

[日付]

はじめてのモダンデータ解析基盤

[文書のサブタイトル]

KAMEDA, HARUNOBU

AMAZON CORPORATE

目次

目次.....	1
1. はじめに.....	3
1.1. 本ハンズオンのゴール.....	3
1.2. 準備事項.....	3
2. ハンズオンの概要.....	4
2.1. ハンズオン全体を通しての注意事項.....	4
2.2. ハンズオンの構成.....	4
3. 準備.....	6
3.1. 事前準備.....	6
3.1.1. キーペアの作成.....	6
3.2. S3 の設定.....	7
3.2.1. S3 バケットの作成.....	7
3.3. CloudFormation の実行.....	7
3.3.1. VPC, EC2, Kinesis Data Firehose を CloudFormation で構築.....	7
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.....	10
4.1. Glue Crawler, Athena の設定変更.....	10
4.1.1. IAM ロールのポリシー追加.....	10
4.1.2. Glue Crawler を使ったスキーマの自動作成.....	12
4.1.3. Athena でクエリ実行.....	15
4.1.4. Athena で CTAS (Create Table AS) クエリの実行.....	17
4.1.5. S3 Select でクエリ実行.....	19
4.2. まとめ.....	20
5. クラウド DWH を使用したデータ分析.....	21
5.1. Redshift の構築.....	21
5.2. Redshift への接続.....	24
5.2.1. Redshift への接続.....	24
5.2.2. Redshift にデータロード.....	25
5.2.3. Redshift Spectrum の使用.....	29
5.2.4. Redshift からデータエクスポート.....	31
5.3. まとめ.....	33
6. サーバーレスでデータの ETL 処理.....	33
6.1. Glue の ETL 処理.....	34
6.1.1. IAM ロールにポリシーを追加.....	34
6.1.2. Glue で ETL ジョブ作成と実行.....	35
6.1.3. Glue クローラの作成と実行.....	38
6.1.4. Athena でクエリ比較.....	39
6.1.5. Glue ジョブで Parquet とパーティショニングを実行.....	41

6.1.6. Athena でクエリ比較	44
6.2. まとめ	44
7. 後片付け.....	46

1. はじめに

1.1. 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。

1.2. 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)

2. ハンズオンの概要

2.1. 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注意ください。

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。

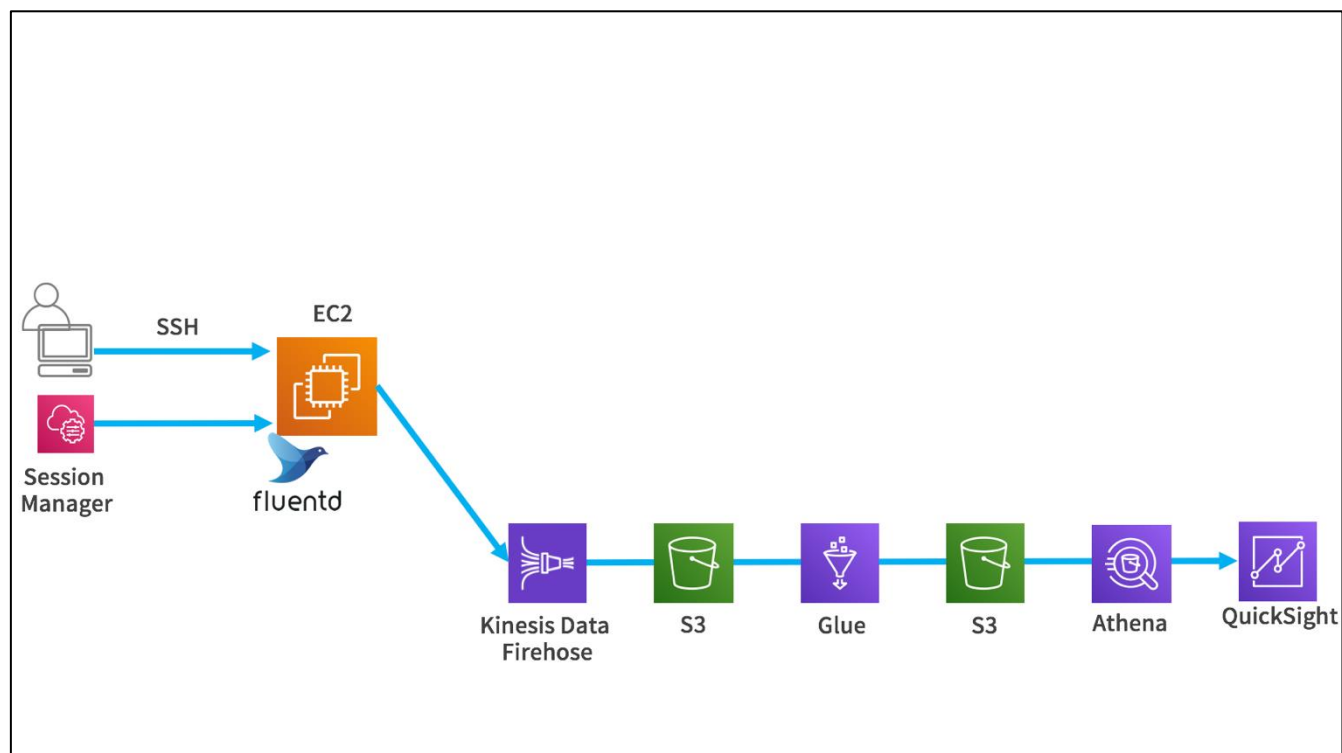
2.2. ハンズオンの構成

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
- 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
- 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。



