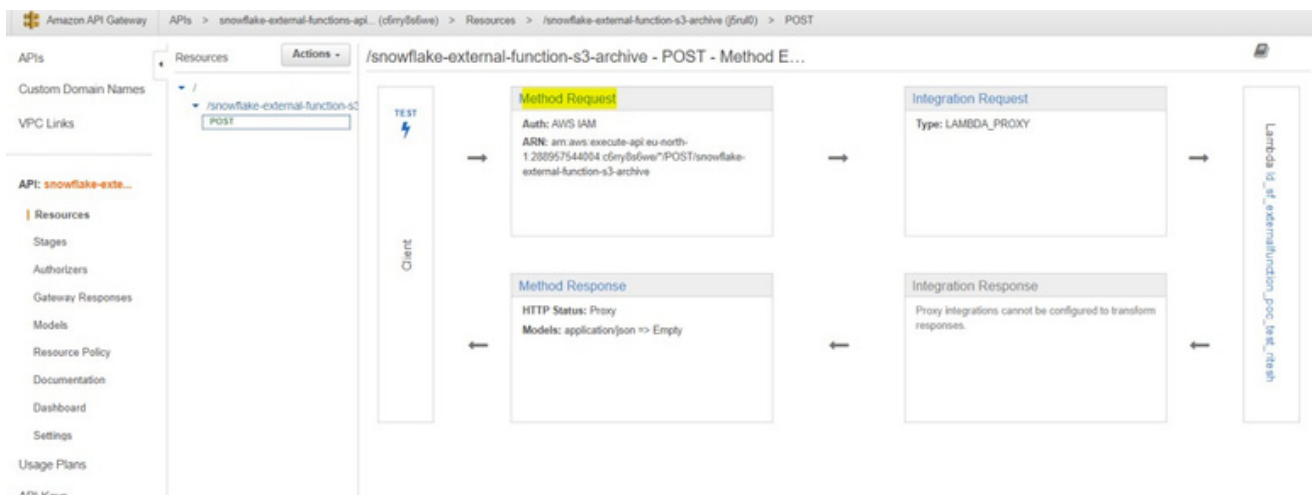


iv. Create Method.

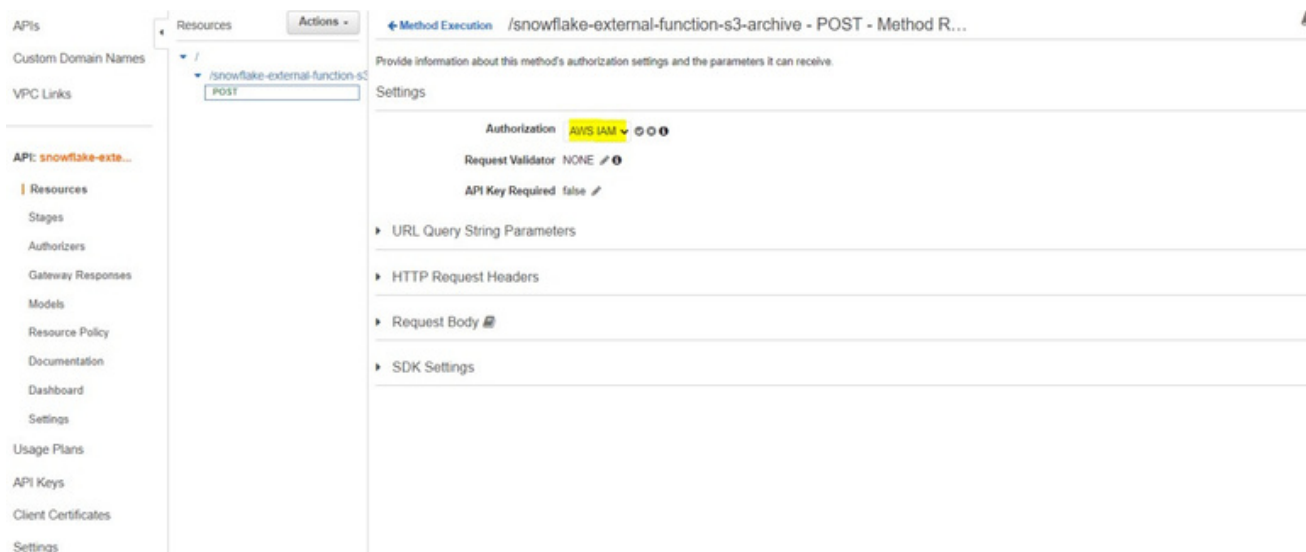
After creating the method, a warning will pop up. Click on Ok.



