

Spotfiy Data Pipeline - End To End Data Engineering Project

Overview:

Implement End to End Data Pipeline using Spotify API. The project aims to integrate into a Spotify API and use AWS services to extract the messed up data, transform and load the data, to be query'd by the serverless analytics platform, AWS Athena for analysis. This pipeline should be automated in 1 day interval.

Objective:

- Integrating with Spotify API and extracting Data
- Deploying code on AWS Lambda for Data Extraction
- Adding trigger to run the extraction automatically
- Writing transformation function for the extracted data
- Building automated trigger on transformation function
- Store files on S3
- Building Analytics Tables on data files using AWS Glue and Athena

API Used:

This API contains information about music artist, albums and songs

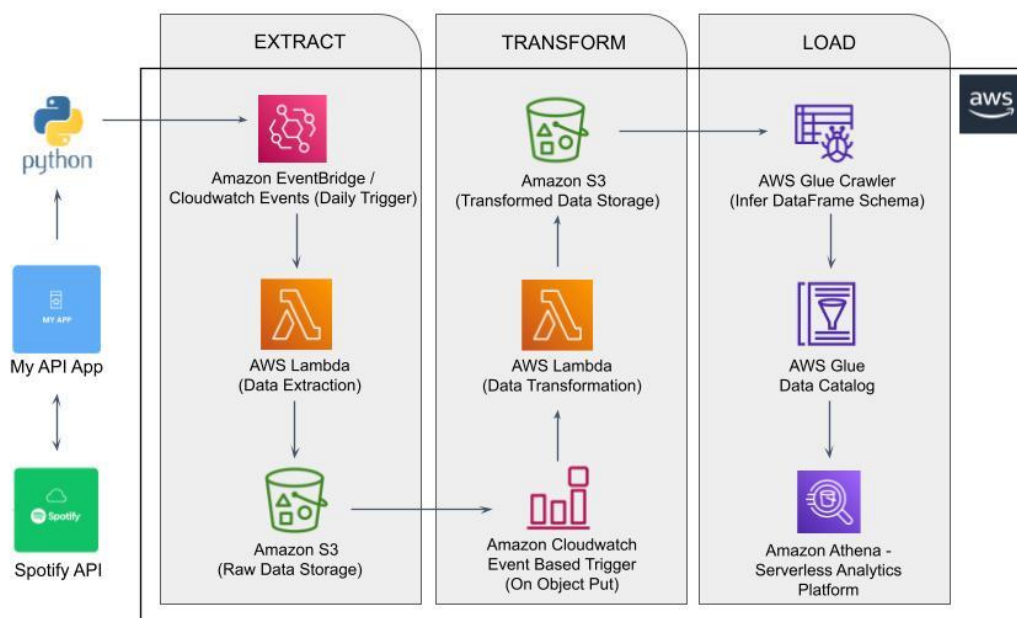
AWS Services Used:

1. Amazon S3: Amazon S3 is an object storage service that provides manufacturing scalability, data availability, security, and performance.
2. AWS Lambda: Lambda is a computing service that allows programmers to run code without creating or managing servers.
3. AWS Athena: Athena is an interactive query service for S3 in which there is no need to load data it stays in S3. Glue Crawler is a fully managed service that

automatically crawls your data sources, identifies data and infer schemas to create an Glue Data Catalog.

4. AWS Glue: A serverless data integration service that makes it easy to discover, prepare, and combine data for analytics and application development.
5. AWS IAM: This is an identity and access management service which enables us to manage access to AWS services and resources securely.
6. AWS Cloudwatch Events/Event Bridge : It helps in creating a CloudWatch Events Rule That Triggers on an Event or a particular set interval of time.

ETL Pipeline - Architectural Diagram



Requirements

Python 3
AWS Services

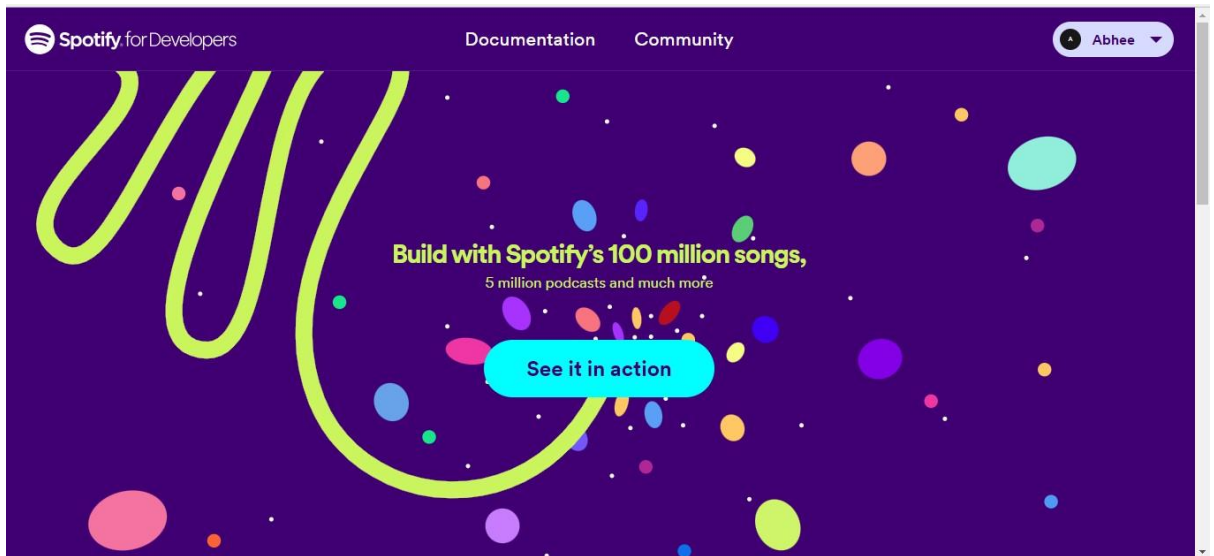
Python Packages -

```
import spotipy
import pandas
import numpy
import json
import os
import boto3
import datetime
import io.StringIO
```

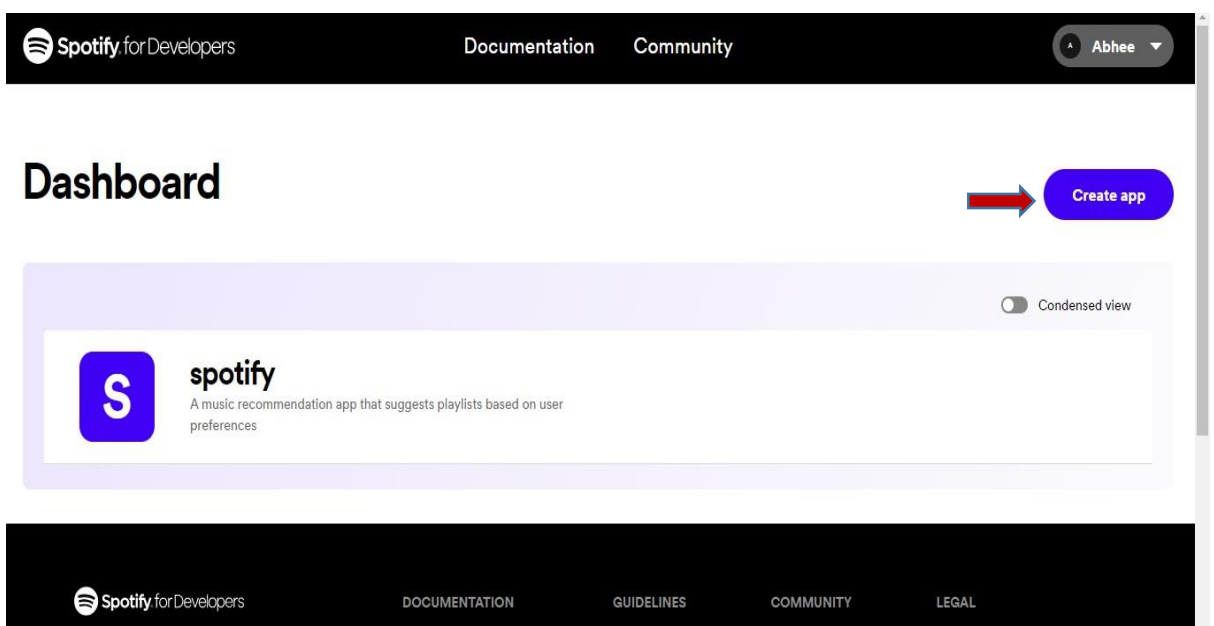
Spotify Developer API

To create a Spotify Developer API application, follow these steps:

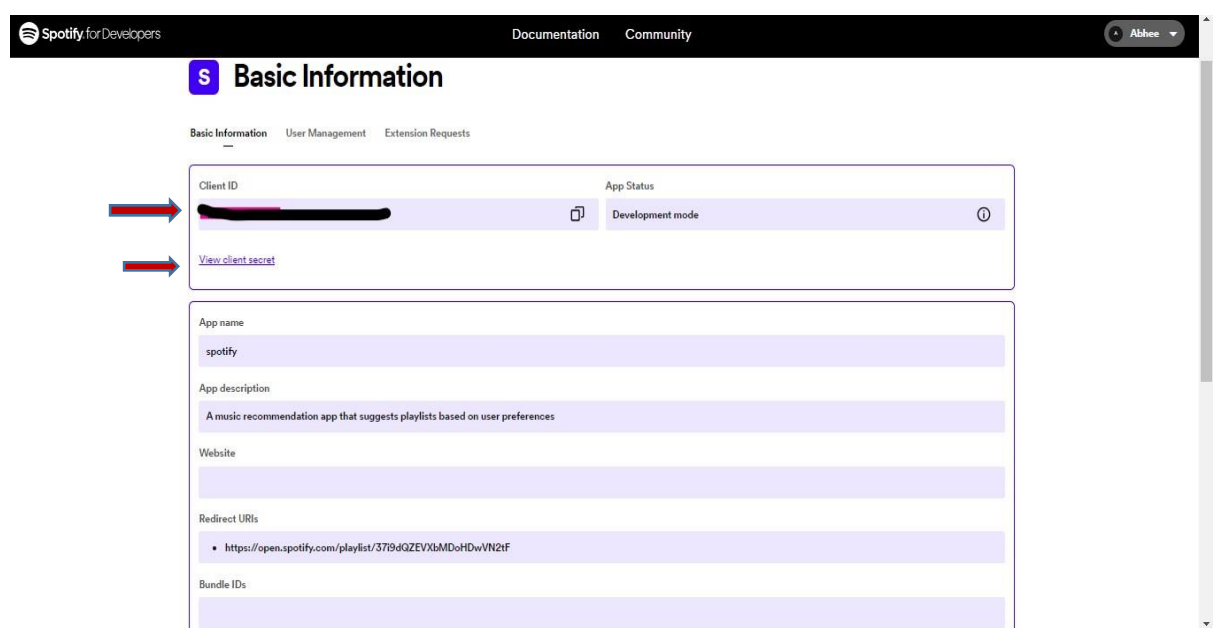
1. Visit the Spotify Developer Dashboard: Go to the Spotify Developer Dashboard at <https://developer.spotify.com/dashboard/> and log in using your Spotify account credentials. If you don't have an account, create one for free.



2. Create a new application: Once you're logged in, click on the "Create an App" button.



3. Fill in the application details: Provide the required information for your application, including the name, description, and other relevant details. It's essential to accurately describe your app's purpose, as this information will help Spotify review and approve your application.
4. Configure your application settings: After creating the application, you'll be taken to the application dashboard. Here, you can configure various settings for your app, such as adding redirect URIs, setting permissions, and managing other advanced options. Make sure to carefully set up the settings according to your application's requirements.
5. Obtain your Client ID and Client Secret: In the application dashboard, you'll find your Client ID and Client Secret. These are unique identifiers that your app will use to authenticate with the Spotify API. Keep these credentials secure and never share them publicly.
6. Set up authentication: Spotify uses the OAuth 2.0 protocol for authentication. You'll need to implement this authentication flow in your application to obtain access tokens that allow you to interact with the Spotify API on behalf of your users. Refer to Spotify's documentation for details on implementing OAuth 2.0 authentication.



7. Explore the API documentation: Familiarize yourself with the Spotify API documentation to understand the available endpoints, data models, and how to make API requests. The documentation provides examples and guidance for various functionalities, including searching for tracks, retrieving playlists, and more.
8. Develop and test your application: Start building your application using the Spotify API. You can use your preferred programming language or framework to integrate with the API and create functionality based on your app's purpose. Test your application thoroughly to ensure it functions as expected.

9. Submit your application for review: If your application intends to access user-specific data or perform certain actions, you may need to submit it for review by Spotify. This step ensures compliance with Spotify's guidelines and policies. Follow the submission process outlined in the Spotify Developer Dashboard.
10. Deploy and distribute your application: Once your application is approved, you can deploy it to a server or distribute it to your users, depending on your application's nature. Make sure to comply with Spotify's terms and guidelines when distributing your app.

Remember to regularly update and maintain your application to ensure compatibility with any changes or updates to the Spotify API.

AWS S3 (Amazon Simple Storage Service)

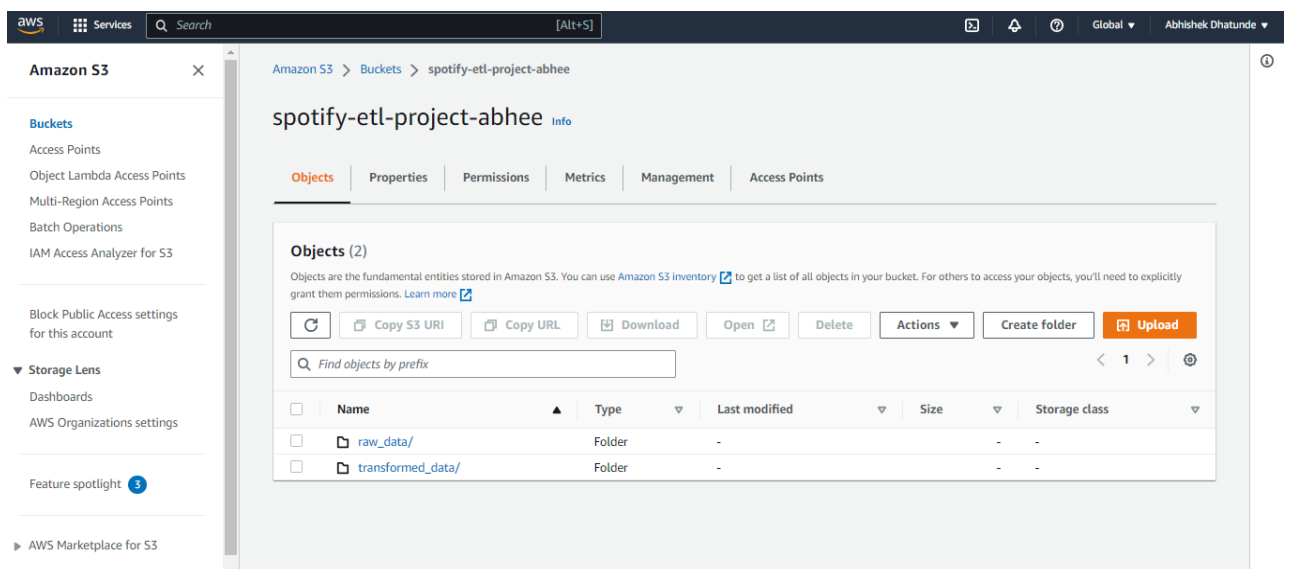
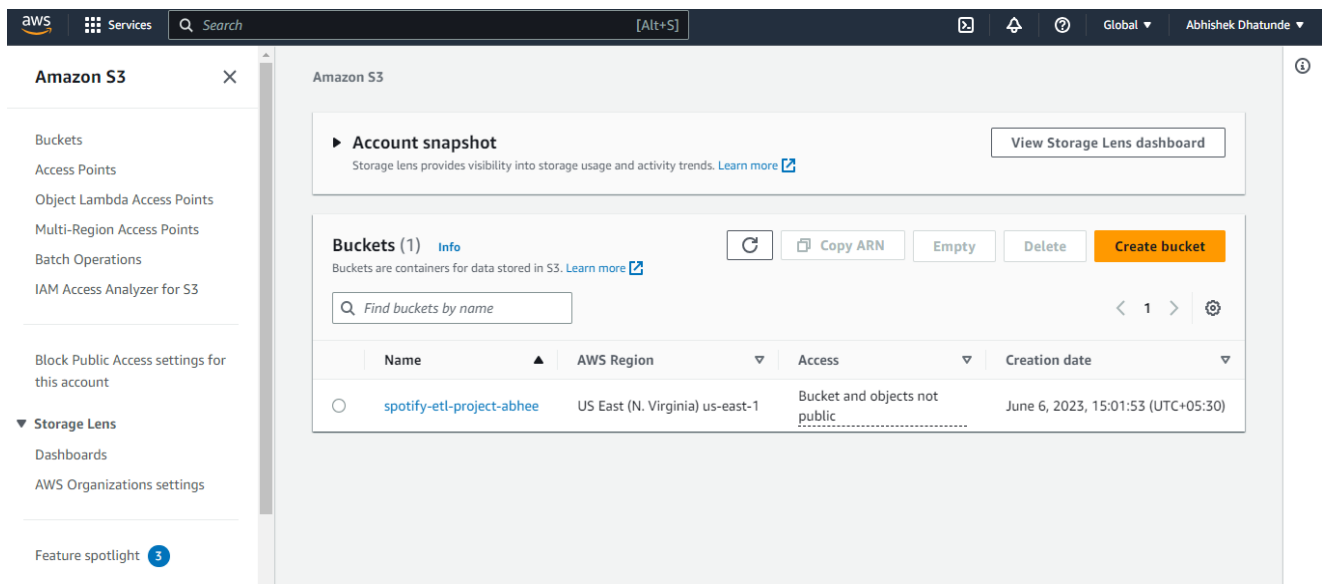
AWS S3 (Amazon Simple Storage Service) is a cloud-based object storage service provided by Amazon Web Services (AWS). It allows you to store and retrieve large amounts of data, such as files, images, videos, and backups, in a highly scalable and durable manner. Here's an overview of AWS S3 and how to get started with it:

1. **Sign up for an AWS account:** If you don't already have an AWS account, visit the AWS website (<https://aws.amazon.com/>) and sign up. You will need to provide your payment information, but AWS offers a Free Tier that includes some S3 usage for the first year.
2. **Access the AWS Management Console:** Once you have an AWS account, log in to the AWS Management Console using your account credentials.
3. **Navigate to the S3 service:** In the AWS Management Console, search for "S3" or find it under the "Storage" category. Click on the S3 service to access the S3 dashboard.
4. **Create an S3 bucket:** In the S3 dashboard, click the "Create bucket" button. Provide a unique bucket name, choose the region where you want to store your data, and configure any additional settings as per your requirements. Review the settings and create the bucket.
5. **Configure bucket permissions:** By default, your bucket is private, meaning only the bucket owner has access to its contents. If you want to grant access to other AWS accounts or make your objects publicly accessible, you can configure bucket policies and access control lists (ACLs) accordingly.
6. **Upload files to your bucket:** Once your bucket is created, you can upload files to it. In the S3 dashboard, select your bucket and click the "Upload" button. You can either upload files one by one or in batches. You can also use AWS CLI, SDKs, or other tools to upload files programmatically.
7. **Manage your files and buckets:** In the S3 dashboard, you can perform various operations on your files and buckets. These include renaming files, moving files between folders, deleting files, creating folders (known as "prefixes" in S3), and managing bucket properties and configurations.
8. **Set up permissions and access controls:** S3 provides several mechanisms to control access to your data. You can define access policies using AWS Identity and Access Management (IAM) to manage user permissions. You can also generate pre-signed URLs to provide time-limited access to specific objects.
9. **Enable versioning and logging (optional):** S3 supports versioning, which allows you to store multiple versions of an object and track changes over time. You can also enable server access logging to log detailed information about requests made to your S3 bucket.
10. **Monitor and manage your S3 usage:** AWS provides monitoring and management tools for S3, including CloudWatch metrics, S3 event notifications, and AWS S3

Analytics. These tools can help you track storage usage, monitor access patterns, and optimize your S3 infrastructure.

AWS S3 offers a wide range of features and capabilities beyond the basic steps outlined above. It's recommended to refer to the AWS documentation and explore additional features such as lifecycle policies, cross-region replication, encryption, and integration with other AWS services for more advanced use cases.

After completing these steps, you should have an S3 bucket named "spotify-etl-project-abhee" with the subfolders "raw_data" and "transformed_data" inside it. You can now upload files to these subfolders or perform other operations as needed, such as setting permissions, configuring access control, or integrating the S3 bucket with your ETL (Extract, Transform, Load) project.



Amazon S3

Buckets

Access Points

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

AWS Organizations settings

Feature spotlight 3

AWS Marketplace for S3

Amazon S3 > Buckets > spotify-etl-project-abhee > raw_data/ > to_be_processed/

to_be_processed/

Copy S3 URI

ObjectsProperties

Objects (1)

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

Refresh

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"/>	spotify_raw_2023-03-27_18_24_36.954996.json	json	June 6, 2023, 15:09:09 (UTC+05:30)	225.6 KB	Standard

Amazon S3

Buckets

Access Points

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

AWS Organizations settings

Feature spotlight 3

AWS Marketplace for S3

Amazon S3 > Buckets > spotify-etl-project-abhee > raw_data/

raw_data/

Copy S3 URI

ObjectsProperties

Objects (2)

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

Refresh

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"/>	processed/	Folder	-	-	-
<input type="checkbox"/>	to_be_processed/	Folder	-	-	-

AWS Lambda

A Lambda function, or AWS Lambda, is a serverless compute service provided by Amazon Web Services (AWS). It allows you to run your code without provisioning or managing servers. With Lambda, you can execute your code in response to events, such as changes to data in an Amazon S3 bucket, updates in a database, or incoming HTTP requests.

Key features of AWS Lambda include:

1. **Serverless Execution:** Lambda abstracts the underlying infrastructure, automatically managing the provisioning, scaling, and maintenance of servers for you. You can focus on writing code and defining triggers without worrying about server management.
2. **Event-driven Triggers:** Lambda functions are triggered by various events from AWS services or custom events you define. These triggers can be configured to execute your code in response to specific events, enabling event-driven architectures and automating workflows.
3. **Wide Language Support:** Lambda supports multiple programming languages, including Node.js, Python, Java, C#, Ruby, Go, and PowerShell. You can write your functions in the language of your choice.
4. **Pay-per-Use Billing:** With Lambda, you pay only for the actual compute time consumed by your code. You are billed based on the number of executions and the duration of each execution, measured in milliseconds.
5. **Scalability and High Availability:** Lambda automatically scales your functions in response to incoming request rates. It can handle thousands or even millions of requests concurrently. AWS takes care of the infrastructure scaling, ensuring high availability and fault tolerance.
6. **Integration with AWS Services:** Lambda seamlessly integrates with other AWS services, allowing you to build serverless architectures and create powerful workflows. For example, you can combine Lambda with Amazon S3, DynamoDB, API Gateway, SNS, SQS, and many other services to create robust and scalable applications.
7. **Easy Deployment and Management:** AWS provides a user-friendly console, SDKs, and command-line tools to deploy and manage your Lambda functions. You can version your functions, configure environment variables, monitor their execution, and set up logging and error handling.

Lambda functions are typically used for event-driven or serverless architectures, where you want to run code in response to specific events without managing server infrastructure. They are well-suited for tasks like data processing, file conversions, real-time stream processing, web and mobile backends, chatbots, and more.

AWS Lambda offers a flexible and cost-effective way to execute your code with high scalability and minimal operational overhead.

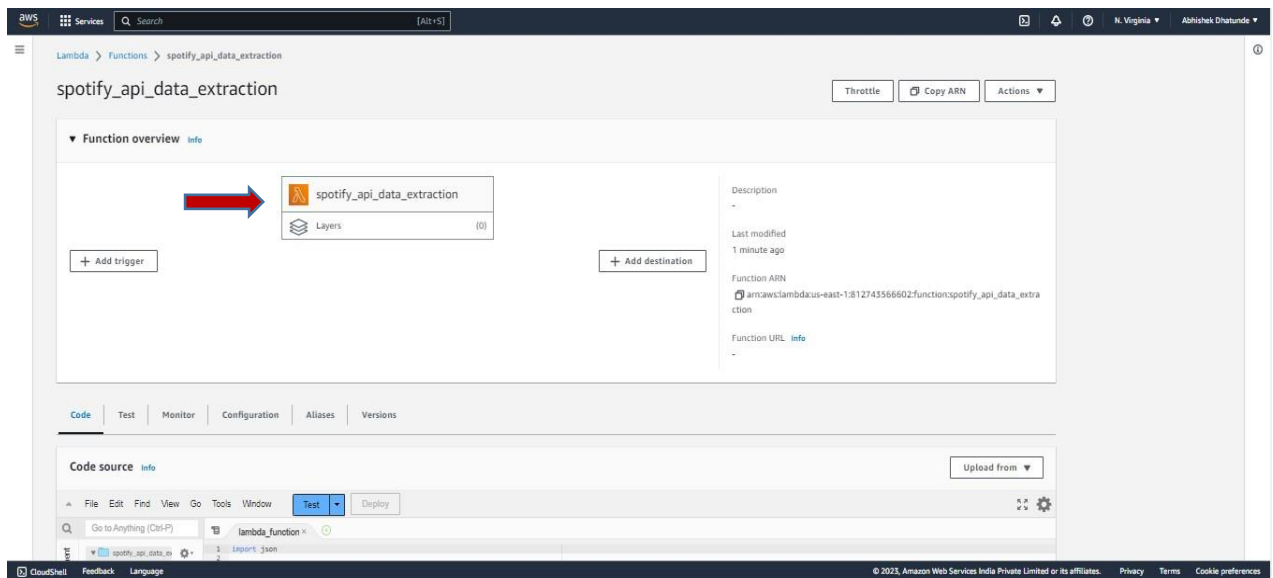
To create a Lambda function for the ETL (Extract, Transform, Load) phase of extracting Spotify data, follow these steps:

1. Create a new Lambda function: In the Lambda console, click the "Create function" button.
2. Choose the function blueprint: Select the appropriate blueprint for your ETL process. You can choose a basic blueprint or search for relevant templates such as "spotify-api-data-extraction" or "Spotify API integration" to get a head start. You can also create a function from scratch using the "Author from scratch" option.
3. Configure the function details:
 - Provide a name for your Lambda function, such as "spotify-api-data-extraction".
 - Select the runtime environment based on your programming language preference. Node.js is a commonly used runtime for Lambda functions.
 - Choose an existing or create a new execution role that grants necessary permissions to access S3 and any other AWS services required for your ETL process.
4. Write or upload the function code: In the code editor section, write your code logic to extract data from the Spotify API. You can use the Spotify API client libraries or HTTP requests to interact with the API and retrieve the desired data. Ensure you handle authentication and pagination as necessary.
5. Configure the trigger: Specify the trigger that will invoke the Lambda function to initiate the ETL process. For example, you can set up a time-based trigger using Amazon CloudWatch Events to execute the function periodically or use other event sources such as Amazon S3 or Amazon SQS.
6. Configure function environment variables: If your code requires any environment variables such as Spotify API credentials or S3 bucket details, define them in the function's environment variables section. These variables can be accessed within your Lambda function code.
7. Set up function timeout and other settings: Adjust the function timeout according to the expected duration of your ETL process. Configure other advanced settings, such as memory allocation and VPC (Virtual Private Cloud) settings, if needed.
8. Save and test your Lambda function: Once you've configured all the necessary settings, click the "Save" button. You can then test your function using sample test events or by triggering it manually, depending on your requirements. Ensure the function executes successfully and retrieves the desired Spotify data.
9. Monitor and troubleshoot: Enable logging in your Lambda function and utilize AWS CloudWatch Logs to monitor the execution logs for any errors or issues. This will help you troubleshoot and optimize your ETL process.

