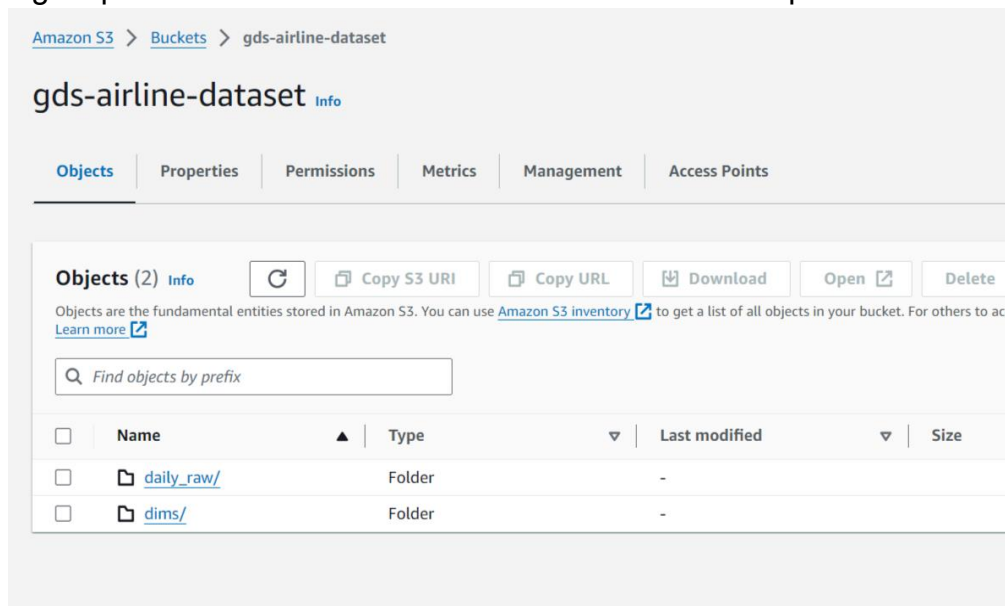


## Airline Data Ingestion

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 foreign 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
0	DL	10397	10423	33	34
1	DL	11278	10397	323	322
2	DL	14107	13487	-7	-13
3	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			
0	11630	Fairbanks	AK	Fairbanks International			
1	11997	Gustavus	AK	Gustavus Airport			
2	12523	Juneau	AK	Juneau International			
3	12819	Ketchikan	AK	Ketchikan International			

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 'dms' 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
);
```

- 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.

[AWS Glue](#) > Crawlers

## Crawlers

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

**Crawlers (3)** Info Last updated (UTC) June 12, 2024 at 19:37:41 Refresh Action Run

View and manage all available crawlers.

<input type="checkbox"/>	Name	State	Schedule	Last run	Last run time	Log	Table changes
<input type="checkbox"/>	<a href="#">dim_airport_crawler</a>	Ready		Succeeded	June 8, 2024 at 16:49:49	<a href="#">View log</a>	1 created
<input type="checkbox"/>	<a href="#">fact_flights_destination</a>	Ready		Succeeded	June 8, 2024 at 16:49:49	<a href="#">View log</a>	1 created
<input type="checkbox"/>	<a href="#">raw_daily_file_crawler</a>	Ready		Succeeded	June 12, 2024 at 19:37:41	<a href="#">View log</a>	-

[AWS](#) [Services](#)  [Alt+S] N. Virginia SB FT AWS

**airlines** Last updated (UTC) June 12, 2024 at 19:39:43 Refresh Edit Delete

**Database properties**

Name	Description	Location	Created on (UTC)
airlines	-	-	June 8, 2024 at 16:49:49

**Tables (3)** Last updated (UTC) June 12, 2024 at 19:39:46 Refresh Delete Add tables using crawler Add table

View and manage all available tables.

