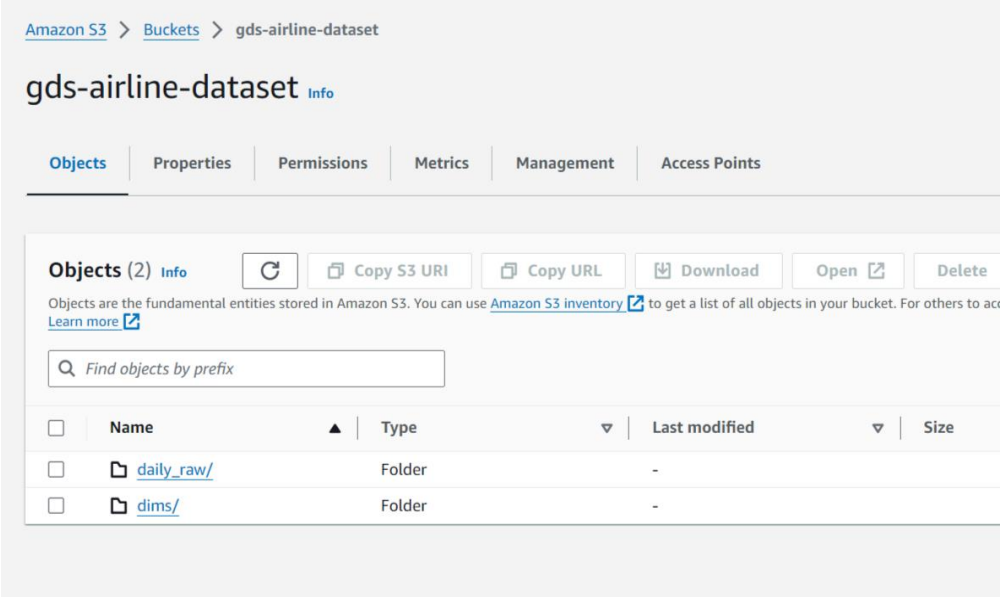
Airline Data Ingestion

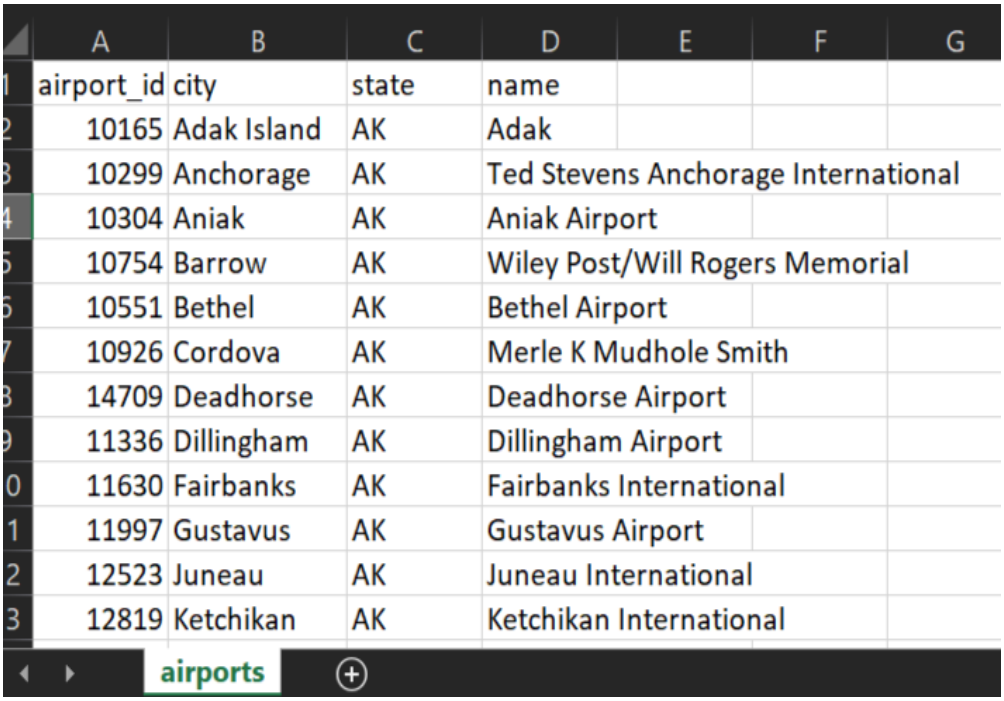
1. this project was about achieving daily incremental data load in redshift fact table(destination table) i.e as soon as S3 bucket receives any file we needed to start the process.
2. created two separate folders in s3 bucket- one for daily raw where we have flights partitioned data and other for dimension table airport data.



