AWS Black Belt Online Seminar

Amazon Athena

Athena SQL 編

久保 和隆

Solutions Architect
2023/11

自己紹介

名前：久保 和隆（くぼ かずたか）

所属：技術統括本部
西日本ソリューション部
ソリューションアーキテクト

経歴：金融機関にて規制対応に伴う、
システム開発・運用に従事

好きな AWS サービス：

AWS Glue & AWS Lake Formation

アジェンダ

1. Amazon Athena の概要
2. Athena SQL の基本
3. Athena SQL が提供する追加機能
4. Athena SQL のパフォーマンス関連Tips
5. ユースケース
6. 料金
7. まとめ

Amazon Athena の概要

データ分析の現状

増加するデータ量

多様な
データソース

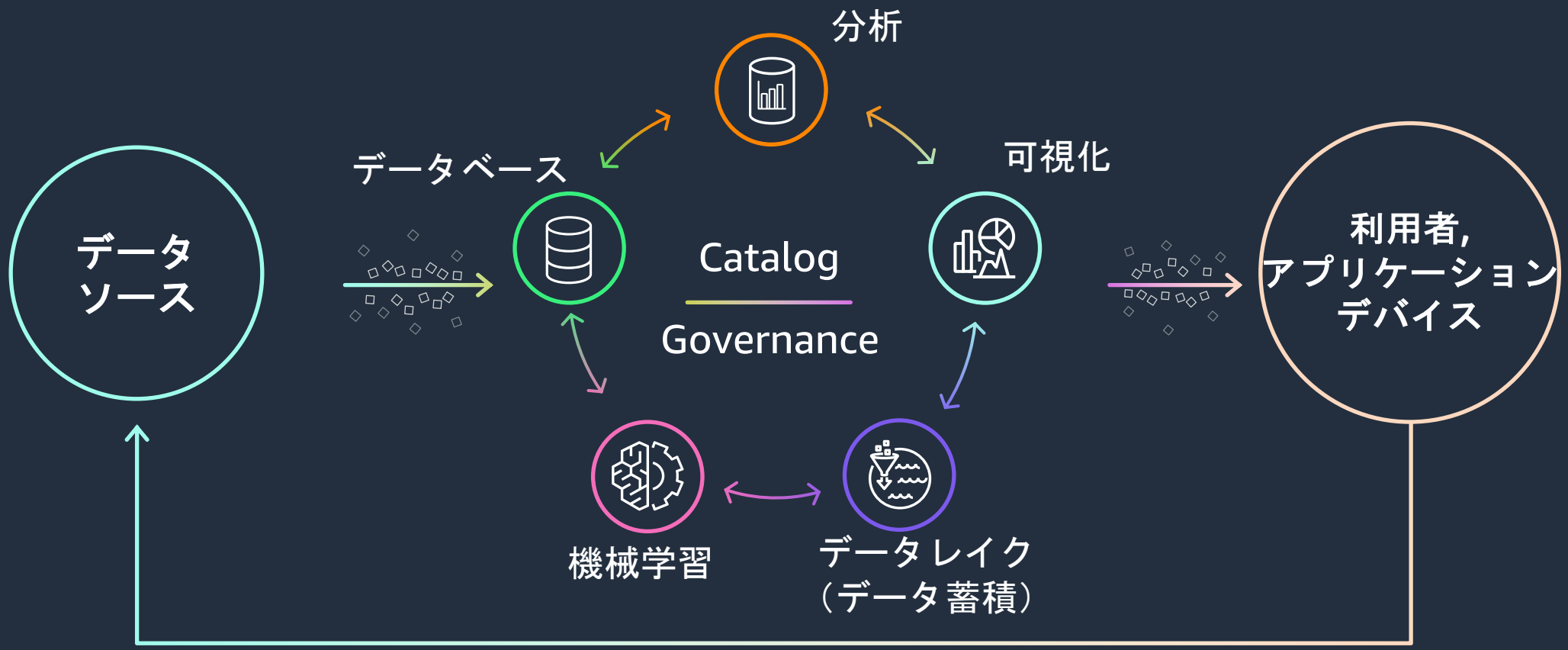
多様化する
データ形式

利用者が持つ
多様な目的

データ分析や
機械学習への活用

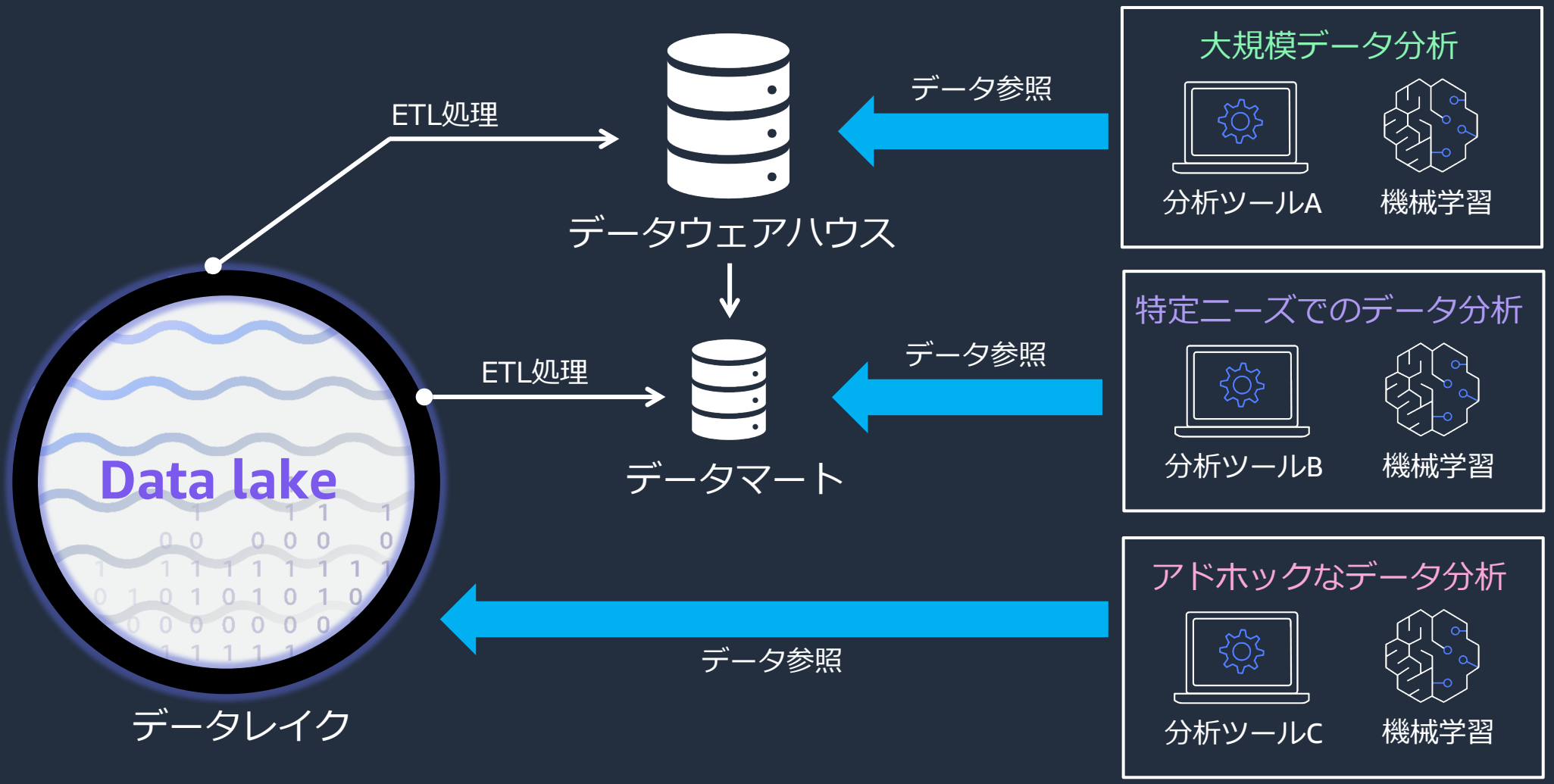
モダンデータアーキテクチャというアプローチ

- データを有効活用するためのアーキテクチャ

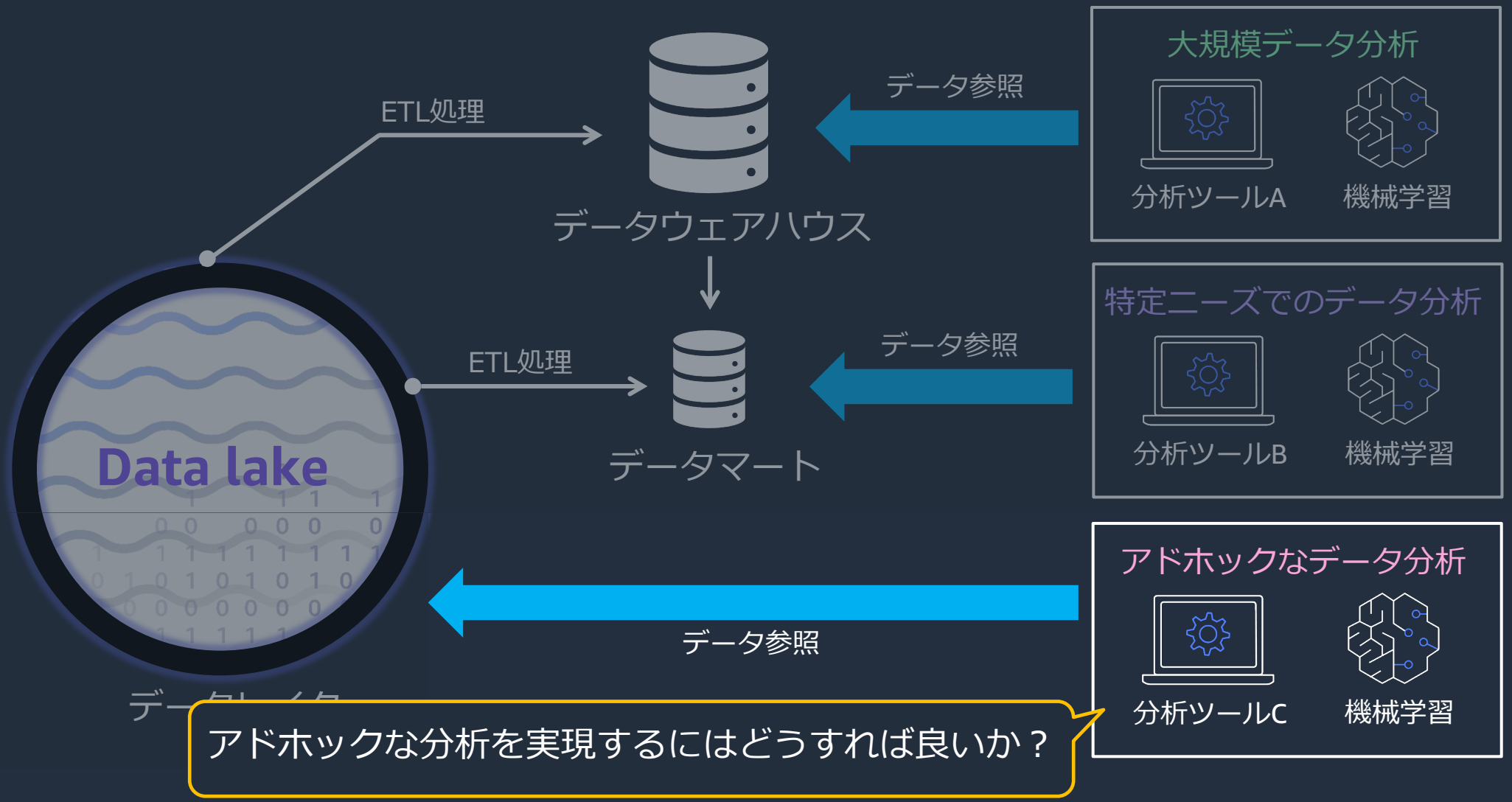


データを使用する目的に合わせ、分析サービスと
データの格納先を使い分けることでデータ活用を促進

代表的なデータアクセスパターン



代表的なデータアクセスパターン



Amazon Athena の特徴

簡単

- サーバーレスで
  セットアップが不要
- 最適化された環境を
  すぐに使用可能

インタラクティブ

- ユーザーにインタラク
  ティブな分析環境を提供
- 25以上のデータストアへ
  クエリを実行可能

高い柔軟性

- オープンソースを
  AWS に最適化
- 複数のフォーマット、
  圧縮タイプ、データ形式
  をサポート

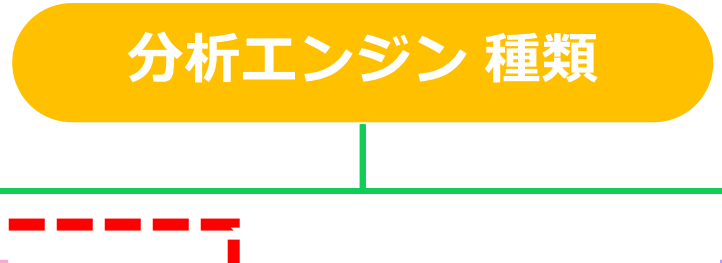
高い費用対効果

- 従量課金
- データ圧縮により、
  30% ~ 90% の
  コスト削減が可能

S3 を中心に様々なデータストアに対して、
アドホックでインタラクティブな分析が可能

Amazon Athena が持つ2種類の分析エンジン

- 分析エンジンとして、Athena は2種類のエンジンを提供



SQL (Presto/Trino)

- 標準的な SQL を使用して データソースにクエリを実行
- SQL実行時のスキャンデータ量による料金と、キャパシティ予約で指定したコンピューティングリソースに対する料金

Apache Spark

- Athena管理コンソールに組み込まれた Jupyter Notebook を使用してSpark アプリケーションを実行
- 1つのセッション内で「コードブロック」を実行、その処理に消費された コンピューティング能力による料金

本セッションではAthena SQLにフォーカスして解説

Athena SQL の基本

コンソールを使用したSQLの実行

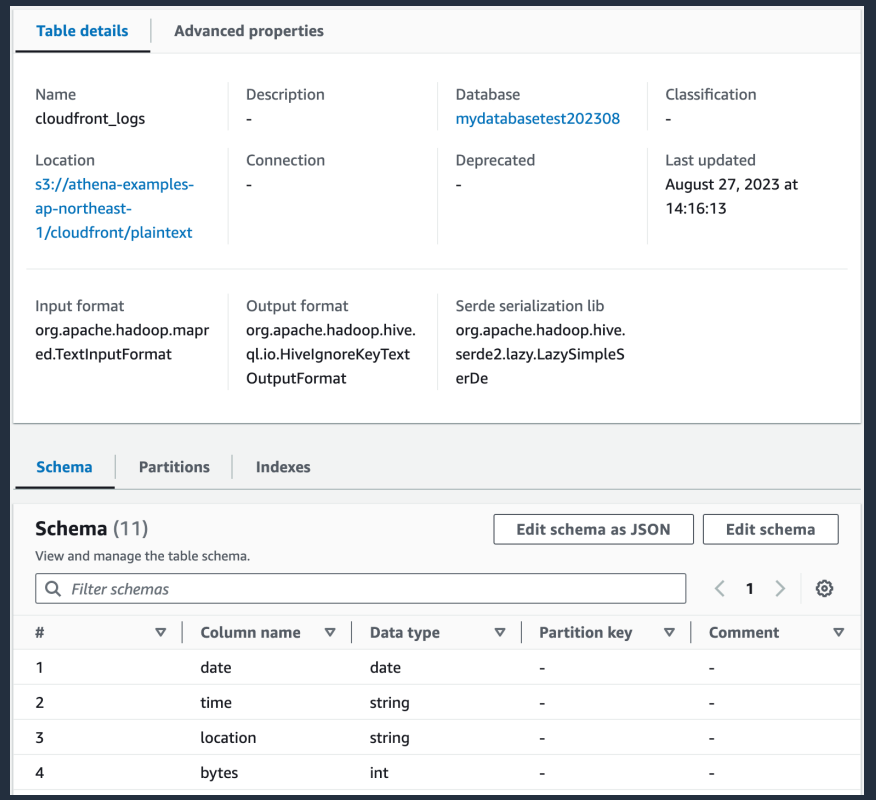
Athena管理コンソールのUIとしてクエリエディタを提供

テーブルの定義

- Athena ではクエリのためにテーブル定義が必要
  - デフォルトで AWS Glue Data Catalog 上のテーブル定義を使用