3. 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。

3.1. 事前準備

3.1.1. 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。

2. 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。

3. 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。

▼ ネットワーク & セキュリティ

セキュリティグループ

New

Elastic IP New

プレースメントグループ

New

キーペア New

ネットワークインターフェイス

[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。

3.2.S3 の設定

3.2.1. 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。



2. 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いち

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。ブロックパブリックアクセスのバケット設定、バケットのバージョンニング、デフォルトの暗号化は、設定変更せず進めます。

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。

3.3.CloudFormation の実行

3.3.1. 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 が見つけれられない場合、検索窓に「cloudform」などを入力し、選択します。
[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。

注：誤って「既存のリソース仕様(リソースをインポート)」を選択した場合、「リソースのインポートではサポートされていません」というエラーが出力されます。

2. [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。
3. 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。
4. [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。
5. 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。
6. 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。

機能

i The following resource(s) require capabilities: [AWS::IAM::Role]

このテンプレートには、Identity and Access Management (IAM) リソースが含まれています。これらのリソースを個別に作成し、それぞれに最小限必要な権限を与えるかどうか確認してください。さらに、カスタム名が付けられているか確認してください。カスタム名が、ご利用の AWS アカウント内で一意のものであることを確認してください。 [詳細はこちら](#)

☐ 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。

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。

handson-minilake

削除更新するスタックアクション ▼

スタックの情報イベントリソース出力パラメータテンプレート変更セット

概要

スタック ID	説明
arn:aws:cloudformation:ap-northeast-1:294963776963:stack/handson-minilake/c6b33230-c738-11ea-b7f7-063a64457ffa 🔗	-
ステータス	状況の理由
🟢 CREATE_COMPLETE	-

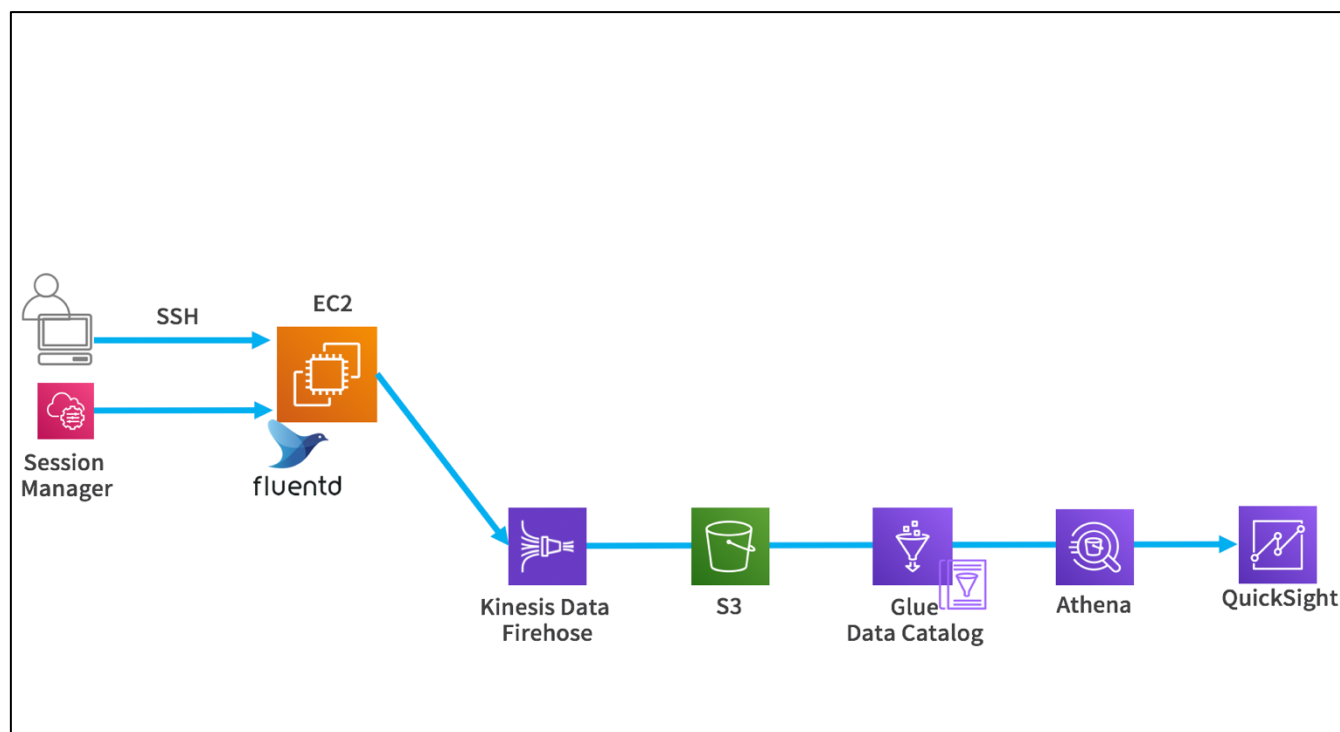
エラーとなった場合は、イベントタブを開き、赤字で【FAILED】などとなっている部分を質問窓口にコピーしてください。

7. 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）

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

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ます。



4.1. Glue Crawler, Athena の設定変更

4.1.1. 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。

Identity and Access Management (IAM)

ダッシュボード

▼ アクセス管理

グループ

ユーザー

ロール

ポリシー

ID プロバイダー

アカウント設定

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。

許可 | 信頼関係 | タグ | アクセスアドバイザー | セッションを取り消す

許可ポリシー (1)
最大 10 個の管理ポリシーを添付できます。

シミュレート 削除

アクセス許可を追加 ▲

ポリシーをアタッチ
インラインポリシーを作成

Q ポリシーをプロパティまたはポリシー名でフィルタし、Enter キーを押します。

<input type="checkbox"/>	ポリシー名	タイプ	説明
<input type="checkbox"/>	AmazonKinesisFirehoseFullAccess	AWS 管理	Provides full access to