3. here flights data acting as fact table and airports data as dimension table. Fact table contains numerical data and primary keys for referenced dimension tables. Dimension tables contain descriptive information. Like here its containing information for each airport.

	A	B	C	D	E
	Carrier	OriginAirp	DestAirpo	DepDelay	ArrDelay
2	DL	11433	13303	-3	1
3	DL	14869	12478	0	-8
4	DL	14057	14869	-4	-15
5	DL	15016	11433	28	24
6	DL	11193	12892	-6	-11
7	DL	10397	15016	-1	-19
8	DL	15016	10397	0	-1
9	DL	10397	14869	15	24
10	DL	10397	10423	33	34
11	DL	11278	10397	323	322
12	DL	14107	13487	-7	-13
13	DL	11433	11298	22	41

flights



	A	B	C	D	E	F	G
1	airport_id	city	state	name			
2	10165	Adak Island	AK	Adak			
3	10299	Anchorage	AK	Ted Stevens Anchorage International			
4	10304	Aniak	AK	Aniak Airport			
5	10754	Barrow	AK	Wiley Post/Will Rogers Memorial			
6	10551	Bethel	AK	Bethel Airport			
7	10926	Cordova	AK	Merle K Mudhole Smith			
8	14709	Deadhorse	AK	Deadhorse Airport			
9	11336	Dillingham	AK	Dillingham Airport			
10	11630	Fairbanks	AK	Fairbanks International			
11	11997	Gustavus	AK	Gustavus Airport			
12	12523	Juneau	AK	Juneau International			
13	12819	Ketchikan	AK	Ketchikan International			

