[日付]

Kameda, Harunobu

Amazon Corporate

[会社の住所]

はじめてのモダンデータ解析基盤

[文書のサブタイトル]

# 目次

[目次 1](#_Toc45219270)

[1. はじめに 3](#_Toc45219271)

[1.1. 本ハンズオンのゴール 3](#_Toc45219272)

[1.2. 準備事項 3](#_Toc45219273)

[2. ハンズオンの概要 4](#_Toc45219274)

[2.1. ハンズオン全体を通しての注意事項 4](#_Toc45219275)

[2.2. ハンズオンの構成 4](#_Toc45219276)

[3. はじめの準備 6](#_Toc45219277)

[3.1. 事前準備 6](#_Toc45219278)

[3.1.1. キーペアの作成 6](#_Toc45219279)

[3.2. S3 の設定 6](#_Toc45219280)

[3.2.1. S3 バケットの作成 6](#_Toc45219281)

[3.3. CloudFormationの実行 7](#_Toc45219282)

[3.3.1. VPC, EC2, Kinesis Data Firehoseを CloudFormation で構築 7](#_Toc45219283)

[4. アプリケーションログの永続化と長期間データの分析と可視化 9](#_Toc45219284)

[4.1. Glue Crawler, Athena の設定変更 9](#_Toc45219285)

[4.1.1. IAM ロールのポリシー追加 9](#_Toc45219286)

[4.1.2. Glue Crawler を使ったスキーマの自動作成 10](#_Toc45219287)

[4.1.3. Athena でクエリ実行 11](#_Toc45219288)

[4.1.4. QuickSight の設定 12](#_Toc45219289)

[4.2. まとめ 14](#_Toc45219290)

[5. クラウドDWHを使用したデータ分析 16](#_Toc45219291)

[5.1. Redshiftの構築 16](#_Toc45219292)

[5.2. Redshiftへの接続 17](#_Toc45219293)

[5.2.1. Redshiftへの接続 17](#_Toc45219294)

[5.2.2. Redshiftにデータロード 18](#_Toc45219295)

[5.2.3. Redshift Spectrumの使用 20](#_Toc45219296)

[5.2.4. QuickSight の設定 22](#_Toc45219297)

[5.3. まとめ 25](#_Toc45219298)

[6. サーバーレスでデータのETL処理 27](#_Toc45219299)

[6.1. GlueのETL処理 27](#_Toc45219300)

[6.1.1. IAMロールにポリシーを追加 27](#_Toc45219301)

[6.1.2. GlueでETLジョブ作成と実行 27](#_Toc45219302)

[6.1.3. Glueクローラの作成と実行 30](#_Toc45219303)

[6.1.4. Athenaでクエリ比較 31](#_Toc45219304)

[6.1.5. GlueジョブでParquetとパーティショニングを実行 32](#_Toc45219305)

[6.1.6. Athenaでクエリ比較 34](#_Toc45219306)

[6.2. まとめ 34](#_Toc45219307)

[7. 後片付け 36](#_Toc45219308)

# はじめに

## 本ハンズオンのゴール

幅広いデータソースからの構造化データ、または非構造化データの集中リポジトリとして使用できる Data Lake は、データの保存と分析の方法として多くの企業に取り入れられています。

AWS のビッグデータ関連サービスを使用して実際に分析パイプラインを構築することを通して、Data Lake とビッグデータ分析基盤構築の実感を持って頂くことをゴールとしています。

## 準備事項

* AWS を利用可能なネットワークに接続された PC（Windows, Mac OS, Linux等）
* 事前に用意していただいたAWSアカウント
* ブラウザ（Firefox もしくは Chrome を推奨）
* SSH クライアント（Windows 環境では Tera Term を推奨）

# ハンズオンの概要

## ハンズオン全体を通しての注意事項

本ハンズオンは、基本的に「東京」、「バージニア北部」、「オレゴン」、「シンガポール」を前提に記載されています。リソースなどの上限に引っかかってしまった場合は、上記のリージョンのどれかの環境で作成することが可能です。作業を行うリージョンは講師の指示に従ってください。

注：CloudFormation内で利用するAMIが上記４つのリージョンのみ用意されています。

各章で配置されている「補足説明」につきましては、本ハンズオンを進めていただく上では必須手順ではありません。参考資料としてください。

同じ AWS アカウントで複数人が同時に本ハンズオンを実施される場合、適宜名前などが重複しないようにご留意ください。

各手順において、「任意」と記載のあるものについては自由に名前を変更いただくことができますが、ハンズオン中に指定した名前がわからなくならないように、ハンズオン実施中はS3の名前以外、基本的にはそのままの名前で進めることを推奨いたします。

## ハンズオンの構成

本ハンズオンは4つのラボで構成されています。

* 準備
  + AWSサービス：Amazon S3、Amazon EC2、AWS CloudFormation
* アプリケーションログの永続化と長期間データの分析と可視化
  + AWSサービス：Amazon Kinesis Data Firehose、Amazon S3、Amazon Athena、Amazon QuickSight
* クラウドDWHを使用したデータ分析
  + AWSサービス：Amazon Kinesis Data Firehose、Amazon S3、Amazon Redshift、Amazon Redshift Spectrum, Amazon QuickSight
* サーバーレスでデータのETL処理
  + AWSサービス：AWS Glue, Amazon Athena

これらのハンズオンを通じて、長期間のデータをバッチ分析する環境の構築と、パフォーマンスコストの最適化について、実施することができます。

ログデータを安価に長期保存しながら、必要に応じて ETL 処理を行った上で、アドホックにログデータに直接クエリしながら分析すべきデータを見極めつつ、 DWH で細かく分析を行うと同時に、 BI ツールで可視化する構成を、ほぼサーバレスで実現することが可能です。

スクリーンショットの画面

自動的に生成された説明

# 準備

本日の ハンズオンで必要となる共通の環境を構築します。

Amazon S3（以降、S3）のバケットを作成します。

次に AWS CloudFormation（以降、CloudFormation）にて、 次の内容を構築します。

Amazon VPC（以降、VPC）および、ログ収集ソフトウェアの Fluentd がインストールされたAmazon EC2（以降、EC2）を構築します。Amazon Kinesis Data Firehose（以降、Kinesis Data Firehose）を構築し、ログを送信し、S3 に保存されるようにします。

## 事前準備

### キーペアの作成

1. AWS マネジメントコンソールにログインします。ログイン後、画面右上のヘッダー部のリージョン選択にて、 **利用を指示されたリージョン** となっていることを確認します。

注：[東京]、[バージニア] 、[オレゴン] 、[シンガポール] のどれかになります。

1. 既存のキーペアを利用する場合は、下記の手順はスキップして下さい。
2. AWS マネジメントコンソールのサービス一覧から **EC2** を選択します。

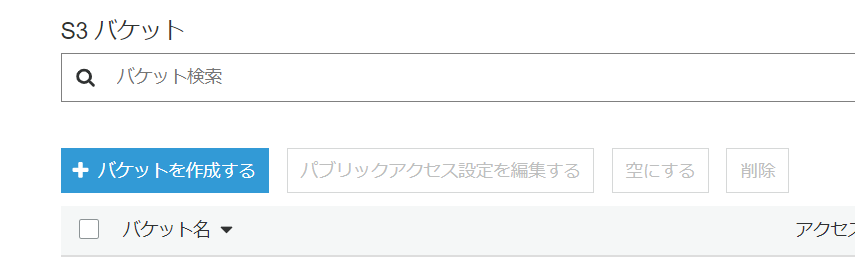


**[EC2 ダッシュボード]** の左ペインから **[キーペア]** をクリックし、 **[キーペア作成]** ボタンをクリックし、**[キーペア名]** に任意の値（例：handson）を入力し、pem形式を指定し**[作成]** をクリックします。操作しているパソコンに秘密鍵（例：handson.pem）をダウンロードします。