<input type="checkbox"/>	Name	Database	Location	Classification	Deprecated	View data	Data quality
<input type="checkbox"/>	<a href="#">daily_raw</a>	airlines	s3://gds-airline-dataset/c	CSV	-	<a href="#">Table data</a>	<a href="#">View data quality</a>
<input type="checkbox"/>	<a href="#">dev_airlines_airports_dim</a>	airlines	dev.airlines.airports_dim	redshift	-	-	<a href="#">View data quality</a>
<input type="checkbox"/>	<a href="#">dev_airlines_daily_flights</a>	airlines	dev.airlines.daily_flights_1	redshift	-	-	<a href="#">View data quality</a>

- we used a visual ETL job 'airline\_data\_ingestion' where we start reading daily raw data received from glue catalog table 'daily\_raw'. We also parallelly 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.

The screenshot displays the AWS CloudTrail console for the 's3-event-trail'. The interface is divided into several sections:

- General details:** Contains fields for Trail logging (enabled), Trail log location (aws-CloudTrail-logs-339713057891-f19bc600/AWSLogs/339713057891), Log file validation (enabled), SNS notification delivery (disabled), Trail name (s3-event-trail), Multi-region trail (yes), Apply trail to my organisation (not enabled), Last log file delivered (June 13, 2024, 00:45:12 UTC+05:30), Log file SSE-KMS encryption (enabled), AWS KMS key (arn:aws:kms:us-east-1:339713057891:key/818f975e-a783-42e7-9eca-c01a43fb954b), and AWS KMS key alias (s3-cloudtrail-encryption-key).
- CloudWatch Logs:** Shows the Log group (aws-cloudtrail-logs-339713057891-6009e2f7) and IAM Role (arn:aws:iam::339713057891:role/service-role/CloudTrailRoleForCloudWatchLogs\_s3-event-trail).
- Tags:** A table showing no tags are associated with this trail.
- Management events:** Includes API activity (All) and options to exclude AWS KMS events (No) and Amazon RDS Data API events (No).
- Data events:** Shows Data events: S3, Log selector template (Log all events), and Selector name (All events).
- Insights events:** A section indicating that insights events are not configured for this trail.

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.

Execution: 644a7dd3-ec3e-8775-565f-c8d4a68f9379\_3af738d3-7c0e-d205-a3e4-a711be04facd

Execution status: **Succeeded**

Execution type: Standard

Execution ARN: arn:aws:states:us-east-1:339713057891:execution:airline-ingestion-stepfunction:644a7dd3-ec3e-8775-565f-c8d4a68f9379\_3af738d3-7c0e-d205-a3e4-a711be04facd

IAM role ARN: arn:aws:iam::339713057891:role/service-role/StepFunctions-airline-ingestion-stepfunction-role-0fh3n0f31

State transitions: [Learn more](#)

Execution Logs: [Learn more](#)

CloudWatch Logs: [Learn more](#)

Start time: Jun 13, 2024, 00:08:14.249 (UTC+05:30)

End time: Jun 13, 2024, 00:13:25.029 (UTC+05:30)

Duration: 00:05:10.780

Alias: -

Version: -

**Graph view** | Table view

**Graph view**

Start

StartCrawler

GetCrawler

Crawler Status Check

Wait

Glue StartJobRun

Glue Job Status Check

success\_notification

failed\_notification

End

**Glue Job Status Check**

Test state

Input

Output

Details

Definition

Events

```
10: {
11:   "JobName": "airline_data_ingestion",
12:   "JobRunState": "SUCCEEDED",
13:   "LastModifiedOn": 171821744164,
14:   "LogGroupName": "/aws-glue/jobs",
15:   "MaxCapacity": 2,
16:   "NumberOfWorkers": 2,
17:   "PredecessorRuns": [],
18:   "StartedOn": 1718217568166,