Once the Method is created, Click on Method Request



Inside Method request, select Authorization as IAM



Once authorization is set, return to Method overview screen, and you can see the ARN populated.

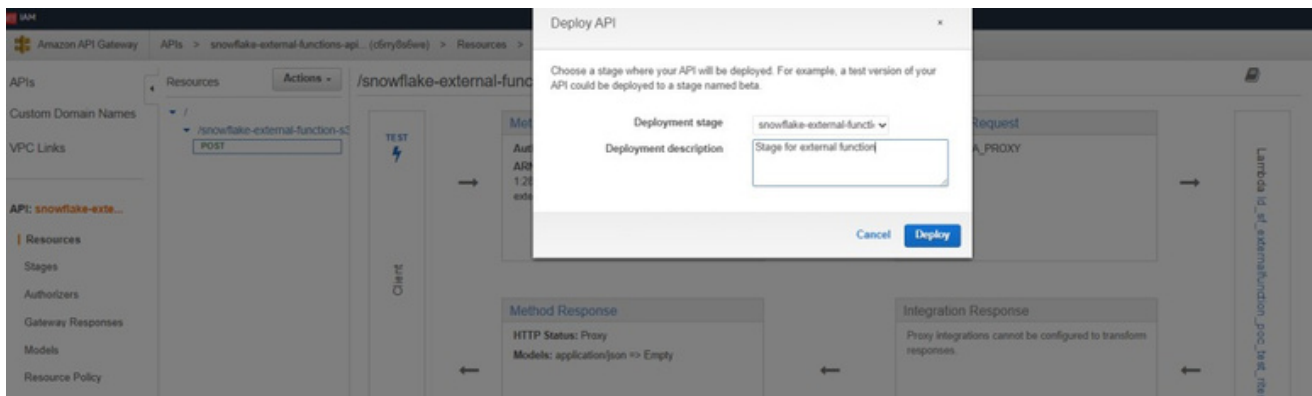


Note: Add this ARN to the notepad Documentation

Lambda Service Role: Snowflake-S3-Access-Role
Lambda FunctionId_sf_externalfunction_poc
Lambda Function ARN: arn:aws:lambda:eu-north-1:288957544004:function:ld_sf_externalfunction_poc_test
Lambda Method ARN : arn:aws:execute-api:eu-north-1:*****:*****/*/POST/snowflake-external-functions3-archive

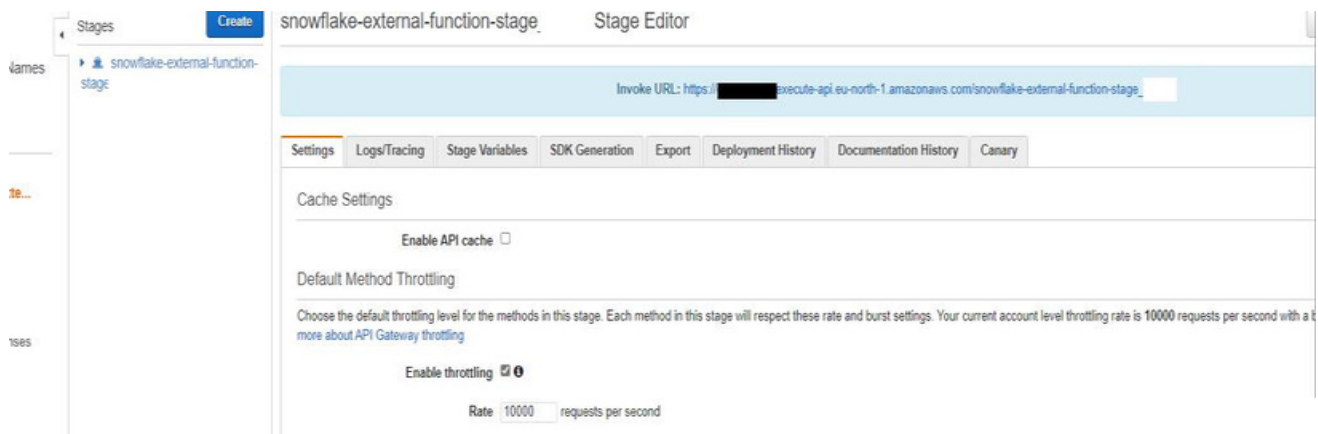
b) Deploy API created to Stage

i. Provide the Deployment stage name and description



ii. Once deployed, we get the Stage invoke URL. This is the URL through which the API is accessed by the authorized roles.