## S3 の設定

### S3 バケットの作成

1. AWS マネジメントコンソールのサービス一覧から **S3** を選択し、画面の **[バケットを作成する]** をクリックします。



1. バケット名を以下のルールに従って入力し、画面左下の **[作成]** をクリックします。

* バケット名：[YYYYMMDD]-handson-minilake-[Your Name][Your Birthday]
* [YYYYMMDD]：ハンズオン実施日
* [Your Name]：ご自身のお名前
* [Your Birthday]：ご自身の誕生日の日にち

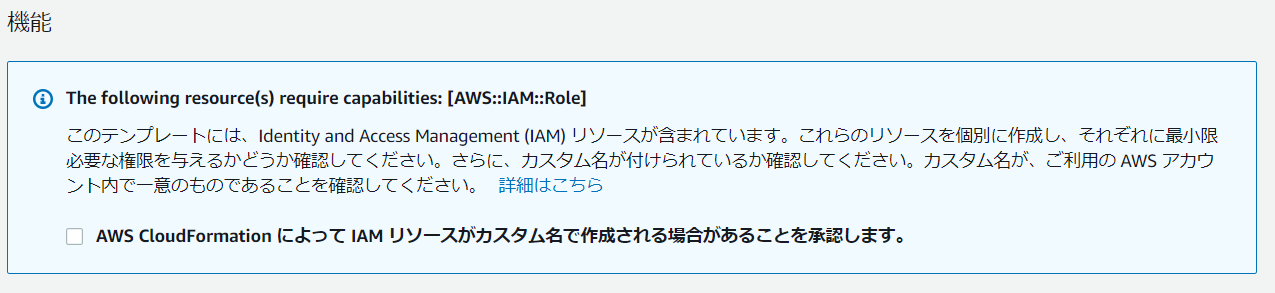
注：S3 バケット名はグローバルで一意である必要がありますが、バケットが作成できればバケット名は任意で構いません。

バケット名はCloudFormation実行時に必要になります。メモを取ってください。

## CloudFormationの実行

### VPC, EC2, Kinesis Data Firehoseを CloudFormation で構築

CloudFormation を使い、VPC を作成し、作成された VPC にログを出力し続ける EC2 を構築します。 ログは2分おきに10件前後出力され、10分おきに300件のエラーログを出力します。また、プライベートサブネットを作成し、 Amazon Redshift（以降、Redshift）に設定するセキュリティグループも同時に作成します。

1. AWS マネジメントコンソールのサービス一覧から **CloudFormation** を選択します。CloudFormation が見つけられない場合、検索窓に「cloudform」などと入力し、選択します。
2. **[CloudFormation]** の画面において、画面右上の **[スタックの作成]** をクリックし、 **[テンプレートの準備完了]** を選択します。
3. **[スタックの作成]** 画面の **[前提条件 - テンプレートの準備]** において、**[テンプレートの準備完了]** を選択します。デフォルトで選択されている場合、そのまま進めます。
4. 続いて、 **[スタックの作成]** 画面の **[テンプレートの指定]** において、 **[テンプレートファイルのアップロード]** を選択し、 **[ファイルの選択]** をクリックし、ダウンロードしたテンプレート「 **3-minilake.yaml** 」を指定し、 **[次へ]** をクリックします。
5. **[スタックの名前]** に 「 **handson-minilake**（任意）」、 **[パラメータ]** の **[InstanceType]** に「 **t2.micro** 」を選択し、 **[KeyPair]** に 3.1.1 で作成したキーペア「**handson.pem**（任意）」、もしくは既に作成済みの場合はそのキーペアを指定し、**[S3BucketName]** に3.2.1で作成したバケット名を入力し、**[次へ]** をクリックします。
6. オプションの **タグ** で、**キー** に「**Name**」 、**値** に「**handson-minilake**（任意）」 と入力し、**[次へ]**をクリックします。
7. 最後の確認ページの内容を確認し、最下部の **[AWS CloudFormation によってIAMリソースがカスタム名で作成される場合があることを承認します。]** にチェックを入れ、 **[スタックの作成]** をクリックします。

5分ほど待つ**[スタック情報]**のステータスが緑色のCREATE\_COMPLETEとなり、 EC2 一台ができあがり、 **/root/es-demo/testapp.log** にログ出力が始まります。



エラーとなった場合は、イベントタブを開き、赤字で【FAILED】などとなっている部分を質問窓口にコピペしてください。

1. S3 にデータが出力されていることを確認します。

注：数分かかります。（S3 のパスの例：20200721-handson-minilake-xxxx/minilake-in1/2020/07/21/06）

# アプリケーションログの永続化と長期間データの分析と可視化

ストリームデータを Amazon Kinesis Data Firehose（以降、Kinesis Data Firehose）に送信後、 Amazon S3（以降、S3）に長期保存しています。その後、 Amazon Athena（以降、Athena）を用いて、アドホックな分析を行い、 Amazon QuickSight（以降、QuickSight）で可視化します。

スクリーンショットの画面

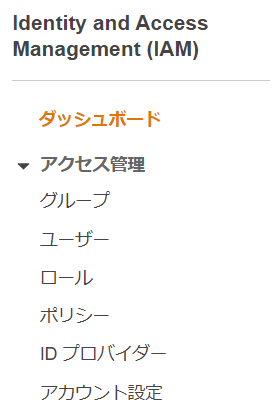
自動的に生成された説明

## Glue Crawler, Athena の設定変更

### IAM ロールのポリシー追加

作成済の「 **handson-minilake** 」のIAMロールに以下のようにポリシーを追加します。

1. AWS マネジメントコンソールのサービス一覧から **IAM** を選択し、 **[Identity and Access Management (IAM)]** 画面の左ペインから **[ロール]** を選択し、「 **handson-minilake** 」のロール名をクリックします。



1. **[アクセス権限]** タブを選択し、 **[ポリシーのアタッチ]** をクリックします。



1. 検索窓で「 **awsgluese** 」と入れ検索し **[AWSGlueServiceRole]** にチェック、再度、検索窓で「 **amazons3** 」と入れ検索し、 **[AmazonS3ReadOnlyAccess]** にチェックを入れ、 **[ポリシーのアタッチ]** をクリックします。
2. **[AWSGlueServiceRole]** と **[AmazonS3ReadOnlyAccess]** がアタッチされたことを確認します。