19:   "Timeout": 2880,
20:   "WorkerType": "G.1X"
21: }
```

**Event view** | State view

Events (81)

Filter by properties or search by keyword

Filter by a date and time range

ID	Type	Step	Resource	Started After	Timestamp
1	ExecutionStarted			0	Jun 13, 2024, 00:08:14.249 (UTC+05:30)
2	TaskStateEntered	StartCrawler		00:00:00.042	Jun 13, 2024, 00:08:14.291 (UTC+05:30)
3	TaskScheduled	StartCrawler	aws-sdk:glue:startCrawler	00:00:00.042	Jun 13, 2024, 00:08:14.291 (UTC+05:30)
4	TaskStarted	StartCrawler	aws-sdk:glue:startCrawler	00:00:00.124	Jun 13, 2024, 00:08:14.373 (UTC+05:30)
5	TaskSucceeded	StartCrawler	aws-sdk:glue:startCrawler	00:00:11.549	Jun 13, 2024, 00:08:25.798 (UTC+05:30)
6	TaskStateExited	StartCrawler		00:00:11.579	Jun 13, 2024, 00:08:25.828 (UTC+05:30)
7	TaskStateEntered	GetCrawler		00:00:11.579	Jun 13, 2024, 00:08:25.828 (UTC+05:30)
8	TaskScheduled	GetCrawler	aws-sdk:glue:getCrawler	00:00:11.579	Jun 13, 2024, 00:08:25.828 (UTC+05:30)
9	TaskStarted	GetCrawler	aws-sdk:glue:getCrawler	00:00:11.666	Jun 13, 2024, 00:08:25.915 (UTC+05:30)
10	TaskSucceeded	GetCrawler	aws-sdk:glue:getCrawler	00:00:11.771	Jun 13, 2024, 00:08:26.020 (UTC+05:30)
11	TaskStateExited	GetCrawler		00:00:11.806	Jun 13, 2024, 00:08:26.055 (UTC+05:30)
12	ChoiceStateEntered	Crawler Status Check		00:00:11.806	Jun 13, 2024, 00:08:26.055 (UTC+05:30)
13	ChoiceStateExited	Crawler Status Check		00:00:11.806	Jun 13, 2024, 00:08:26.055 (UTC+05:30)
14	WaitStateEntered	Wait		00:00:11.806	Jun 13, 2024, 00:08:26.055 (UTC+05:30)
15	WaitStateExited	Wait		00:00:21.890	Jun 13, 2024, 00:08:36.139 (UTC+05:30)

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.

The screenshot shows the AWS EventBridge console for the 'airline-stepfunction-trigger' rule. The rule is enabled and has a status of 'Enabled'. The event bus name is 'default' and the type is 'Standard'. The rule ARN is 'arn:aws:events:us-east-1:3397130578:rule:airline-stepfunction-trigger' and the event bus ARN is 'arn:aws:events:us-east-1:3397130578:bus:default'. The event pattern is defined as follows:

```
1 {
2   "source": ["aws.s3"],
3   "detail-type": ["AWS API Call via CloudTrail"],
4   "detail": {
5     "eventSource": ["s3.amazonaws.com"],
6     "eventName": ["PutObject", "CompleteMultipartUpload"],
7     "requestParameters": {
8       "bucketName": ["gds-airline-dataset"],
9       "key": {
10        "suffix": "/flights.csv"
11      }
12    }
13  }
14 }
```

The console also shows tabs for 'Event pattern', 'Targets', 'Monitoring', and 'Tags'. The 'Event pattern' tab is selected, and the event pattern is displayed in a code editor with a 'Copy' button.

The screenshot shows the AWS IAM console for the role 'Amazon\_EventBridge\_Invoke\_Step\_Functions\_1811165352'. The role was created on June 12, 2024, at 22:32 UTC+05:30. The role ARN is 'arn:aws:iam:339713057891:role/service-role/Amazon\_EventBridge\_Invoke\_Step\_Functions\_1811165352' and the maximum session duration is 1 hour. The role has three permissions policies attached: 'Amazon\_EventBridge\_Invoke\_Step...', 'AWSStepFunctionsConsoleFullA...', and 'AWSStepFunctionsFullAccess'. The permissions boundary is not set. The console also shows tabs for 'Permissions', 'Trust relationships', 'Tags', 'Access Advisor', and 'Revoke sessions'. The 'Permissions' tab is selected, and the permissions policies are listed in a table.