- AWS Glue Data Catalog は、Apache Hive Metastore という OSS と互換性のある、メタデータを管理するためのリポジトリ

- AWS Glue Data Catalog にテーブル定義を作成する方法は以下
  - AWS Glue Crawler
  - Athena テーブル作成フォーム
  - Hive DDL
  - AWS Glue Catalog API



AWS Glue Data Catalog で定義されたテーブルの例

DDL を用いたテーブル作成

- Athena の DDL は HiveQL 形式で記述

- 標準的なテーブル定義ステートメントの後に、パーティション定義、データ形式、データの場所、圧縮形式などを指定

```
CREATE EXTERNAL TABLE IF NOT EXISTS action_log (
      user_id string,
      action_category string,
      action_detail string
)
  PARTITIONED BY (year int, month int)
  STORED AS PARQUET
  LOCATION 's3://athena-examples/action-log/'
  TBLPROPERTIES ('PARQUET.COMPRESS'='SNAPPY');
```

Amazon Athena で使用可能なデータ形式

- SerDe ※1ライブラリを指定することで様々なファイル形式に対応
- ファイル形式に対応する圧縮形式※2を選択可能

#	対応ファイル形式	利用するSerDe
1	**Amazon Ion**	• Ion Hive SerDe
2	**Apache Avro**	• Avro SerDe
3	**Apache Parquet**	• Parquet SerDe (+ SNAPPY圧縮)
4	**Apache Web Server ログ**	• Grok SerDe or Regex SerDe
5	**CloudTrail ログ**	• Hive JSON SerDe
6	**CSV**	• LazySimple SerDe or OpenCSVSerDe
7	**カスタム区切り**	• LazySimple SerDe
8	**JSON**	• Hive JSON SerDe or OpenX JSON SerDe
9	**ORC**	• ORC SerDe (& ZLIB圧縮)
10	**TSV**	• LazySimple SerDe (& FIELDS TERMINATED BY '¥t' 指定)

※1 SerDe = Serializer/Deserializer
クエリ時に指定することでテーブル定義で指定したSerDeを無視可能

#	対応圧縮形式	補足
1	**BZIP2**	• Burrows-Wheelerアルゴリズム圧縮形式
2	**DEFLATE**	• Deflate圧縮形式 (Avroが利用)
3	**GZIP**	• Deflateベース(.tar.gz形式には未対応)
4	**LZ4**	• LZ7形式の一種。3つの実装(Raw, Framed, hadoop互換)
5	**LZO**	• LZO圧縮形式。2つの実装(Standard/Hadoop互換)
6	**SNAPPY**	• LZ7形式の一種
7	**ZLIB**	• zlibライブラリ、Deflateベース
8	**ZSTD**	• Zstandard圧縮形式