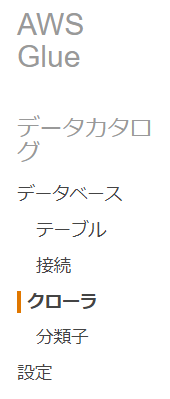
1. **[信頼関係]** タブをクリックし、 **[信頼関係の編集]** ボタンをクリックします。
2. **[信頼関係の編集]** 画面において、**”Service”: “ec2.amazonaws.com”** の箇所に **glue** を追記します。 **[]でくくり、 カンマ で区切り、 glue.amazonaws.com を追記し、[信頼ポリシーの更新]** をクリックします。

**[記入例]　エラーが出る方は以下をコピペしてみてください。**

|  |
| --- |
| {  "Version": "2012-10-17",  "Statement": [  {  "Effect": "Allow",  "Principal": {  "Service": [  "glue.amazonaws.com",  "ec2.amazonaws.com"  ]  },  "Action": "sts:AssumeRole"  }  ]  } |

### Glue Crawler を使ったスキーマの自動作成

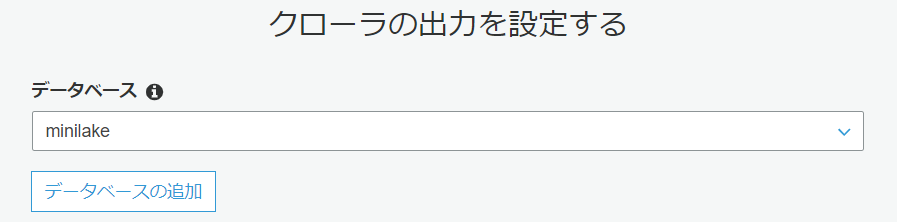
1. AWS マネジメントコンソールのサービス一覧で **AWS Glue** を選択し、**[AWS Glue]** 画面の左ペインにおいて、**[クローラ]** を選択し、**[クローラの追加]** をクリックします。



1. **[クローラの名前]** に「**minilake-in1**（任意）」と入力し、**[次へ]** をクリックします。

次の画面もそのまま **[次へ]** をクリックします。

1. **[データストアの追加]** 画面において、**[データストアの選択]** は **[S3]** 、**[クロールするデータの場所]** は **[自分のアカウントで指定されたパス]** を選択し、**[インクルードパス]** を右のフォルダアイコンから、最初に作成したバケットの「 **s3://[S3 BUCKET NAME]/minilake-in1**（任意）」を選択し、 **[次へ]** をクリックします。
2. **[別のデータストアの追加]** 画面においても、そのまま**[次へ]** をクリックします。
3. **[IAM ロールの選択]** 画面において、**[既存のIAMロールを選択]** にチェックを入れ、作成したロール「 **handson-minilake**（任意）」を選択し、**[次へ]** をクリックします
4. 続いての画面も **[頻度]** も **[オンデマンドで実行]** のままにし、 **[次へ]** をクリックします。
5. **[クローラの出力を設定する]** 画面において、 **[データベースの追加]** をクリックし、ポップアップした画面で **[データベース名]** に「 **minilake**（任意）」と入力し、 **[作成]** をクリックします。



1. **少し待ちますが上記の画面のようにデータベース名が指定されたら、[クローラの出力を設定する]** 画面に戻り、 **[次へ]** をクリックします。
2. 続いての画面の内容を確認し、 **[完了]** をクリックします。
3. **[クローラ]** の画面において、作成したクローラの「**minilake-in1**（任意）」にチェックを入れ、 **[クローラの実行]** をクリックします。



ステータスが **[Starting]** になるので、数分待ちます。

ステータスが **[Stopping]** から **[Ready]** に戻ったら、左ペインの **[テーブル]** をクリックします。

1. 「 **minilake\_in1**（任意）」のテーブルが作成されていることを確認し、テーブル名の箇所をクリックし、スキーマ定義を確認します。

列名に、「partition\_0」, 「partition\_1」等があるのが確認できます。



### Athena でクエリ実行

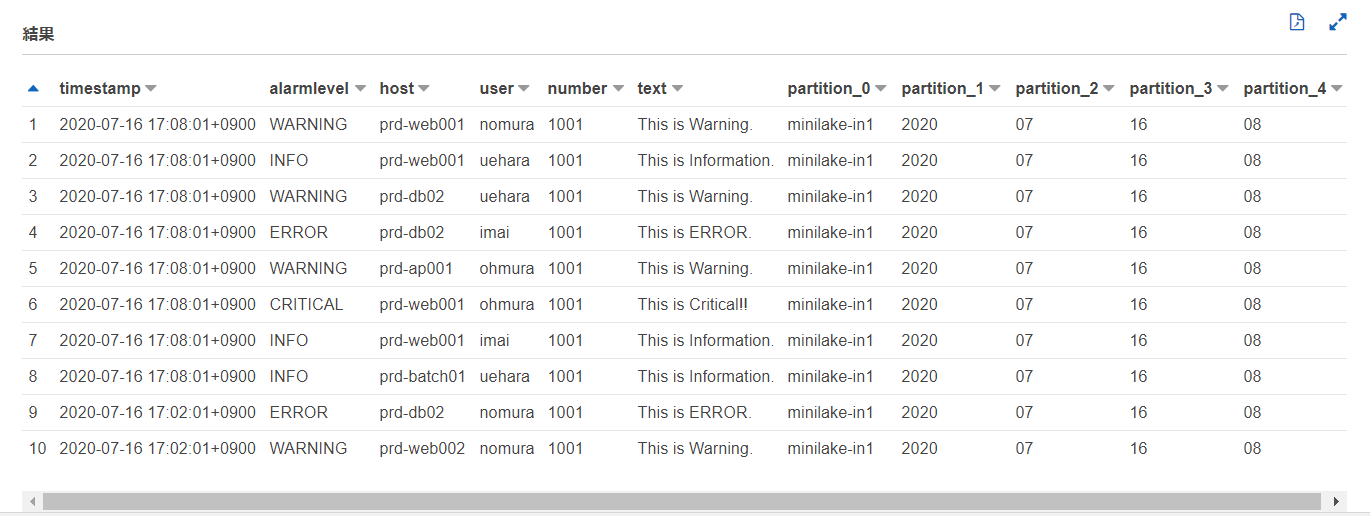
1. AWS マネジメントコンソールのサービス一覧で、 **Athena** を選択します。
2. **画面右上の [設定]** を選択し **[クエリの結果の場所]** に「 **s3://[S3 BUCKET NAME]/result/**（任意）」を入力し、**[保存]** をクリックします。（一番最後の/を忘れないでください）

**注：[S3 BUCKET NAME] には、ご自身で作成されたS3バケットの名前を入力ください。**

1. **[データベース]** において「 **minilake**（任意）」を選び、テーブルは先程作成した「 **minilake\_in1**（任意）」を選び、テーブル名の右端の **[点マーク]** をクリックし、 **[テーブルのプレビュー]** をクリックします。



