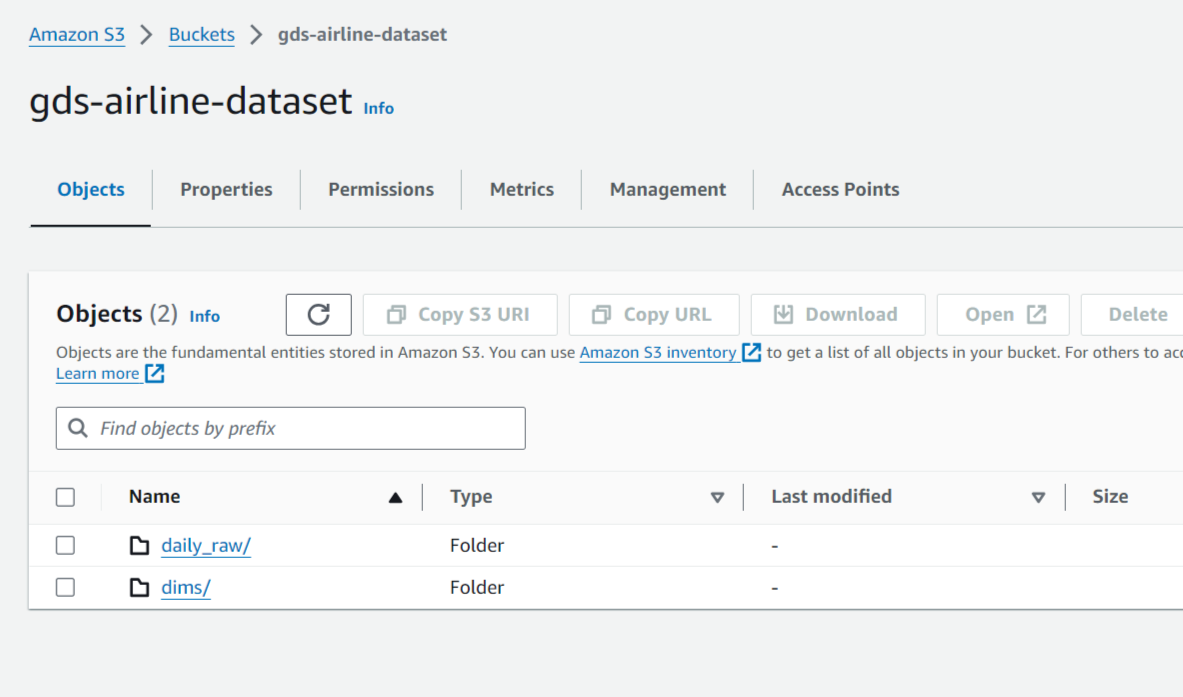
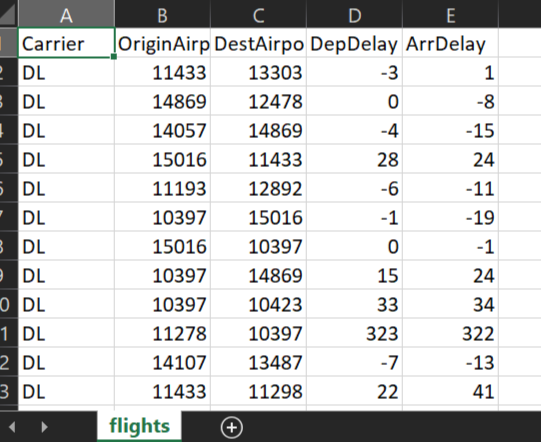
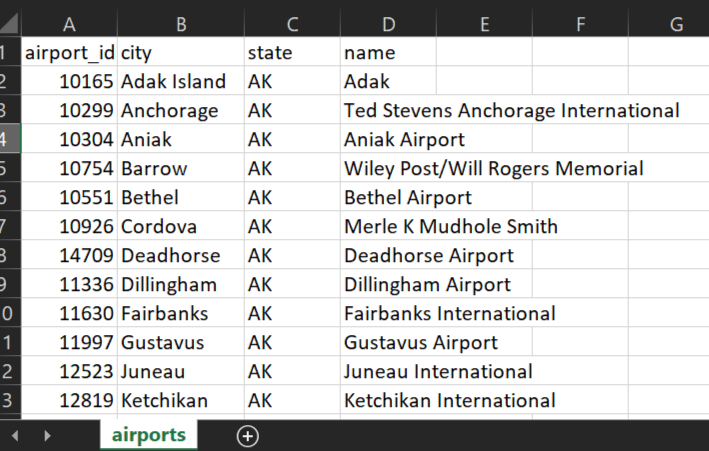
Airline Data Ingestion