3. 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。
4. **[AWSGlueServiceRole]** と **[AmazonS3ReadOnlyAccess]** がアタッチされたことを確認します。

アクセス権限

信頼関係

タグ (1)

アクセスアドバイザー

セッションの無効化

▼ Permissions policies (3 適用済みポリシー)

ポリシーをアタッチします

ポリシー名 ▼

▶  [AWSGlueServiceRole](#)

▶  [AmazonS3ReadOnlyAccess](#)

さらに 1 を表示

5. **[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。
6. **[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ーしてみてください。

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

4.1.2. 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] Crawlers** を選択し、**[クローラの追加]** をクリックします。

AWS Glue

データカタログ

データベース

テーブル

接続

| クローラ

分類子

設定

2. **[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(Specify crawler source type)もそのまま **[次へ]** をクリックします。
3. **[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。
4. **[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。
5. **[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
6. 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。
7. **[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。

クローラの出力を設定する

データベース ⓘ

minilake

データベースの追加

8. 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。
9. 続いての画面の内容を確認し、**[完了]** をクリックします。
10. **[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。

クローラの追加

クローラの実行

アクション ▼

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。

11. 「**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。

スキーマ

	列名	データ型	パーティションキー	コメント
1	timestamp	string		
2	alarmlevel	string		
3	host	string		
4	user	string		
5	number	string		
6	text	string		
7	partition_0	string	Partition (0)	
8	partition_1	string	Partition (1)	
9	partition_2	string	Partition (2)	
10	partition_3	string	Partition (3)	

4.1.3. 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。
3. **クエリエディタを開き[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。

データソース データソースを接続する

AwsDataCatalog

データベース

minilake

テーブルとビューのフィルタリング...

▼ テーブル (1) テーブルの作成

▼ 0200716_handson_minilake_hkameda18730105... ⋮

partition_0 (string) (パーティション化)

partition_1 (string) (パーティション化)

partition_2 (string) (パーティション化)

partition_3 (string) (パーティション化)

partition_4 (string) (パーティション化)

4. クエリ結果が画面下部に表示されることを確認します。

結果

	timestamp ▼	alarmlevel ▼	host ▼	user ▼	number ▼	text ▼	partition_0 ▼	partition_1 ▼	partition_2 ▼	partition_3 ▼	partition_4 ▼
1	2020-07-16 17:08:01+0900	WARNING	prd-web001	nomura	1001	This is Warning.	minilake-in1	2020	07	16	08
2	2020-07-16 17:08:01+0900	INFO	prd-web001	uehara	1001	This is Information.	minilake-in1	2020	07	16	08
3	2020-07-16 17:08:01+0900	WARNING	prd-db02	uehara	1001	This is Warning.	minilake-in1	2020	07	16	08
4	2020-07-16 17:08:01+0900	ERROR	prd-db02	imai	1001	This is ERROR.	minilake-in1	2020	07	16	08
5	2020-07-16 17:08:01+0900	WARNING	prd-ap001	ohmura	1001	This is Warning.	minilake-in1	2020	07	16	08
6	2020-07-16 17:08:01+0900	CRITICAL	prd-web001	ohmura	1001	This is Critical!!!	minilake-in1	2020	07	16	08
7	2020-07-16 17:08:01+0900	INFO	prd-web001	imai	1001	This is Information.	minilake-in1	2020	07	16	08
8	2020-07-16 17:08:01+0900	INFO	prd-batch01	uehara	1001	This is Information.	minilake-in1	2020	07	16	08
9	2020-07-16 17:02:01+0900	ERROR	prd-db02	nomura	1001	This is ERROR.	minilake-in1	2020	07	16	08
10	2020-07-16 17:02:01+0900	WARNING	prd-web002	nomura	1001	This is Warning.	minilake-in1	2020	07	16	08

