

Spotify Playlist ETL Pipeline on AWS

Objective

The main objective of this project is to design and implement a **serverless ETL pipeline** that automates the collection, transformation, and analysis of Spotify playlist data. The goal is to enable efficient storage, schema inference, and analytical querying on playlist data without manual intervention.

Specifically, the pipeline aims to:

- **Automate data ingestion** from Spotify API on a weekly basis.
- **Transform raw JSON data** into structured, query-ready formats.
- **Leverage AWS services** (Lambda, S3, Glue, Athena) for scalability and cost-efficiency.
- **Enable SQL-based analytics** on playlist data for insights into tracks, artists, and genres.

Introduction

The Spotify API, specifically the Spotify Web API, is a RESTful API that allows developers to interact with Spotify's vast data catalog and services. It provides a programmatic interface to access and manage various aspects of Spotify.

Once upon a time, a music lover's Spotify playlist grew so big that keeping track of songs, artists, and trends became overwhelming. Manual tracking took hours, and insights were nearly impossible to get. That's why we need an ETL pipeline—to automatically extract, clean, and load playlist data into reports and dashboards, turning chaos into clear insights.

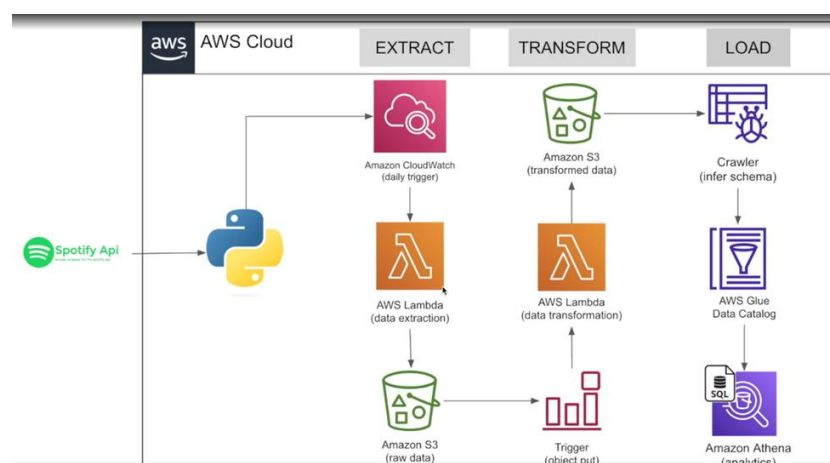
The goal of this project is to build a fully automated ETL pipeline for Spotify playlist data using AWS services. The pipeline is designed to:

- Extract playlist data from the Spotify API on a scheduled basis.
- Transform the raw JSON into structured, analysis-ready data.
- Load the transformed data into Amazon S3, with schema management in AWS Glue, for easy SQL querying through Amazon Athena.

By achieving this, the project enables seamless weekly updates, automated schema inference, and powerful analytics on Spotify playlists without manual intervention.

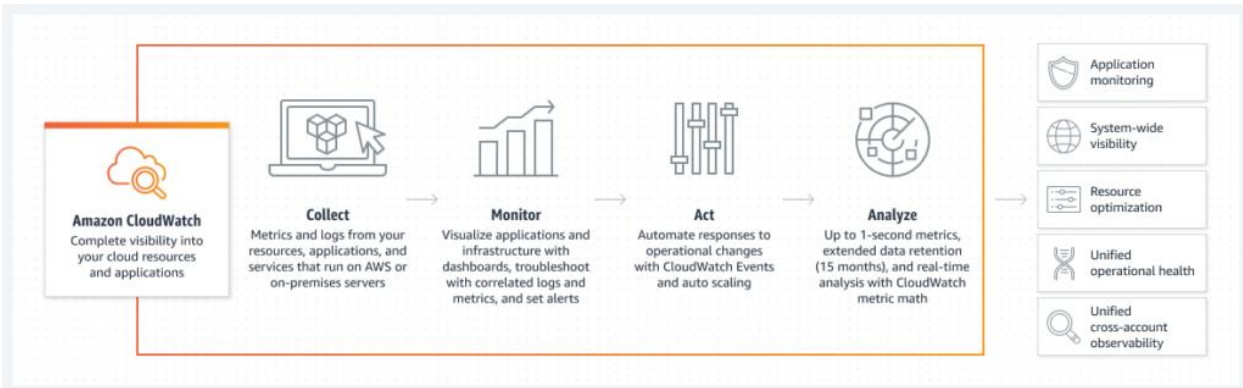
Architecture & Workflow

Architecture



Amazon CloudWatch:

Amazon CloudWatch is a monitoring and observability service that collects and tracks metrics, collects and monitors log files, and sets alarm to notify you of certain conditions. It provides a unified view of your resources and applications running on AWS and on-premises, helping you monitor performance, detect anomalies, troubleshoot issues, and automate responses. Key features include CloudWatch Logs for log analysis, CloudWatch Metrics for performance data visualization, alarms to trigger automated actions, and Container Insights for containerized applications.