1. this project was about daily incremental data load into redshift fact table(destination table).
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.

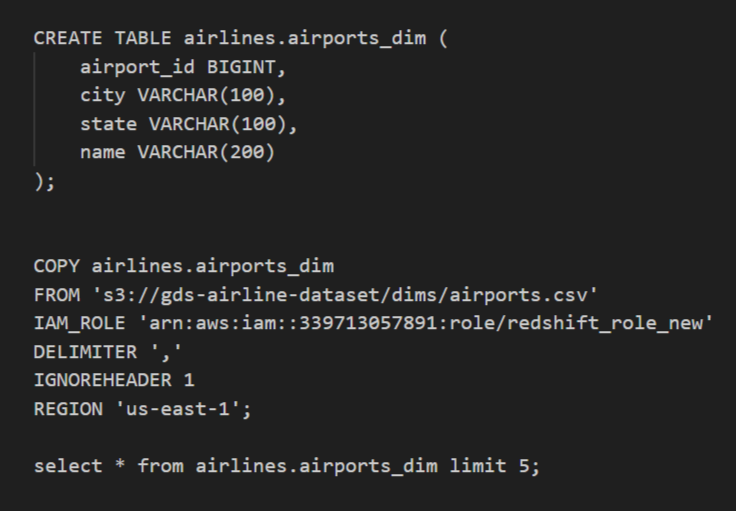


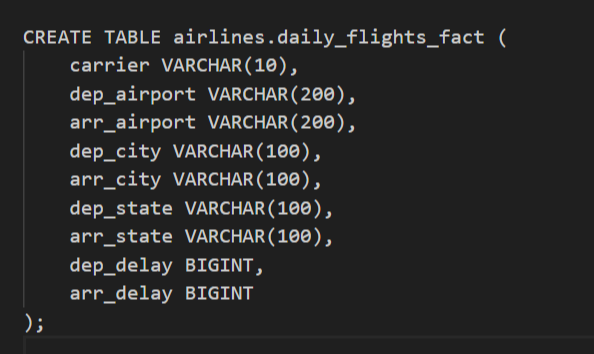
1. here flights data acting as fact table and airports data as dimension table. Fact table contains numerical data and foreign keys for referenced dimension tables. Dimension tables contain descriptive information. Like here its containing information for each airport.