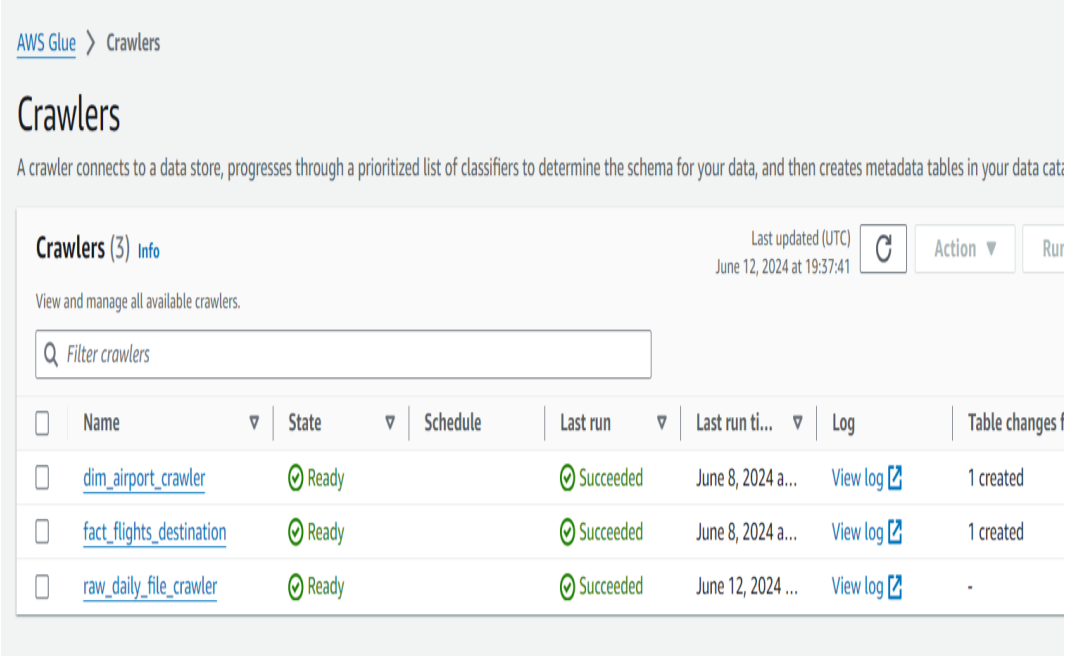
airports

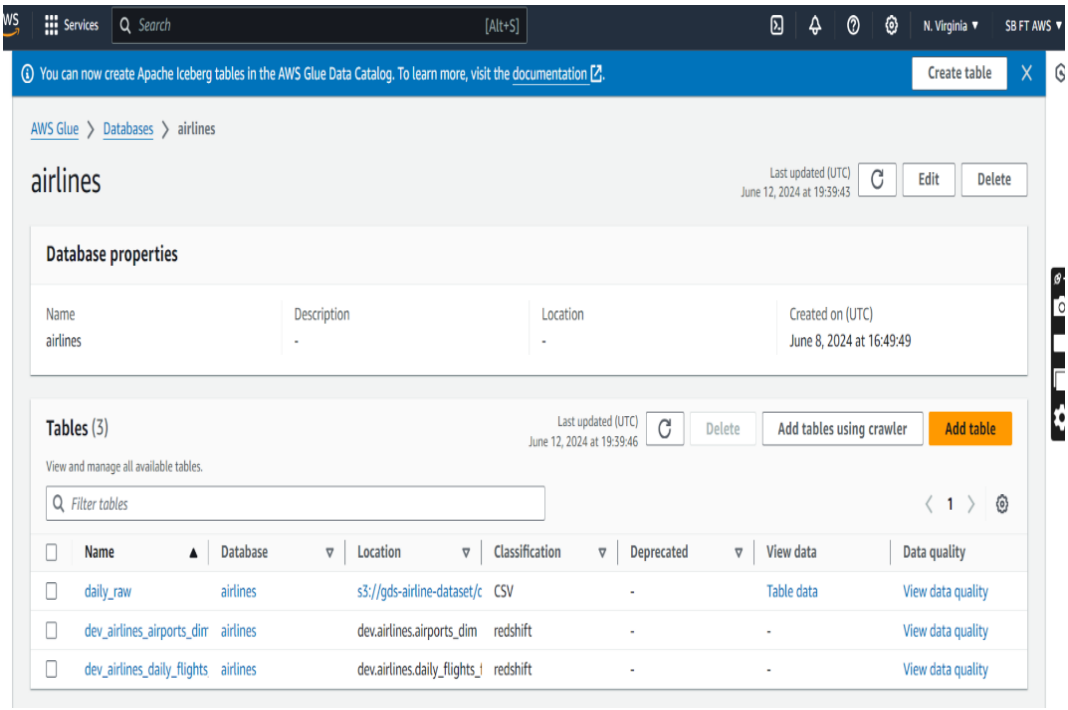
4. On the redshift warehouse, we created 2 tables. One is redshift dimension table 'airports_dim' where we load data from s3 bucket dimension data 'dims' folder. Other is redshift fact table 'daily_flights_fact' as destination table.

```
CREATE TABLE airlines.airports_dim (
    airport_id BIGINT,
    city VARCHAR(100),
    state VARCHAR(100),
    name VARCHAR(200)
);


COPY airlines.airports_dim
FROM 's3://gds-airline-dataset/dims/airports.csv'
IAM_ROLE 'arn:aws:iam::339713057891:role/redshift_role_new'
DELIMITER ','
IGNOREHEADER 1
REGION 'us-east-1';

select * from airlines.airports_dim limit 5;
```

```
CREATE TABLE airlines.daily_flights_fact (
    carrier VARCHAR(10),
    dep_airport VARCHAR(200),
    arr_airport VARCHAR(200),
    dep_city VARCHAR(100),
    arr_city VARCHAR(100),
    dep_state VARCHAR(100),
    arr_state VARCHAR(100),
    dep_delay BIGINT,
    arr_delay BIGINT
);
```

5. next we create crawlers over s3 daily raw flights data, redshift dimension table and redshift destination table which will be creating glue catalog metadata tables.