AWS Lambda:

AWS Lambda is a serverless, event-driven compute service provided by Amazon Web Services (AWS). It allows users to run code without provisioning or managing servers. Users only pay for the compute time consumed when their code is executed.

Simple Storage Service (S3):

Amazon S3, or Amazon Simple Storage Service, is an object storage service provided by Amazon Web Services (AWS). It is designed to offer industry-leading scalability, data availability, security, and performance for storing and retrieving any amount of data from anywhere on the web.

AWS Crawler:

AWS Glue Crawler is a feature of AWS Glue that automatically scans data repositories like S3, DynamoDB, and on-premises systems, extracts metadata and schema information, and populates the AWS Glue Data Catalog.

Amazon Athena:

Amazon Athena is a serverless, interactive query service provided by Amazon Web Services (AWS) that enables users to analyze data directly in Amazon Simple Storage Service (Amazon S3) using standard SQL.

Workflow

Data Extraction:

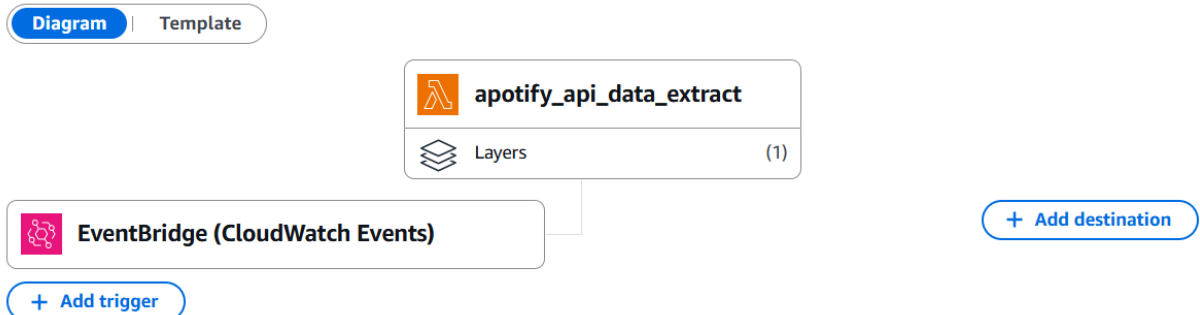
1. Amazon watch will trigger the AWS Lambda contains the data extraction code, which basically extracts data from Spotify API and then store the extracted raw data into S3 as a JSON format.

Code:

```
lambda_function.py X
lambda_function.py
1 import json
2 import os
3 import spotipy
4 from spotipy.oauth2 import SpotifyClientCredentials
5 import boto3
6 from datetime import datetime
7
8 def lambda_handler(event, context):
9
10     client_id = os.environ.get('client_id')
11     client_secret = os.environ.get('client_secret')
12
13     client_credentials_manager = SpotifyClientCredentials(client_id=client_id, client_secret=client_secret)
14     sp = spotipy.Spotify(client_credentials_manager = client_credentials_manager)
15     playlists = sp.user_playlists('spotify')
16
17     playlist_link = "https://open.spotify.com/playlist/2SM6rniz184FeyMCBSKMQB"
18     playlist_URI = playlist_link.split("/")[-1].split("?")[0]
19
20     spotify_data = sp.playlist_tracks(playlist_URI)
21
22     client = boto3.client('s3')
23
24     filename = "spotify_raw_" + str(datetime.now()) + ".json"
25
26     client.put_object(
27         Bucket="spotify-etl-thrividram",
28         Key="raw_data/to_processed/" + filename,
29         Body=json.dumps(spotify_data)
30     )
```

2. EventBridge (CloudWatch Events) will trigger the Lambda function which has extract code. This EventBridge will execute the program every week or every month. The below figure is the diagrammatic representation of the trigger connected to Data Extract

▼ Function overview [Info](#)



3. Below is the configuration of the same trigger. For testing purpose this trigger is set to trigger everyone minute but in real time this won't be necessary.



