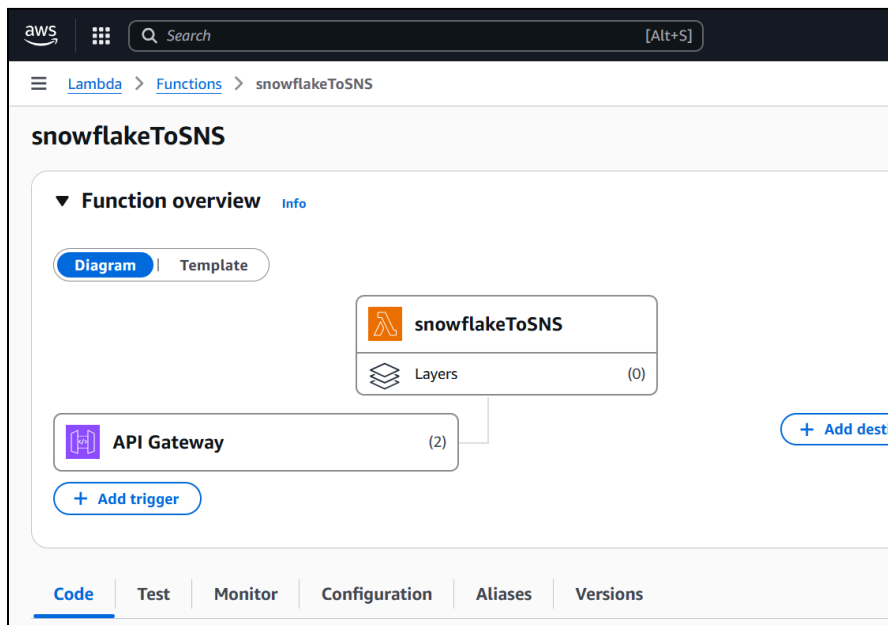
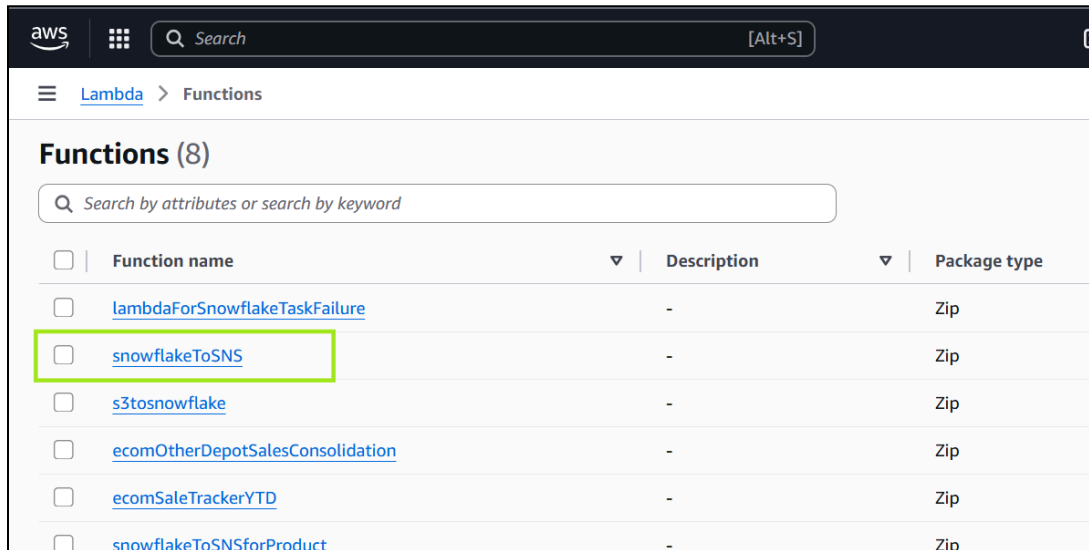