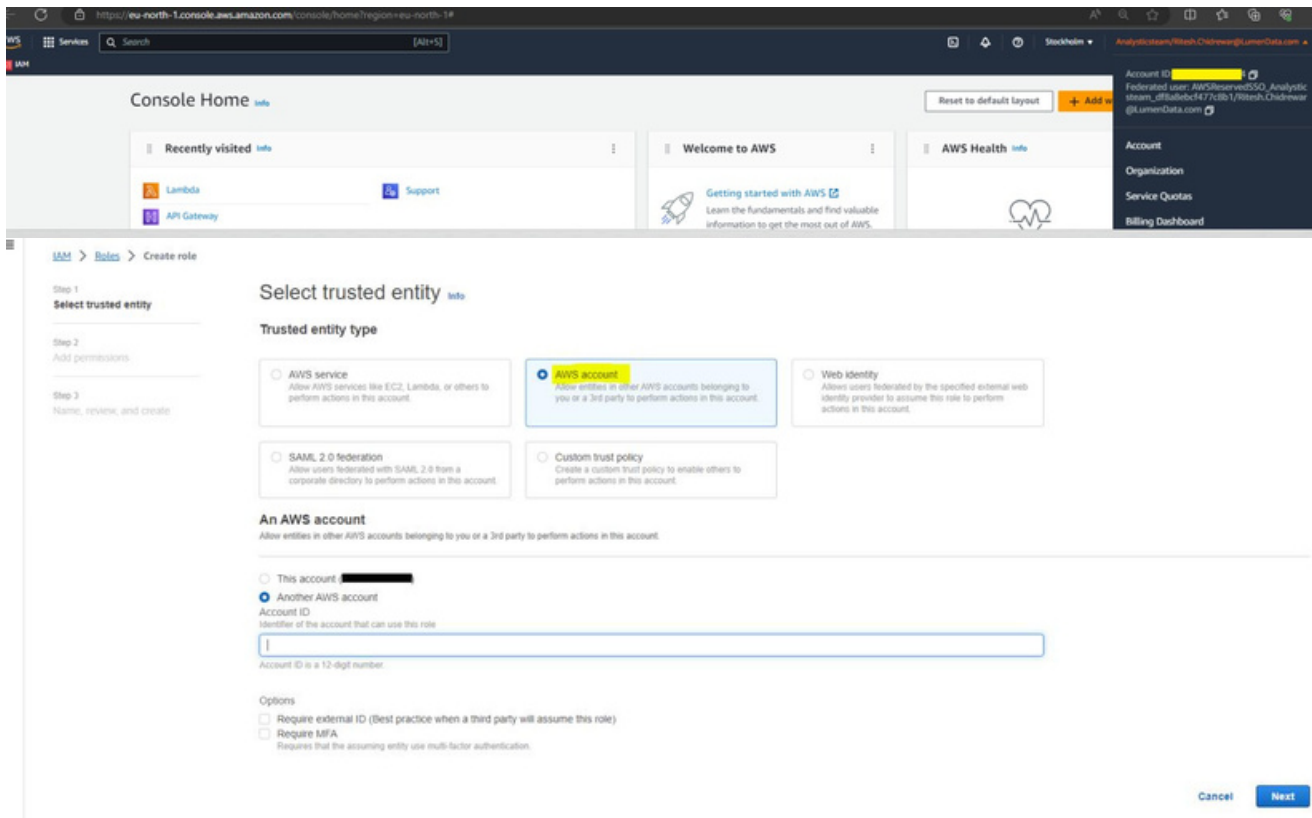
Note this in notepad documentation.