EventBridge (CloudWatch Events): [every_1_min](#)

arn:aws:events:us-east-2:124695121231:rule/every_1_min

Rule state: **ENABLED**

▼ Details



Event bus: **default**

isComplexStatement: **No**

Schedule expression: **rate(1 minute)**

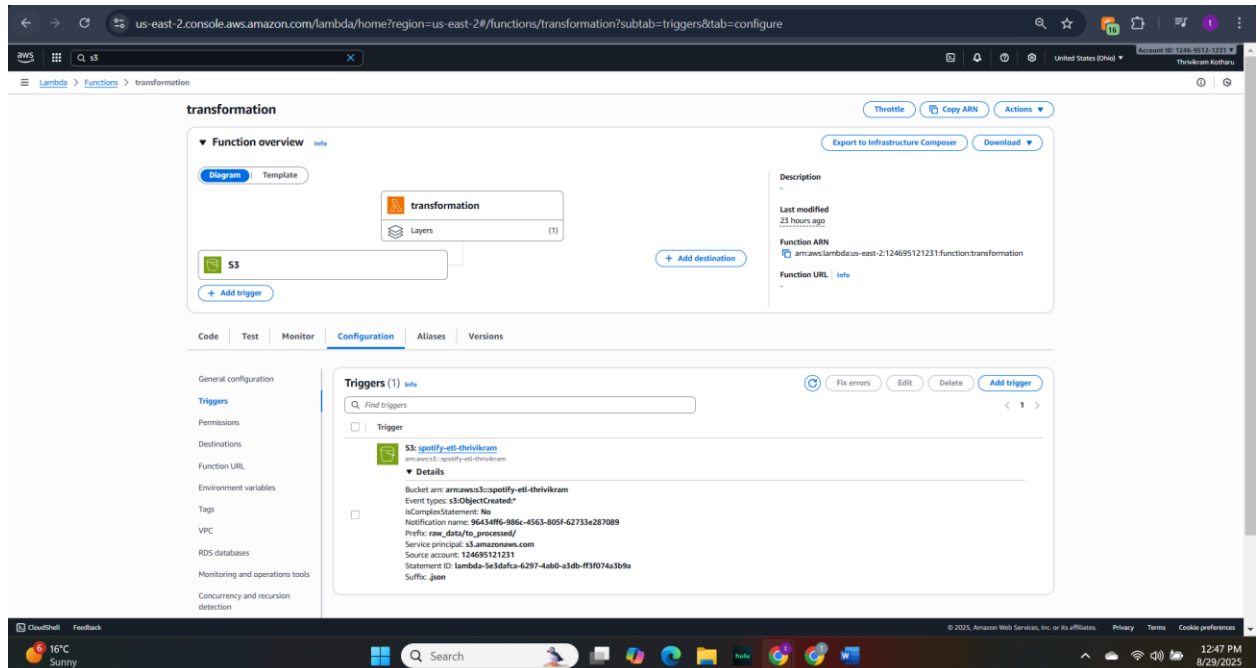
Service principal: **events.amazonaws.com**

Statement ID: **lambda-2ba2ee43-f34e-4ac8-8527-6809c4cecde6**

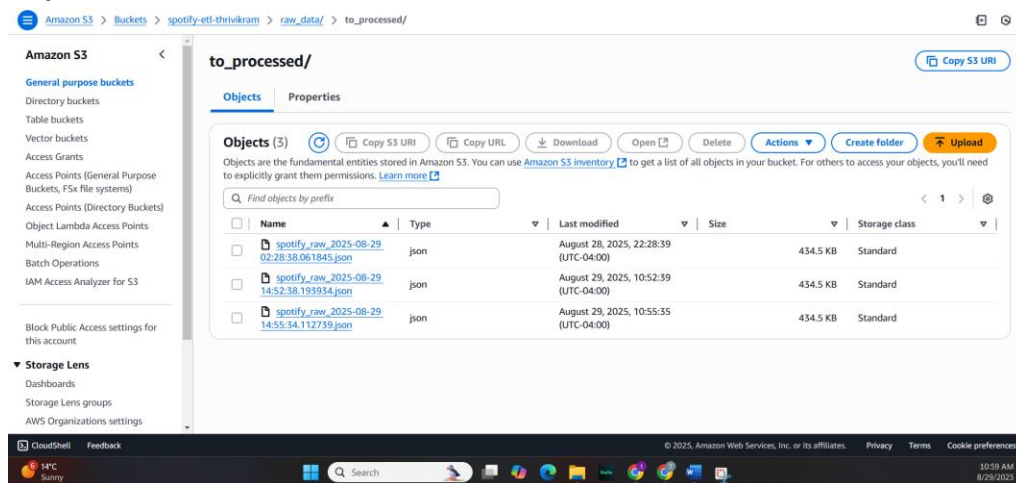
Data Transformation and Loading:

4. Now, Once this is triggered the extracted data will be staged in S3, the main reason to use it this way is because I am using trial version of lambda for this project which has very little RAM, to perform both Extract and transform together, so I am using the same S3 bucket but a different file to store both to_processed data(extracted data) and processed data(transformed data).