Remember to manage any necessary authentication, authorization, and rate limits imposed by the Spotify API. You may need to store temporary data in AWS services like Amazon S3 or

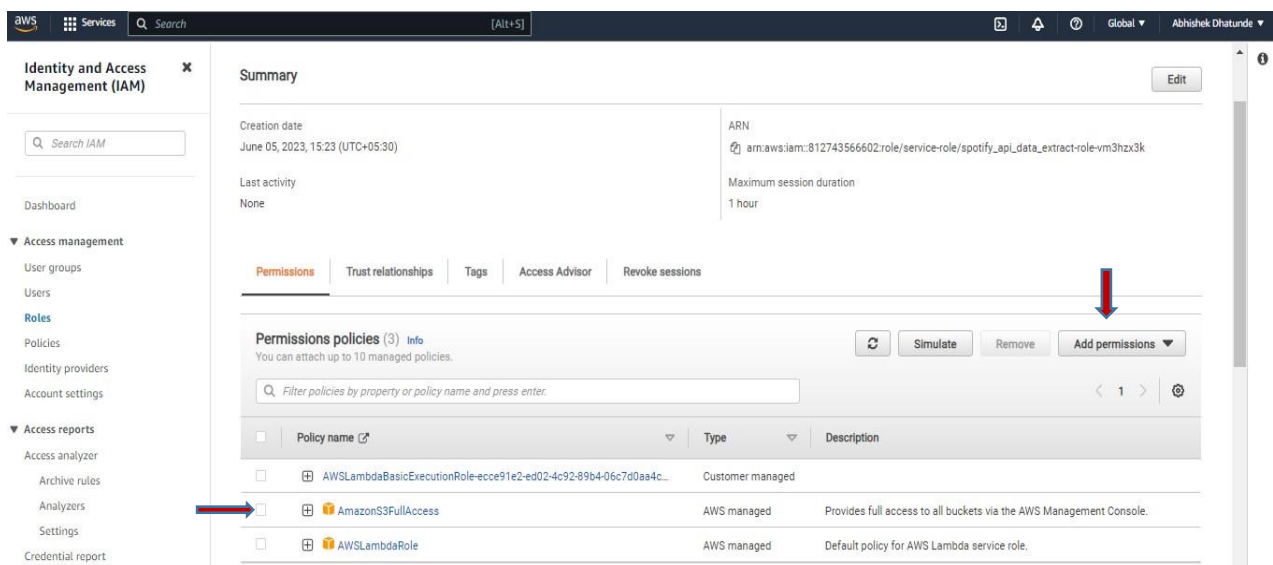
a database during the transformation phase before loading the transformed data into the final destination.

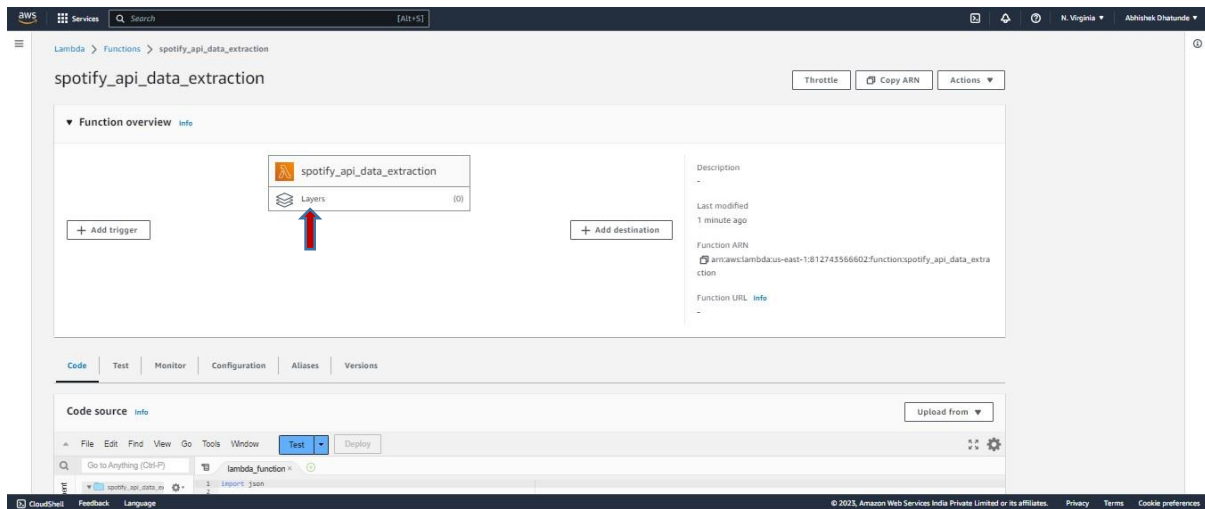
These steps provide a general guideline for creating a Lambda function for Spotify data extraction in the ETL phase. Depending on your specific requirements and architecture, you may need to modify the process accordingly.



Set up the IAM role:

- In the "spotify_api_data_extraction-role-d6bzvau0" dialog, you can optionally customize the role name and adjust other settings if needed.
- Under "Permissions Policies", click on the "Add Permissions" button to add a new policy AmazonS3FullAccess and AWSLambdaRole for accessing S3.
-





Adding Layer

1. Select the Lambda function: Locate and select the existing Lambda function to which you want to add a layer or create a new Lambda function.
2. Navigate to the Layers section: Scroll down to the "Function overview" section, and you will find the "Layers" tab.
3. Click on "Add a layer": In the Layers tab, click on the "Add a layer" button.
4. Select the layer option:
 - Custom Layers: You can choose a custom layer if you have already created and published a layer in your AWS account. Select the desired custom layer from the list of available layers in your account.
 - AWS Managed Layers: AWS provides several managed layers, such as AWS SDKs, popular libraries, and runtime-specific dependencies. You can select the desired managed layer from the list.
5. Configure the layer: Once you select the layer, you can optionally provide a compatible runtime if prompted. The layer should be compatible with the runtime of your Lambda function.
6. Save the changes: After selecting the layer and making any necessary configurations, click on the "Save" or "Add" button to add the layer to your Lambda function.
7. Verify the layer: The layer should now appear in the Layers section of your Lambda function's overview. You can see the details of the added layer, including its ARN and version.

By adding a layer to your Lambda function, you can include additional libraries, dependencies, or custom code without increasing the size of your function's deployment package. Layers help separate reusable components from your function's code, making it easier to manage and update.

Note that layers can be created and managed separately in the Lambda service or using the AWS CLI or SDKs. You can create your own custom layers by packaging the necessary files and dependencies, or you can use pre-built layers published by others.

Ensure that the added layer is compatible with the runtime and version of your Lambda function. Layers can provide functionality like SDKs, custom code, or shared resources that your Lambda function can utilize during execution.

Add layer

Function runtime settings

Runtime: Python 3.10 | Architecture: x86_64

Choose a layer

Layer source [Info](#)
Choose from layers with a compatible runtime and instruction set architecture or specify the Amazon Resource Name (ARN) of a layer version. You can also create a new layer.

☒ **AWS layers**
Choose a layer from a list of layers provided by AWS.

☐ Custom layers
Choose a layer from a list of layers created by your AWS account or organization.

☐ Specify an ARN
Specify a layer by providing the ARN.

AWS layers
Layers provided by AWS that are compatible with your function's runtime.

Choose

Cancel Add

Successfully updated the function spotify_api_data_extraction

Code properties [Info](#)

Package size: 299.0 byte | SHA256 hash: F06Z0RH/XN6RA3trvd8IUyXav7827pDgNWNKKh | Last modified: June 5, 2023 at 04:54 PM GMT+5:30

Runtime settings [Info](#) [Edit](#) [Edit runtime management configuration](#)

Runtime: Python 3.10 | Handler: lambda_function.lambda_handler | Architecture: x86_64

[Runtime management configuration](#)

Layers [Info](#) [Edit](#) [Add a layer](#)

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN
1	AWS SDK for Python 3.10	2	python3.10	x86_64	arn:aws:lambda:us-east-1:356392948345:layer:AWS SDK for Python 3.10:2