Snowflake to SNS Notification Pipeline

Distributor & Product Addition

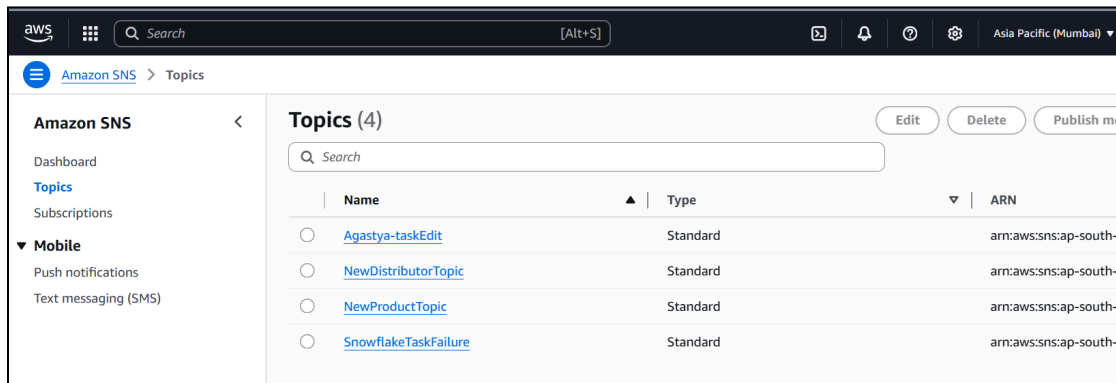
AWS Set-up:

- Navigate to AWS Lambda Console. Function name: 'snowflakeToSNS'.

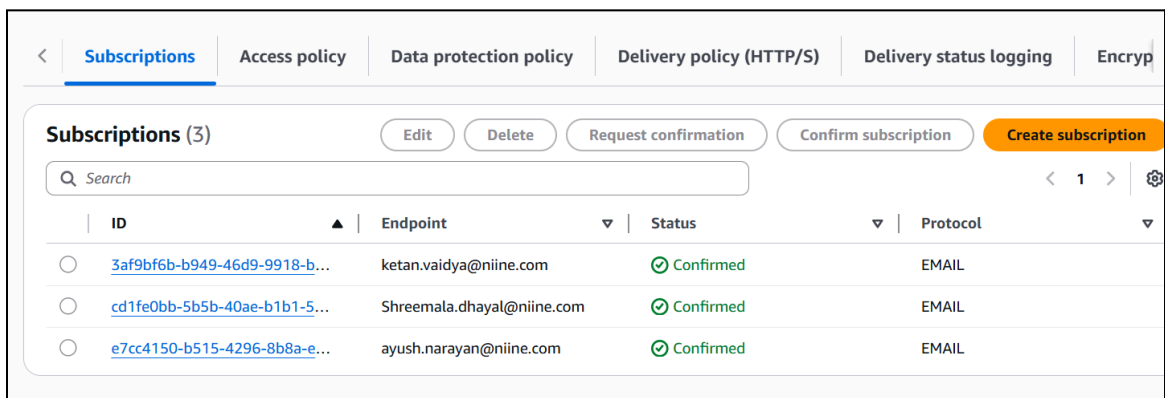


This function contains python code in the default code editor. The purpose of the code/lambda function is to receive messages from snowflake external function (mentioned later in this document), parse it and forward it to AWS SNS topic (in next step).

- Open the AWS SNS console. Go to topic '[NewDistributorTopic](#)' or '[NewProductTopic](#)'

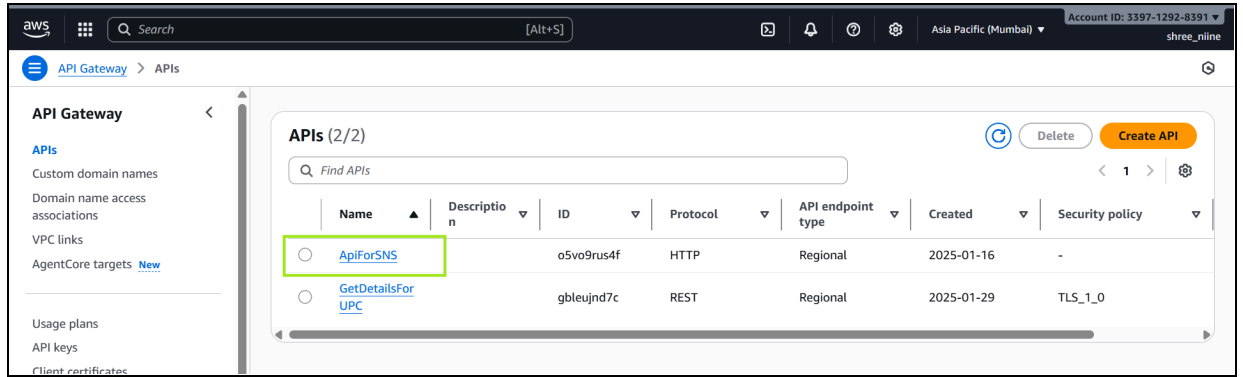


This topic is of type: “*Standard*” (not FIFO), everything else is default. Subscriptions are created which are subscribed to this topic. Below are the subscriptions (of type Email) which are subscribed to this topic:

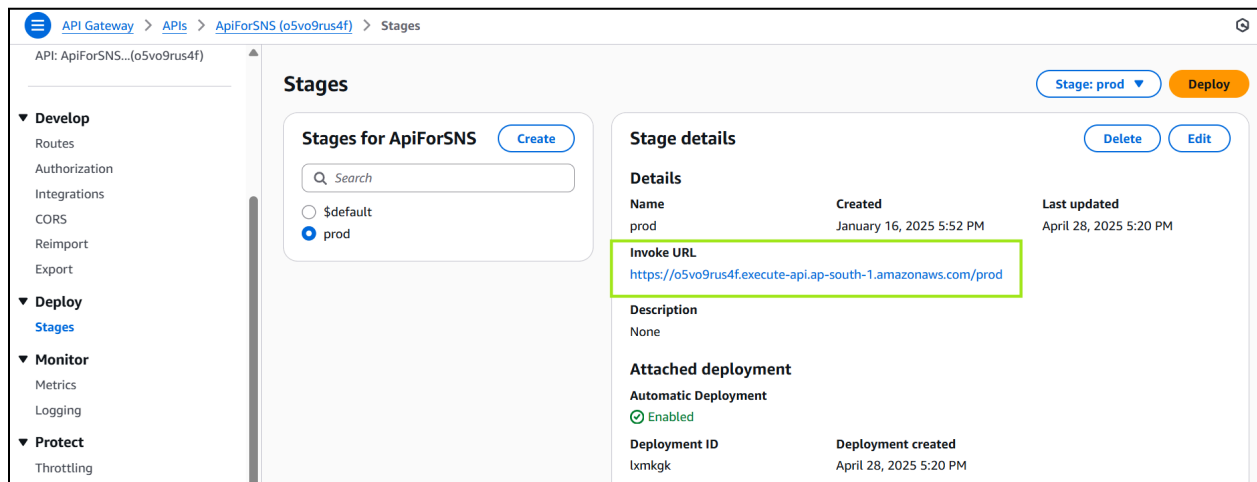


All the subscribed people will receive the mail when the pipeline is triggered from snowflake.

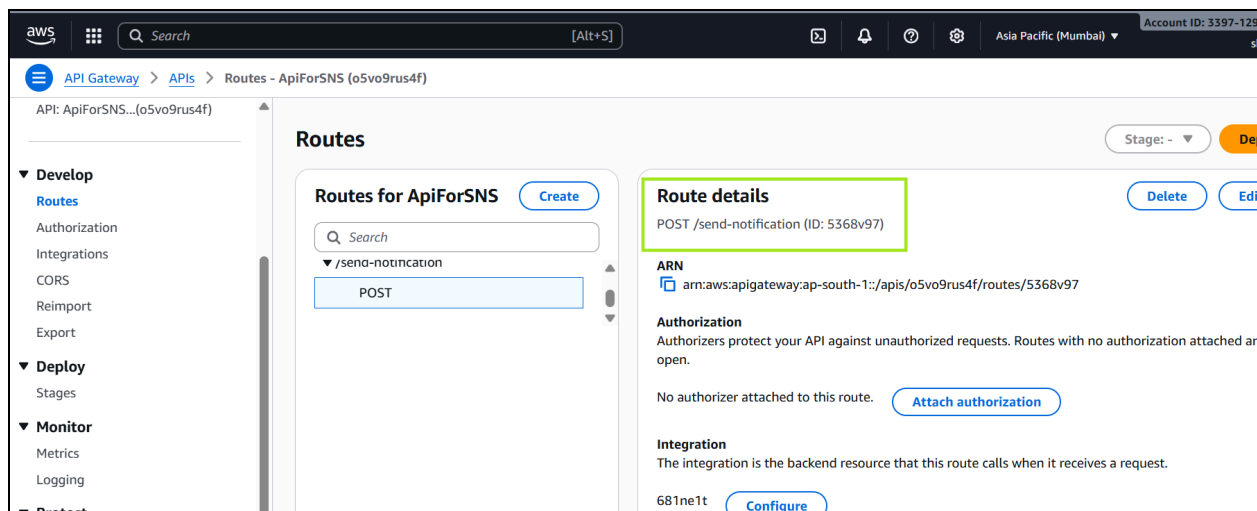
- Now navigate to AWS API Gateway console, specifically to '[ApiForSNS](#)' API.