5. So, for transformation we will use a separate lambda function in AWS, we are going to trigger this lambda function once there is some new data or files in the to_processed folder in S3. Below is a screenshot of code and the diagrammatic representation of trigger and its configuration.



to_processed data



processed data

Amazon S3 > Buckets > spotify-etl-thrivikram > transformed_data/

Amazon S3

- General purpose buckets
- Directory buckets
- Table buckets
- Vector buckets
- Access Grants
- Access Points (General Purpose Buckets, FSx file systems)
- Access Points (Directory Buckets)
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3

Block Public Access settings for this account

▼ Storage Lens

- Dashboards
- Storage Lens groups
- AWS Organizations settings

transformed_data/

Copy S3 URI

Objects (3)

Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

Name	Type	Last modified	Size	Storage class
album_data/	Folder	-	-	-
artist_data/	Folder	-	-	-
songs_data/	Folder	-	-	-

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

11:01 AM 8/29/2025

us-east-2.console.aws.amazon.com/s3/buckets/spotify-etl-thrivikram?region=us-east-2&bucketType=general&prefix=transformed_data/album_data/&showversions=false

Search [Alt+S]

United States (Ohio) Account ID: 1246-9512-1231 Thrivikram Kotharu

Amazon S3 > Buckets > spotify-etl-thrivikram > transformed_data/ > album_data/

Amazon S3

- General purpose buckets
- Directory buckets
- Table buckets
- Vector buckets
- Access Grants
- Access Points (General Purpose Buckets, FSx file systems)
- Access Points (Directory Buckets)
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3

Block Public Access settings for this account

▼ Storage Lens

- Dashboards
- Storage Lens groups
- AWS Organizations settings

album_data/

Copy S3 URI

Objects (3)

Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

Name	Type	Last modified	Size	Storage class
album_transformed_20250828T172905Z.csv	csv	August 28, 2025, 13:29:08 (UTC-04:00)	10.4 KB	Standard
album_transformed_20250829T150247Z.csv	csv	August 29, 2025, 11:02:51 (UTC-04:00)	10.4 KB	Standard
album_transformed_20250829T155241Z.csv	csv	August 29, 2025, 11:52:44 (UTC-04:00)	10.4 KB	Standard

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

11:54 AM 8/29/2025

us-east-2.console.aws.amazon.com/s3/buckets/spotify-etl-thrividram?region=us-east-2&bucketType=general&prefix=transformed_data/artist_data/&showversions=false

Search [Alt+S]

United States (Ohio)

Account ID: 1246-9512-1231

Thrivikram Kotharu

Amazon S3

General purpose buckets

Directory buckets

Table buckets

Vector buckets

Access Grants

Access Points (General Purpose Buckets, FSx file systems)

Access Points (Directory Buckets)

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

Dashboards

Storage Lens groups

AWS Organizations settings

artist_data/

Copy S3 URI

Objects (3)

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Find objects by prefix

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	artist_transformed_20250828T172905Z.csv	csv	August 28, 2025, 13:29:08 (UTC-04:00)	11.2 KB	Standard
<input type="checkbox"/>	artist_transformed_20250829T150247Z.csv	csv	August 29, 2025, 11:02:51 (UTC-04:00)	11.2 KB	Standard
<input type="checkbox"/>	artist_transformed_20250829T155241Z.csv	csv	August 29, 2025, 11:52:44 (UTC-04:00)	11.2 KB	Standard

CloudShell

Feedback

© 2025, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

Rainy days ahead 16°C

Search

11:54 AM 8/29/2025

us-east-2.console.aws.amazon.com/s3/buckets/spotify-etl-thrividram?region=us-east-2&bucketType=general&prefix=transformed_data/songs_data/&showversions=false

Search [Alt+S]

United States (Ohio)

Account ID: 1246-9512-1231

Thrivikram Kotharu

Amazon S3

General purpose buckets

Directory buckets

Table buckets

Vector buckets

Access Grants

Access Points (General Purpose Buckets, FSx file systems)

Access Points (Directory Buckets)

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

Dashboards

Storage Lens groups

AWS Organizations settings

songs_data/

Copy S3 URI

Objects (3)

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

Create folder

Upload

Find objects by prefix

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	songs_transformed_20250828T172905Z.csv	csv	August 28, 2025, 13:29:08 (UTC-04:00)	17.5 KB	Standard
<input type="checkbox"/>	songs_transformed_20250829T150247Z.csv	csv	August 29, 2025, 11:02:51 (UTC-04:00)	17.5 KB	Standard
<input type="checkbox"/>	songs_transformed_20250829T155241Z.csv	csv	August 29, 2025, 11:52:44 (UTC-04:00)	17.5 KB	Standard

CloudShell

Feedback

© 2025, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

Rainy days ahead 16°C

Search

11:54 AM 8/29/2025

Analysis using AWS Glue, Athena:

6. After, transformation we got the files ready to perform any analysis required, one of the way we do it is by creating a AWS Glue Database(spotify_db).