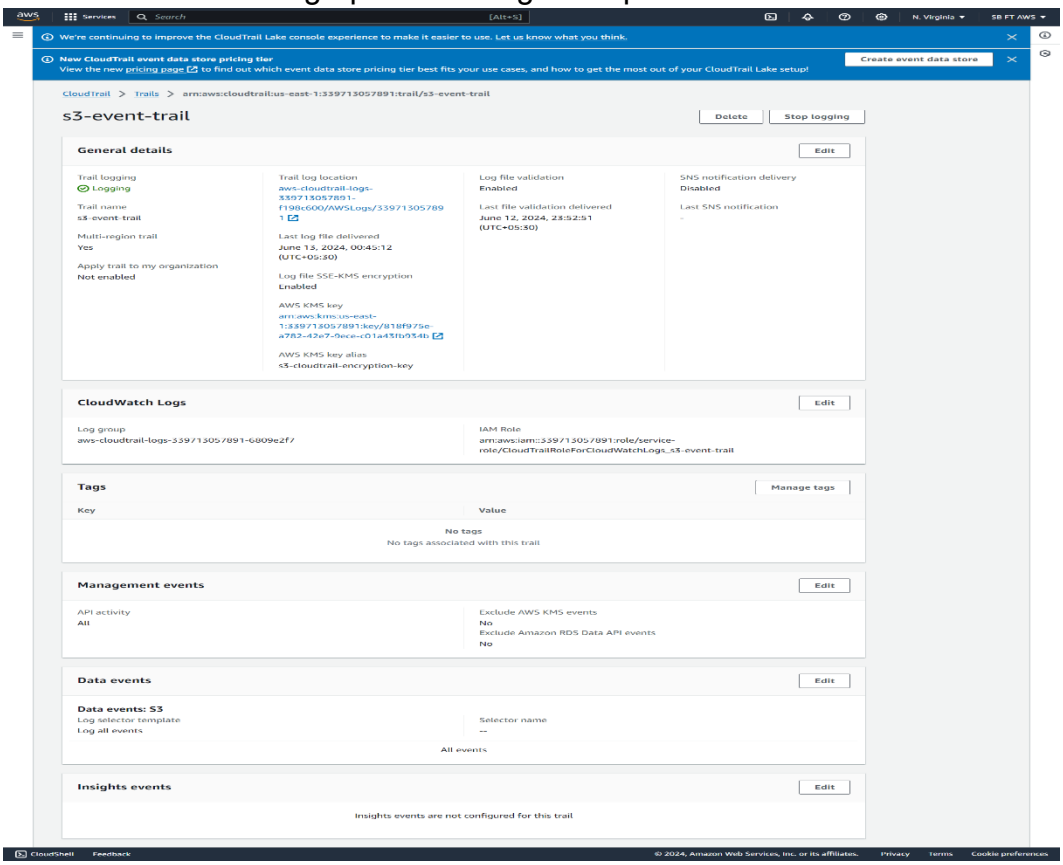


6. we used an visual ETL job 'airline_data_ingestion' where we start reading daily raw data received from glue catalog table 'daily_raw'. We also parallely reading glue catalog dimension table. In the next transformation we performed joining of these two tables and following this joining result we changed the schema same matching with destination redshift table. Once 'change schema' part is done, we wrote the final output to the redshift destination table