Adding Trigger

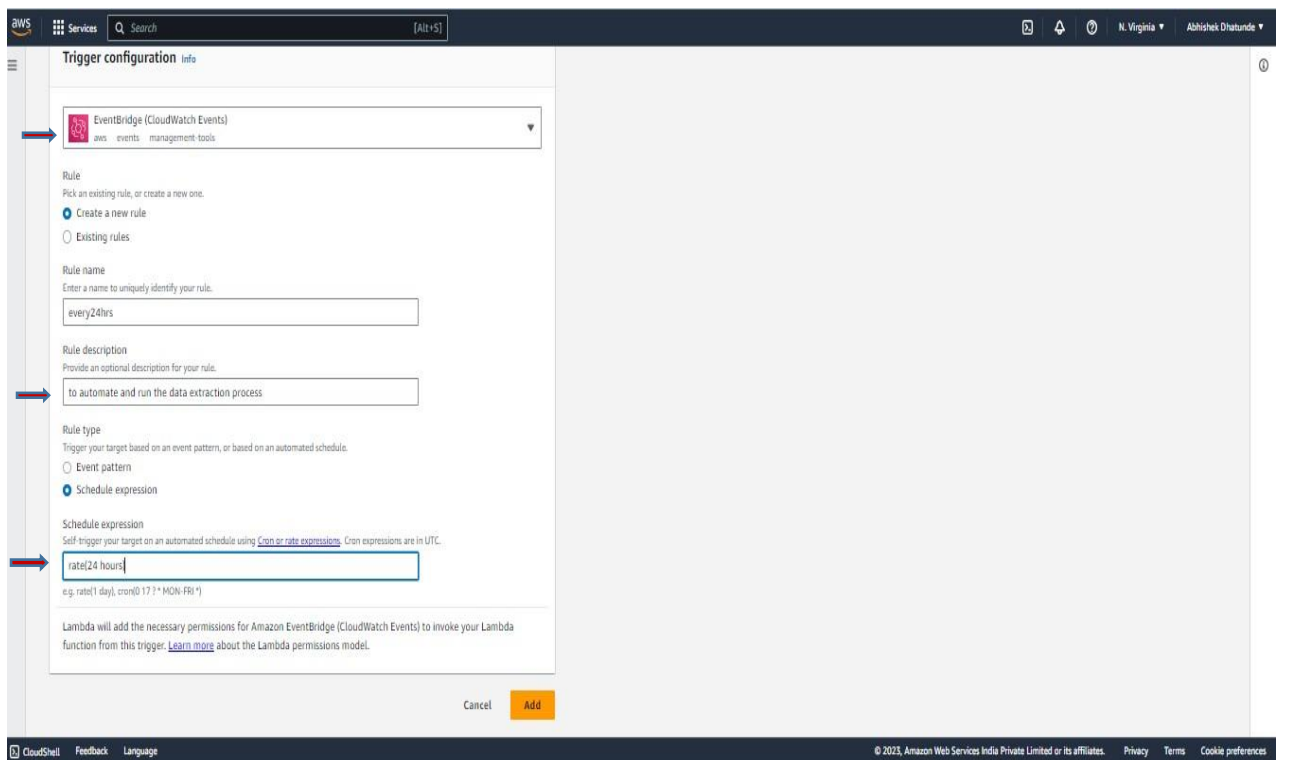
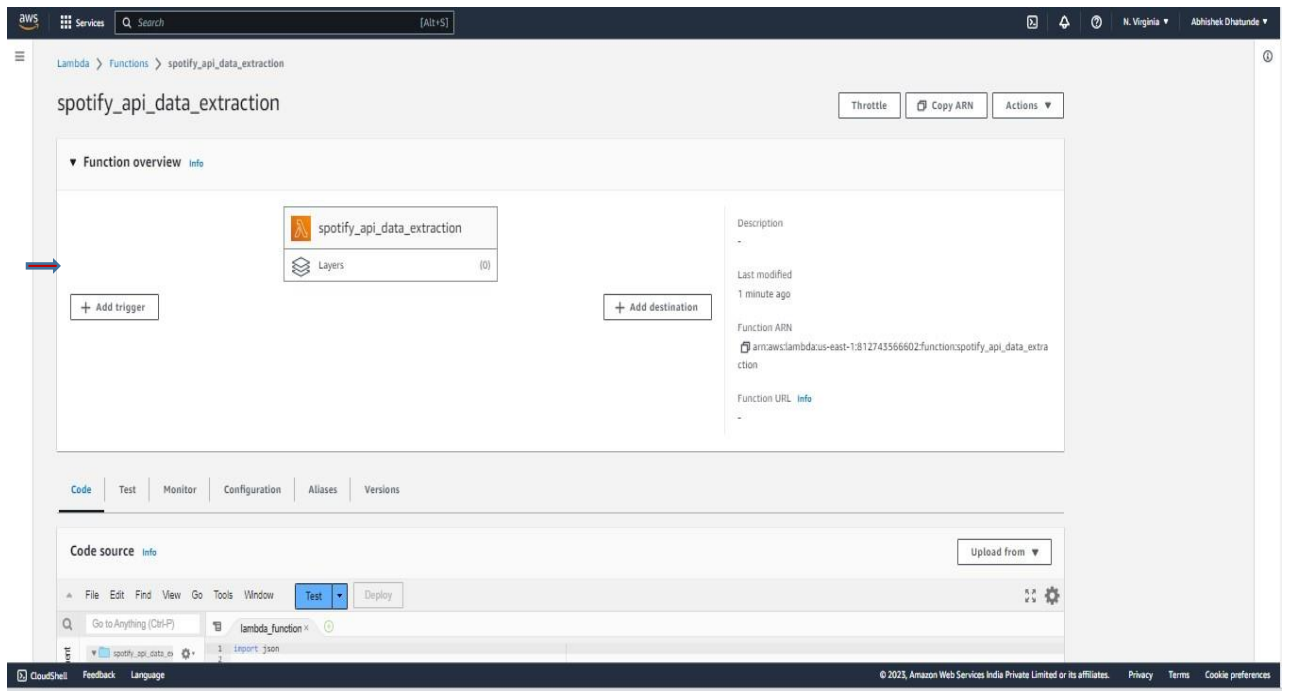
1. Navigate to the Triggers section: Scroll down to the "Function overview" section, and you will find the "Triggers" tab.
2. Click on "Add trigger": In the Triggers tab, click on the "Add trigger" button.
3. Configure the trigger:
 - Trigger configuration: In the "Select a trigger" screen, choose "EventBridge (CloudWatch Events)" from the list of available triggers.
 - Rule: If you already have an EventBridge rule, you can select it from the dropdown list. Otherwise, click on the "Create a new rule" link to create a new rule.
 - Rule configuration: Provide a name and description for the rule. Configure the event pattern or schedule expression based on your requirements.
 - Click on the "Add" button to proceed.
4. Configure permissions: If this is your first time adding an EventBridge trigger, you may need to grant permission for Lambda to receive events from EventBridge. Follow the instructions to automatically create the necessary permissions.
5. Save the changes: After configuring the trigger and permissions, click on the "Save" or "Add" button to add the EventBridge trigger to your Lambda function.
6. Verify the trigger: The EventBridge trigger should now appear in the Triggers section of your Lambda function's overview. You can see the details of the trigger, including the associated rule and its configuration.

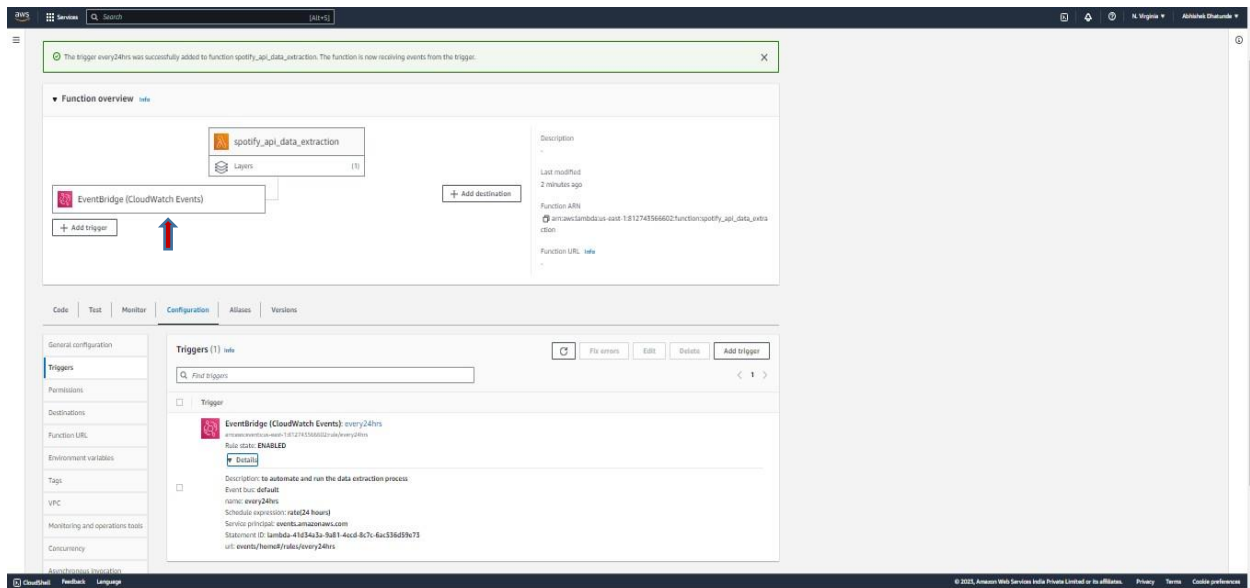
By adding an EventBridge trigger to your Lambda function, you can define rules that trigger the function based on events from various AWS services or custom events. EventBridge allows you to create event-driven architectures, schedule periodic invocations, or respond to specific events within your AWS environment.

Make sure you configure the event pattern or schedule expression in the EventBridge rule to specify the events or schedule that should trigger your Lambda function. You can configure rules to trigger based on events from services like AWS CloudTrail, AWS S3, AWS CloudFormation, and more.

Note that EventBridge also supports custom events, enabling you to publish events from your own applications or systems and trigger Lambda functions based on those events.

Ensure that your Lambda function code handles the incoming event data appropriately based on the event structure and content. You can access the event data within your Lambda function code to perform the necessary processing or take appropriate actions based on the triggered event.