The screenshot shows the AWS Glue console interface. On the left, the navigation pane is open, showing the 'Data Catalog' section with 'Databases' selected. The main content area displays the 'spotify_db' database properties. A blue banner at the top announces enhancements to Glue statistics. Below the database properties, there is a section for 'Tables (3)' with a search bar and a table listing the tables.

Database properties

Name	Description	Location	Created on (UTC)
spotify_db	-	-	July 11, 2025 at 17:02:14

Tables (3)

View and manage all available tables.

<input type="checkbox"/>	Name	Database	Location	Classification	Deprecated	View data	Data quality	Column statis...
<input type="checkbox"/>	albumalbum_data	spotify_db	s3://spotify-etl-thri	CSV	-	Table data	View data quality	View statistics
<input type="checkbox"/>	artist_data	spotify_db	s3://spotify-etl-thri	CSV	-	Table data	View data quality	View statistics
<input type="checkbox"/>	songs_data	spotify_db	s3://spotify-etl-thri	CSV	-	Table data	View data quality	View statistics

7. Once the database is created, we will use AWS Glue Crawler to extract metadata from the transformed data in the S3. The Crawler needs to be configured with the required IAM role's. Below are the screenshots of Crawler's configuration and I used different crawlers for different folders in S3.

The screenshot shows the AWS Glue console interface for the 'Crawlers' section. The left navigation pane shows 'Data Catalog' with 'Crawlers' selected. The main content area displays the 'Crawlers (3)' section with a search bar and a table listing the 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

View and manage all available crawlers.

<input type="checkbox"/>	Name	State	Schedule	Last run	Last run times...	Log	Table changes fr...
<input type="checkbox"/>	spotify_album	Ready		Succeeded	August 28, 2025 a...	View log	1 created
<input type="checkbox"/>	spotify_artist	Ready		Succeeded	August 28, 2025 a...	View log	1 created
<input type="checkbox"/>	spotify_songs_cra...	Ready		Succeeded	August 28, 2025 a...	View log	1 updated

us-east-2.console.aws.amazon.com/glue/home?region=us-east-2#/data-catalog/crawlers/view/spotify_album

AWS Glue > Crawlers > spotify_album

spotify_album Last updated (UTC) August 28, 2025 at 16:11:00 [Run crawler](#) [Edit](#) [Delete](#)

Crawler properties

Name	spotify_album	IAM role	AWSGlueServiceRole-s3-album	Database	spotify_db	State	READY
Description	-	Security configuration	-	Lake Formation configuration	-	Table prefix	-
Maximum table threshold	-						

► **Advanced settings**

[Crawler runs](#) [Schedule](#) [Data sources](#) [Classifiers](#) [Tags](#)

Crawler runs (3) [Stop run](#) [View CloudWatch logs](#) [View run details](#)

The list of crawler runs for this crawler.

Filter data

Filter by a date and time range

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPH hours	Table changes
August 28, 2025 at 21:37:12	August 28, 2025 at 21:37:58	45 s	Completed	0.132	1 table change, 0 partition changes
August 28, 2025 at 21:28:05	August 28, 2025 at 21:28:51	45 s	Completed	0.133	-
August 28, 2025 at 21:24:44	August 28, 2025 at 21:25:28	44 s	Completed	0.132	-

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

12:11 PM 8/29/2025

us-east-2.console.aws.amazon.com/glue/home?region=us-east-2#/data-catalog/crawlers/edit/spotify_album

AWS Glue > Crawlers > Edit crawler

Review and update

Step 1: Set crawler properties

Set crawler properties

Name	spotify_album	Description	-	Tags	-
------	---------------	-------------	---	------	---

Step 2: Choose data sources and classifiers

Data sources (1)

The list of data sources to be scanned by the crawler.