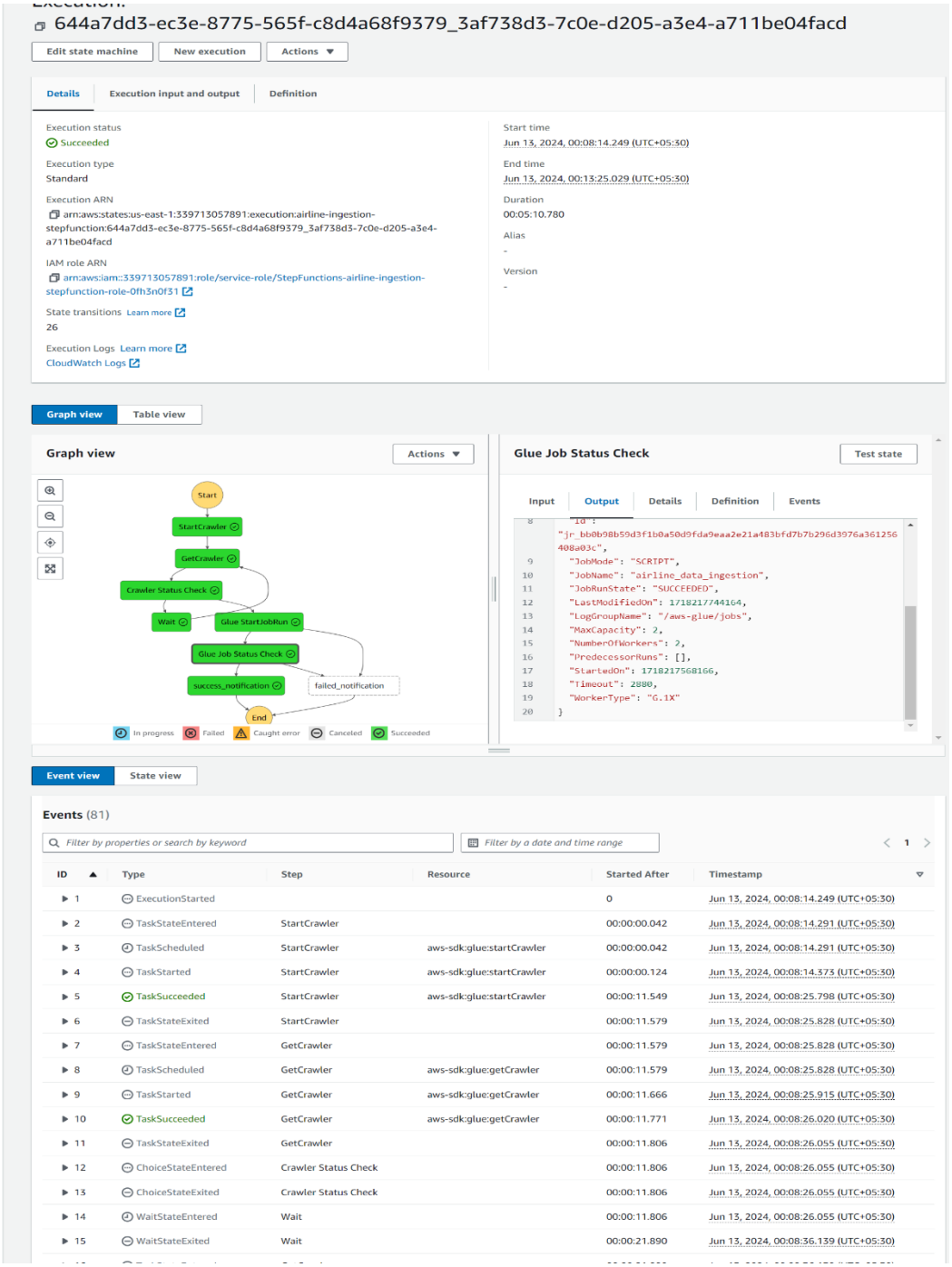
'dev_airlines_daily_flights'. We also kept 'Job Bookmark' enabled to receive only new or updated data.

```
# Script generated for node dim_airport_code_read
dim_airport_code_read_node1717951465216 = glueContext.create_dynamic_frame.from_catalog(database="airlines", table_name="dev_airlines_airports_dim",
redshift_tmp_dir="s3://gds-temp-2", transformation_ctx="dim_airport_code_read_node1717951465216")

# Script generated for node daily_raw_flight_data_from_s3
daily_raw_flight_data_from_s3_node1717951141158 = glueContext.create_dynamic_frame.from_catalog(database="airlines", table_name="daily_raw",
transformation_ctx="daily_raw_flight_data_from_s3_node1717951141158")

# Script generated for node Join
Join_node1718104899234 = Join.apply(frame1=daily_raw_flight_data_from_s3_node1717951141158, frame2=dim_airport_code_read_node1717951465216, keys1=["originairportid"],
keys2=["airport_id"], transformation_ctx="Join_node1718104899234")

# Script generated for node detp_airport_schema_changes
detp_airport_schema_changes_node1718105092428 = ApplyMapping.apply(frame=Join_node1718104899234, mappings=[("carrier", "string", "carrier", "string"), ("destairportid",
"long", "destairportid", "long"), ("depdelay", "long", "dep_delay", "bigint"), ("arrdelay", "long", "arr_delay", "bigint"), ("city", "string", "dep_city", "string"),