1. クエリ結果が画面下部に表示されることを確認します。



1. クエリエディタで下記 SQL を実行します。（limit部分を取ります）

|  |
| --- |
| SELECT \* FROM "minilake"."minilake\_in1"; |

[実行結果例]

|  |
| --- |
| (実行時間: 2.89 秒, スキャンしたデータ: 140.47 KB) |

1. Where 句をつけたクエリを実行してみます。

|  |
| --- |
| SELECT \* FROM "minilake"."minilake\_in1"  where partition\_0 = '2020'  AND partition\_1 = '07'  AND partition\_2 = '21'  AND partition\_3 = '06'; |

**注**：Where句の日付はデータが存在するものを入力して下さい。上記4番の実行結果を見て、partitionの値に合致したものを指定してみてくさい。

### S3 Selectでクエリ実行

1. AWS マネジメントコンソールのサービス一覧で、 **S3** を選択します。
2. 本日作成したバケットを選択し、ログのある階層まで移動します。

例：）20200721-handson-minilake-xxxx/minilake-in1/2020/07/21/06

1. S3 Selectを行うログのチェックボックスにチェックを入れます。

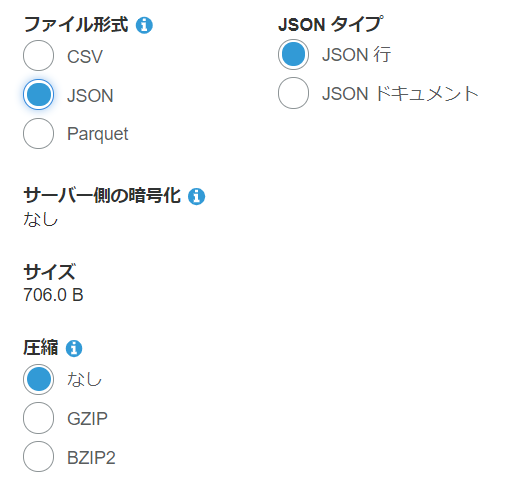
スクリーンショットの画面

自動的に生成された説明

1. **[アクション]** → **[S3 Select]をクリックします**

ファイル形式は[JSON]、JSONタイプは[JSON行]、圧縮は[なし]を選択します。

1. **[ファイルプレビューの表示]**をクリックし、**[次へ]**をクリックします。



1. SQLを入力し、**[SQLの実行]**をクリックします。

|  |
| --- |
| select \* from s3object s limit 5; |

|  |
| --- |
| select s."host" from s3object s; |

## まとめ

ストリーミングデータを直接データストアに永続化し、長期間の保存を可能にした上で、アドホックな分析・可視化を行う基盤ができました。

# クラウドDWHを使用したデータ分析

ストリームデータを Kinesis Data Firehose に送信後、 Amazon S3（以下、S3）に保存することで長期保存しています。その後、 Amazon Redshift Spectrum（以降、Redshift Spectrum）を用いて、クエリを実行します。QuickSightなどで可視化も可能です。

スクリーンショットの画面

自動的に生成された説明

## Redshiftの構築

1. Redshift を作成するVPCと配置先サブネットを確認します。

**CloudFormation** のコンソールに移動し、このハンズオンの最初に作ったスタックの「**出力**」タブを確認します。

「**VPCID**」と「 **サブネット ID** 」をメモします。



1. AWS マネジメントコンソールのサービス一覧から **Amazon Redshift** を選択し、左ペインの **[設定]** を選択します。



1. **[サブネットグループ]** タブをクリックし、 **[クラスターサブネットグループの作成]** をクリックします。
2. 以下の値を入力します。

* 名前：handson-minilake-dwh（任意）
* 説明：dwh in my vpc（任意）
* VPC ：「**handson-minilake** 」を選択
* アベイラビリティゾーン：ap-northeast-1a（東京リージョンと違う場合は、各リージョンの1aを選択）
* サブネットID：メモした「 サブネット ID 」を選択 し、**[サブネットを追加] をクリック**



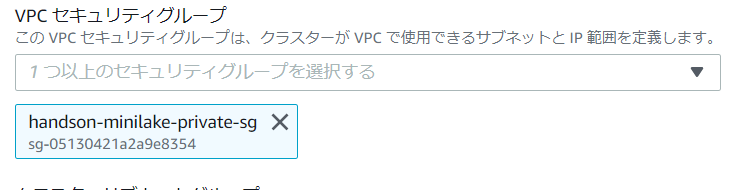
1. **[クラスターサブネットグループの作成]** をクリックします。
2. 左ペインから **[ダッシュボード]** を選択し、 **[クラスターを作成]** をクリックします。
3. 以下の値を入力します。

* クラスター識別子：handson-minilake-dwh（任意）
* ノードの種類 : dc2.large を選択
* データベース名：db（任意）
* データベースポート：5439 （デフォルト）
* マスターユーザー名：admin（任意）
* マスターユーザーのパスワード：MyPassword1（任意）

注：パスワードは8~64文字、大文字、小文字、数字が含まれている必要があります。

1. **[追加設定]** の **[デフォルトを使用]** のラジオボタンをオフにし、**[ネットワークとセキュリティ]** にて以下の値を入力します。

* Virtual Private Cloud (VPC) : **handson-minilake**
* VPC セキュリティグループ : **handson-minilake-private-sg**



1. Redshift の利用には課金が発生します。**[設定の概要]** において推定コンピューティング料金を確認し、先に進みます。

**[クラスターを作成]** をクリックします。クラスターの起動完了までに数分時間がかかります。

**注**：課金に関する注意事項は必ずご確認ください。

## Redshiftへの接続

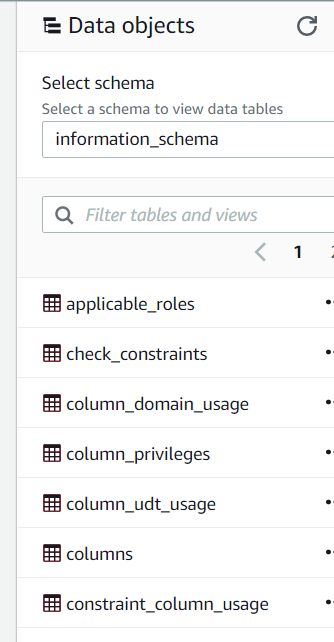
### Redshiftへの接続

クラスターの状態がAvailableになった後、Redshiftに接続できるか確認します。

1. **[Redshift]** の 左ペインから **[エディタ]** を選択します。
2. 作成したインスタンスへの接続情報を入力し、 **[データベースに接続]** をクリックします。

* クラスター：handson-minilake-dwh（任意）
* データベース名：db（任意）
* データベースユーザー：admin（任意）
* パスワード：設定済みパスワード（MyPassword1）