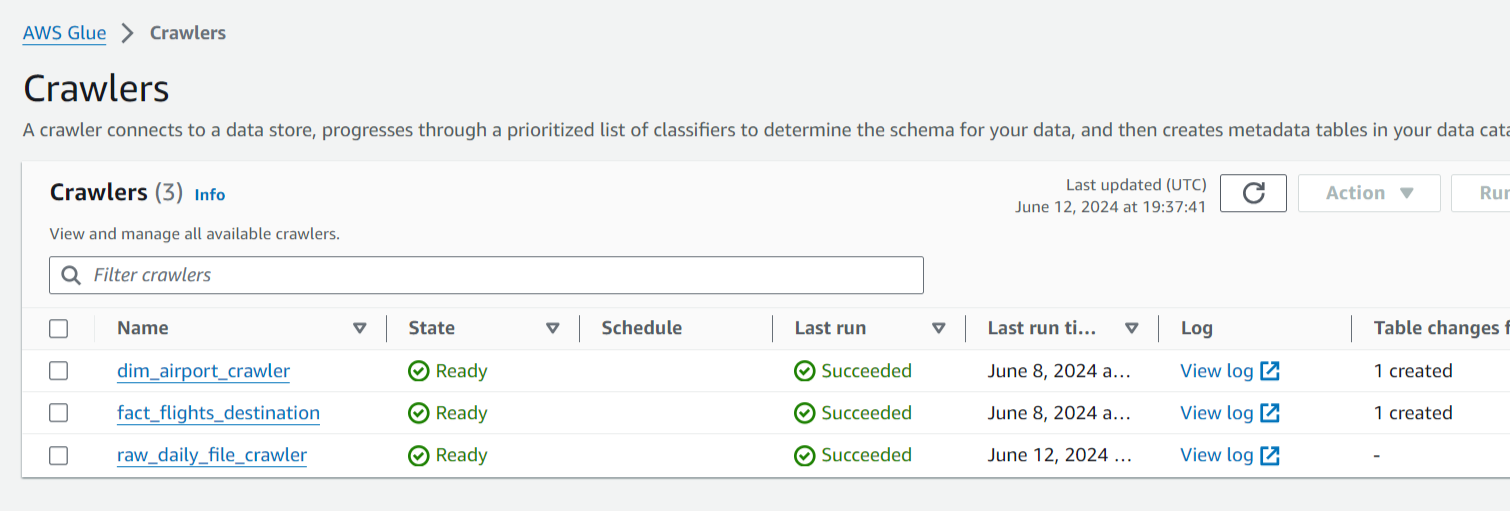


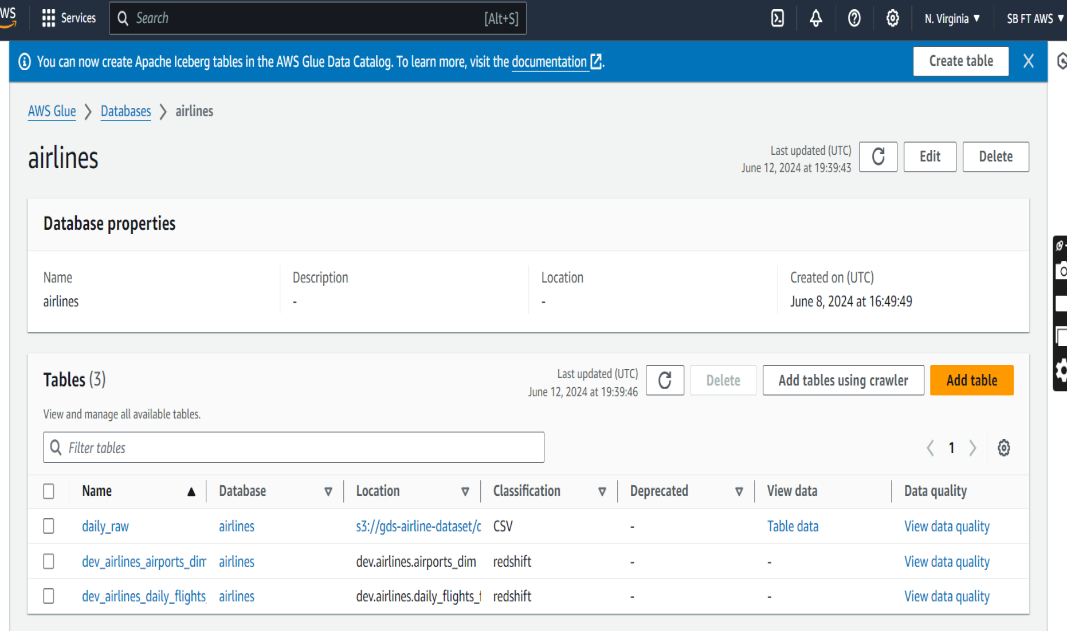
1. On the redshift warehouse, we created 2 tables. One is redshift dimension table 'airports\_dim' where we load data from s3 bucket dimension data 'dims' folder. Other is redshift fact table 'daily\_flights\_fact' as destination table.