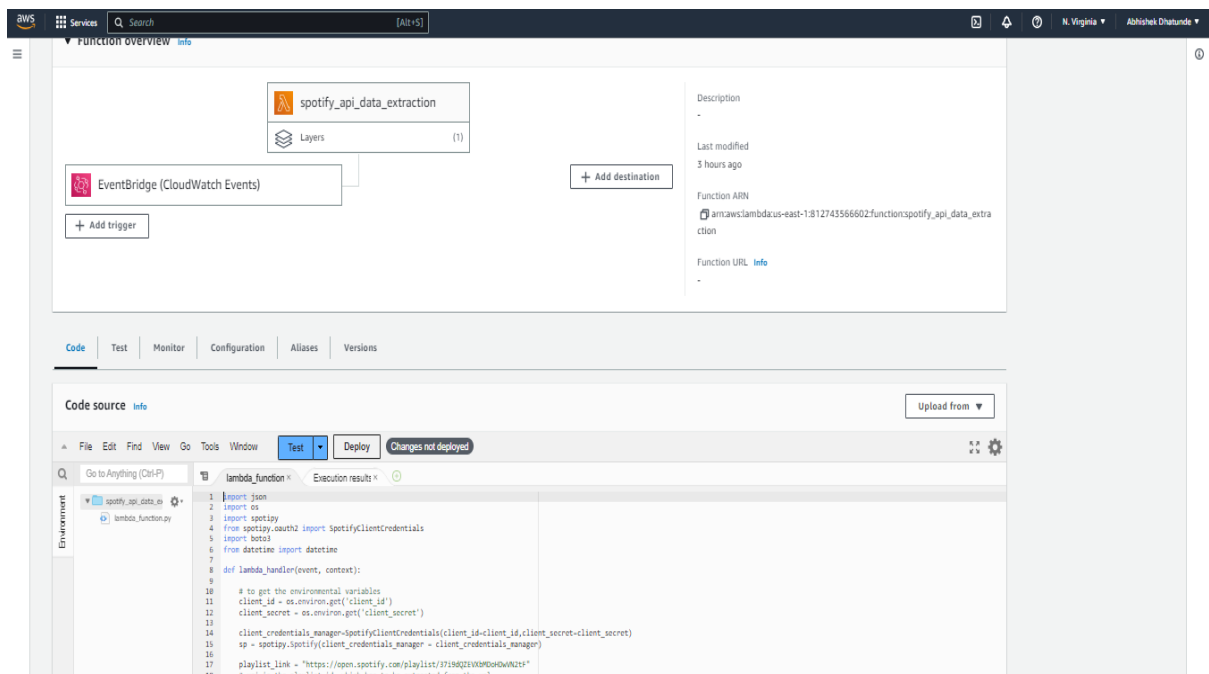
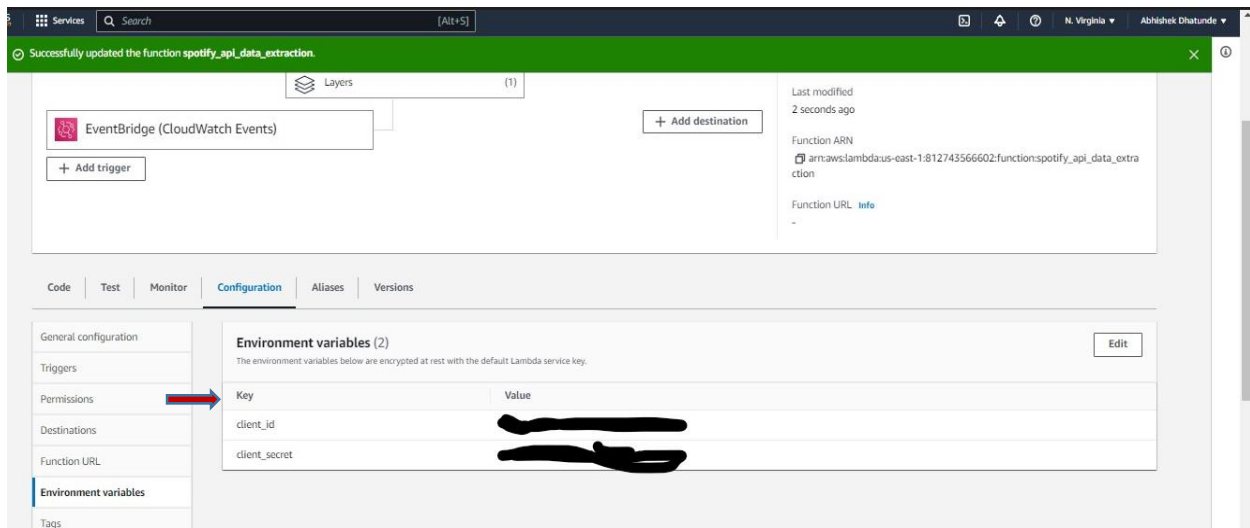
To add environment variables to an AWS Lambda function, you can follow these steps:

1. Navigate to the Configuration section: Scroll down to the "Function overview" section, and you will find the "Configuration" tab.
2. Click on "Configuration": In the Configuration tab, click on the "Configuration" button to access the function configuration settings.
3. Configure environment variables:
 - Under the "General configuration" section, scroll down to the "Environment variables" section.
 - Click on the "Edit" button to edit the environment variables.
4. Add environment variables: In the "Environment variables" editor, you can add or modify environment variables for your Lambda function. Each variable has a key-value pair.
 - To add a new environment variable, click on the "Add environment variable" button.
 - Provide the variable name (key) and its value. For example, you could set the key as **API_KEY** and the value as your API key value.
5. Save the changes: After adding or modifying the environment variables, click on the "Save" button to save the changes to your Lambda function.

By adding environment variables to your Lambda function, you can securely pass configuration values or sensitive information, such as API keys, database connection strings, or custom settings, without hard-coding them in your code. Environment variables provide flexibility and allow you to modify the behavior of your Lambda function without redeploying the code.

Ensure that you handle environment variables securely and avoid exposing any sensitive information in your Lambda function's logs or outputs.

Note: If you are using infrastructure-as-code tools like AWS CloudFormation or AWS Serverless Application Model (SAM), you can define environment variables in the template file and deploy the stack using those configurations.



Create a new function with an S3 trigger for the transform operation and add it to the specified S3 bucket

The screenshot shows the AWS Lambda console interface. At the top, the function name is `spotify_transform_load_function`. Below it, the 'Layers' section shows one layer. The 'Triggers' section is expanded, showing a single trigger of type 'S3' with the name `spotify-etl-project-abhee`. The trigger details include the bucket ARN `arn:aws:s3::spotify-etl-project-abhee`, event types `s3:ObjectCreated:*`, notification name `cf8a756a-2a06-4153-8360-f41a58df5239`, prefix `raw_data/to_be_processed/`, service principal `s3.amazonaws.com`, source account `812743566602`, and statement ID `lambda-50167ad4-09f6-4274-adf9-56bb4acbe3af`. The right sidebar shows the function's description, last modified time (2 minutes ago), and function ARN `arn:aws:lambda:us-east-1:812743566602:function:spotify_transform_load_function`. The bottom navigation bar includes tabs for Code, Test, Monitor, Configuration (selected), Aliases, and Versions.

The screenshot shows the AWS IAM console interface. The left sidebar contains navigation links for Identity and Access Management (IAM), Access management, Access reports, and Related consoles. The main content area displays the role `spotify_transform_load_function-role-fh17a3sv`. The 'Summary' section shows the creation date (June 06, 2023, 15:27 UTC+05:30), last activity (None), and ARN `arn:aws:iam::812743566602:role/service-role/spotify_transform_load_function-role-fh17a3sv`. The 'Permissions' tab is selected, showing a list of permissions policies. The table lists three policies: `AWSLambdaBasicExecutionRole-cbd7c0b3-96fb-42cb-bfee-2349ae5cbbd3` (Customer managed), `AmazonS3FullAccess` (AWS managed), and `AWSLambdaRole` (AWS managed). The 'Permissions boundary' section is also visible, indicating that no boundary is set.

Policy name	Type	Description
<code>AWSLambdaBasicExecutionRole-cbd7c0b3-96fb-42cb-bfee-2349ae5cbbd3</code>	Customer managed	
<code>AmazonS3FullAccess</code>	AWS managed	Provides full access to all buckets via the AWS Management Console.
<code>AWSLambdaRole</code>	AWS managed	Default policy for AWS Lambda service role.


Create a new custom layer by adding given zip file spotipy_layer_aws_lambda.zip

Layer configuration

Name
spotipy_layer

Description - optional
Description

☒ Upload a .zip file
☐ Upload a file from Amazon S3


 Upload spotipy_layer_aws_lambda.zip (1.5 MB)
For files larger than 10 MB, consider uploading using Amazon S3.


Compatible architectures - optional [Info](#)
Choose the compatible instruction set architectures for your layer.

☒ x86_64
☐ arm64

Compatible runtimes - optional [Info](#)
Choose up to 15 runtimes.

Runtimes
Python 3.10

spotipy_layer ARN -  arn:aws:lambda:ap-northeast-1:812743566602:layer:spotipy_layer:1 Delete Download Create version

 Successfully created layer spotipy_layer version 1.

Version details

Version 1	Description -	Created now
License -	Compatible runtimes python3.10	Compatible architectures x86_64

Versions | Functions using this version

All versions

Version	Version ARN	Description
1	arn:aws:lambda:ap-northeast-1:812743566602:layer:spotipy_layer:1	-

Services

Search

[Alt+S]

Tokyo

Abhishek Dhatunde

Successfully updated the function `spotify_transform_load_function`.