1. 左ペインにテーブル一覧が表示されれば、ログイン成功です。

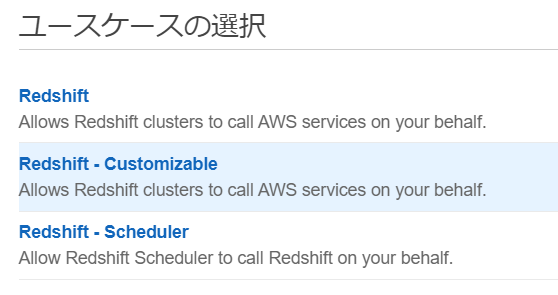
****

### Redshiftにデータロード

Redshiftにデータをロードします。

1. Redshift から S3 にアクセスして、データをロードするための IAM ロールを作成します。 AWS マネジメントコンソールのサービス一覧から **IAM** を選択し、 **[Identity and Access Management (IAM)]** 画面の左ペインから **[ロール]** を選択し、 **[ロールの作成]** をクリックします。
2. **[AWSサービス]** をクリックし、 **[Redshift]** をクリック、 **[ユースケースの選択]** では、 **[Redshift - Customizable]** を選択します。続いて、 **[次のステップ：アクセス権限]** をクリックします。

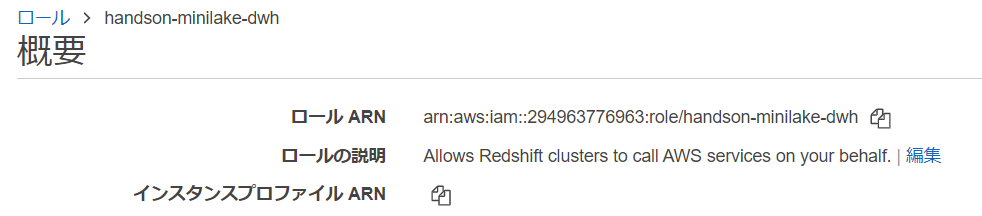




1. **[AmazonS3ReadOnlyAccess]** を選択して、 **[次のステップ：タグ]** をクリックします。

次の画面で **[次のステップ：確認]** をクリックします。

1. **[ロール名]** に「 **handson-minilake-dwh**（任意）」と入力し、 **[ロールの作成]** をクリックします。
2. 作成したロールの ARN は後で使用するのでメモしておきます。ARNは作成されたロールを特定しクリックすることで遷移する詳細画面で確認ができます。

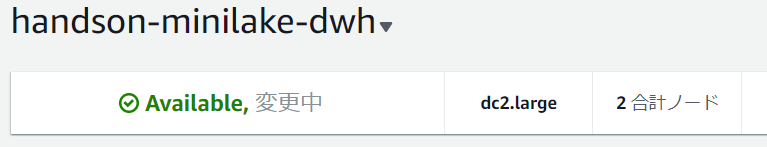


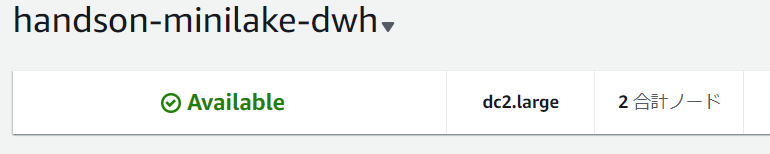
1. **[Redshift]** に戻り、画面左ペインの **[クラスター]** を選択し、作成した Redshift クラスター名（例：handson-minilake-dwh）をクリックします。 **[アクション]** から **[IAMロールの管理]** をクリックします。



1. さきほど作成した IAM ロール「 **handson-minilake-dwh**（任意）」を選択し、 **[IAMロールを追加]** をクリックし **[Done]** をクリックします。

次の画面で **[クラスターステータス]** が **[Available]** になるまで待ちます。変更中が出ている間はまだ適応作業中ですので数分間待ちます。





1. 左ペインから **[エディタ]** を選択し、 Redshift に接続し、先にロード対象のテーブルを作成します。以下のクエリを入力し、 **[実行]** をクリックし、クエリを実行します。

|  |
| --- |
| create table ec2log ( timestamp varchar, alarmlevel varchar, host varchar, number int2, text varchar ); |

1. S3 からデータを COPY します。以下のクエリを入力し、 **[実行]** をクリックし、クエリを実行します。

|  |
| --- |
| copy ec2log from 's3://[S3 BUCKET NAME]/minilake-in1' format as json 'auto' iam\_role '[IAM ROLE ARN]'; |

**注**：**[S3 BUCKET NAME]** と **[IAM ROLE ARN]** には、実際の値を入力し、実行します。

例：copy ec2log from 's3://20200716-handson-minilake-hkameda18730105/minilake-in1' format as json 'auto' iam\_role 'arn:aws:iam::294963776963:role/handson-minilake-dwh';

例は講師の検証環境なので実際の値は異なります。

1. **ec2log** テーブルに対して、自由にクエリを実行してみましょう。

(例1) テーブルの全件数が返ってくるクエリです。

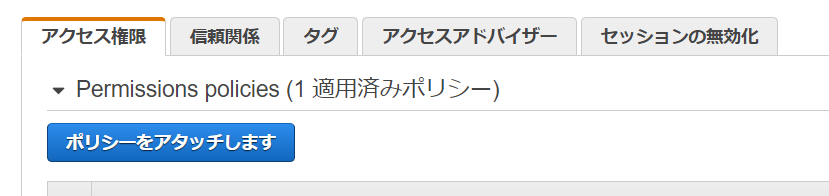
|  |
| --- |
| select count(timestamp) from ec2log; |

(例2) アラームレベル毎の集計値が返ってくるクエリです。

|  |
| --- |
| select alarmlevel, count(alarmlevel) from ec2log group by alarmlevel; |

### Redshift Spectrumの使用

1. Redshift Spectrum のスキーマとデータベースを作成することができるように、作成した IAM ロールにポリシーを追加します。 AWS マネジメントコンソールのサービス一覧から **IAM** を選択し、 **[Identity and Access Management (IAM)]** 画面の左ペインから **[ロール]** を選択し、「 **handson-minilake-dwh**（任意）」のロール名をクリックします。
2. **[アクセス権限]** タブを選択し、 **[ポリシーをアタッチします]** をクリックします。



1. **[AWSGlueServiceRole]** にチェックを入れ、 **[ポリシーのアタッチ]** をクリックします。

**注：Redshift Spectrum で外部スキーマとDBを作成する際、データカタログには AWS Glue のデータカタログが使用されます。そのため、 AWS Glue の操作認可（厳密には resource \* に対する glue:CreateDatabase のみ）が必要となります。**

1. **[Redshift]** に戻り、画面左ペインの **[エディタ]** からSQLを実行します。

Redshift に腹持ちしているスキーマとDB内には外部テーブルを作成することはできないため、別途、外部テーブル用の外部スキーマと、 DB を作成します。

|  |
| --- |
| create external schema my\_first\_external\_schema from data catalog database 'spectrumdb' iam\_role '[IAM ROLE ARN]' create external database if not exists; |

**注**：**[IAM ROLE ARN]** には、 「 **handson-minilake-dwh**（任意）」の値を入力します。

1. 外部テーブルを作成します。

|  |
| --- |
| create external table my\_first\_external\_schema.ec2log\_external ( timestamp varchar(max), alarmlevel varchar(max), host varchar(max), number int2, text varchar(max) )  partitioned by (year char(4), month char(2), day char(2), hour char(2))  ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'  WITH SERDEPROPERTIES ( 'paths'='timestamp,alarmlevel,host,number,text')  STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'  OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'  location 's3://[S3 BUCKET NAME]/minilake-in1/'; |