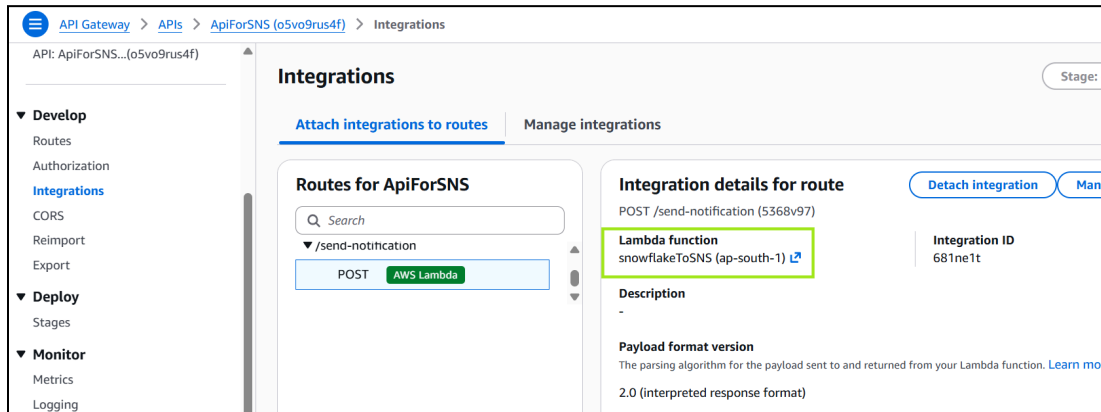
Go to the “Stages” section, stage is where the base url is present.



Then in the routes section, further url and API method (POST) is defined.



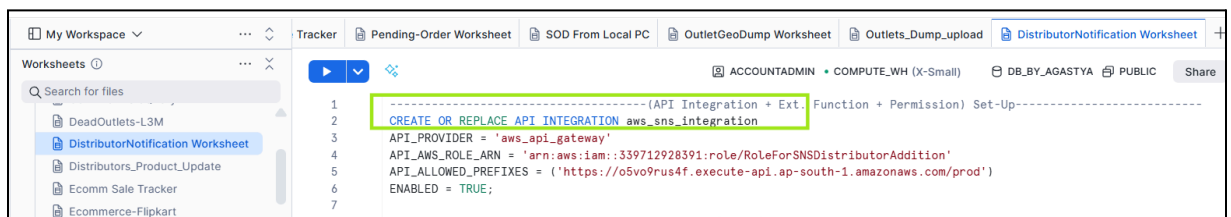
After routes, integration is attached to the route, which in this case is Lambda integration (the 'snowflakeToSNS' lambda function in above steps). When a request is made to the complete url of this API (<https://o5vo9rus4f.execute-api.ap-south-1.amazonaws.com/prod/send-notification>), it will call the attached lambda function.



AWS Set-up is complete, now snowflake needs to be configured to be able to call 'snowflakeToSNS' lambda function so that it can forward the message from snowflake to subscribed users of SNS Topic.

Snowflake Set-up:

- API Integration is created with API provider as *aws_api_gateway* and base url of our API as allowed prefix. Follow this [documentation](#).



- External function is then created with String type parameter and complete url of our AWS API route.