("name", "string", "dep_airport", "string"), ("state", "string", "dep_state", "string")], transformation_ctx="detp_airport_schema_changes_node1718105092428")

# Script generated for node Join
Join_node1718105521733 = Join.apply(frame1=detp_airport_schema_changes_node1718105092428, frame2=dim_airport_code_read_node1717951465216, keys1=["destairportid"], keys2=
["airport_id"], transformation_ctx="Join_node1718105521733")

# Script generated for node Change Schema
ChangeSchema_node1718105692873 = ApplyMapping.apply(frame=Join_node1718105521733, mappings=[("carrier", "string", "carrier", "string"), ("dep_state", "string",
"dep_state", "string"), ("state", "string", "arr_state", "string"), ("arr_delay", "bigint", "arr_delay", "long"), ("city", "string", "arr_city", "string"), ("name",
"string", "arr_airport", "string"), ("dep_city", "string", "dep_city", "string"), ("dep_delay", "bigint", "dep_delay", "long"), ("dep_airport", "string", "dep_airport",
"string")], transformation_ctx="ChangeSchema_node1718105692873")

# Script generated for node redshift_fact_table_Write
redshift_fact_table_Write_node1718105875563 = glueContext.write_dynamic_frame.from_catalog(frame=ChangeSchema_node1718105692873, database="airlines",
table_name="dev_airlines_daily_flights_fact", redshift_tmp_dir="s3://gds-temp-2",additional_options={"aws_iam_role": "arn:aws:iam::339713057891:role/
redshift_role_new"}, transformation_ctx="redshift_fact_table_Write_node1718105875563")

job.commit()
```