Type	Data source	Parameters
S3	s3://spotify-etl-thivukram/transformed_data/album_data/	Recrawl all

Step 3: Configure security settings

Configure security settings

IAM role	AWSGlueServiceRole-s3-album	Security configuration	-	Lake Formation configuration	-
----------	-----------------------------	------------------------	---	------------------------------	---

Step 4: Set output and scheduling

Set output and scheduling

Database	spotify_db	Table prefix - optional	-	Maximum table threshold - optional	-	Schedule	On demand
----------	------------	-------------------------	---	------------------------------------	---	----------	-----------

[Cancel](#) [Previous](#) [Update](#)

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

12:16 PM 8/29/2025

us-east-2.console.aws.amazon.com/glue/home?region=us-east-2#/data-catalog/crawlers/view/spotify_artist

AWS Glue > Crawlers > spotify_artist

spotify_artist Last updated (UTC) August 28, 2025 at 16:14:00 [Run crawler](#) [Edit](#) [Delete](#)

Crawler properties

Name	spotify_artist	IAM role	AWSGlueServiceRole-s3-artist	Database	spotify_db	State	READY
Description	-	Security configuration	-	Lake Formation configuration	-	Table prefix	-
Maximum table threshold	-						

► **Advanced settings**

[Crawler runs](#) [Schedule](#) [Data sources](#) [Classifiers](#) [Tags](#)

Crawler runs (3) [Stop run](#) [View CloudWatch logs](#) [View run details](#)

The list of crawler runs for this crawler.

Filter data

Filter by a date and time range

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPH hours	Table changes
August 29, 2025 at 16:12:40	August 29, 2025 at 16:13:24	44 s	Completed	0.127	-
August 29, 2025 at 16:11:32	August 29, 2025 at 16:12:17	45 s	Completed	0.130	1 table change, 0 partition changes
August 28, 2025 at 21:43:57	August 28, 2025 at 21:44:41	44 s	Completed	0.136	1 table change, 0 partition changes

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

12:14 PM 8/29/2025

us-east-2.console.aws.amazon.com/glue/home/region-us-east-2#/data-catalog/crawlers/edit/spotify_artist

AWS Glue

Getting started
ETL jobs
Visual ETL
Notebooks
Job run monitoring
Data Catalog tables
Data connections
Workflows (orchestration)
Zero-ETL integrations

Data Catalog
Databases
Tables
Stream schema registries
Schemas
Connections
Crawlers
Classifiers
Catalog settings

Data Integration and ETL
Legacy pages
What's New
Documentation
AWS Marketplace

Enable compact mode
Enable new navigation

Review and update

Step 1: Set crawler properties

Set crawler properties

Name: spotify_artist
Description: -
Tags: -

Step 2: Choose data sources and classifiers

Data sources (1)

The list of data sources to be scanned by the crawler.

Type	Data source	Parameters
S3	s3://spotify-etl-thrividam/transformed_data/artist_data/	Recrawl all

Step 3: Configure security settings

Configure security settings

IAM role: AWSGlueServiceRole-s3-artist
Security configuration: -
Lake Formation configuration: -

Step 4: Set output and scheduling

Set output and scheduling

Database: spotify_db
Table prefix - optional: -
Maximum table threshold - optional: -
Schedule: On demand

Cancel Previous Update

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

12:16 PM 8/29/2025

us-east-2.console.aws.amazon.com/glue/home/region-us-east-2#/data-catalog/crawlers/view/spotify_songs_crawler

AWS Glue

Getting started
ETL jobs
Visual ETL
Notebooks
Job run monitoring
Data Catalog tables
Data connections
Workflows (orchestration)
Zero-ETL integrations

Data Catalog
Databases
Tables
Stream schema registries
Schemas
Connections
Crawlers
Classifiers
Catalog settings

Data Integration and ETL
Legacy pages
What's New
Documentation
AWS Marketplace

Enable compact mode
Enable new navigation

spotify_songs_crawler

Last updated (UTC): August 28, 2025 at 16:15:05

Run crawler Edit Delete

Crawler properties

Name: spotify_songs_crawler
Description: -
Maximum table threshold: -

IAM role: AWSGlueServiceRole-aws-spotify-s4-glue-role
Security configuration: -

Database: spotify_db
Lake Formation configuration: -

State: READY
Table prefix: -

Advanced settings

Crawler runs (6)

The list of crawler runs for this crawler.

Filter date

Filter by a date and time range

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
August 28, 2025 at 17:29:39	August 28, 2025 at 17:30:24	45 s	Completed	0.135	1 table change, 0 partition changes
August 28, 2025 at 17:09:26	August 28, 2025 at 17:10:11	45 s	Completed	0.133	1 table change, 0 partition changes
August 28, 2025 at 16:53:59	August 28, 2025 at 16:54:43	44 s	Completed	0.128	1 table change, 0 partition changes
August 28, 2025 at 16:50:48	August 28, 2025 at 16:51:35	46 s	Completed	0.144	1 table change, 0 partition changes
July 11, 2025 at 17:04:12	July 11, 2025 at 17:04:57	45 s	Completed	0.137	-
July 11, 2025 at 17:02:50	July 11, 2025 at 17:03:35	44 s	Completed	0.140	1 table change, 0 partition changes

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

12:15 PM 8/29/2025

us-east-2.console.aws.amazon.com/glue/home/region-us-east-2#/data-catalog/crawlers/edit/spotify_songs_crawler