The screenshot shows the Snowflake SQL Editor interface. The left sidebar displays a list of worksheets, with 'DistributorNotification Worksheet' selected. The main editor area shows a SQL query to create or replace an external function named 'send_sns_notification'. The function takes a 'message' parameter of type 'STRING' and returns a 'VARIANT'. It is configured to use the 'aws_sns_integration' and points to the AWS API endpoint 'https://o5vo9rus4f.execute-api.ap-south-1.amazonaws.com/prod/send-notification'.

```

8 CREATE OR REPLACE EXTERNAL FUNCTION send_sns_notification(message STRING)
9 RETURNS VARIANT
10 API_INTEGRATION = aws_sns_integration
11 AS 'https://o5vo9rus4f.execute-api.ap-south-1.amazonaws.com/prod/send-notification';
12

```

- Snowflake query is written as such that it executes the above function and the query for new distributor addition is passed as parameter to the function.

The screenshot shows the Snowflake SQL Editor with a more complex query. The left sidebar shows the 'DistributorNotification Worksheet' selected. The main editor area shows a query that calls the 'send_sns_notification' function. The query is enclosed in a green box. It starts with a comment '-- 2) Distributor Update:'. It then defines a CTE 'sns_result' which calls 'send_sns_notification' with a concatenated string of customer data. The string is built using 'LISTAGG' to concatenate customer codes, names, and state names. The data is sourced from 'S3_DATA_NIINE.SAP_PRIMARY.SAP_PRIMARY' and joined with 'S3_DATA_NIINE.NIINE_DATA.DB_MASTER'. The query filters for distributors where 'DISTRIBUTORERPID' is NULL. Finally, it selects the 'result' from 'sns_result'.

```

80 -- 2) Distributor Update:
81
82 WITH sns_result AS (
83     SELECT send_sns_notification(
84         -- Concatenate the DISTINCT result into a single string
85         (SELECT LISTAGG(
86             customer_data.CUSTOMER_CODE || ',' || customer_data.customer_name || ',' ||
87             customer_data.customer_state_name || ',' || customer_data.GROUPNAME, ';'
88         ) WITHIN GROUP (ORDER BY customer_data.CUSTOMER_CODE) -- Order by CUSTOMER_CODE
89         FROM (
90             SELECT DISTINCT a.CUSTOMER_CODE, a.customer_name, a.customer_state_name, a.G
91             FROM S3_DATA_NIINE.SAP_PRIMARY.SAP_PRIMARY a
92             LEFT JOIN S3_DATA_NIINE.NIINE_DATA.DB_MASTER b
93                 ON a.CUSTOMER_CODE = b.DISTRIBUTORERPID
94             WHERE b.DISTRIBUTORERPID IS NULL
95             ) AS customer_data)
96         ) AS result
97 )
98 SELECT
99     result
100 FROM sns_result;

```

- Snowflake task is then created for this query to be executed at a specific schedule.

```
100
101
102 -----Task for Email Update-----
103 --
104 CREATE OR REPLACE TASK send_sns_notification_task
105 WAREHOUSE = COMPUTE_WH
106 SCHEDULE = 'USING CRON 30 7 * * * UTC'
107 -- SCHEDULE = '15 MINUTE'
108 -- SCHEDULE = '60 MINUTE'
109 -- SCHEDULE = '3 HOURS'
110
111 AS
112 WITH sns_result AS (
113     SELECT send_sns_notification(
114         -- Concatenate the DISTINCT result into a single string
115         (SELECT LISTAGG(
116             customer_data.CUSTOMER_CODE || ',' || customer_data.customer_name || ',' ||
117             customer_data.customer_state_name || ',' || customer_data.GROUPNAME, ';')
118         ) WITHIN GROUP (ORDER BY customer_data.CUSTOMER_CODE) -- Order by CUSTOMER_CODE
119     FROM (
120         SELECT DISTINCT a.CUSTOMER_CODE, a.customer_name, a.customer_state_name, a.GROUPNAME
121         FROM S3_DATA_NIINE.SAP_PRIMARY.SAP_PRIMARY a
122         LEFT JOIN S3_DATA_NIINE.NIINE_DATA.DB_MASTER b
123             ON a.CUSTOMER_CODE = b.DISTRIBUTORERPID
124         WHERE b.DISTRIBUTORERPID IS NULL
125     ) AS customer_data)
126     ) AS result
127
128 SELECT
129     result
130 FROM sns_result;
```

For pipeline flow, view the “Snowflake to SNS Pipeline.drawio” diagram.

– By Agastya Singh