1. 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.

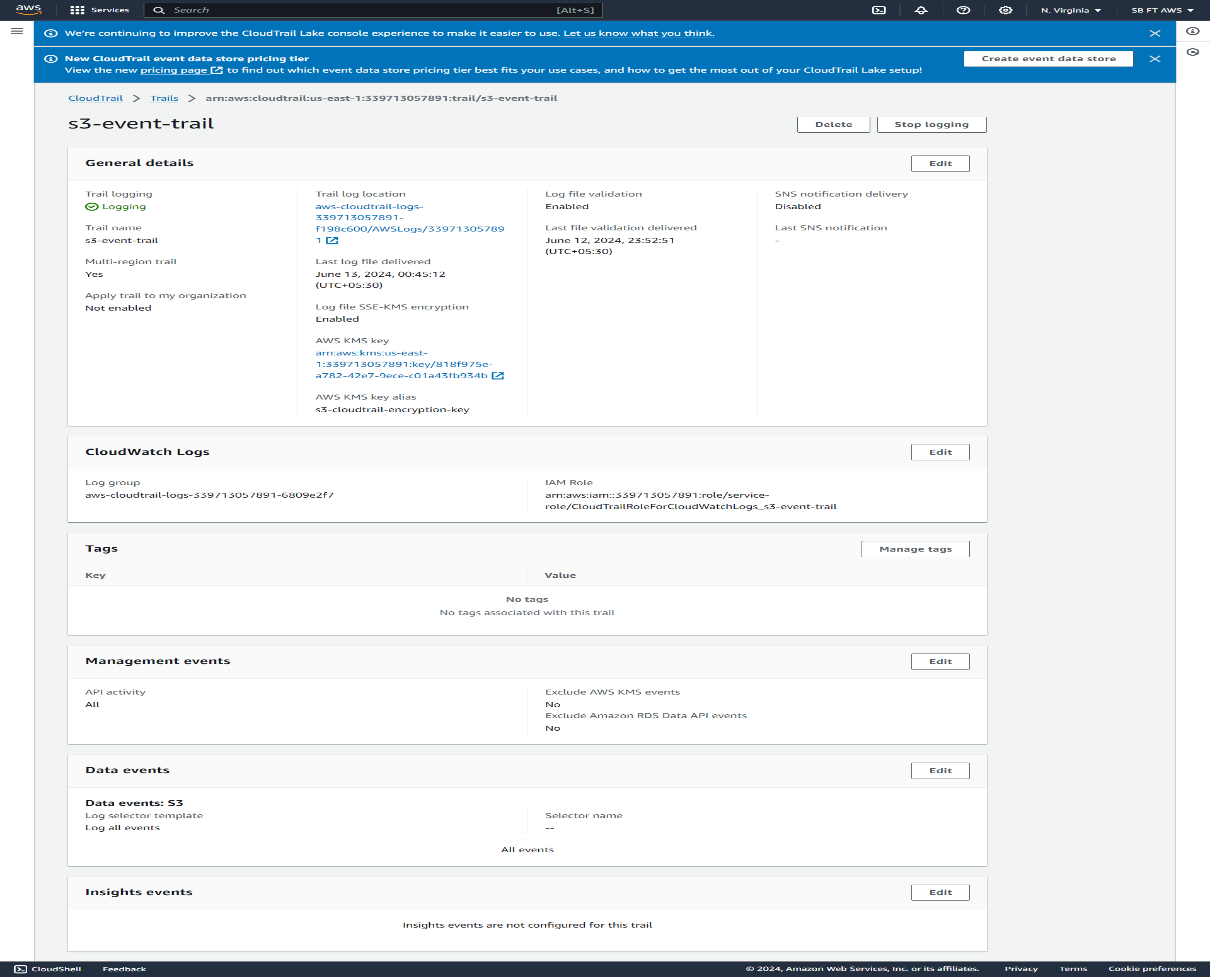




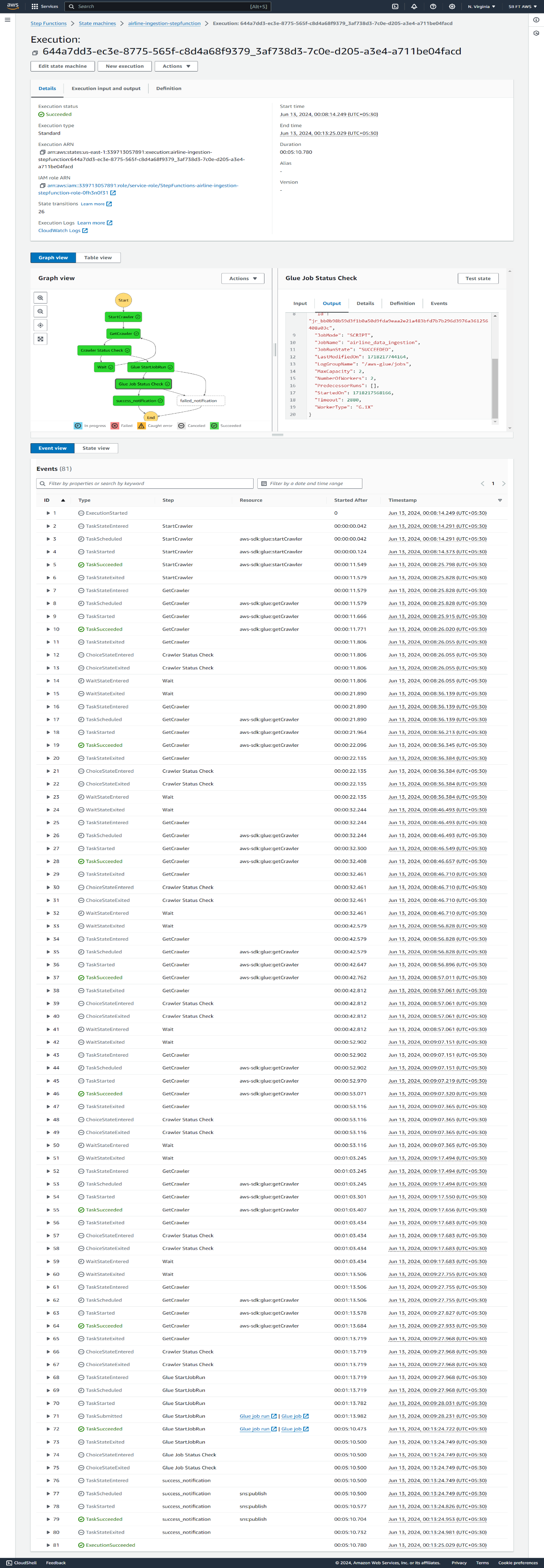
1. we used a visual ETL job 'airline\_data\_ingestion' where we start reading daily raw data received from glue catalog table 'daily\_raw'. We also parallely reading glue catalog dimension table. In the next transformation we performed joining of these two tables and following this joining result we 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.