5. 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";
```

（テーブル名 : minilake_in1 が異なる場合は置き換えて下さい）

[実行結果例]

(実行時間: 2.89 秒, スキャンしたデータ: 140.47 KB)

6. 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ださい。

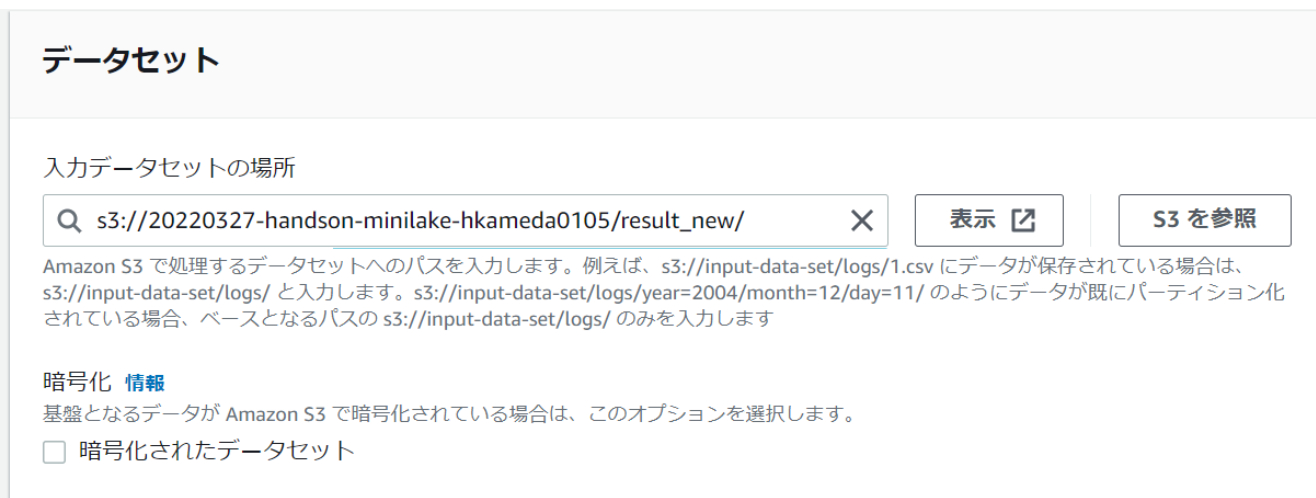
作業日によっては、partition_0 にテーブル名が挿入されている場合があります。

4.1.4. Athena で CTAS (Create Table AS) クエリの実行

1. 上記、クエリエディタの [名前を付けて保存] のとなりの [作成] をクリックし、[クエリからテーブルを作成] を続けてクリックします。



2. テーブル名に「minilake_in1_new」、出力場所に「s3://[S3 BUCKET NAME]/result_new/ (任意)」と入力します。



3. データ形式で Parquet を指定します

データ形式

データ形式

Parquet ▼

4. 次の画面で、内容を確認し、**[作成]**をクリックします。

5. 左側に「minilake_in1_new」が作成されたことが確認できます。

データソース データソース
を接続する

AwsDataCatalog ▼

データベース

minilake ▼

テーブルとビューのフィル

▼ テーブル (2)
テーブルの作成

▶ minilake_in1 (パーテ... ⓘ

▶ minilake_in1_new ⓘ

6. クエリエディタで下記 SQL を実行します。「クエリの実行」ボタンがクリック出来ない場合はブラウザをリロードしてみてください。

```
SELECT * FROM "minilake"."minilake_in1_new"
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ださい。

Parquet 形式で作成した結果、スキャンしたデータが少なくなっていることが分かります。

4.1.5. 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

3. 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。



4. [アクション] → [S3 Select を使用したクエリ]をクリックします

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

形式

- ☐ CSV
- ☒ JSON
- ☐ Apache Parquet

JSON コンテンツタイプ

- ☒ 行
入力データの各行には 1 つの JSON オブジェクトが含まれています
- ☐ ドキュメント
1 つの JSON オブジェクトを入力複数の行にまたがることができます。

圧縮

- ☒ なし
- ☐ GZIP
- ☐ BZIP2

5. SQL を入力し、**[SQL の実行クエリ]**をクリックします。

```
select * from s3object s limit 5;
```

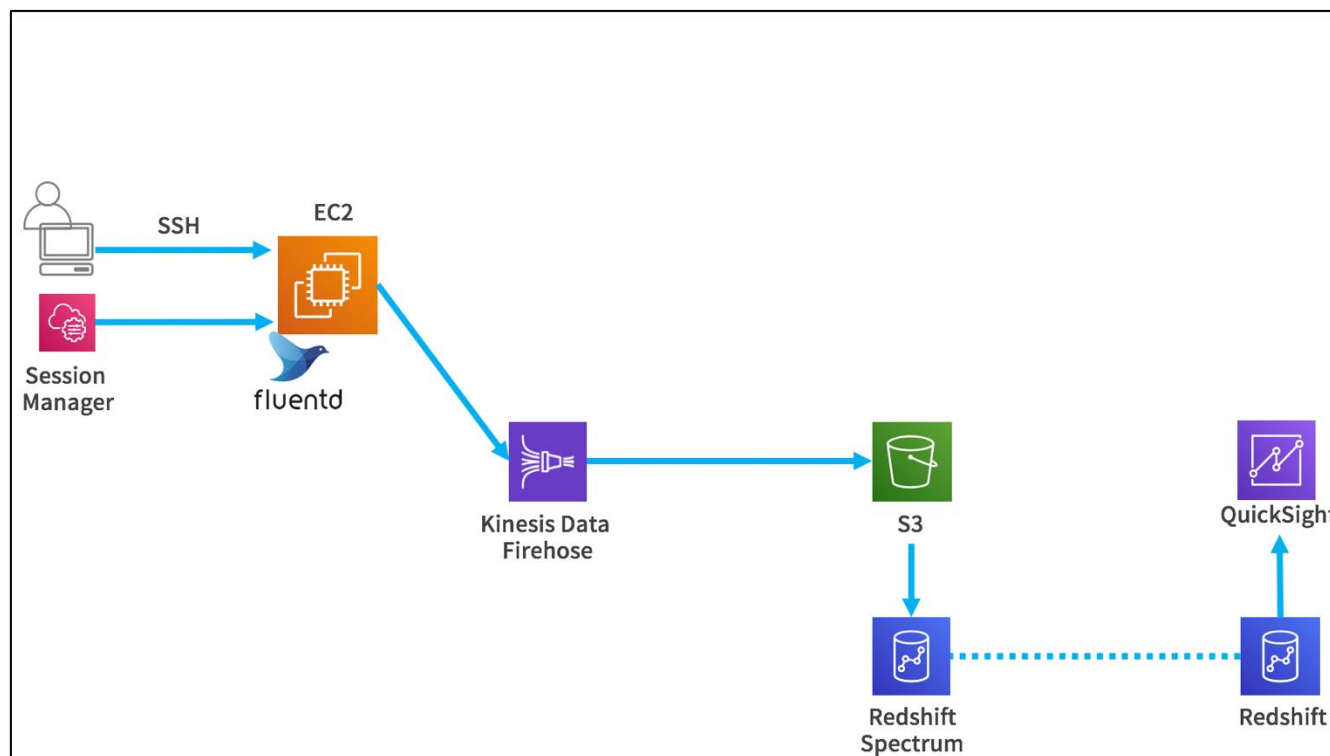
```
select s."host" from s3object s;
```