**注**：**[S3 BUCKET NAME]** には、作成済みの S3 バケット名を入力します。

1. 下記コマンドを実行し、スキーマ 、DB、テーブルを確認します。

**[スキーマの確認]**

|  |
| --- |
| select \* from svv\_external\_schemas; |

**[DBの確認]**

|  |
| --- |
| select \* from svv\_external\_databases; |

**[テーブルの確認]**

|  |
| --- |
| select \* from svv\_external\_tables; |

1. 下記 SQL を実行して、外部テーブルにパーティションを追加します。

|  |
| --- |
| ALTER TABLE my\_first\_external\_schema.ec2log\_external  ADD PARTITION (year='2020', month='07', day='21', hour='14')  LOCATION 's3://[S3 BUCKET NAME]/minilake-in1/2020/07/21/14'; |

**注**：ADD PARTITION句の各パーティションの値は、 LOCATION 句の S3 バケットのパスのパーティションの値と合わせてください。また、LOACATION句のパスはS3を確認して下さい。

皆さんが作業を行う時間で、2020,07,21,14の値は変わりますので、S3バケットを実際に確認し、存在しているフォルダの値を設定してください。

1. AWS マネジメントコンソールのサービス一覧から **Athena** を選択し、データベースとして先程作成した「 **spectrumdb**（任意）」を選択すると、先程作成したテーブル「**ec2log\_external**（任意）」が存在することが確認できます。

**注**：このまま Athena からクエリすることもできます。

1. AWS マネジメントコンソールのサービス一覧から **AWS Glue** を選択し、画面左ペインの **[テーブル]** をクリックすると、先程作成したテーブル「 **ec2log\_external**（任意）」が存在することが確認できます。
2. Redshift のクエリエディタを使って、外部テーブル（ec2log\_externalテーブル）に対して、自由にクエリしてみてください。

[実行例１]

|  |
| --- |
| select count(\*) from my\_first\_external\_schema.ec2log\_external; |

[実行例2]

|  |
| --- |
| select count(\*) from ec2log; |
|  |

## まとめ

ストリーミングデータを直接データストアに永続化し、長期間の保存を可能にした上で、 DWH に読み込み分析を行う基盤ができました

# サーバーレスでデータのETL処理

ストリームデータを Amazon Kinesis Data Firehose（以降、Kinesis Data Firehose）に送信後、 Amazon S3（以降、S3）に保存することで長期保存します。

その後、 AWS Glue（以下、Glue）を使って、①ファイルフォーマットを Apache Parquet 形式に変換します。②ファイルをパーティショングして配置するための処理、を実行し、その結果を S3 に保存します。

その後、 Amazon Athena（以降、Athena）や Amazon Redshift Spectrum（以降、Redshift Spectrum）を用いて、クエリを実行します。

スクリーンショットの画面

自動的に生成された説明

## GlueのETL処理

### IAMロールにポリシーを追加

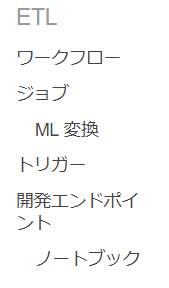
作成済の「 handson-minilake（任意）」の IAM ロールに以下のようにポリシーを追加します。

1. AWS マネジメントコンソールのサービス一覧から IAM を選択し、 [Identity and Access Management (IAM)] 画面の左ペインから [ロール] を選択し、「 handson-minilake（任意）」のロール名をクリックします。
2. **[アクセス権限]** タブを選択し、 **[ポリシーをアタッチします]** をクリックします。
3. 検索窓で「 **amazons3** 」と入れ検索し、 **[AmazonS3FullAccess]** にチェックを入れ、 **[ポリシーのアタッチ]** をクリックします。
4. 変更実施したロールの **[アクセス権限]** タブを選択し、 **[AmazonS3FullAccess]** がアタッチされたことを確認します。

****

### GlueでETLジョブ作成と実行

1. AWS マネジメントコンソールのサービス一覧から **S3** を選択し、今回のハンズオンで作成したバケットを選択します。 **[フォルダの作成]** をクリックし、フォルダ名に「 **minilake-out1**（任意）」と入力し、 **[保存]** をクリックします。
2. AWS マネジメントコンソールのサービス一覧から **AWS Glue** を選択します。画面左ペインから **[ジョブ]** をクリックし、 **[ジョブの追加]** をクリックします。



1. **[名前]** に「 **minilake1**（任意）」、 **[IAMロール]** に「 **handson-minilake**（任意）」を入力し、 **[詳細プロパティ]** をクリック、 **[モニタリングオプション]** をクリック、 **[セキュリティ設定、スクリプトライブラリおよびジョブパラメータ（任意）]** をクリックします。
2. **[モニタリングオプション]** セクション内の **[ジョブメトリクス]** にチェックを入れます。

**注**：リソースのモニタリングが可能になります。

1. **[セキュリティ設定、スクリプトライブラリおよびジョブパラメータ（任意）]** セクション内の **[最大容量]** を「 **2** 」と入力し、画面下の **[次へ]** をクリックします。
2. **[データソースの選択]** の画面で「 **minilake\_in1**（任意）」にチェックを入れ、 **[次へ]** をクリックします。
3. **[変換タイプを選択します。]** の画面はそのまま **[次へ]** をクリックし進みます。
4. **[データターゲットの選択]** 画面にて、下記を設定し、 **[次へ]** をクリックします。

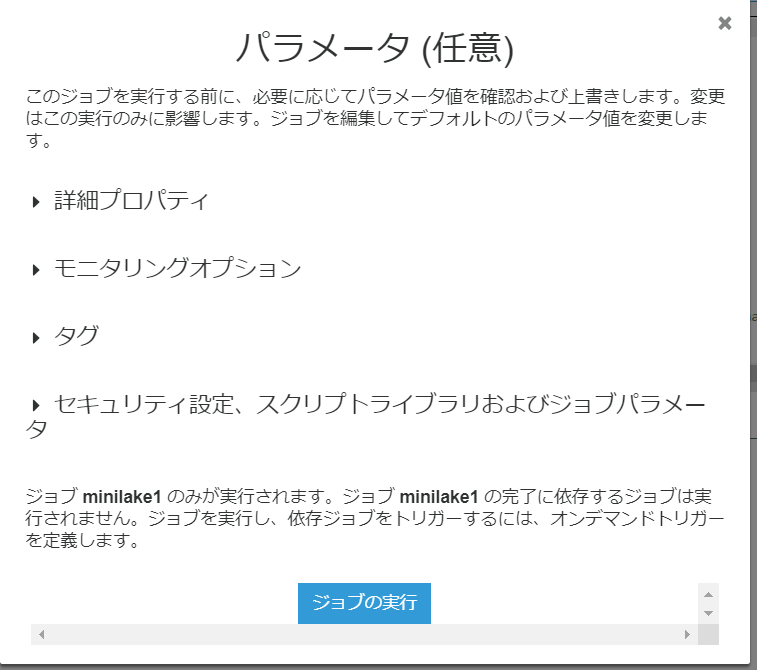
* **[データターゲットでテーブルを作成する]**にチェック
* **データストア**：Amazon S3
* **フォーマット**：Parguet
* **ターゲットパス**：「s3://**[S3 BUCKET NAME]**/minilake-out1（任意）」を選択