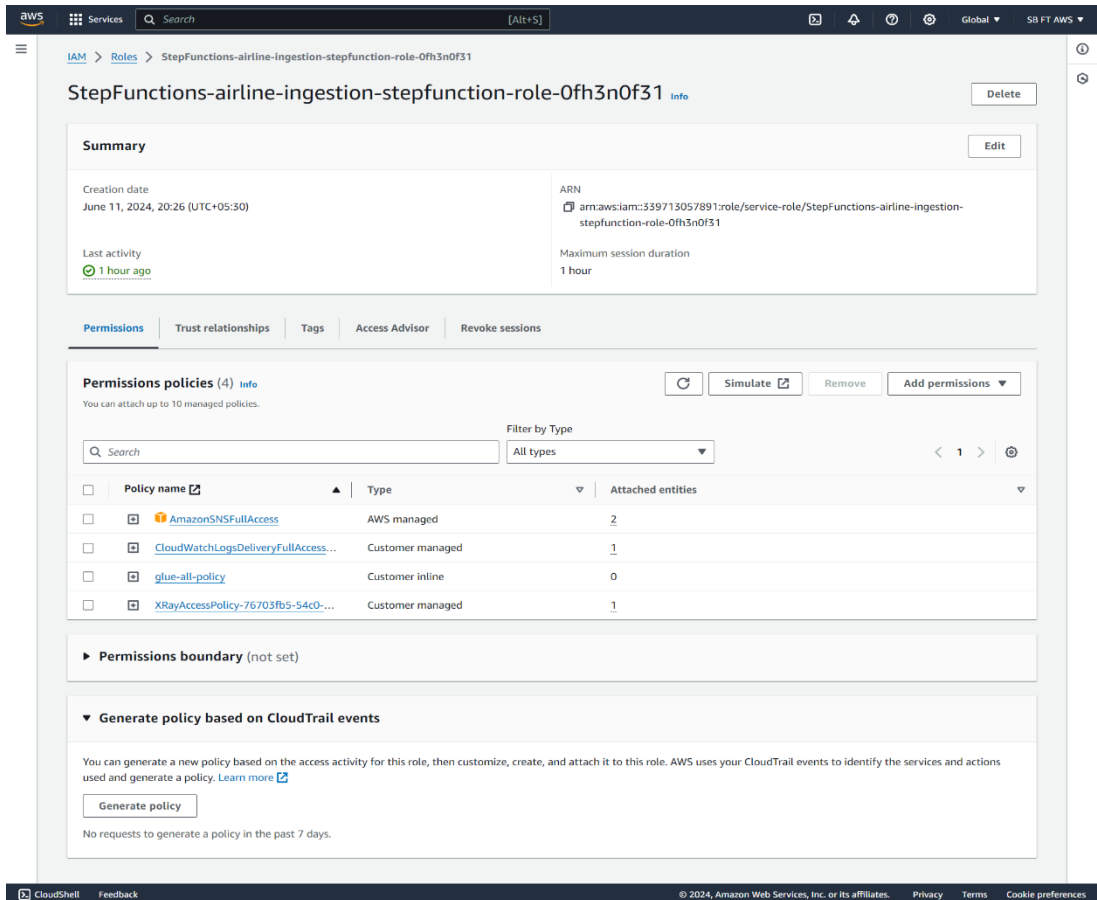
Policy name	Type	Attached entities
Amazon_EventBridge_Invoke_Step...	Customer managed	1
AWSStepFunctionsConsoleFullA...	AWS managed	1
AWSStepFunctionsFullAccess	AWS managed	1

The console also shows a section for 'Generate policy based on CloudTrail events' with a 'Generate policy' button.

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.