4.2.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。

5. 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 などでも可視化も可能です。



5.1.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。

出力 (3)

Q 検索結果の出力

キー ▲	値 ▼	説明 ▼	エクス
AllowIPAddress	54.150.62.87	EC2 PublicIP	-
PrivateSunetID	subnet-0ec47de2c34f515d6	Private Subnet ID	-
VPCID	vpc-055145742c67204ef	VPC ID	-

- 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。
- [サブネットグループ]** 項目より、**[サブネットグループの管理]**をクリックします。クラスターサブネットグループ画面より、右上の**[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。
- 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

サブネットを追加

VPC
クラスターサブネットグループに使用するサブネットを含む VPC の識別子を選択します。

handson-minilake
vpc-055145742c67204ef ▼

この VPC のすべてのサブネットを追加する

アベイラビリティゾーン サブネット

ap-northeast-1a (2) ▼ subnet-0ec47de2c34f515d6 ▼ サブネットを追加

このクラスターサブネットグループのサブネット Remove all

アベイラビリティゾーン	サブネット ID	CIDR ブロック	アクション
ap-northeast-1a	subnet-0ec47de2c34f515d6	10.0.0.32/27	削除

5. [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。

6. 左ペインから [クラスター] を選択し、[クラスターを作成] をクリックします。

Redshift serverless **New**

Provisioned clusters dashboard

▼ クラスター

- リザーブドノード
- スナップショット

クエリエディタ

Query editor v2 [🔗](#)

クエリとロード

クラスター (0) 情報 🔄 Query data ▼ アクション ▼ クラスターを作成

🔍 プロパティまたは値でクラスターをフィルタリング < 1 > ⚙️

クラスター	クラスター名前空間	状態	使用されたストレージ	CPU 使用率
クラスターがありません Amazon Redshift クラスターを作成する				

クラスターを作成

7. 以下の値を入力します。

- クラスター識別子 : handson-minilake-dwh (任意)
- 本番稼働
- ノードの種類 : dc2.large を選択
- ノード : 1
- 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)

- 管理者ユーザー名 : admin (任意)
- 管理者のパスワード : 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。

8. **[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**

VPC セキュリティグループ

この VPC セキュリティグループは、クラスターが VPC で使用できるサブネットと IP 範囲を定義します。

1 つ以上のセキュリティグループを選択する ▼

handson-minilake-private-sg ✕
sg-05130421a2a9e8354

9. 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。

5.2.Redshift への接続

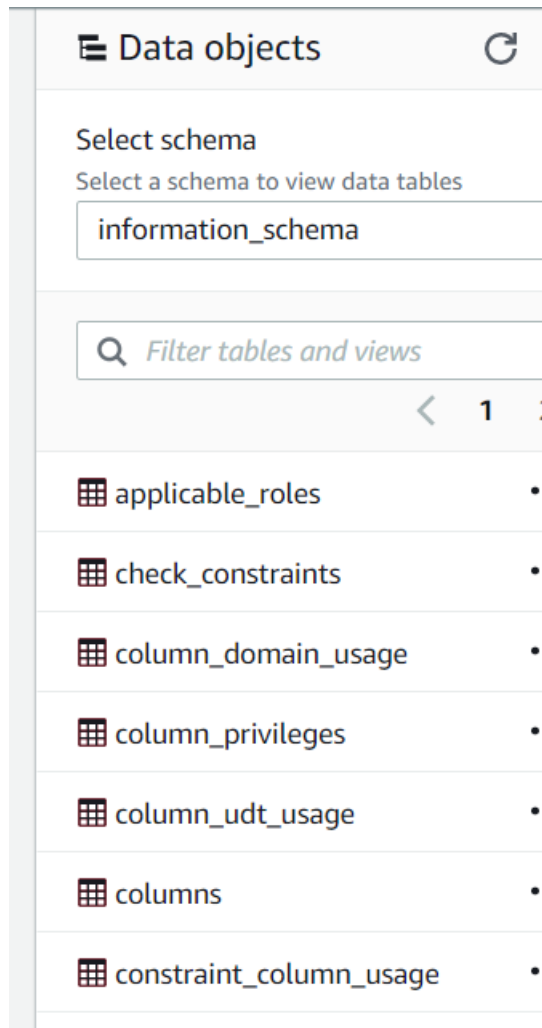
5.2.1. Redshift への接続

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 **[クエリエディタ]** 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 : dev (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）（セッションが維持されていれば求められません）

3. 左ペインの[select schema]より[information_schema]を選択し、にテーブル一覧が表示されれば、ログイン成功です。



5.2.2. 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、**[次へ]** をクリックします。

ユースケースの選択

Redshift

Allows Redshift clusters to call AWS services on your behalf.

Redshift - Customizable

Allows Redshift clusters to call AWS services on your behalf.



Redshift - Scheduler

Allow Redshift Scheduler to call Redshift on your behalf.