**注**：**[S3 BUCKET NAME]** には、ご自身で作成されたS3バケットの名前を入力ください。

1. **[ソース列をターゲット列にマッピングします。]** 画面にて、不要なカラムを排除したり、データ型の変更を行うことができます。今回はこのまま **[ジョブを保存してスクリプトを編集する]** をクリックします。
2. 続いての画面にて、**[ジョブの実行]** をクリックし、ポップアップ画面で **[ジョブの実行]** を2回クリックします。

スクリーンショットの画面

自動的に生成された説明



1. 別タブでもう一つAWS マネジメントコンソールのサービス一覧から **AWS Glue** を選択し、 **[AWS Glue]** 画面の左ペインから **[ジョブ]** を選択し、今回のジョブ「 **minilake1**（任意）」にチェックを入れます。 10分程かかる場合もありますが、問題がなければ **[実行ステータス]** が **[Succeeded]** となります。

**注**：時間があれば **[メトリクス]** タブでリソースモニタリングも確認してください。表示まで少し時間かかります。**[追加メトリクスの表示]** をクリックすると詳細が確認できますが、本ハンズオンでは値が小さすぎて見づらいことが予想されます。



1. S3の「 s3://**[S3 BUCKET NAME]**/minilake-out1/」にファイルが出力されていることを確認します。

**注**：**[S3 BUCKET NAME]** には、ご自身で作成されたS3バケットの名前を入力ください。

### Glueクローラの作成と実行

Glue で出力データ用のクローラを作成します。

1. **[AWS Glue]** 画面の左ペインから **[クローラ]** をクリックし、 **[クローラの追加]** をクリックします。
2. **[クローラの名前]** に「 **minilake-out1**（任意）」と入力し、 **[次へ]** をクリックします。
3. 続いての画面もそのまま **[次へ]** をクリックします。
4. **[データストアの追加]** 画面において、 **[インクルードパス]** に「s3://**[S3 BUCKET NAME]**/minilake-out1（任意）」を設定し、 **[次へ]** をクリックします。

**注：[S3 BUCKET NAME] には、ご自身で作成されたS3バケットの名前を入力ください。**

1. 次の画面もそのままで **[次へ]** をクリックします。
2. **[既存の IAM ロールを選択]** にチェックを入れ、作成済みロールの「 **handson-minilake**（任意）」を選び、 **[次へ]** をクリックします。
3. **[このクローラのスケジュールを設定する]** 画面において、 **[オンデマンドで実行]** のままにし、 **[次へ]** をクリックします。
4. **[データベース]** に「 **minilake**（任意）」を選び 、 **[次へ]** をクリックし、確認画面の内容を確認し、 **[完了]** をクリックします。
5. 「 **minilake-out1**（任意）」にチェックを入れ、 **[クローラの実行]** をクリックし、クローラを実行します。ステータスがStartingになりますので、完了するまで待ちます。Readyになったら完了です。



1. 左ペインの **[テーブル]** をクリックし、テーブル名「 **minilake\_out1**（任意）」をクリックし、スキーマ情報を確認します。

### Athenaでクエリ比較

1. 以下のクエリを実行し、入力データと出力データで比較を行います。

**[入力データ]** ：CSV形式で日付でパーティション

|  |
| --- |
| SELECT count(user) FROM "minilake"."minilake\_in1" where user='uchida' and timestamp >= '2020-07-21 13%' AND timestamp <= '2020-07-21 21%'; |

**注**：

「uchida（任意）」さんのデータをカウントしているクエリになります。本日のログが出た時間がすべて入るように設定してください。実行結果例は、しばらく時間が経ってデータが溜まった状態のものとなります。

上記のクエリ例は、「minilake\_out1（任意）」テーブルとの比較のため、 Where 句でパーティションの指定カラムを使用していない例となりますが、パーティションの指定カラムを使用すると、スキャンデータ量が減り、実行時間が短縮することが確認できます。

**実行結果例**：

|  |
| --- |
| (実行時間: 2.71 秒, スキャンしたデータ: 141.16 KB) |

**[出力データ]** ：Parquet形式

|  |
| --- |
| SELECT count(user) FROM "minilake"."minilake\_out1" where user='uchida' and timestamp >= '2020-07-21 13%' AND timestamp <= '2020-07-21 21%'; |

**注**：

入力データ同様、本日のログが出た時間がすべて入るように設定してください。こちらも「uchida（任意）」さんのデータをカウントしているクエリになります。

実行結果例は、しばらく時間が経ってデータが溜まった状態のものとなります。

**実行結果例**：

|  |
| --- |
| (実行時間: 2.32 秒, スキャンしたデータ: 6.75 KB) |

**注**：Parquet形式の場合、不要なカラムのデータを読み込まない分、スキャン容量を少なくすることができます。

**上記2パターンを比較すると、不要なカラムを読まない分、Parquet形式の方が、スキャン量が少ないことが分かります。**

|  |  |  |
| --- | --- | --- |
|  | **実行時間** | **スキャンしたデータ** |
| CSVパーティション（日付） | 2.71秒 | 141.16 KB |
| Parquet | 2.32秒 | 6.75 KB |

### GlueジョブでParquetとパーティショニングを実行

1. AWS マネジメントコンソールのサービス一覧から **S3** を選択し、今回作成したバケットを選択します。 **[フォルダ作成]** をクリックし、フォルダ名に「 **minilake-out2**（任意）」と入力し、 **[保存]** をクリックします。
2. AWS マネジメントコンソールのサービス一覧から **AWS Glue** を選択し、 **[AWS Glue]** の画面の左ペインから **[ジョブ]** を選択し、 **[ジョブ]** 画面から「 **minilake1**（任意）」ジョブにチェックを入れ、 **[アクション]** から **[スクリプトの編集]** をクリックします。



1. 続いての画面の右側のスクリプト編集箇所で「 **applymapping1** 」と「 **datasink4** 」の2箇所に対して編集を加えます。既存の「 **applymapping1** 」と「 **datasink4** 」の行をコメントアウトし、目印に「###1」の行を入れ、それぞれの行の下にコマンドリファレンスのコードをコピーアンドペーストします。

* **applymapping1**：Partition\_0などは分かりづらいので、applymapping関数でyearやmonthに変換しています。

**[コメントアウト対象]**

|  |
| --- |
| #applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [("timestamp", "string", "timestamp", "string"), ("alarmlevel", "string", "alarmlevel", "string"), ("host", "string", "host", "string"), ("user", "string", "user", "string"), ("number", "string", "number", "string"), ("text", "string", "text", "string")], transformation\_ctx = "applymapping1") |

**[追記内容]**

|  |
| --- |
| applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [("timestamp", "string", "timestamp", "string"), ("alarmlevel", "string", "alarmlevel", "string"), ("host", "string", "host", "string"), ("user", "string", "user", "string"), ("number", "string", "number", "string"), ("text", "string", "text", "string"),("partition\_0", "string", "year", "string"), ("partition\_1", "string", "month", "string"), ("partition\_2", "string", "day", "string"), ("partition\_3", "string", "hour", "string")], transformation\_ctx = "applymapping1") |