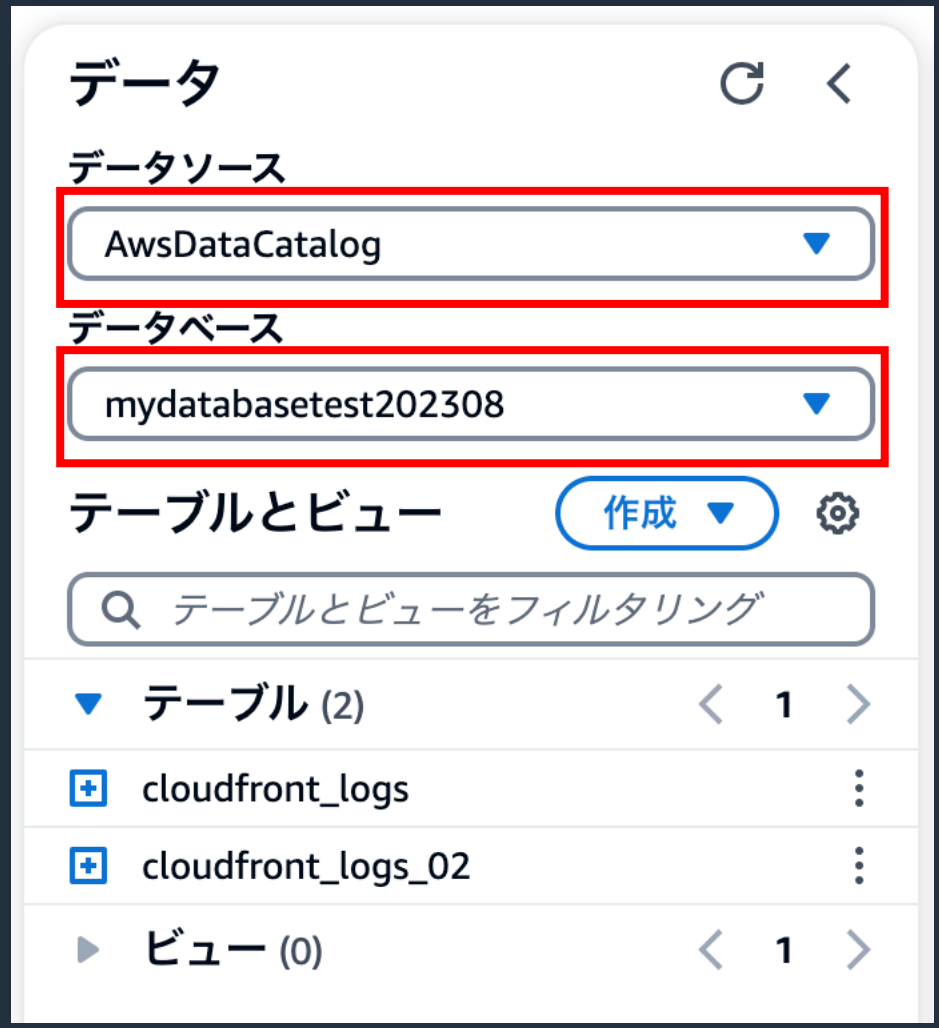
※2 利用可能な組み合わせはドキュメントご参照

https://docs.aws.amazon.com/ja_jp/athena/latest/ug/supported-serdes.html

データソース

- Athena では、データを記述する
  データカタログ、およびそこに含まれる
  データをあわせて**データソース**と定義

- AWS Glue Data Catalog のデータ内で、
  参照権限を持っているもののみが表示

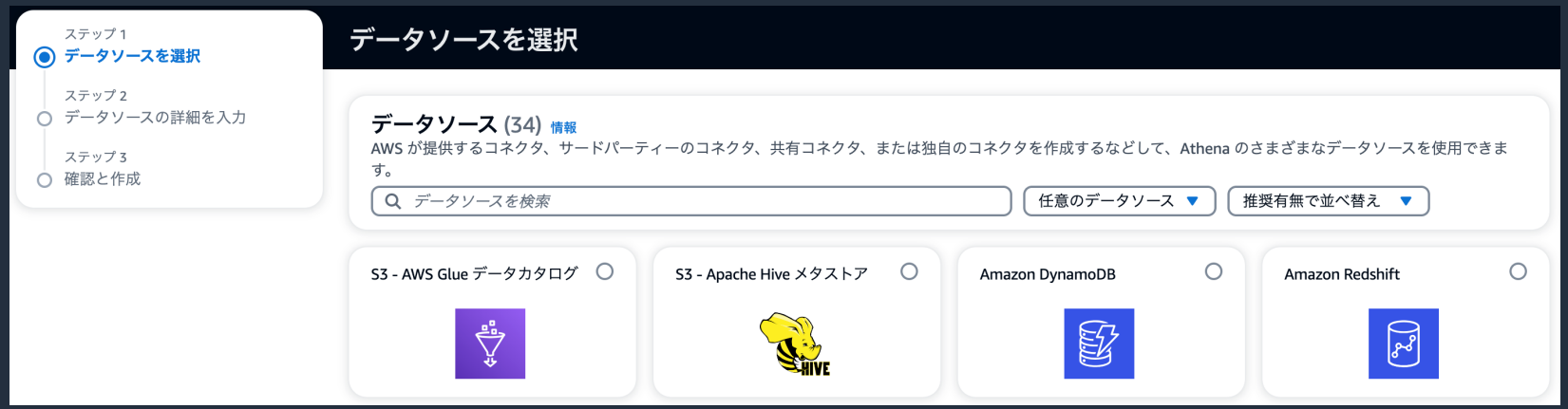


データソースの追加

- AWS Glue Data Catalog 以外のデータソースを、新たに追加することが可能

- 既存の Hive Metastore を Athena でも利用可能
  - 既存資産の（ Hive Metastore）移行不要

- 外部メタストアへの接続には、Lambda 関数として実行されるコネクター (Athena Data Connector for External Hive Metastore) を利用



https://docs.aws.amazon.com/athena/latest/ug/connect-to-data-source-hive.html

Amazon Athena のクエリ

- 標準 ANSI SQL に準拠したクエリ
- WITH句、Window関数、JOINなどに対応
- 基本的には 後述するバージョンでサポートしているクエリエンジンに準拠
  （一部サポートしていない機能等の詳細は以下ドキュメント参照）

https://docs.aws.amazon.com/ja_jp/athena/latest/ug/other-notable-limitations.html

```
[ WITH with_query [, ...] ]
SELECT [ ALL | DISTINCT ] select_expression [, ...]
[ FROM from_item [, ...] ]
[ WHERE condition ]
[ GROUP BY [ ALL | DISTINCT ] grouping_element [, ...] ]
[ HAVING condition ]
[ UNION [ ALL | DISTINCT ] union_query ]
[ ORDER BY expression [ ASC | DESC ] [ NULLS FIRST | NULLS LAST] [, ...] ]
[ LIMIT [ count | ALL ] ]
```

Athena エンジンバージョン 3 （現行バージョン）

- オープンソースの Trino ベースのクエリエンジン
  - 50 を超える新しい 新機能
  - 90 以上のクエリパフォーマンス向上
  - クエリ結果の再利用が可能

- 追加機能
  - Apache Spark バケットアルゴリズムのサポート
  - HyperLogLog関数-高速で巨大なデータセット内の個別要素数を推定
  - 地理空間関数-最適化された地理空間クエリの実行
  - T-digest関数-分位数を正確に推定

（その他機能については、ドキュメントご参照）

https://docs.aws.amazon.com/ja_jp/athena/latest/ug/engine-versions-reference-0003.html

Athena エンジンバージョン 2 （旧バージョン）

- Presto 0.217 をサポートしたエンジン
  - Amazon Athena フェデレーテッドクエリ
  - User Defined Function （ユーザー定義関数）
  - Amazon Athena ML
  - 地理空間関数
  - EXPLAIN および EXPLAIN ANALYZE ステートメント

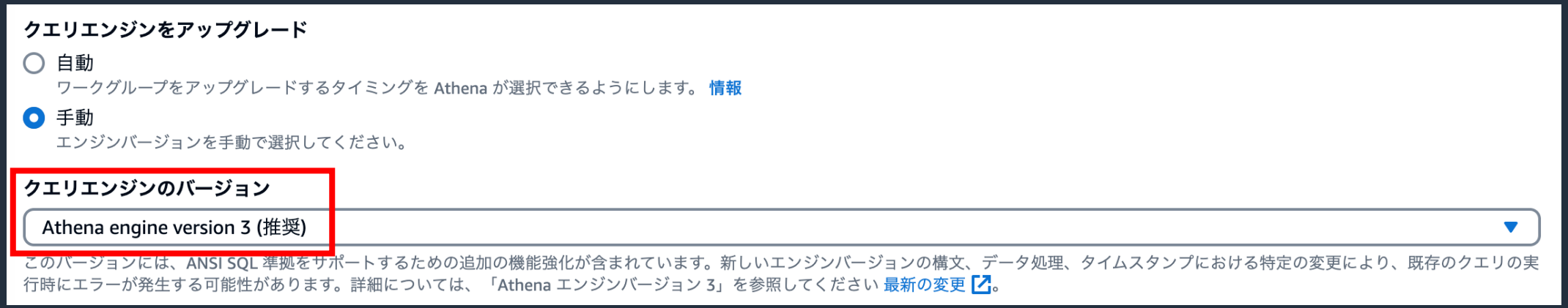
- 性能改善
  - JOIN, ORDER BY, AGGREGATE 操作, Spill to disk

https://docs.aws.amazon.com/ja_jp/athena/latest/ug/engine-versions-reference-0002.html