3. **[AmazonS3ReadOnlyAccess]** を選択して、**[次へ]** をクリックします。
4. **[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。
5. 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。

[ロール](#) > handson-minilake-dwh

概要

ロール ARN	arn:aws:iam::294963776963:role/handson-minilake-dwh 
ロールの説明	Allows Redshift clusters to call AWS services on your behalf. 編集
インスタンスプロファイル ARN	

6. **[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。



7. さきほど作成した IAM ロール「**handson-minilake-dwh**（任意）」を選択し、**[IAM ロールを関連づける]**をおし**[変更を保存]**をクリックします。



クラスター一覧画面で **[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。

<input type="checkbox"/>	クラスター ▲	状態 ▼
<input type="checkbox"/>	handson-minilake-dwh dc2.large 1 個のノード 1...	Available 変更中

<input type="checkbox"/>	クラスター ▲	状態 ▼
<input type="checkbox"/>	handson-minilake-dwh dc2.large 1 個のノード 1...	Available

8. 左ペインから **[クエリエディタ]** 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 );
```

再接続の画面が出た場合、「最近の・・・」を選んでください。

9. 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。

10. **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;
```

5.2.3. 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。(dwh の方です。とても間違いが多いので注意)
2. **[許可]タブ**から**[アクセス許可を追加]** を選んで、**[ポリシーをアタッチ]** を選びます。
3. **[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。
4. **[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 'spectrumbd'  
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。

5. 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。

6. 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;
```

7. 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、LOCATION 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。

8. 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。

9. 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。
10. 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;
```

5.2.4. Redshift からデータエクスポート

1. Redshift から S3 にアクセスして、データをエクスポートするために 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**」のロール名をクリックします。

Identity and Access Management (IAM)

ダッシュボード

▼ アクセス管理

グループ

ユーザー

ロール

ポリシー

ID プロバイダー

アカウント設定

2. [アクセス許可を追加] から、[ポリシーをアタッチ] を選びます。



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。(バケットに対する GET, LIST, PUT 権限があれば十分です)
4. [**AmazonS3FullAccess**] がアタッチされたことを確認します。
5. [**Redshift**] に戻り、画面左ペインの [**エディタ**] から SQL を実行します。

```
UNLOAD ('SELECT * FROM ec2log')
TO 's3://[S3 BUCKET NAME]/export/'
FORMAT AS PARQUET
CREDENTIALS 'aws_iam_role=[IAM ROLE ARN]';
```

注：[**S3 BUCKET NAME**] には、作成済みの S3 バケット名を、[**IAM ROLE ARN**] には、「**handson-minilake-dwh**（任意）」の値を入力します。

6. 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。
7. 出力先(S3 バケット名/export)のバケット、パスを確認します。ファイルを選択し、「S3 Select」タブを選択します。

0000_part_00.parquet 最新バージョン ▼

概要

プロパティ

アクセス権限

S3 Select

S3 Select では、SQL 式を使用して 1 つの CSV、JSON、または Parc
ファイルおよびサーバー側の暗号化ファイルがサポートされます。コ
抽出できます。より大きなファイルまたはより多くのレコードを操作

より複雑な SQL 式を必要とするデータを S3 で分析する場合は、以下

ファイル形式 ⓘ

- ☐ CSV
- ☐ JSON
- ☒ Parquet

サーバー側の暗号化 ⓘ

AES-256

サイズ

4.3 KB

圧縮 ⓘ

なし

- デフォルトで記載されているクエリを実行し、内容を確認します。(エクスポートデータの内容
確認)

5.3.まとめ

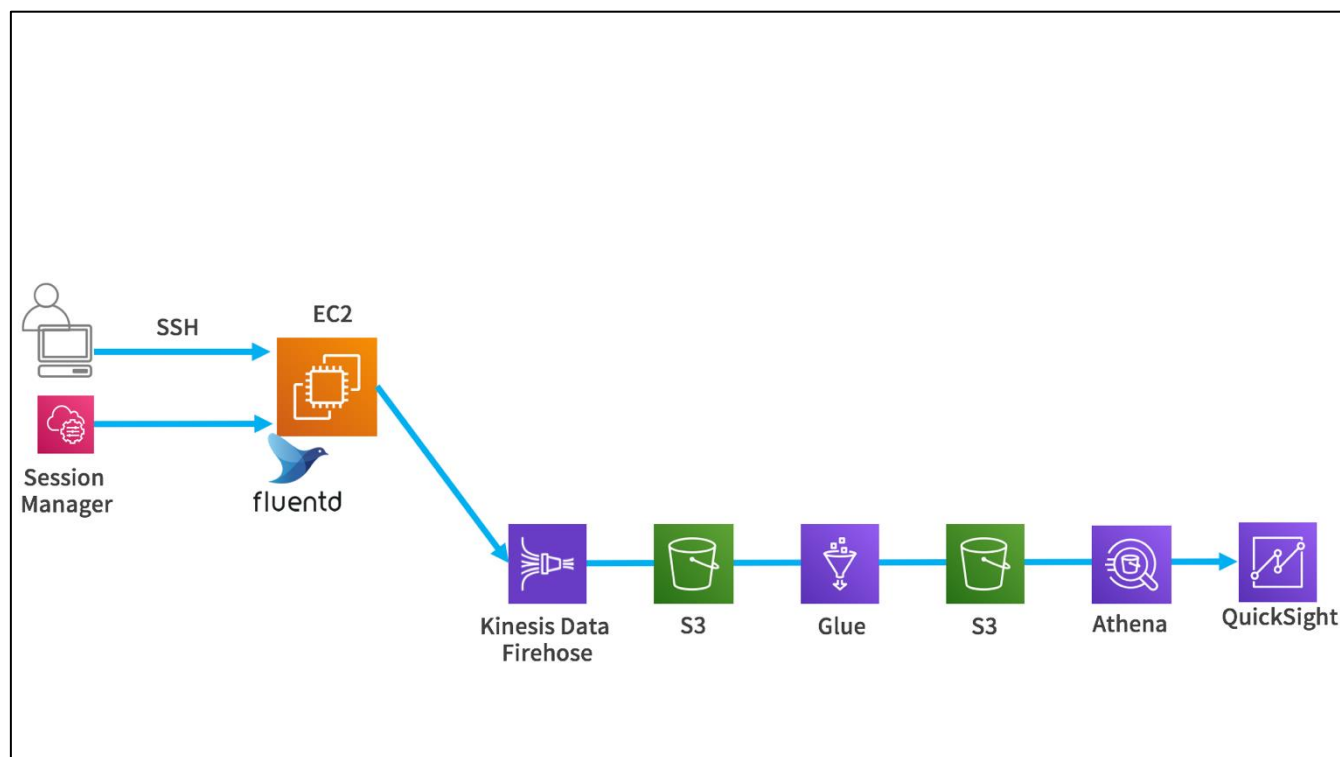
Redshift Spectrum のスキーマとデータベースを作成することができるように、作成した IAM
ストリーミングデータを直接データストアに永続化し、長期間の保存を可能にした上で、 DWH に読
み込み分析を行う基盤ができました