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

Code properties

Info

Package size

299.0 byte

SHA256 hash

f106ZIRH/KN6Ra3twvdRIUyYxv182Tjx0qNWNiKih=

Last modified

June 6, 2023 at 09:43 AM GMT+5:30

Runtime settings

Info

Edit

Edit runtime management configuration

Runtime

Python 3.10

Handler

Info

lambda_function.lambda_handler

Architecture

Info

x86_64

Runtime management configuration

Layers

Info

Edit

Add a layer

Merge order

Name

Layer version

Compatible runtimes

Compatible architectures

Version ARN

1

spotify_layer

1

python3.10

x86_64

arn:aws:lambda:ap-northeast-1:812743566602:layer:spotify_layer:1

Services

Search

[Alt+S]

N. Virginia

Abhishek Dhatunde

Function overview

Info

spotify_transform_load_function

Layers

(1)

+ Add destination

S3

+ Add trigger

Description

-

Last modified

3 hours ago

Function ARN

arn:aws:lambda:us-east-1:812743566602:function:spotify_transform_load_function

Function URL

Info

Code

Test

Monitor

Configuration

Aliases

Versions

Code source

Info

Upload from

File

Edit

Find

View

Go

Tools

Window

Test

Deploy

Changes not deployed

Go to Anything (Ctrl-P)

lambda_function

Execution results

Environment

spotify_transform_load_function

lambda_function.py

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

AWS Glue

AWS Glue is a fully managed extract, transform, and load (ETL) service provided by Amazon Web Services (AWS). It is designed to make it easier for users to prepare and load their data for analytics, data warehousing, and other data-related tasks.

Here are some key features and components of AWS Glue:

1. **Data Catalog:** AWS Glue provides a centralized metadata repository called the Data Catalog. It stores metadata information about various data sources, such as tables, schemas, and partitions, making it easier to discover and manage data.
2. **Crawlers:** AWS Glue includes crawlers that automatically scan and catalog data from various sources, including Amazon S3, databases, and data warehouses. The crawlers infer schemas and create tables in the Data Catalog based on the discovered data.
3. **ETL Jobs:** AWS Glue allows you to define ETL jobs using a visual interface or by writing custom code in Python or Scala. These jobs enable you to transform and clean your data, making it suitable for analysis and loading into target destinations.
4. **Data Preparation and Transformation:** AWS Glue provides a range of built-in transforms to perform data transformations, such as filtering, mapping, joining, and aggregating. You can also create custom transformations using code.
5. **Integration with Other AWS Services:** AWS Glue seamlessly integrates with other AWS services. For example, you can use AWS Glue to load transformed data into Amazon Redshift, Amazon S3, or Amazon RDS for further analysis using services like Amazon Athena, Amazon EMR, or Amazon QuickSight.
6. **Serverless Execution:** AWS Glue is serverless, meaning that you don't need to provision or manage infrastructure. AWS Glue automatically scales resources based on the processing demands of your ETL jobs.
7. **Data Quality and Deduplication:** AWS Glue includes features to help with data quality and deduplication. It provides visual tools to create data quality rules and identify duplicate records during the ETL process.
8. **Data Lake and Data Warehouse Integration:** AWS Glue supports both data lake and data warehouse use cases. You can use it to prepare data for loading into a data warehouse, or to catalog and transform data stored in a data lake architecture.

Overall, AWS Glue simplifies the process of building and managing data pipelines by providing an automated and scalable solution for data preparation and transformation tasks.

Add new tables to the Data Catalog

To add new tables to the Data Catalog in AWS Glue, you have a few options depending on your specific use case. Here's an overview of the steps involved:

1. **Create a Crawler:** The first step is to create a crawler in AWS Glue, which will scan your data source and automatically infer the schema and create the table in the Data Catalog. You can create a crawler through the AWS Management Console or by using the AWS Glue API or AWS Command Line Interface (CLI). The crawler will connect to your data source, such as an Amazon S3 bucket or a database, and crawl the data to determine its structure.
2. **Configure the Crawler:** When setting up the crawler, you will need to provide details about the data source, including the location, access credentials, and any custom classifiers or schema mappings. You can specify whether the crawler should run on a schedule or be triggered manually.
3. **Run the Crawler:** After configuring the crawler, you can run it to scan the data source and create the table in the Data Catalog. The crawler will analyze the data and infer the schema, including column names, data types, and partitions if applicable.
4. **Review and Edit Table Details:** Once the crawler has completed its job, you can review the generated table in the AWS Glue Data Catalog. You may need to review and edit the table details, such as adjusting column names, data types, or adding custom metadata.
5. **Query and Use the Table:** With the table added to the Data Catalog, you can now use it for querying and data processing. You can use various AWS services, such as Amazon Athena or Amazon Redshift Spectrum, to query the data directly from the Data Catalog.

Alternatively, if you have a predefined schema or want to manually create a table in the Data Catalog without using a crawler, you can do so through the AWS Glue console, API, or CLI. In this case, you would provide the table schema manually, specifying the column names, data types, and other relevant information.

Remember that AWS Glue provides flexibility in terms of data sources, allowing you to add tables from various sources like Amazon S3, databases, and data warehouses.

Note : Please refer below images

Services

Search

[Alt+S]

N. Virginia

Abhishek Dhatwade

AWS Glue

Tables

Add table

Step 1

Set table properties

Step 2

Choose or define schema

Step 3

Review and create

Getting started

ETL jobs

Visual ETL

Notebooks

Job run monitoring

Data Catalog tables

Data connections

Workflows (orchestration)

Data Catalog

Databases

Tables

Stream schema registries

Schemas

Connections

Classifiers

Classifiers

Catalog settings

Data Integration and ETL

Legacy pages

What's New

Documentation

AWS Marketplace

Enable compact mode

Enable new navigation

Set table properties

Table details

Name

album_data

If you plan to access the table from Amazon Athena, then the name should be under 255 characters and contain only lowercase letters (a-z), numbers (0-9), and underscore (_). For more information, see [Athena names](#).

Database

spotify_db

Create database

Description - optional

Enter a description

Descriptions can be up to 2048 characters long.

Data store

Select the type of source

☒ S3

☐ Kinesis

☐ Kafka

Data location is specified in

☒ my account

☐ another account

Include path

s3://spotify-etl-project-abhee/transformed_data/album_data

View

Browse S3

Paths must be in the form s3://bucket/prefix/. It must end with a slash (/) and not include any files.

Data format

Classification

Choose the format of the data in your table.

☐ Avro

☒ CSV

☐ JSON

☐ XML

☐ Parquet

☐ ORC

Delimiter

Comma (,)

Services

Search

[Alt+S]

N. Virginia

Abhishek Dhatwade

AWS Glue

Tables

album_data

One table successfully created

The following table is now created: "album_data (db:spotify_db)"

Getting started

ETL jobs

Visual ETL

Notebooks

Job run monitoring

Data Catalog tables

Data connections

Workflows (orchestration)

Data Catalog

Data Integration and ETL

Legacy pages

What's New

Documentation

AWS Marketplace

Enable compact mode

Enable new navigation

album_data

Last updated (UTC)

June 6, 2023 at 10:15:14

Version 0 (Current version)

Actions

Table overview

Data quality

Table details

Advanced properties

Name	album_data	Description	-	Database	spotify_db	Classification	csv
Location	s3://spotify-etl-project-abhee/transformed_data/album_data	Connection	-	Deprecated	-	Last updated	June 6, 2023 at 10:15:14
Input format	org.apache.hadoop.mapred.TextInputFormat	Output format	org.apache.hadoop.hive.qjo.HiveIgnoreKeyTextOutputForma	Serde serialization lib	org.apache.hadoop.hive.serde2.OpenCSVSerde		

Schema

Partitions

Indexes

Schema (0)

View and manage the table schema.

Edit schema as JSON

Edit schema

Filter schemas

< 1 >

#	Column name	Data type	Partition key	Comment
No table schema				

Services

Search

[Alt+S]

N. Virginia

Abhishek Dhatunde

AWS Glue

Getting started

ETL jobs

Visual ETL

Notebooks

Job run monitoring

Data Catalog tables

Data connections

Workflows (orchestration)

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

AWS Glue

Crawlers

Add crawler

Step 1

Set crawler properties

Step 2

Choose data sources and classifiers

Step 3

Configure security settings

Step 4

Set output and scheduling

Step 5

Review and create

Configure security settings

IAM role

info

Existing IAM role

AWSGlueServiceRole-spotify-c3-glue-role

Create new IAM role

Update chosen IAM role

Only IAM roles created by the AWS Glue console and have the prefix "AWSGlueServiceRole-" can be updated.

Lake Formation configuration - optional

Allow the crawler to use Lake Formation credentials for crawling the data source. [Learn more.](#)

☐ Use Lake Formation credentials for crawling S3 data source

Checking this box will allow the crawler to use Lake Formation credentials for crawling the data source. If the data source is registered in another account, you must provide the registered account ID. Otherwise, the crawler will crawl only those data sources associated to the account. Only applicable to S3, Glue Catalog and iceberg data sources.

Security configuration - optional

Enable at-rest encryption with a security configuration.