AWS Glue

Getting started
ETL jobs
Visual ETL
Notebooks
Job run monitoring
Data Catalog tables
Data connections
Workflows (orchestration)
Zero-ETL integrations

Data Catalog
Databases
Tables
Stream schema registries
Schemas
Connections
Crawlers
Classifiers
Catalog settings

Data Integration and ETL
Legacy pages
What's New
Documentation
AWS Marketplace

Enable compact mode
Enable new navigation

Review and update

Step 1: Set crawler properties

Set crawler properties

Name: spotify_songs_crawler
Description: -
Tags: -

Step 2: Choose data sources and classifiers

Data sources (1)

The list of data sources to be scanned by the crawler.

Type	Data source	Parameters
S3	s3://spotify-etl-thrividam/transformed_data/songs_data/	Recrawl all

Step 3: Configure security settings

Configure security settings

IAM role: AWSGlueServiceRole-aws-spotify-s4-glue-role
Security configuration: -
Lake Formation configuration: -

Step 4: Set output and scheduling

Set output and scheduling

Database: spotify_db
Table prefix - optional: -
Maximum table threshold - optional: -
Schedule: On demand

Cancel Previous Update

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

12:17 PM 8/29/2025

8. Now, that the tables are added to the database we created in the AWS glue, we can now perform different Query's to perform different analytics on this data. Below is the screenshot of the AWS Athena in action performing various data analysis.

The screenshot shows the AWS Athena Query Editor interface. On the left, the 'Data' panel shows the 'Data source' as 'AwsDataCatalog', 'Catalog' as 'None', and 'Database' as 'spotify_db'. The 'Tables and views' panel lists 'album_data', 'artist_data', and 'songs_data'. The 'Query editor' shows a SQL query: `SELECT * FROM album_data`. The 'Query results' tab shows the query was completed. The 'Results (291)' table has the following data:

#	album_id	name	release_date	total_tracks	url
1	7q2B4M5E8kq1aW8B7N	2001	1999-11-16	23	https://open.spotify.com/album/7q2B4M5E8kq1aW8B7N
2	3c4f55k8ffncbtKjKw	*Honesty	Nevermind	14	
3	6PzYUR9OCYCF7Inq4Phd	Rapalutin	2021-02-26	1	https://open.spotify.com/album/6PzYUR9OCYCF7Inq4Phd
4	2Z1gnUf3bn6DhwZ5UH54	MILLION DOLLAR BABY	2024-04-26	2	https://open.spotify.com/album/2Z1gnUf3bn6DhwZ5UH54
5	2Q5DPv9ulinOB5dNooke3	Vegas (From the Original Motion Picture Soundtrack ELVIS)	2022-05-06	1	https://open.spotify.com/album/2Q5DPv9ulinOB5dNooke3
6	1BNOKL3ZETa4WwLMmOUZ	UTOPIA	2023-07-28	19	https://open.spotify.com/album/1BNOKL3ZETa4WwLMmOUZ

The screenshot shows the AWS Athena Query Editor interface. On the left, the 'Data' panel shows the 'Data source' as 'AwsDataCatalog', 'Catalog' as 'None', and 'Database' as 'spotify_db'. The 'Tables and views' panel lists 'album_data', 'artist_data', and 'songs_data'. The 'Query editor' shows a SQL query: `SELECT * FROM "spotify_db"."album_data" limit 10;`. The 'Query results' tab shows the query was completed. The 'Results (10)' table has the following data:

#	album_id	name	release_date	total_tracks	url
1	7q2B4M5E8kq1aW8B7N	2001	1999-11-16	23	https://open.spotify.com/album/7q2B4M5E8kq1aW8B7N
2	3c4f55k8ffncbtKjKw	*Honesty	Nevermind	14	
3	6PzYUR9OCYCF7Inq4Phd	Rapalutin	2021-02-26	1	https://open.spotify.com/album/6PzYUR9OCYCF7Inq4Phd
4	2Z1gnUf3bn6DhwZ5UH54	MILLION DOLLAR BABY	2024-04-26	2	https://open.spotify.com/album/2Z1gnUf3bn6DhwZ5UH54
5	2Q5DPv9ulinOB5dNooke3	Vegas (From the Original Motion Picture Soundtrack ELVIS)	2022-05-06	1	https://open.spotify.com/album/2Q5DPv9ulinOB5dNooke3
6	1BNOKL3ZETa4WwLMmOUZ	UTOPIA	2023-07-28	19	https://open.spotify.com/album/1BNOKL3ZETa4WwLMmOUZ

us-east-2.console.aws.amazon.com/athena/home?region=us-east-2#/query-editor/history/71a091cf-5934-47a4-a8d3-5daebd503fda

Amazon Athena > Query editor

Athena now supports typeahead code suggestions to speed up SQL query development. Typeahead suggestions are turned on by default. You can change this setting in query editor preferences.