6. 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ニ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。

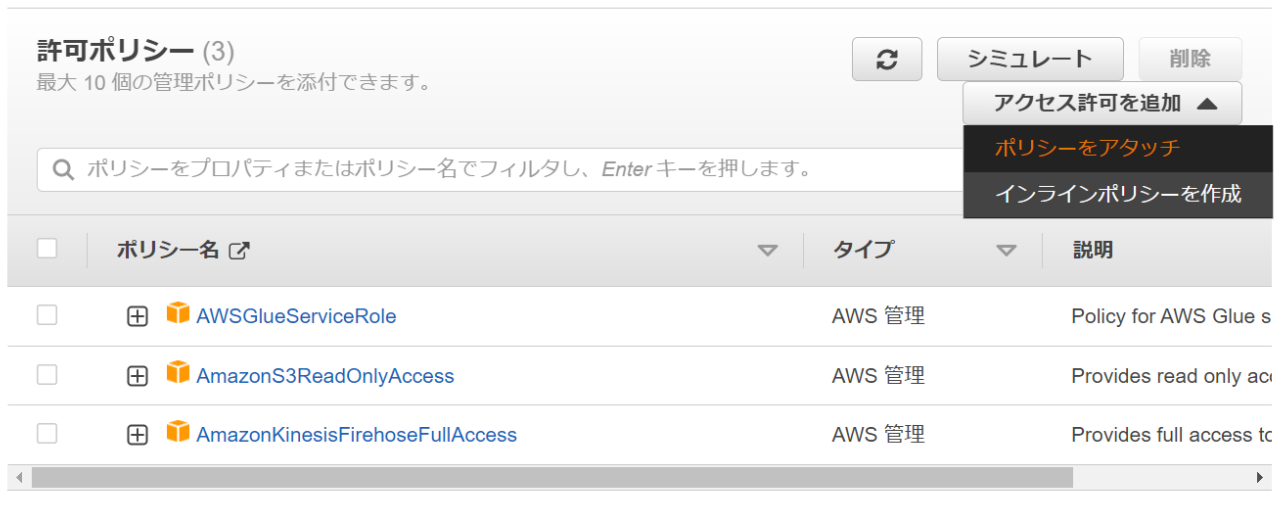


6.1.Glue の ETL 処理

6.1.1. 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。
（注意：dwh じゃない方です。）

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. [許可] タブを選択し、[ポリシーをアタッチ] 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 **[許可]** タブを選択し、**[AmazonS3FullAccess]** がアタッチされたことを確認します。



6.1.2. 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。(minilake-in1 と同じ場所に作成します)

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(legacy)]** をクリックし、**[ジョブの追加]** をクリックします。
3. **[名前]** に「**minilake1**（任意）」、**[IAM ロール]** に「**handson-minilake**（任意）」を入力します。
4. **[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。
5. **[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 **[最大同時実行数]** を「**2**」と入力し、画面下の **[次へ]** をクリックします。
6. **[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。
7. **[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。
8. **[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
 - **形式** : Parquet
 - **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。
9. **[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。
10. 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]** をクリックします。

ジョブ: minilake アクション ▼ 保存 ジョブの実行 ダイアグラムの生成 カールソにテンプレートを挿入 ソース ターゲット ターゲット位置 トランスフォーム 検

データベースの名前 minilake
テーブル名 minilake_minilake_in

変換の名前 ApplyMapping

変換の名前 ResolveChoice

変換の名前 DropNullFields

パス s3://20180819-handson-minilake-ueharaxx/minilake-out

SCRIPT

```

1  #!python
2  from aws glue.transforms import *
3  from aws glue.utils import getResolvedOptions
4  from pyspark.context import SparkContext
5  from aws glue.context import GlueContext
6  from aws glue.job import Job
7
8  ## @params: [JOB_NAME]
9  args = getResolvedOptions(sys.argv, ['JOB_NAME'])
10
11  sc = SparkContext()
12  glueContext = GlueContext(sc)
13  spark = glueContext.spark_session
14  job = Job(glueContext)
15  job.init(args['JOB_NAME'], args)
16
17  ## @args: [database = "minilake", table_name = "minilake_minilake_in", transformation_ctx = "datasource0"]
18  ## @return: datasource0
19  ## @inputs: []
20  datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "minilake", table_name = "minilake_minilake_in", transformation_ctx = "datasource0")
21
22  ## @type: ApplyMapping
23  ## @args: [mapping = [{"timestamp", "string", "timestamp", "string"}, {"alarmlevel", "string", "alarmlevel", "string"}, {"host", "string", "host", "string"}]]
24  ## @return: applymapping1
25  ## @inputs: [frame = datasource0]
26  applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [{"timestamp", "string", "timestamp", "string"}, {"alarmlevel", "string", "alarmlevel", "string"}, {"host", "string", "host", "string"}])
27
28  ## @type: ResolveChoice
29  resolvechoice1 = ResolveChoice.apply(applymapping1, transform = "ResolveChoice", allow_unresolved_columns = True)
30
31  ## @type: DropNullFields
32  dropnullfields1 = DropNullFields.apply(resolvechoice1)
33
34  dropnullfields1.write.saveAsTextOutput(s3://20180819-handson-minilake-ueharaxx/minilake-out, format = "text")
35
36  job.commit()