Cancel

Previous

Next

Services

Search

[Alt+S]

N. Virginia

Abhishek Dhatunde

AWS Glue

Getting started

ETL jobs

Visual ETL

Notebooks

Job run monitoring

Data Catalog tables

Data connections

Workflows (orchestration)

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

AWS Glue

Crawlers

Add crawler

Step 1

Set crawler properties

Step 2

Choose data sources and classifiers

Step 3

Configure security settings

Step 4

Set output and scheduling

Step 5

Review and create

Set output and scheduling

Output configuration

info

Target database

spotify_db

Clear selection

Add database

Table name prefix - optional

Type a prefix added to table names

Maximum table threshold - optional

This field sets the maximum number of tables the crawler is allowed to generate. In the event that this number is surpassed, the crawl will fail with an error. If not set, the crawler will automatically generate the number of tables depending on the data schema.

Type a number greater than 0

Advanced options

Crawler schedule

You can define a time-based schedule for your crawlers and jobs in AWS Glue. The definition of these schedules uses the Unix-like [cron](#) syntax. [Learn more.](#)

Frequency

On demand

Cancel

Previous

Next

AWS Services Search [Alt+S]

N. Virginia Abhishek Dhatunde

AWS Glue

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

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.

Last updated (UTC) June 6, 2023 at 10:47:37 Action Run [Create crawler](#)

<input type="checkbox"/>	Name	State	Schedule	Last run	Last run timestamp	Log	Table changes from last run
<input type="checkbox"/>	spotify_album_crawler	Ready		Succeeded	June 6, 2023 at 10:39:36	View log	-
<input type="checkbox"/>	spotify_artist_crawler	Ready		Succeeded	June 6, 2023 at 10:42:01	View log	-
<input type="checkbox"/>	spotify_top50tracks_crawler	Ready		Succeeded	June 6, 2023 at 10:43:03	View log	-

After 3 day We Use Amazon Athena is a serverless interactive query service provided by Amazon Web Services (AWS) that allows you to analyze data directly in Amazon S3 using SQL. It does not require any infrastructure setup or data movement.

Amazon Athena

Amazon Athena is a serverless query service provided by Amazon Web Services (AWS). It allows you to run interactive SQL queries on data stored in Amazon S3 without the need to set up any infrastructure or manage servers.

With Athena, you can query structured, semi-structured, and unstructured data using standard SQL syntax. It supports various file formats, such as CSV, JSON, Parquet, and ORC, making it flexible for analyzing different types of data. You simply define your table schema and point Athena to the location of your data in S3.

Here are some key features of Amazon Athena:

1. **Serverless:** Athena is serverless, which means you don't have to provision or manage any infrastructure. It automatically scales based on your query demands, and you only pay for the queries you run.
2. **Schema-on-Read:** Athena follows the schema-on-read approach. This means that you define the schema when querying the data, rather than upfront during data ingestion. It gives you the flexibility to query different types of data without preprocessing or transforming it beforehand.
3. **Integration with Amazon S3:** Athena seamlessly integrates with Amazon S3 as its primary data source. You can directly query data stored in S3 buckets or query data that is continuously streamed into S3.
4. **Compatibility with SQL:** Athena supports SQL (Structured Query Language), making it easy for users familiar with SQL to analyze and extract insights from their data. It supports a wide range of SQL functions, including joins, aggregations, and window functions.
5. **Query Performance:** Athena is built on Presto, an open-source distributed SQL query engine. It utilizes distributed processing to parallelize queries across multiple nodes, enabling fast and efficient query performance, even on large datasets.
6. **Data Catalog Integration:** Athena can integrate with AWS Glue Data Catalog, which acts as a central metadata repository for your data assets. The Glue Data Catalog helps in managing and organizing your data, making it easier to query and analyze with Athena.

Overall, Amazon Athena provides a powerful and easy-to-use platform for querying and analyzing data stored in Amazon S3 using SQL. It offers the flexibility of a serverless architecture, eliminating the need for infrastructure management, and allows you to derive insights from your data quickly and cost-effectively.

AWS Glue, on the other hand, is a fully managed extract, transform, and load (ETL) service that helps you prepare and transform your data for analysis. It provides capabilities for data cataloging, data discovery, data transformation, and job scheduling.

The AWS Glue Data Catalog is a central metadata repository that stores metadata about your data assets. It acts as a data catalog for AWS Glue, Amazon Athena, and other AWS services. The Glue Data Catalog allows you to create and manage databases, tables, and partitions, and it provides a consistent view of your data across different services.

When using Amazon Athena with the AWS Glue Data Catalog, you can take advantage of the catalog's metadata to easily query and analyze your data. Instead of explicitly defining the schema for your data in Athena, you can use the schema information stored in the Glue Data Catalog. This allows you to query and join tables without needing to define their structure explicitly in Athena.

To use Amazon Athena with the AWS Glue Data Catalog, you need to configure Athena to use the Glue Data Catalog as its metadata store. This can be done through the Athena console or by using the AWS Command Line Interface (CLI). Once configured, you can start querying your data in S3 using SQL statements in Athena, and it will leverage the metadata stored in the Glue Data Catalog for schema information.

In summary, Amazon Athena is a query service for analyzing data in S3 using SQL, while AWS Glue provides ETL capabilities and includes the Glue Data Catalog, which acts as a central metadata repository for various AWS services, including Athena. Using Athena with the Glue Data Catalog allows you to query and analyze your data with the help of the metadata stored in the catalog.

The screenshot displays the Amazon Athena console interface. At the top, there's a navigation bar with various AWS services. The main area is titled 'Amazon Athena > Query editor'. On the left, there's a 'Data' sidebar with 'Data source' set to 'AwsDataCatalog', 'Database' set to 'spotify_db', and a list of tables including 'album_data', 'artist_data', and 'track_data'. The main editor shows a SQL query: `SELECT * FROM "spotify_db"."artist_data" limit 10;`. Below the query, there are buttons for 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create'. The 'Query results' section shows a 'Completed' status with a green bar. Below this, the 'Results (10)' are displayed in a table format. The table has columns for '#', 'artist_id', 'artist_name', and 'external_url'. The first three rows are visible, showing data for Miley Cyrus, SZA, and KAROL G.

#	artist_id	artist_name	external_url
1	5YGY8feqx7naU7z4HrwZM6	Miley Cyrus	https://open.spotify.com/artist/5YGY8feqx7naU7z4HrwZM6
2	7cYKF4w9nC0nq9CsPZTHyP	SZA	https://open.spotify.com/artist/7cYKF4w9nC0nq9CsPZTHyP
3	790FomKXxshlBYZFtlgla	KAROL G	https://open.spotify.com/artist/790FomKXxshlBYZFtlgla

CloudShell interface showing Query 7 results. The query is: `SELECT * FROM "spotify_db"."album_data" limit 10;`

Query results (10):

#	album_id	album_name	release_date	total_tracks	url
1	0HiZ8NkxJOQcrf5ifrdz	Endless Summer Vacation	2023-03-10	13	https://open.spotify.com/album/0HiZ8NkxJOQcrf5ifrdz
2	1nnVofqDRs7cpWXJ49qTnP	SOS	2022-12-08	23	https://open.spotify.com/album/1nnVofqDRs7cpWXJ49qTnP
3	4kS7bSuU0Jm9LYMosFU2x5	MAÑANA SERÁ BONITO	2023-02-24	17	https://open.spotify.com/album/4kS7bSuU0Jm9LYMosFU2x5
4	6cVfHBcp3AdpYY0b8gkLN	Boy's a liar Pt. 2	2023-02-03	2	https://open.spotify.com/album/6cVfHBcp3AdpYY0b8gkLN
5	50uChhk7AKkzDKytdixYW	RR	2023-03-24	3	https://open.spotify.com/album/50uChhk7AKkzDKytdixYW
6	6aBVGuOUeUx18rHxyDWbtI	La Bebe (Remix)	2023-03-17	2	https://open.spotify.com/album/6aBVGuOUeUx18rHxyDWbtI
7	35dut3lCqF3NEDkxfzJJ1	Starboy (Deluxe)	2023-03-14	21	https://open.spotify.com/album/35dut3lCqF3NEDkxfzJJ1

CloudShell interface showing Query 5 results. The query is: `SELECT * FROM "spotify_db"."track_data" limit 10;`

Query results (10):

#	track_id	track_name	track_duration	track_popularity	track_added	track_url
1	4DHcnVTt87F0zZhrPYmZ3B	Flowers	200600	87	2023-03-30 10:22:07+00:00	https://open.spotify.com/track/4DHcnVTt87F0zZhrPYmZ3B
2	1Qrg8KqBpW07V7PNxwwL	Kill Bill	153946	94	2023-03-30 10:22:07+00:00	https://open.spotify.com/track/1Qrg8KqBpW07V7PNxwwL
3	0DWgJ2oZMBFzRsi2Cvzfz	TQG	197933	97	2023-03-30 10:22:07+00:00	https://open.spotify.com/track/0DWgJ2oZMBFzRsi2Cvzfz
4	6AQbmUe0Qw5P2nt4HmTxv	Boy's a liar Pt. 2	131013	97	2023-03-30 10:22:07+00:00	https://open.spotify.com/track/6AQbmUe0Qw5P2nt4HmTxv
5	609E1JlInJcactoMmkDon	BESO	194543	86	2023-03-30 10:22:07+00:00	https://open.spotify.com/track/609E1JlInJcactoMmkDon
6	2UW7JaomAMuX9pZjVpHAU	La Bebe - Remix	234352	85	2023-03-30 10:22:07+00:00	https://open.spotify.com/track/2UW7JaomAMuX9pZjVpHAU