エンジンバージョンのアップグレード

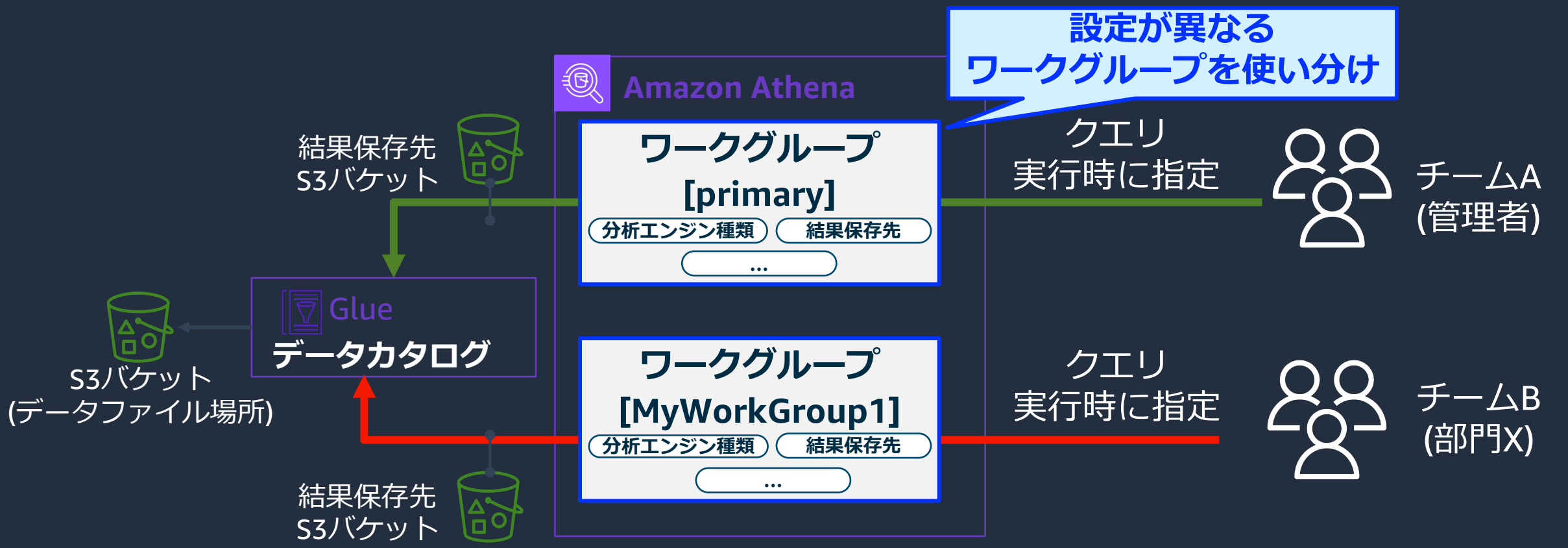
- 自動
  - 自動アップグレードが完了するまでエンジンバージョン2にとどまり、Athenaはワークグループをエンジンバージョンにアップグレードするタイミングを選択

- 手動
  - デフォルトの選択肢はV3に設定され、エンジンバージョン2に切り替えることが可能



ワークグループと権限管理

- 同一アカウント内で、仮想的なワークグループを作成することが可能



(*) 実行ごとに結果保存先S3バケットのロケーションを上書き指定することも可能
※実行時にワークグループ指定を省略した場合はデフォルトで存在する「primary」ワークグループが暗黙的に利用
※ユーザーのIAMポリシーで、利用可能なワークグループを制限可能

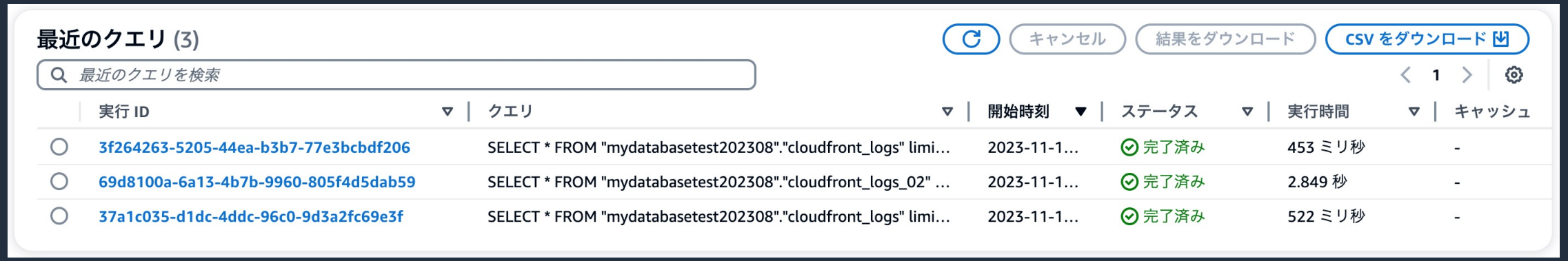
クエリ結果

- 実行される各クエリのクエリ結果とメタデータ情報を、
  指定した **S3 バケット**に自動的に保存
    - この保存自体をオフにすることはできない
    - 必要に応じてこの場所にあるファイルにアクセスして操作可能
    - Athena コンソール履歴画面から、
      クエリ結果ファイルを直接ダウンロードすることも可能

- クエリ出力ファイルへのアクセスには以下の権限が必要
    - クエリ結果の場所の Amazon S3 GetObject アクション
    - Athena GetQueryResults アクション

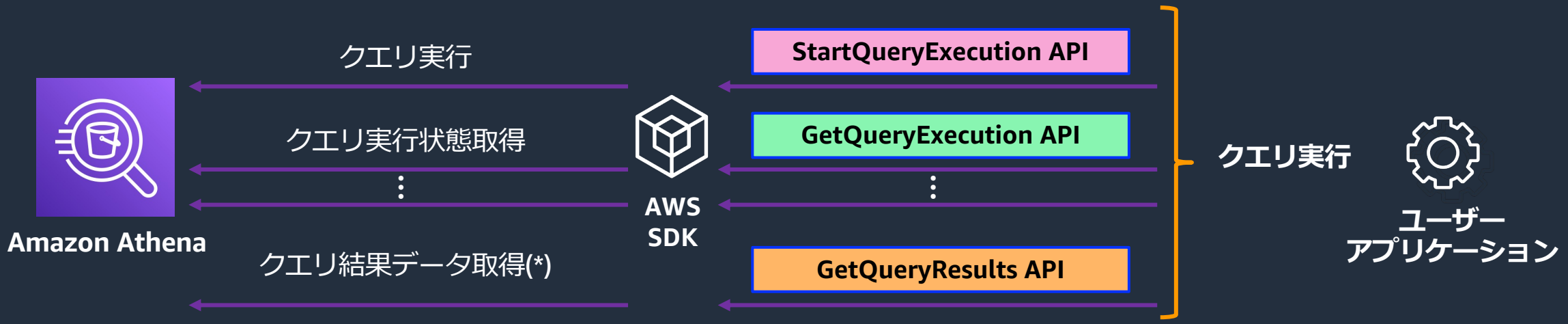
クエリ履歴

- Athena コンソールの履歴画面では下記情報を確認可能（ 45 日間の結果を表示）
  - クエリ送信時刻 / 送信クエリ/ 暗号化タイプ/
    クエリの状態(成功/失敗) / 実行時間 / スキャンしたデータ量 /
    成功したクエリの結果ファイルダウンロード /
    失敗したクエリのエラー情報詳細表示



クエリエディタ以外のクエリ実行 - Athena API

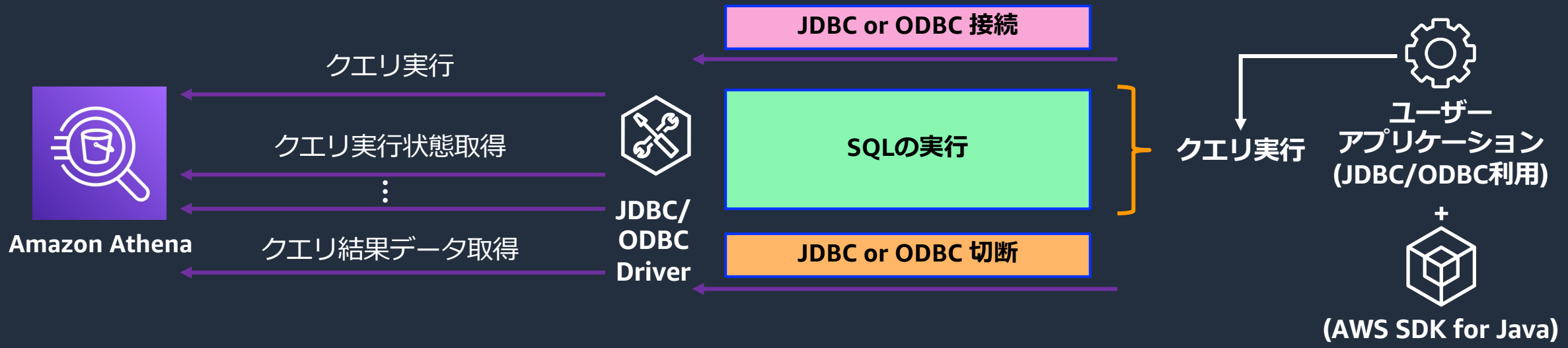
- Athena API を AWS SDK(またはそれを利用したライブラリ)で呼び出し利用可能