```

LOG

```

download: s3://test-glue00/se2/script/minilake to ./script_2018-08-19-06-23-23.py
SCRIPT_URL = /tmp/g-180aa40cdbc4d341ab1aeda282773c93eea6f58-2278422265748487581/script_2018-08-19-06-23-23.py
/usr/lib/spark/bin/spark-submit --conf spark.hadoop.yarn.resourcemanager.hostname=ip-172-31-0-195.ap-northeast-1.compute.internal:8020 --conf spark.hadoop.yarn.resourcemanager.address=ip-172-31-0-195.ap-northeast-1.compute.internal:8032 --conf spark.dynamicAllocation.enabled=true --conf spark.shuffle.service.enabled=true --conf spark.dynamicAllocation.minExecutors=1 --conf spark.dynamicAllocation.maxExecutors=2 --conf spark.executor.memory=5g --conf spark.executor.cores=4 --name tape --master yarn --deploy-mode cluster --jars /opt/amazon/superjar/glue-assembly-jar --files /tmp/glue-default.conf,/tmp/glue-override.conf,/opt/amazon/certs/InternalAndExternalAndAWSTrustStore.jks,/opt/amazon/certs/rds-combined-ca-bundle.pem,/opt/amazon/certs/redshift-ssl-cert.pem,/opt/amazon/certs/RDSTrustStore.jks,/tmp/image-creation-time,/tmp/g-180aa40cdbc4d341ab1aeda282773c93eea6f58-2278422265748487581/script_2018-08-19-06-23-23.py --py-files /tmp/ID/ID-glue-assembly-jar --executor-memory 5g --executor-cores 4 --py-files /tmp/ID/ID-glue-assembly-jar --py-files /tmp/ID/ID-glue-assembly-jar

```

パラメータ (任意)

このジョブを実行する前に、必要に応じてパラメータ値を確認および上書きします。変更はこの実行のみに影響します。ジョブを編集してデフォルトのパラメータ値を変更します。

- ▶ 詳細プロパティ
- ▶ モニタリングオプション
- ▶ タグ
- ▶ 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

ジョブ **minilake1** のみが実行されます。ジョブ **minilake1** の完了に依存するジョブは実行されません。ジョブを実行し、依存ジョブをトリガーするには、オンデマンドトリガーを定義します。

ジョブの実行

- 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。

ンヨノ

ジョブは、作業の抽出、変換、ロード (ETL) を実行するために必要なビジネスロジックです。ジョブ実行は、スケジュールすることで、またはイベントに引き起こされるトリガーによって開始されます。

[ユーザー設定](#)

ジョブの追加 アクション <input type="text" value="タグや属性によるフィルター"/>							表示中: 1 - 2
<input type="checkbox"/> 名前	Type	ETL 言語	スクリプトの場所	最終更新日時	ジョブのブックマーク		
<input type="checkbox"/> firststep-s3s3-hkameda	Spark	python	s3://aws-glue-sc...	18 5月 2020 12:54 午後 U...	無効化		
<input checked="" type="checkbox"/> minilake1	Spark	python	s3://aws-glue-sc...	17 7月 2020 1:06 午後 UT...	無効化		

履歴 詳細 スクリプト メトリクス							
<input type="button" value="実行メトリクスの表示"/>		<input type="button" value="ジョブのブックマークを巻き戻す"/>				表示中: 1 - 1	
実行 ID	実行ステータス	開始時刻	終了時刻	Start-up time	実行時間	タイムアウト	
<input type="radio"/> jr_60f1a8f61e75c6a893a41a5ee0c4b813dc9b...	Running	17 7月 2020 1:07 午...		0 秒	21 秒	2880 分	

12. 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。

6.1.3. 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(Specify crawler source type)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。

5. 次の画面(別のデータストアの追加)もそのまま [次へ] をクリックします。
6. [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。
7. [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。
8. [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。
9. 「**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。

クローラの追加		クローラの実行	アクション ▼	🔍 タグ
<input type="checkbox"/>	名前	スケジュール		
<input type="checkbox"/>	firststep-s3s3in-0518			
<input type="checkbox"/>	minilake-in1			
<input checked="" type="checkbox"/>	minilake-out1			

10. 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。

6.1.4. 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スキャン結果の比較を行います。

[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';
```

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';
```

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

「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

6.1.5. 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。(minilake-in1 と同じ場所です)
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、[**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。



3. 続いての画面の右側のスクリプト編集箇所「**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")
```

4. 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。
5. 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。
6. **[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。続いたの画面もそのまま **[次へ]** をクリックします。
7. **[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。

8. [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。
9. [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。
10. [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。
11. [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。
12. [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。
13. 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。

スキーマ

	列名	データ型	パーティションキー
1	timestamp	string	
2	alarmlevel	string	
3	host	string	
4	number	string	
5	text	string	
6	user	string	Partition (0)
7	year	string	Partition (1)
8	month	string	Partition (2)
9	day	string	Partition (3)
10	hour	string	Partition (4)

6.1.6. 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

6.2.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。

7. 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。