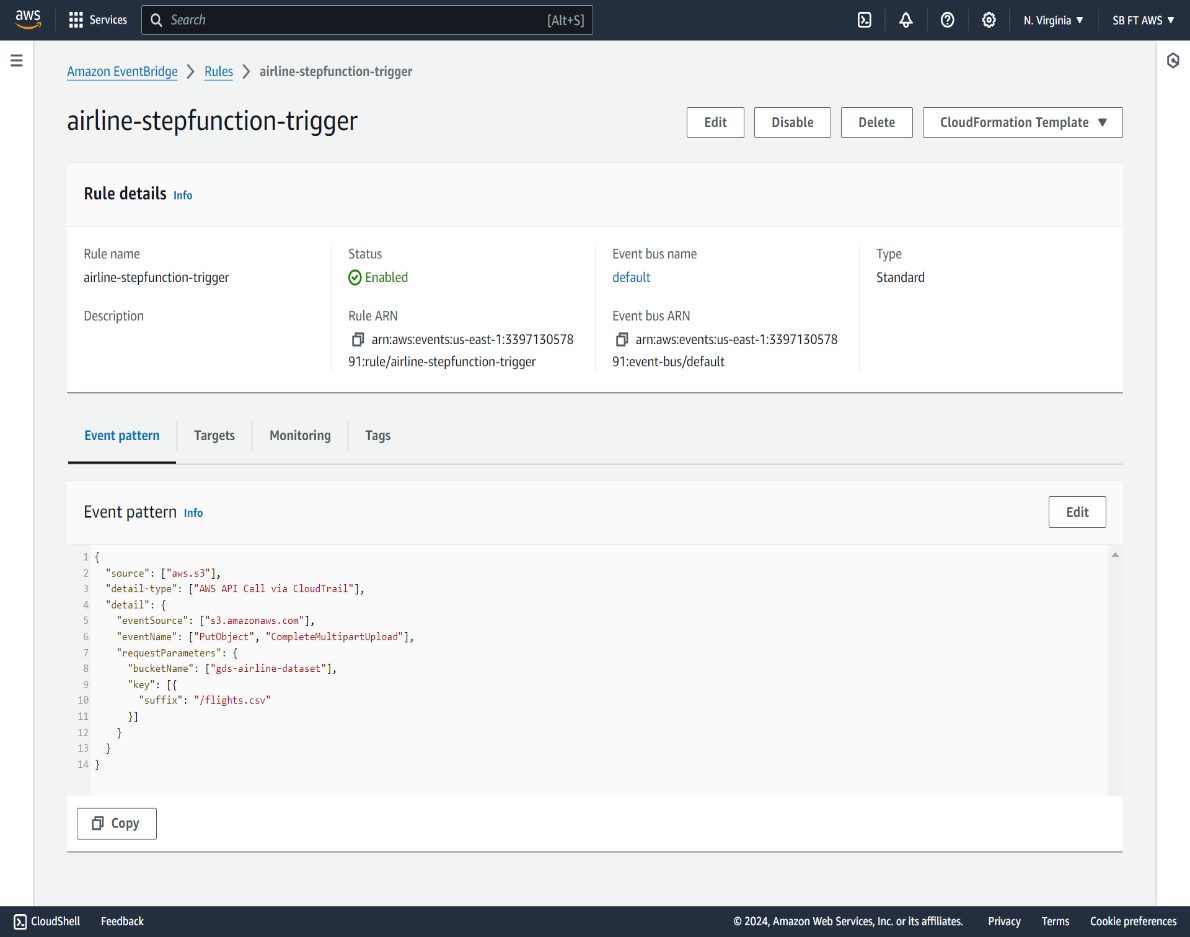
1. We configured CloudTrail data events to log S3 bucket API activity i.e. to get detailed records of actions taken by users, applications, or AWS services. Here S3 events getting passed to cloudtrail and we are receiving API call via Cloudtrail while setting up event bridge rule pattern.