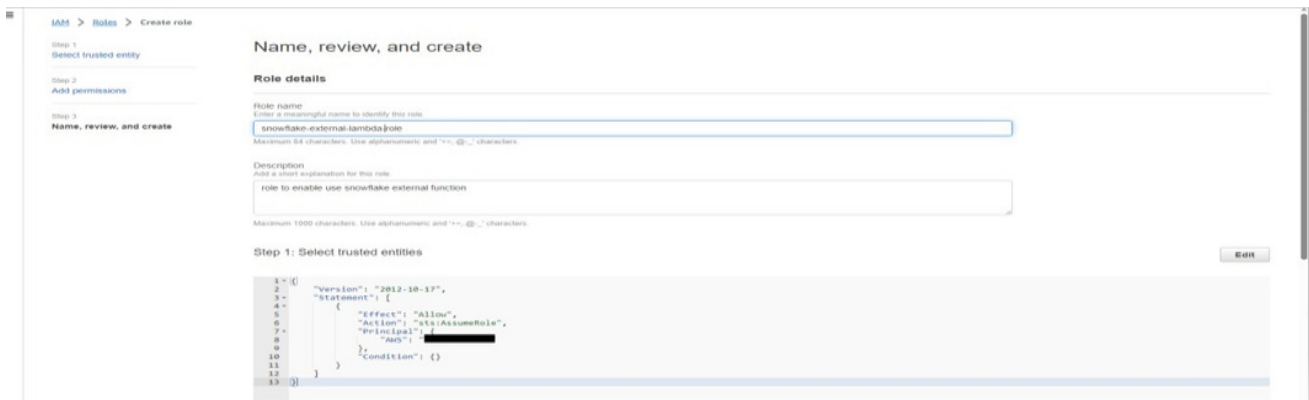
4. Configuring another AWS IAM role to enable handshake between Snowflake and API Integration.

Follow the same steps for IAM creation as mentioned above. Also, attach the same policies

a) For IAM creation, select Another AWS Account. Copy the account ID as shown in the screenshot.



b) Provide the Role name and Description.



Note Role name in our Notepad Documentation.

Lambda Service Role: Snowflake-S3-Access-Role

Lambda FunctionId_sf_externalfunction_poc

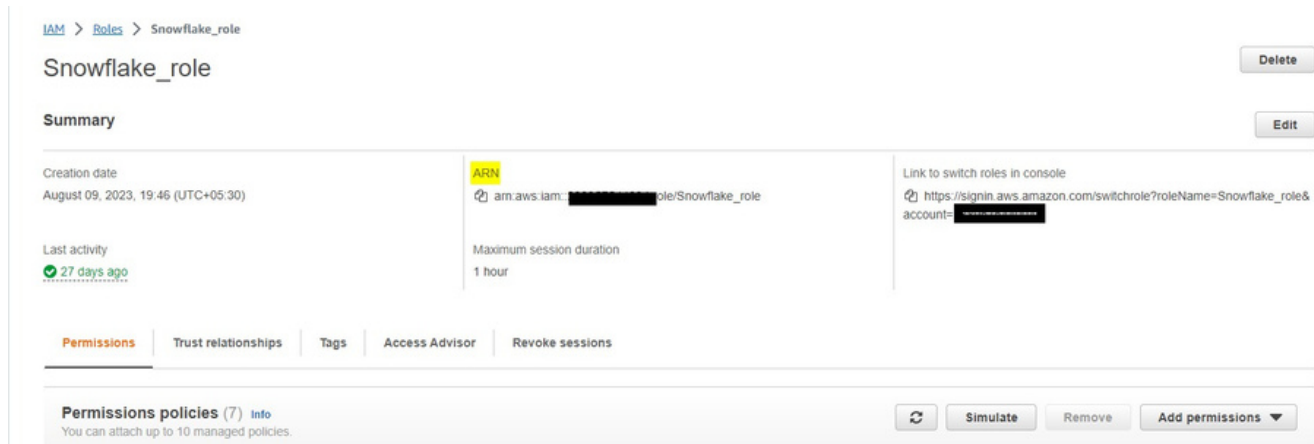
Lambda Function ARN: arn:aws:lambda:eu-north-1:*****:function:ld_sf_externalfunction_poc_test

Lambda Method ARN : arn:aws:execute-api:eu-north1:*****:*****/*/POST/snowflake-external-function-s3-archive

Account ID: 012345678910 IAM

Account Role: snowflake-external-lambda-role

c) Retrieve the Role ARN for the above role that you've created.
Sample screenshot



Note ARN in our Notepad Documentation.