7. We configured CloudTrail data events to log S3 bucket API activity i.e. to get detailed records of actions taken by users, applications, or AWS services. Here S3 events getting passed to cloudtrail and we are receiving API call via Cloudtrail while setting up event bridge rule pattern.

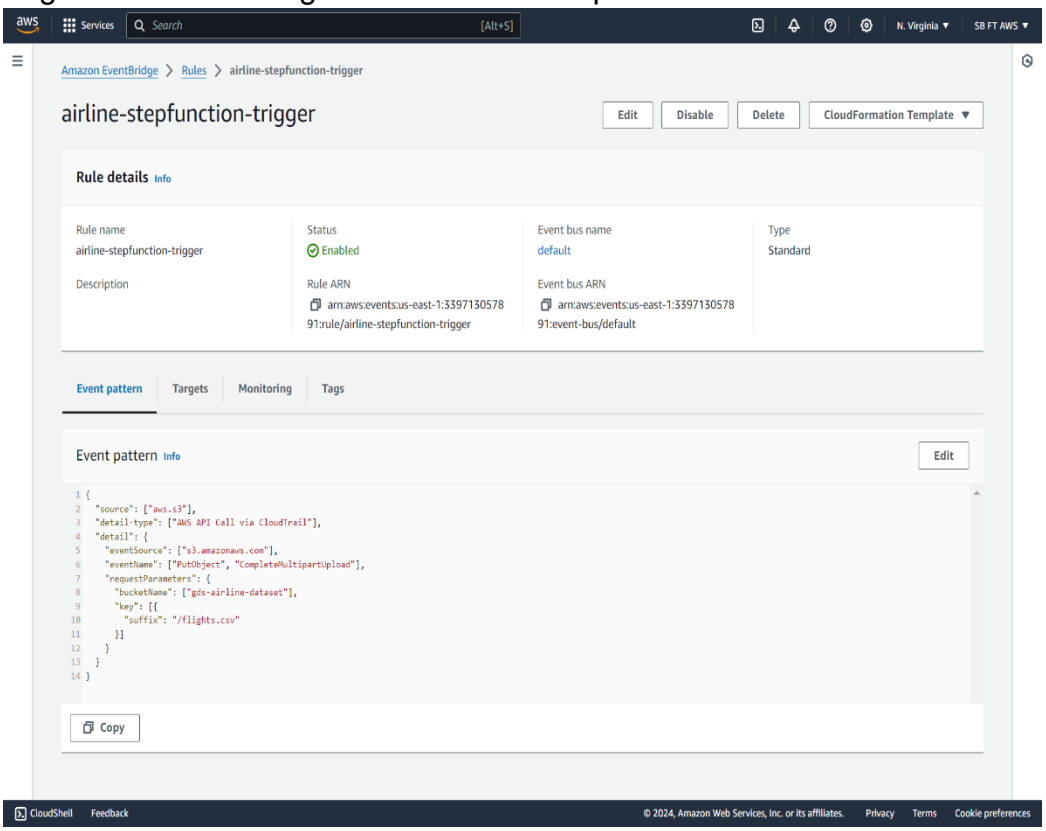


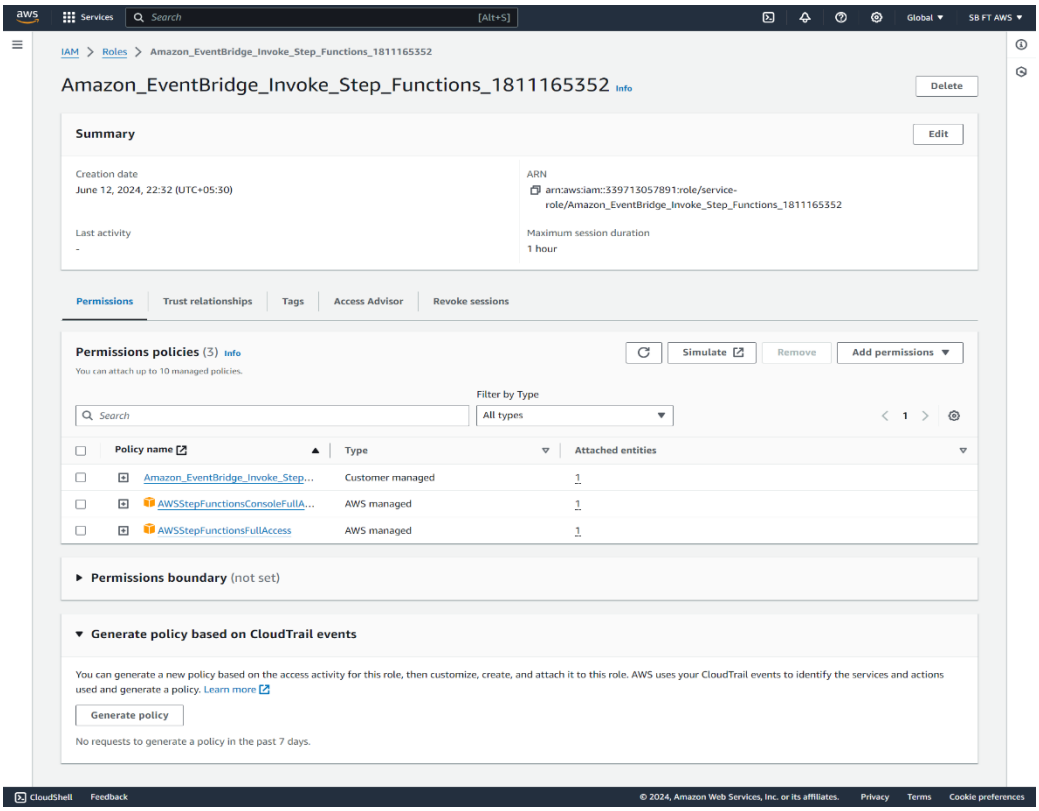
8. Further we created 'airline-ingestion-stepfunction' step function to orchestrate multiple steps in your application workflows. As workflow executes, Step Functions tracks which step is being performed and which data is passed between steps. In case of network failure or any other we were able to check that at which point it failed.