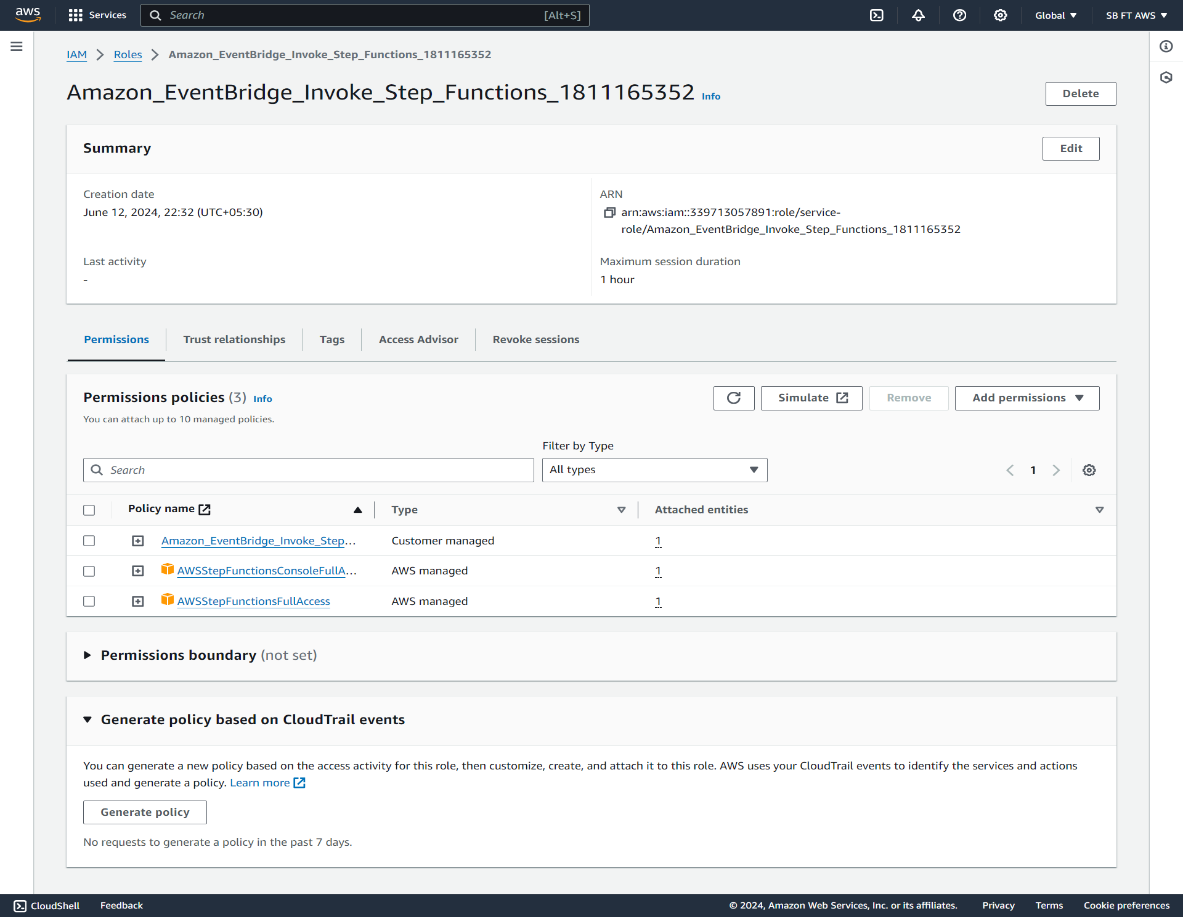


1. Further we created 'airline-ingestion-stepfunction' step function to orchestrate multiple steps in your application workflows. As workflow executes, Step Functions tracks which step is being performed and which data is passed between steps. In case of network failure or any other we were able to check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