Lambda Service Role: Snowflake-S3-Access-Role

Lambda FunctionId_sf_externalfunction_poc

Lambda Function ARN: arn:aws:lambda:eu-north-1:*****:function:ld_sf_externalfunction_poc_test

Lambda Method ARN : arn:aws:execute-api:eu-north-1:*****:*****/*/POST/snowflake-external-function-s3-archive

Account ID: 012345678910 IAM

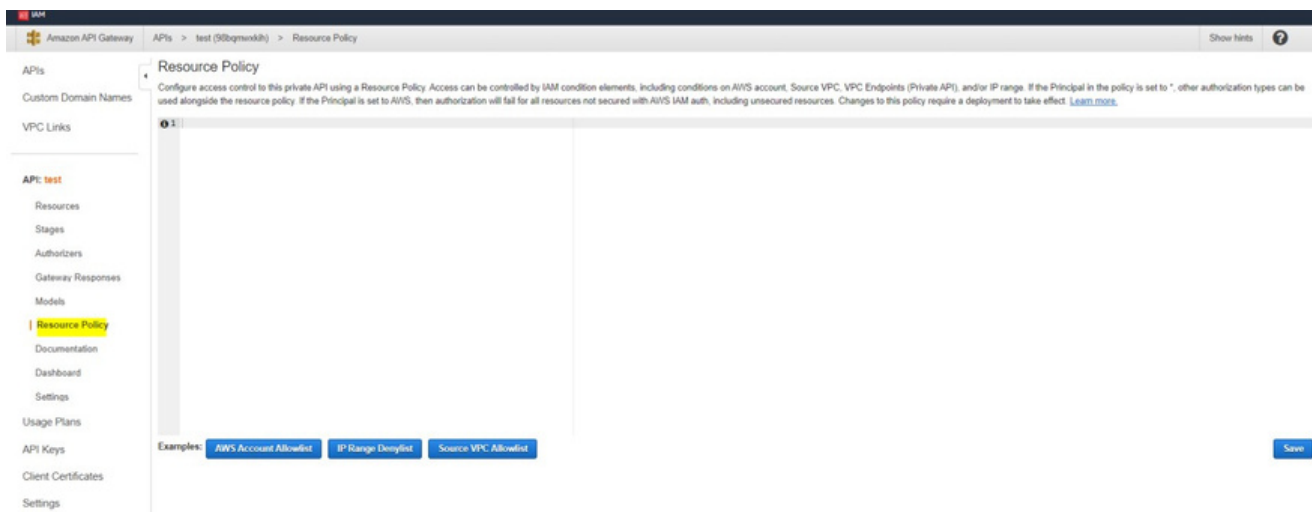
Account Role: snowflake-external-lambda-role

IAM Account Role ARN:

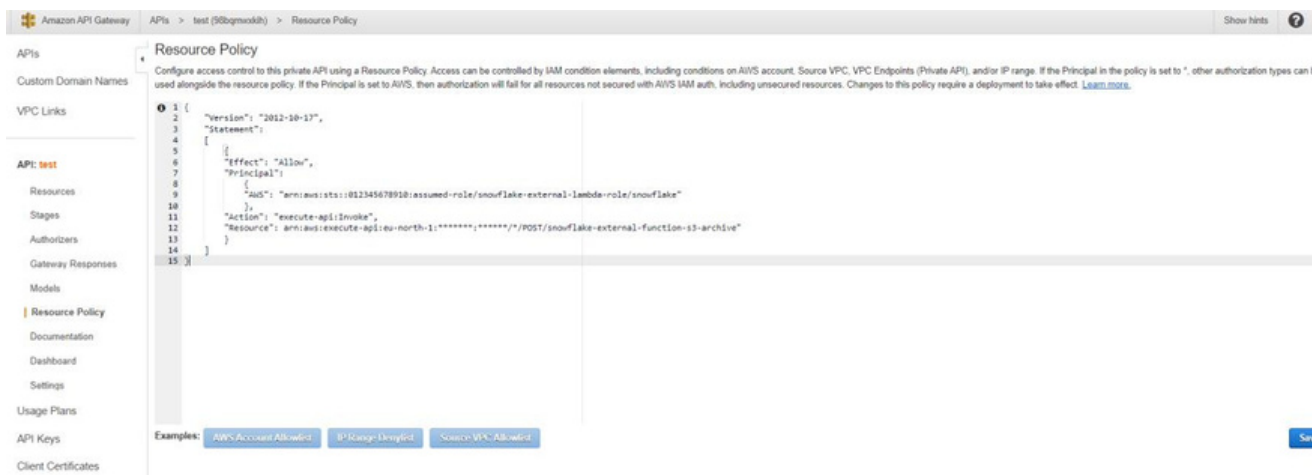
arn:aws:iam::012345678910:role/snowflake-external-lambda-role

d) Updating the Resource policy in API Gateway

To authorize the role created, we have to update the resource policy in the API Gateway.