#	AWS API	説明
1	StartQueryExecution	• SQLクエリを実行し、クエリ実行ID(QueryExecutionId)を得る
2	GetQueryExecution	• クエリ実行ID(QueryExecutionId)を指定して実行状態を得る("SUCCEEDED"など)
3	BatchGetQueryExecution	• 複数のクエリ実行ID(QueryExecutionId)を指定して一括してそれらの実行状態を得る
4	GetQueryResults	• S3上のクエリ結果をAthena API応答として取得 (1API要求での応答は最大1,000件)

クエリエディタ以外のクエリ実行 - JDBC/ODBC

- 3rd Party(Magnitude社)提供の JDBC/ODBC Driver を利用可能能



#	Driver	説明
1	JDBC	• JDBC 4.2に対応 (Java 8.0以降に対応) • アプリケーションは エンドポイントに443/tcp + 444/tcpのアウトバウンド通信が必要 • 認証方式：「IAMクレデンシャル」「Azure ADFS, Okta, Ping 等SAML 2.0 IdPとのIAM連携」 に対応
2	ODBC	• Windows 32/64bit, Linux 32bit/64bit, macOS に対応 • アプリケーションは エンドポイントに443/tcp + 444/tcpのアウトバウンド通信が必要 • 認証方式：「IAMクレデンシャル」「Azure ADFS, Okta, Ping 等SAML 2.0 IdPとのIAM連携」 に対応

クエリエディタ以外のクエリ実行 -連携AWSサービス

- Amazon QuickSight とのネイティブな連携

#	データセット定義	説明
1	**SPICEへのロード**	• 指定したSQLクエリの実行結果を SPICE(QuickSight内部のインメモリDB) にロード • QuickSightデータセットの更新のためには「SPICEのリフレッシュ」が必要
2	**非SPICE(直接クエリ)**	• ユーザーのUI操作(ビジュアル操作)に対応して都度Athenaへクエリを実行

その他にもStep Functions や SageMaker (SameMaker DataWrangler) 等 が
ネイティブでのAthenaへのクエリ実行をサポート

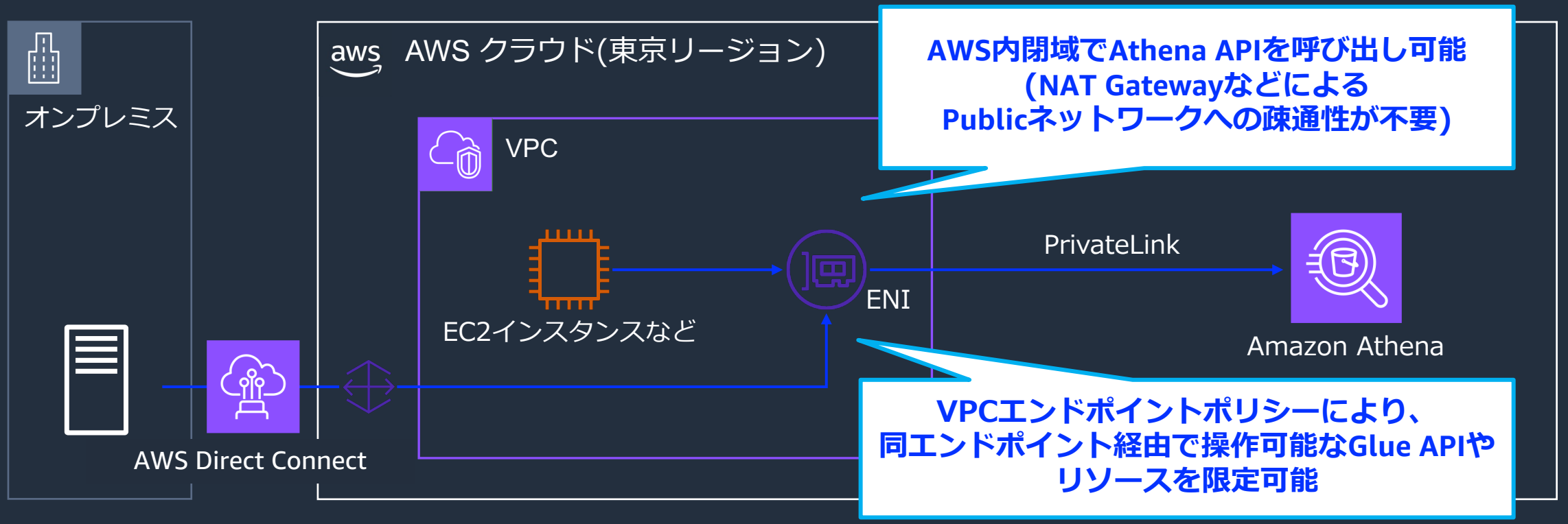
S3 Glacierストレージクラス対応

- テーブルにプロパティを設定 & S3 Glacier (Flexible Retrieval/Deep Archive) のオブジェクトを事前に復元しておくことで、Athenaでのクエリが可能

※「復元」状態にないGlacier系ストレージクラスのオブジェクトはクエリ対象外
※ S3 Glacier Instant Retrieval ストレージクラスの場合、復元操作は元々不要