* **datasink4**：
  + **バケット名は、各自の命名に合わせて修正します。**
  + write\_dynamic\_frame の partitionKeys オプションを使い、パーティション分割して出力指定します。

**[コメントアウト対象]**

|  |
| --- |
| #datasink4 = glueContext.write\_dynamic\_frame.from\_options(frame = dropnullfields3, connection\_type = "s3", connection\_options = {"path": "s3://[S3 BUCKET NAME]/minilake-out"}, format = "parquet", transformation\_ctx = "datasink4") |

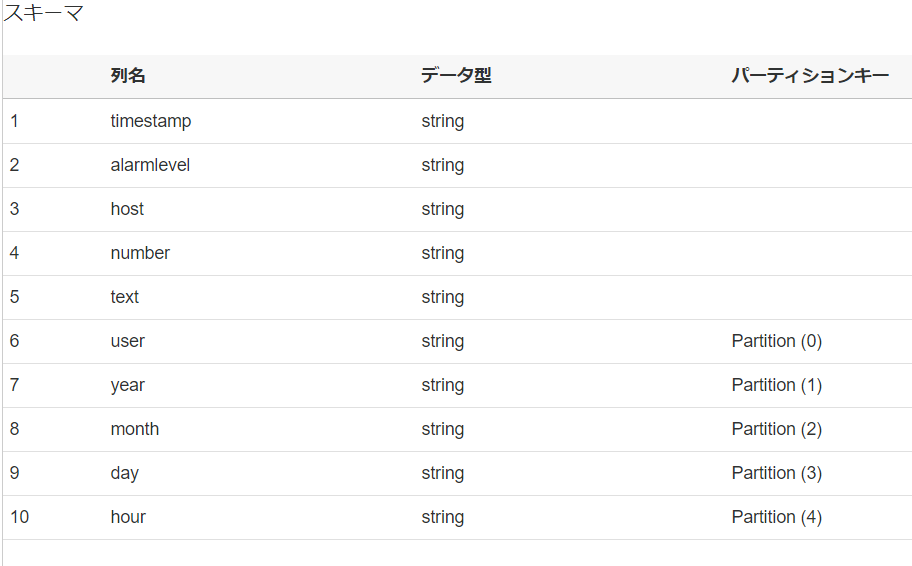
**[追記内容]**

|  |
| --- |
| datasink4 = glueContext.write\_dynamic\_frame.from\_options(frame = dropnullfields3, connection\_type = "s3", connection\_options = {"path": "s3://**[S3 BUCKET NAME]**/minilake-out2", "partitionKeys": ["user", "year", "month", "day", "hour"]},format = "parquet", transformation\_ctx = "datasink4") |

1. Glue のジョブの画面の上部にある **[ジョブの実行]** をクリックし、ポップアップした画面で **[ジョブを今すぐ保存して実行]** をクリックし、 **[ジョブの実行]** をクリックします。別タブでGlueを管理画面を再度開き、ジョブのminilake1(任意)のチェックボックスをクリックするとジョブの実行状況がわかります。作業は数分かかります。ステータスがSucceededになれば完了です。
2. AWS マネジメントコンソールのサービス一覧から **AWS Glue** を選択します。**[AWS Glue]** 画面の左ペインから **[クローラ]** を選択し、 **[クローラの追加]** をクリックします。
3. **[クローラの名前]** に「 **minilake-out2**（任意）」と入力し、 **[次へ]** をクリックします。続いての画面もそのまま **[次へ]** をクリックします。
4. **[インクルードパス]** に「 s3:// **[S3 BUCKET NAME]** /minilake-out2（任意）」を設定し、 **[次へ]** をクリックします。

**注：[S3 BUCKET NAME] には、ご自身で作成されたS3バケットの名前を入力ください。**

1. **[別のデータストアの追加]** 画面もそのままとし、 **[次へ]** をクリックします。
2. **[IAM ロールの選択]** 画面において、 **[既存の IAM ロールを選択]** にチェックを入れ、作成したロールの「 **handson-minilake**（任意）」を選択し、 **[次へ]** をクリックします。
3. **[このクローラのスケジュールを設定する]** 画面において、 **[オンデマンドで実行]** のままにし、そのまま **[次へ]** をクリックします。
4. **[データベース]** に「 **minilake**（任意）」を選び、 **[次へ]** をクリックし、続いての画面で内容を確認し、 **[完了]** をクリックします。
5. **[AWS Glue]** 画面において、左ペインから **[クローラ]** をクリックし、作成したクローラの「 **minilake-out2**（任意）」にチェックを入れ、 **[クローラの実行]** をクリックし、クローラを実行します。ステータスがStartingに変更となりますので数分間待ちます。作業が完了するとReadyに戻るので次に進みます。
6. 左ペインの **[テーブル]** をクリックし、テーブル名「 **minilake\_out2**（任意）」をクリックし、スキーマ情報を確認します。



### Athenaでクエリ比較

1. Parquet 形式でパーティション設定したテーブルに対してクエリを実行します。

**[実行クエリ例]**

|  |
| --- |
| SELECT count(user) FROM "minilake"."minilake\_out2" where user='uchida' and timestamp >= '2020-07-21 13%' AND timestamp <= '2020-07-21 18%'; |

**注**：本日のログが出た時間が入るように設定して下さい。

[実行結果例]

|  |
| --- |
| (実行時間: 2.63 秒, スキャンしたデータ: 1.83 KB) |

**注**：今回の例では、Parquet 形式でパーティションを設定した場合において、スキャン量の減少は確認できますが、個々のファイルサイズが小さいこともあり、実行時間の改善は見られないでしょう

## まとめ

実行したいクエリに対して最適なデータの形への変換処理をするために、サーバーレスでのETL処理パイプラインを実行しました。

これですべてのハンズオンが終了しました。環境を削除される際は、下記を確認下さい。

# 後片付け

下記、後片付けを行います。

**この作業まで完了していない場合、継続して課金が発生しますので、必ず実施してください。**

* Glue のジョブ（ジョブ名:minilake1）
* Glue からクローラの削除（クローラ名:minilake1、minilake-out1、minilake-out2等）
* Glue からテーブルの削除（テーブル名:minilake\_out1、minilake\_out2等）
* Glue からデータベースの削除（データベース名:spectrumdb）
* IAMロール handson-minilake、handson-minilake-dwhの削除
* S3 バケットの削除（バケット名: [ご自身で作成されたS3バケット名]）
* Redshift クラスターの削除
* Redshiftのクラスターサブネットグループの削除（handson-minilake-dwh）
* Cloudwatch Logs の削除（ロググループ名以下）
* リージョン
  + /aws-glue/crawlers/の”minilake”関連ストリーム
  + /aws-glue/jobs/error、 /aws-glue/jobs/outputのストリームはジョブIDごとに作成されます。Glueをいままで使っていなければロググループごと消してください。
* CloudFormation でスタック削除（VPC、EC2 やEIP の削除）
* （作成いただいた場合）キーペアの削除（キーペア名：handson）

後片付けは以上です。