Replace the account_id, IAM account Role, and Method ARN with the values from our documentation.



5. Snowflake Configuration and Setup

a) Attached below is the doc for snowflake commands and queries used for setting up the connection and creation of External function.

i. Create the API Integration object in Snowflake.

Once the DESCRIBE query is executed, we get the below Snowflake credentials .

Copy **API_AWS_IAM_USER_ARN** and **API_AWS_EXTERNAL_ID** in our notepad documentation sheet.



	property	property_type	property_value	property_default
1	ENABLED	Boolean	true	false
2	API_KEY	String		
3	API_PROVIDER	String		
4	API_AWS_IAM_USER_ARN	String		
5	API_AWS_ROLE_ARN	String		
6	API_AWS_EXTERNAL_ID	String		
7	API_ALLOWED_PREFIXES	List		[]
8	API_BLOCKED_PREFIXES	List		[]
9	COMMENT	String		

Lambda Service Role: Snowflake-S3-Access-Role

Lambda FunctionId_sf_externalfunction_poc

Lambda Function ARN: arn:aws:lambda:eu-north-1:*****:function:ld_sf_externalfunction_poc_test

Lambda Method ARN : arn:aws:execute-api:eu-north1:*****:*****/*/POST/snowflake-external-functions3-archive

Account ID: 012345678910

IAM Account Role: snowflake-external-lambda-role

IAM Account Role ARN:

arn:aws:iam::012345678910:role/snowflake-externallambda-role

API Integration Name:

ld_sf_aws_external_function_api_integration

api_aws_iam_user_arn:

arn:aws:iam::109876543210:user/xxxx-x-xxxxxxx

api_aws_external_id:

DEMO_SFCTestRole=0_x0xxxxxxxxxxxxxxxxxxx+xx0xx0xxx=

ii. Update the Trust Relationship in AWS IAM Role

The screenshot shows the AWS IAM console interface for the 'Snowflake_role'. The 'Trust relationships' tab is highlighted, showing a JSON policy that defines the trust relationship. The policy allows the role to assume another IAM role (arn:aws:iam::901469907738:user/Ag130000-s) under the condition that the sts:ExternalId matches 'KL88311_SFCTestRole-2_Y3mK9vW170F826p218ca8ZHLp8-'.

iii. Once the trust relationship is updated and saved, return to Snowflake and create the external function. (Attached doc above)

```
LD.PUBLIC - Settings +
--
--CREATE External function object
21
22 CREATE [OR REPLACE] external function <database>.<schema>.<function name>(<variables>
23 returns variant
24 api_integration = <api integration object>
25 as 'method invoke url'
26
27 CREATE external function LD.PUBLIC.ld_sf_aws_external_function(client_name string,operation_type string)
28 returns variant
29 MAX_BATCH_ROWS = 100
30 api_integration = ld_sf_aws_external_function_api_integration
31 as 'https://*****.execute-api.eu-north-1.amazonaws.com/snowflake-external-function-stage/snowflake-s3-archive';
32
33
```

iv. Test the function and check the output and file moved successfully to the archive location in S3.

```
--Test/call external function using below query  
select LD.PUBLIC.ld_sf_aws_external_function('client_name','archive');
```

ABOUT LUMENDATA:

LumenData is a leading provider of Enterprise Data Management, Cloud & Analytics solutions. We help businesses navigate their data visualization and analytics anxieties and enable them to accelerate their innovation journeys. Founded in 2008, with locations in multiple countries, LumenData is privileged to serve over 100 leading companies, including KwikTrip, Versant Health, US Food & Drug Administration, US Department of Labor, Cummins Engine, BCG, and others. LumenData is SOC2 certified and has instituted extensive controls to protect client data, including adherence to GDPR and CCPA regulations.

Get in touch to discuss how we can facilitate data-driven transformation for your organization.

MEET OUR AUTHORS



Ritesh Chidrewar
Senior Consultant



Ankit Kumar
Technical Lead