プライベートネットワーク接続

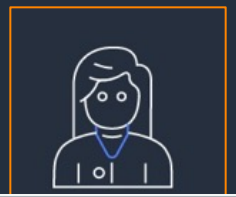
- PrivateLink を使用することで、閉域網でAPIを呼び出し可能



エンドポイントサービス名：
com.amazonaws.<region>.athena

Athena SQL が提供する機能

Amazon Athena SQL で対応可能なニーズ

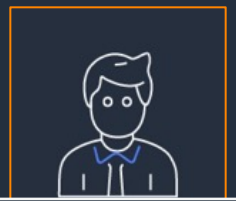


データアナリスト

S3以外のデータソースにもクエリをかけたい。

データ管理者

データ整形のパイプラインを簡素化したい。



データサイエンティスト

機械学習で作成したモデルを簡単に使いたい。

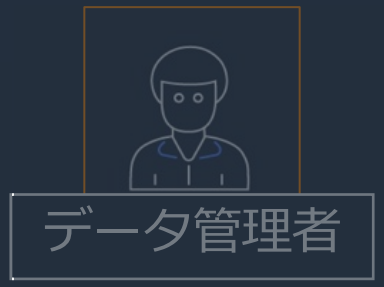
エンジニア

ACID トランザクションを使いたい。

Amazon Athena SQL で対応可能なニーズ

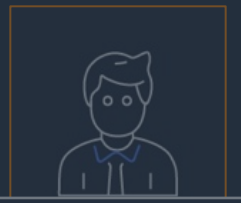
データアナリスト

S3以外のデータソースにもクエリをかけたい。



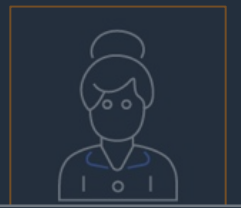
データ管理者

データ整形のパイプラインを簡素化したい。



データサイエンティスト

機械学習で作成したモデルを簡単に使いたい。

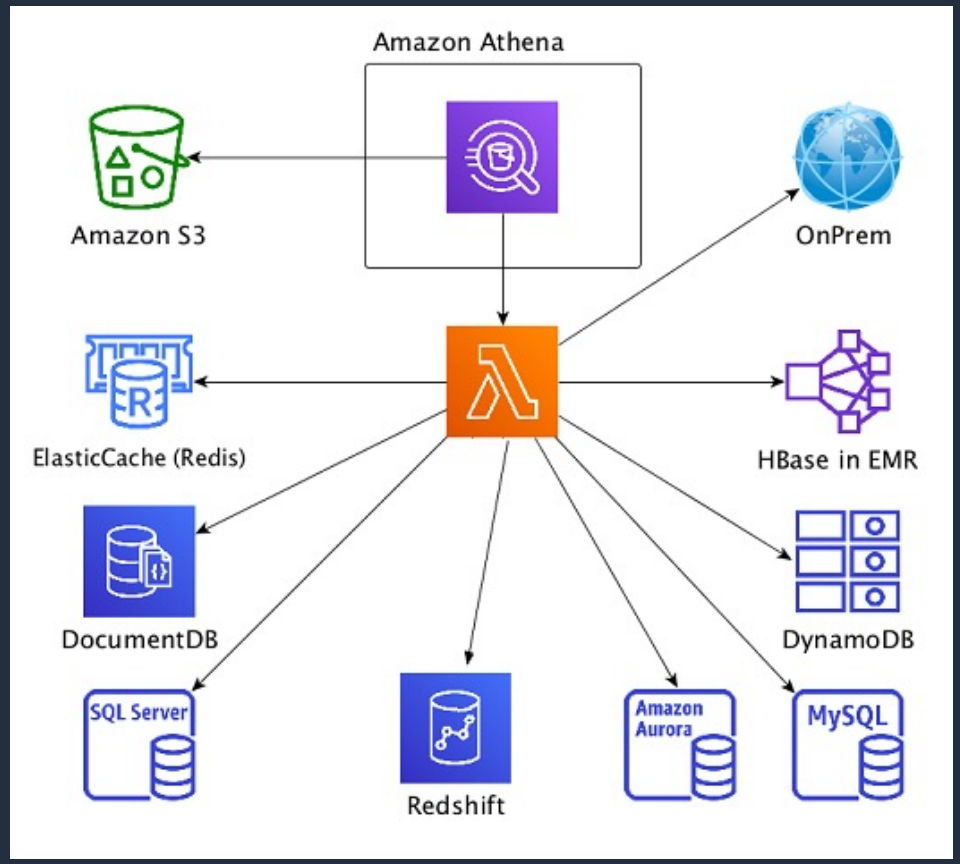


エンジニア

ACID トランザクションを使いたい。

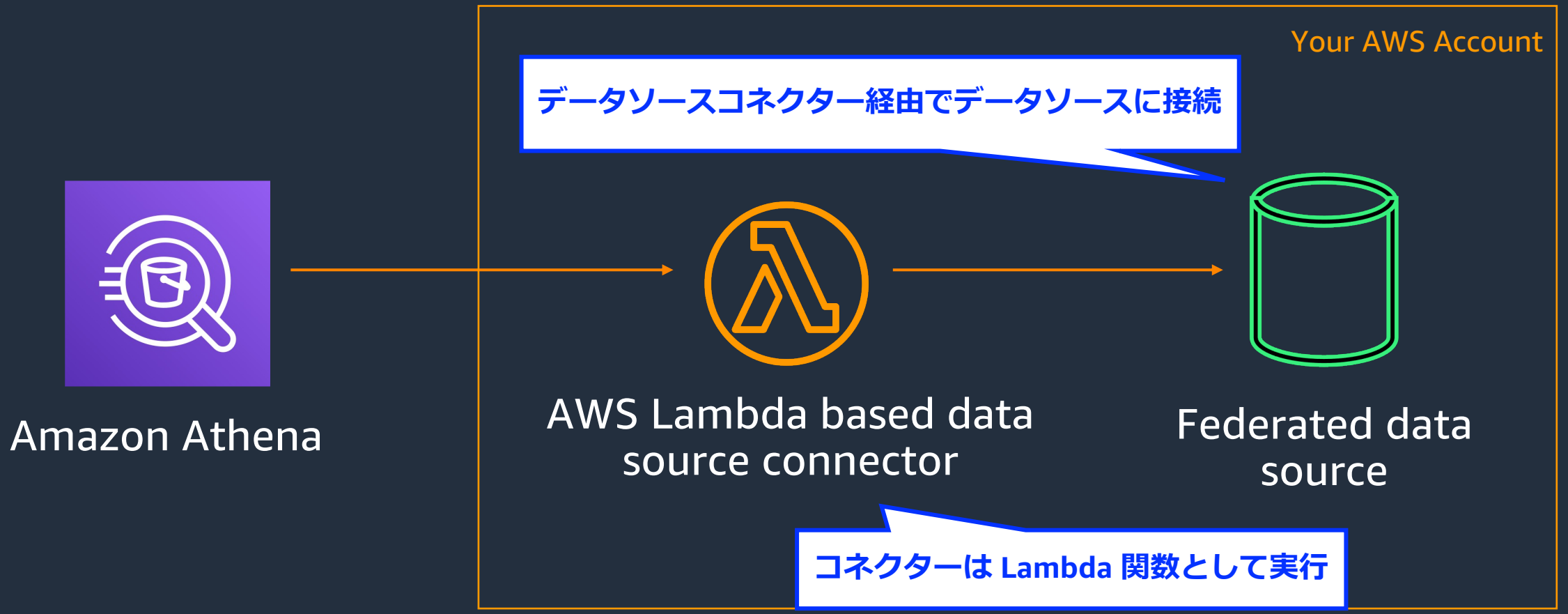
Amazon Athena フェデレーテッドクエリ

- リレーショナル、非リレーショナル、オブジェクト、またはカスタムデータソース間でクエリを実行
- アドホックな調査、複雑なパイプライン、アプリケーションに使用可能
- データソースコネクターを利用して Athena クエリエンジンを拡張可能
- 他の AWS アカウントに保存されているデータにクエリ可能

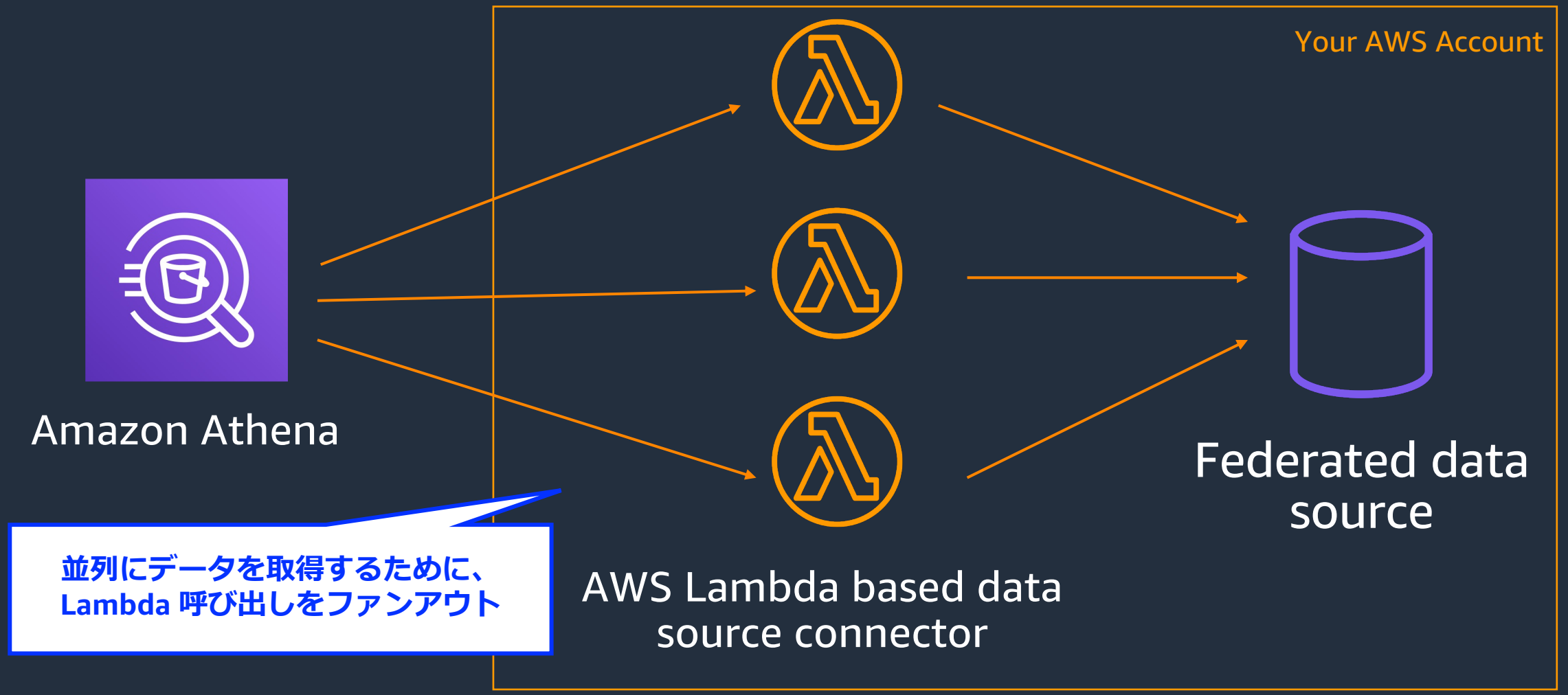


https://github.com/awslabs/aws-athena-query-federation
https://docs.aws.amazon.com/athena/latest/ug/connect-to-a-data-source.html
https://aws.amazon.com/jp/blogs/news/query-any-data-source-with-amazon-athenas-new-federated-query/
https://aws.amazon.com/jp/blogs/news/athena-federated-query-dynamodb-quicksight/