9. So following that created an event bridge rule 'airline-stepfunction-trigger' with a custom event pattern to trigger our created step function. An event pattern is defined in json format where we are passing bucket name and file name as

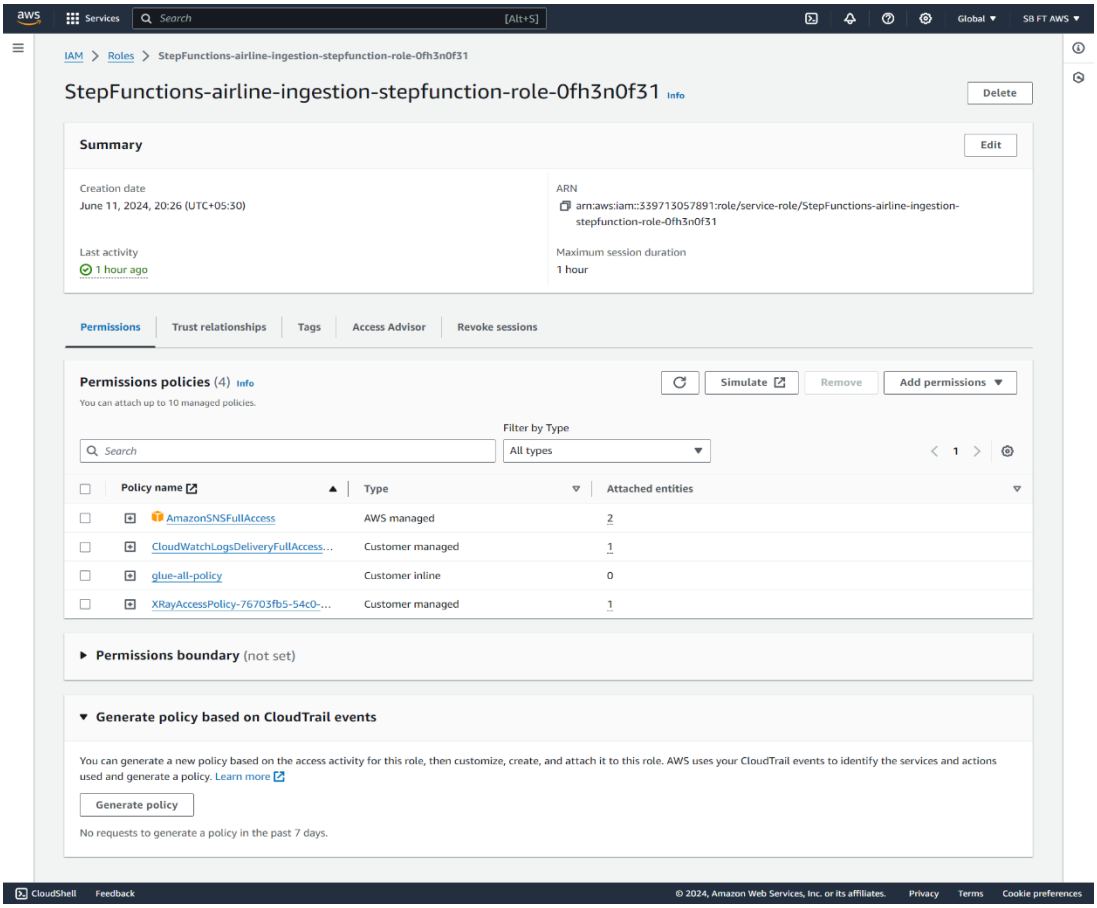
suffix in the 'requestParameters'. Next we select step function to trigger as target with event bridge role access to step function.





10. On the success step function execution, we also set SNS mail notification in the workflow their to send success alert. Hence, to perform this we set up step

function role access to 'AmazonSNSFullAccess' alongwith some other service permission too such as 'CloudwatchDeliveryFullAccess', glue all policy.



11. That's how we will be getting success notification on success ETL job execution.