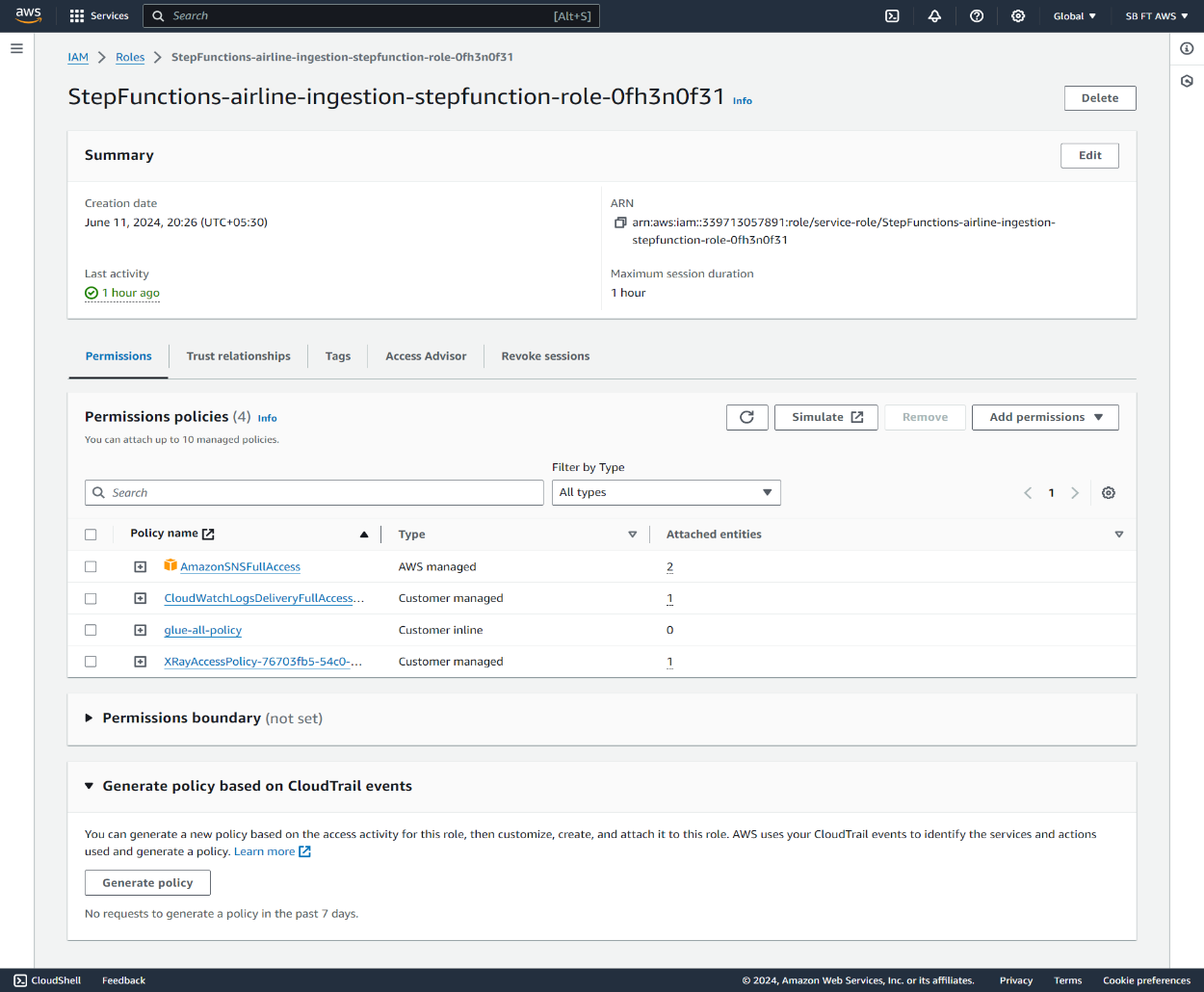


1. So following that created an event bridge rule 'airline-stepfunction-trigger' with a custom event pattern to trigger our created step function. An event pattern is defined in json format where we are passing bucket name and file name as suffix in the 'requestParameters'. Next we select step function to trigger as target with event bridge role access to step function.





1. 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.



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