フェデレーテッドクエリの構成

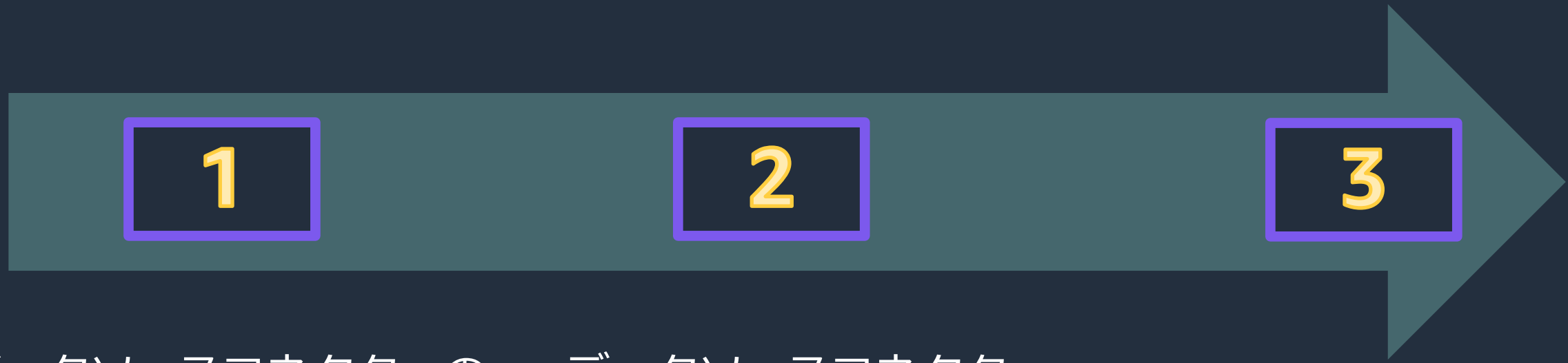


フェデレーテッドクエリの実行



フェデレーテッドクエリの始め方

- 3Stepで簡単に使い始めることが可能

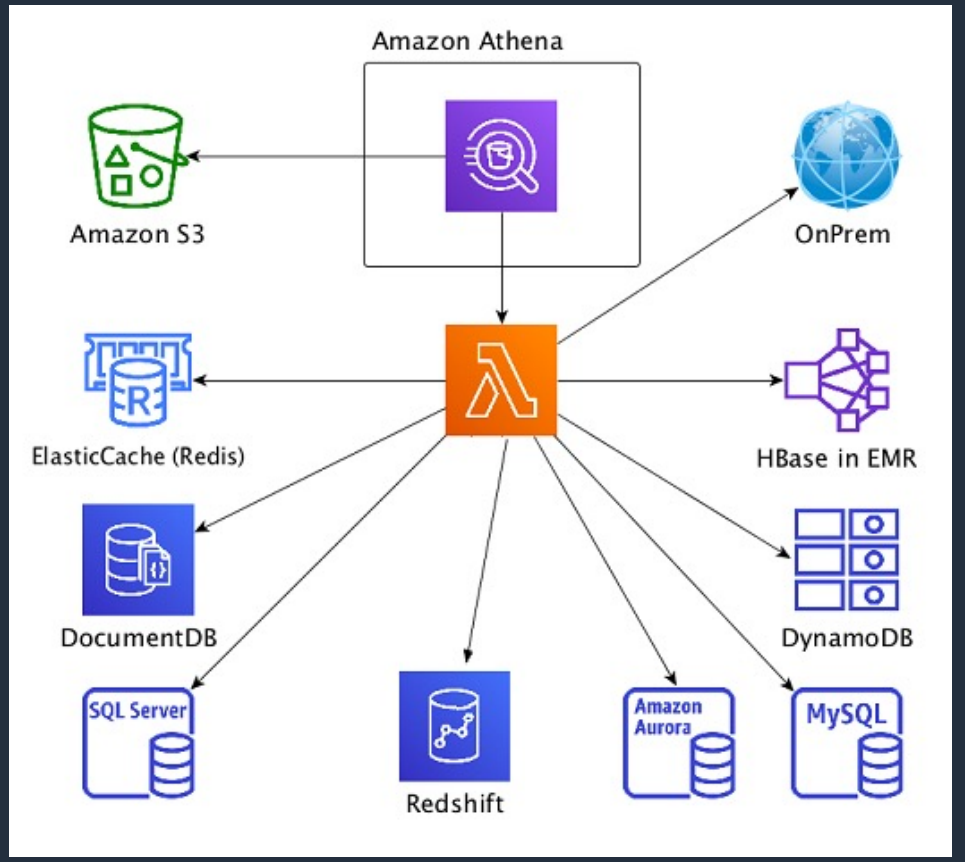


データソースコネクターのデプロイ方法

- AthenaはAWS Lambdaベースの
  データソースコネクタを使用

- コネクタの2つの展開方法
  - AWS Serverless Application Repository
    を使用したワンクリック配備
  - コネクタコードをLambdaに展開

- Lambda 関数をデプロイすると固有の
  Amazon Resource Name（ARN）を取得可能



利用可能なデータソースコネクタ

- 追加費用なしで様々なコネクターをコンソールから選択可能

#	データソースコネクタ (接続先)
1	Apache HBase
2	Azure Data Lake Storage (Gen2)
3	Azure Synapse
4	Cloudera Hive
5	Cloudera Impala
6	Amazon CloudWatch Logs
7	Amazon CloudWatch Metrics
8	AWS CMDB (AWS Resource inventory)
9	IBM Db2
10	Amazon DocumentDB
11	Amazon DynamoDB
12	Google Cloud Storage (CSV/Parquet)
13	Google BigQuery
14	Horonworks
15	Amazon MSK(Managed Streaming for Kafka)

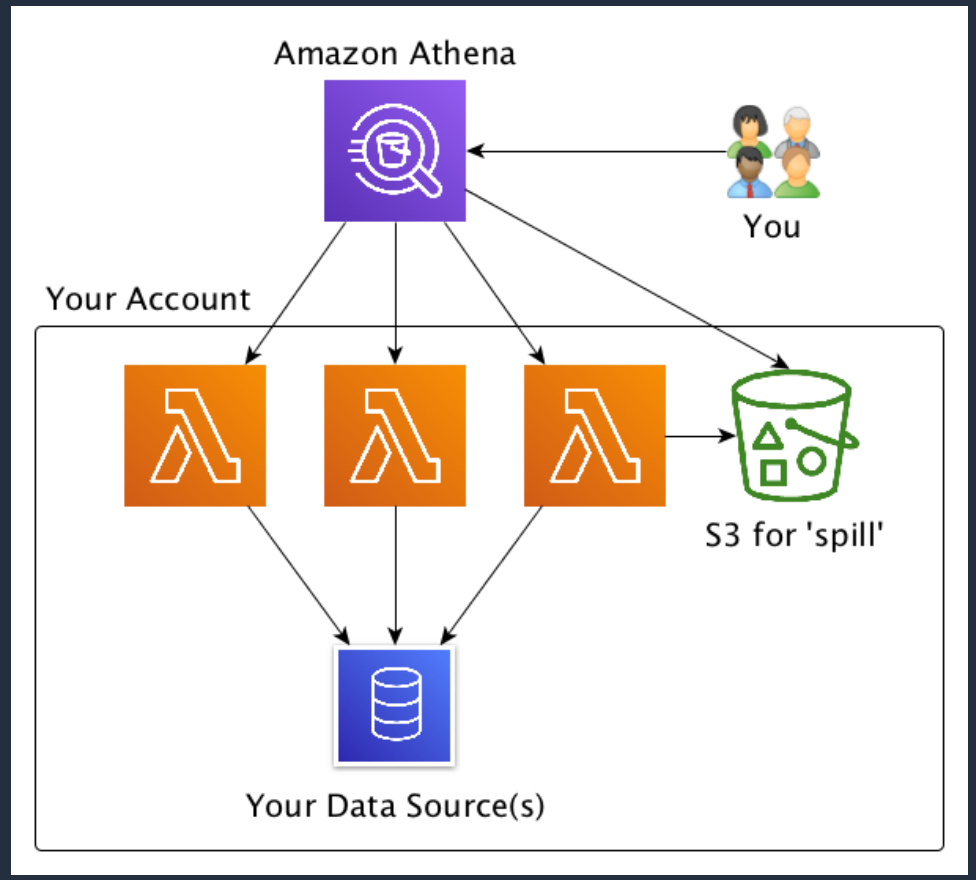
#	データソースコネクタ (接続先)
16	MySQL
17	Neptune
18	OpenSearch
19	Oracle
20	PostgreSQL
21	Redis
22	Redshift
23	SAP HANA
24	Snowflake
25	SQL Server
26	Teradata
27	Amazon Timestream
28	TPS-DS(TPC Benchmark DS data)
29	Vertica

https://docs.aws.amazon.com/ja_jp/athena/latest/ug/connectors-available.html

独自のデータソースコネクタを構築

- Athena Query Federation SDKを使用し、独自のコネクタを作成

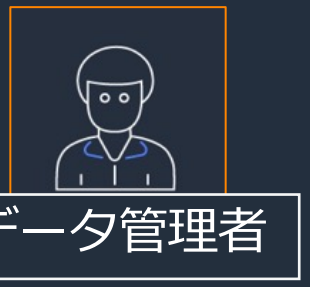
- 特徴
  - S3 spill Partition pruning Parallel scans
  - Portable columnar
  - memory-format (Apache Arrow)
  - Authorization
  - Congestion control/avoidanceAthena



Amazon Athena SQL で対応可能なニーズ

データアナリスト

S3以外のデータソースにもクエリをかけたい。



データ管理者

データ整形のパイプラインを簡素化したい。

データサイエンティスト

機械学習で作成したモデルを簡単に使いたい。

エンジニア

ACID トランザクションを使いたい。

User Defined Functions (UDF) in Athena

- Athena Query Federation SDKを使用したユーザー定義の関数

- 特徴
  - AWS Lambdaを利用したユーザー定義の関数（UDF）
  - APIライクなネットワークコールが可能
  - SELECTおよび/またはFILTERフェーズでUDFを実行

他のAWSサービスを使用することなくETL処理を実行可能

UDF のサンプルコード

- 書き込み、展開、
  呼び出しが簡単
- Scalar 関数も実行可能
- コードはLambda上で起動

```
USING FUNCTION totalprice(quantity int, unitprice DOUBLE)
                RETURN DOUBLE TYPE lambda_udf
        WITH (lambda_udf='ecommerselambdaudf'),
USING FUNCTION isInternational(fullAddress VARCHAR) RETURN BOOLEAN
        TYPE LAMBDA_UDF WITH (lambda_udf='ECommerseLambdaUdf')
SELECT productname,
        productid,
        totalprice(productquantity, unitprice)
FROM    productcatalog
WHERE   isInternational(product.vendor.addr)
```