Data source: AWSDataCatalog, Catalog: None, Database: spotify_db

Tables and views: Filter tables and views

Tables (3): albumalbum_data, artist_data, songs_data

Views (0)

SQL: Ln 1, Col 1

```
SELECT * FROM "spotify_db"."artist_data" limit 10;
```

Run again Explain Cancel Clear Create

Query results: Completed

Time in queue: 91 ms Run time: 414 ms Data scanned: 22.40 KB

Results (10)

#	artist_id	artist_name	external_url
1	6DPYtp5AWVQ54RGwxdPC7	Dr. Dre	https://open.spotify.com/artist/6DPYtp5AWVQ54RGwxdPC7
2	7hJcd9fAakzCq3EaHPuG	Snoop Dogg	https://open.spotify.com/artist/7hJcd9fAakzCq3EaHPuG
3	3TVXNAsR1tunmwj47Z59r4	Drake	https://open.spotify.com/artist/3TVXNAsR1tunmwj47Z59r4
4	1ULRvnhqVAtcrqrcwq10ft	21 Savage	https://open.spotify.com/artist/1ULRvnhqVAtcrqrcwq10ft
5	6QMAbvtzvmxszLYyhgbl	Majestic	https://open.spotify.com/artist/6QMAbvtzvmxszLYyhgbl
6	54R6Y0Y7GUCvDTH21nb	Boney M.	https://open.spotify.com/artist/54R6Y0Y7GUCvDTH21nb

us-east-2.console.aws.amazon.com/athena/home?region=us-east-2#/query-editor/history/16d2965b-fd47-4ad8-b13a-def14a6be42b

Amazon Athena > Query editor

Athena now supports typeahead code suggestions to speed up SQL query development. Typeahead suggestions are turned on by default. You can change this setting in query editor preferences.

Data source: AWSDataCatalog, Catalog: None, Database: spotify_db

Tables and views: Filter tables and views

Tables (3): albumalbum_data, artist_data, songs_data

Views (0)

SQL: Ln 1, Col 1

```
SELECT * FROM "spotify_db"."songs_data" limit 10;
```

Run again Explain Cancel Clear Create

Query results: Completed

Time in queue: 91 ms Run time: 446 ms Data scanned: 17.46 KB

Results (10)

#	song_id	song_name	duration_ms	url	popularity	song_added	album_id
1	503OTs25qe7qk76rgibep	Still D.R.E.	270586	https://open.spotify.com/track/503OTs25qe7qk76rgibep	82	2022-08-29 22:05:55+00:00	7q284M5EIBkqrlaW8B7N
2	3F5CgC3wflRb51Jd8bhe	Jimmy Cooks (feat. 21 Savage)	218364	https://open.spotify.com/track/3F5CgC3wflRb51Jd8bhe	80	2022-08-29 21:38:31+00:00	3c4d55K8ffTndxKjkw
3	0b18g5G5pr42Ckz7Y6Q0Q	Raputin	186209	https://open.spotify.com/track/0b18g5G5pr42Ckz7Y6Q0Q	77	2022-08-29 21:51:10+00:00	6PcyuRk0CVCPTnqyPld
4	SAJ9hqT52wcfQCCLFRO7A	MILLION DOLLAR BABY	155151	https://open.spotify.com/track/SAJ9hqT52wcfQCCLFRO7A	83	2024-11-09 02:21:07+00:00	22TgnUf3dn6DhwZ5UH54
5	0hquQWY3xvYq4qtiquiF	Vegas (From the Original Motion Picture Soundtrack ELVIS)	182906	https://open.spotify.com/track/0hquQWY3xvYq4qtiquiF	72	2022-08-29 21:38:42+00:00	2QSDPvYulInOBSdNook3

Conclusion

This ETL pipeline successfully demonstrates how Spotify playlist data can be ingested, transformed, and analyzed in a fully automated, serverless environment.

Key achievements

- **Automated weekly data refresh** with CloudWatch triggers.
- **Scalable transformations** developed in Jupyter and deployed in Lambda.
- **Schema discovery and analytics** via Glue Data Catalog and Athena.

Future Plans

- In the next phase, this pipeline will be extended to integrate with **Snowflake**.
- Transformed S3 data will be ingested into Snowflake for **Data Warehousing**.
- Snowflake's **OLAP capabilities** will support advanced analytics, faster queries, and complex aggregations.
- This will provide a more robust foundation for **business intelligence dashboards** and reporting tools.