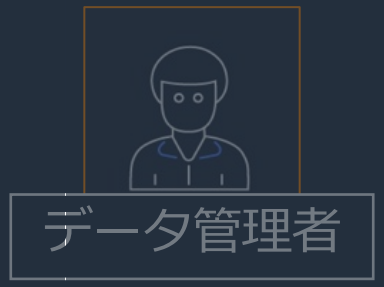
```
public class ECommerceLambdaUdfHandler extends ScalarUdfHandler {

    public double totalPrice(int quantity, double unitPrice) {
        return quantity * unitPrice;
    }

    public boolean isInternational(String encryptedAddress) {
        String customerAddr = cipher.decrypt(encryptedAddress);
        return isInternational(customerAddr);
    }
}
```

Amazon Athena SQL で対応可能なニーズ

S3以外のデータソースにもクエリをかけたい。



データ整形のパイプラインを簡素化したい。

機械学習で作成したモデルを簡単に使いたい。

ACID トランザクションを使いたい。

Machine Learning (ML) with Amazon Athena

- SQLクエリで推論のための機械学習モデルを呼び出し可能
  - MLモデルをAmazon SageMakerに1回展開し、n回使用
  - 任意の場所のデータに対して推論を実行
  - 推論を可能にするアプリケーションの構築が不要
  - 追加のセットアップは不要

Athena を使用したMLモデルの学習

- 学習モデルのトレーニングを行う 3 ステップ

任意のデータソースから
フェデレーテッドクエリ
によりデータを選択

AthenaでUDFを使用して
データを変換

Amazon SageMakerでモデル
をトレーニングして展開

Athena から ML モデルを使用した推論を実行

- 推論実施時の 3 ステップ

SageMakerで
MLモデルを展開

前処理や後処理の為の
UDFを定義

任意のデータソースに
対して推論を実行

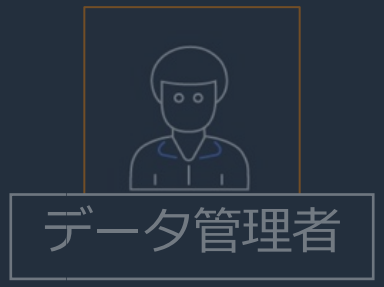
ML を使用する際のユースケース

- 様々な目的に合わせて使い分けることが可能

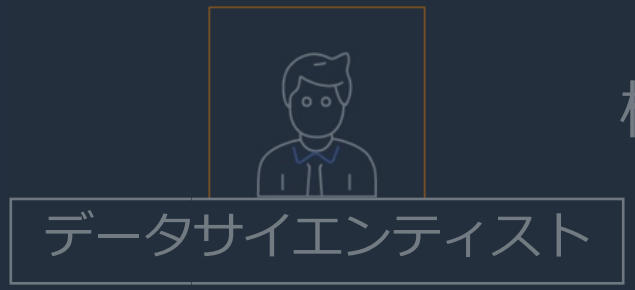
  - アプリケーションログ内の疑わしいアクティビティに関連付けられたIPアドレスを検索
  - 収益異常のある製品の検索 (+/-)
  - トランザクションレコードで詐欺の疑いがあるものを検知
  - 提案された新しいビデオゲームがヒットするかどうかを予測

Amazon Athena SQL で対応可能なニーズ

S3以外のデータソースにもクエリをかけたい。



データ整形のパイプラインを簡素化したい。



機械学習で作成したモデルを簡単に使いたい。

ACID トランザクションを使いたい。

ACIDトランザクション

- Apache ICEBERG を使用することでACIDトランザクションを実現可能

  - 複数の同時接続ユーザが整合性を保ちながら、
    S3 データの行レベルの変更を実行可能

  - 書き込み、削除、更新、タイムトラベルオペレーションを
    Athena のコンソール、API、ODBC/JDBC 経由で実行可能

  - Iceberg のテーブルフォーマットをサポートする
    他システム（EMR や Spark、Flink 等）との互換性

  - Amazon Athena が利用可能な全リージョンで利用可能

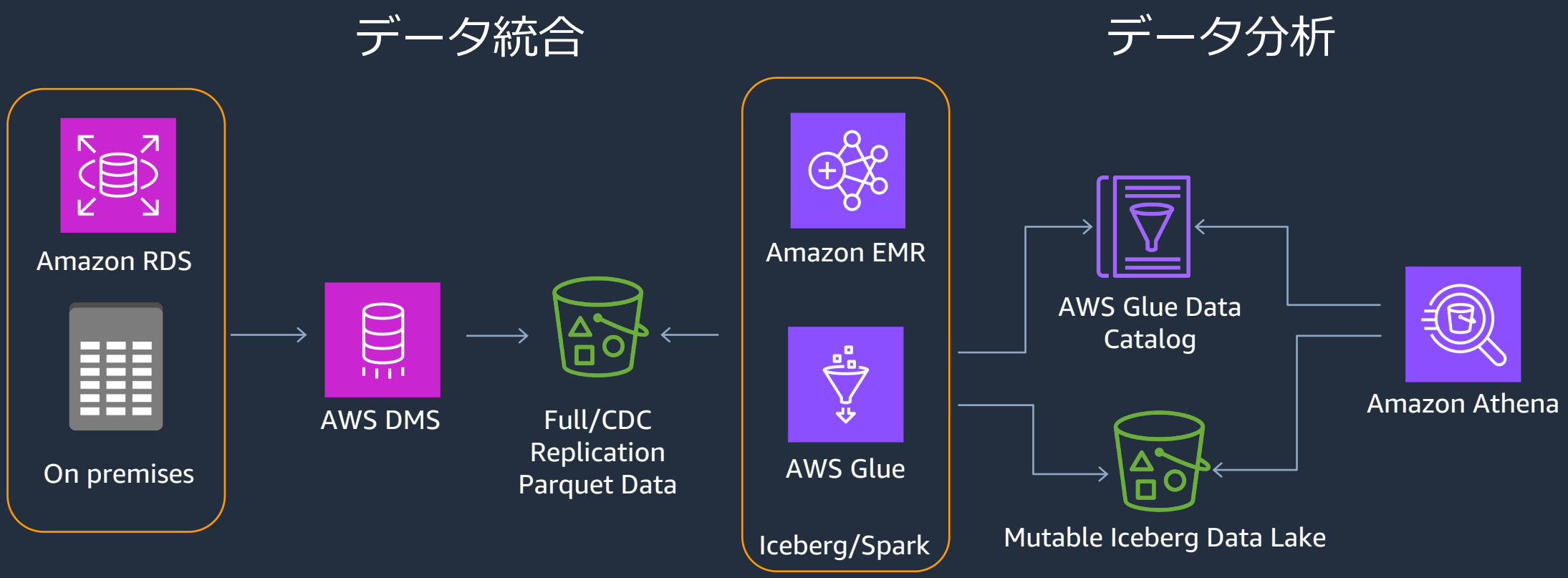
https://docs.aws.amazon.com/ja_jp/athena/latest/ug/querying-iceberg.html

Apache Iceberg テーブルの特徴

- スナップショットベースのテーブルメタデータを使用し、ACIDトランザクションをサポート

Apache IcebergとAthenaの関係

- Apache Iceberg テーブルを作成して Amazon S3 に保存



その他の選択肢

- Apache Hudi
  - 増分データの処理とデータパイプラインの開発をシンプルにするオープンソースのデータ管理フレームワーク
  - Uber 社がオープンソース化
  - Apache Spark、Apache Hive、および Presto と統合

- Delta Lake
  - Delta Lakeとは、信頼性の高い読み書きを高速かつ同時に実行可能な、オープンソースのストレージレイヤソフトウェア
  - Databricksによるオープンソース

Athena SQL の
パフォーマンス関連Tips

列指向フォーマット

- 目的に合わせたデータ形式を選択することが重要

指向	特徴
列指向	• カラムごとにデータをまとめて保存 • 特定の列だけを扱う処理では、ファイル全体を読む必要がない • OLAP向き • ORC, Parquet など
行指向	• レコード単位でデータを保存 • 1カラムのみ必要でも、レコード全体を読み込む必要がある • OLTP向き • AVRO など

https://orc.apache.org/docs/

列指向フォーマットを使うメリット

- OLAP 系の分析クエリを効率的に実行可能
  - たいていの分析クエリは、
    一度のクエリで特定のカラムを使用
  - 単純な統計データなら、メタデータで完結
- I/O 効率が向上
  - 圧縮と同時に使うことで I/O 効率が向上
  - カラムごとに分けられてデータが整列
  - 類似データが続くことで圧縮効率が向上

データ圧縮

- 最低限分割可能な圧縮形式を利用しておくと
  巨大なファイルがあったとしても分散処理することが可能

- 分割不可の圧縮方式では、ファイル単位でしか分散処理できため、
  巨大なファイルは事前に分割してお区ことが必要

	gzip	bzip2	lzo	snappy
file extension	.gzip	.bz2	.lzo	.snappy
Compression Level	High	Highest	Average	Average
Speed	Medium	Slow	Fast	Fast
CPU usage	Medium	High	Low	Low
Is Splittable	No(※)	Yes	Yes, if indexed	No(※)

※ Parquet, Avro などのコンテナフォーマットで利用する場合は分割できる

https://docs.aws.amazon.com/ja_jp/athena/latest/ug/compression-formats.html

パーティション

```
SELECT
    month
    , action_category
    , action_detail
    , COUNT(user_id)
FROM
    action_log
WHERE
    year = 2016
    AND month >= 4
    AND month < 7
GROUP BY
    month
    , action_category
    , action_detail
```

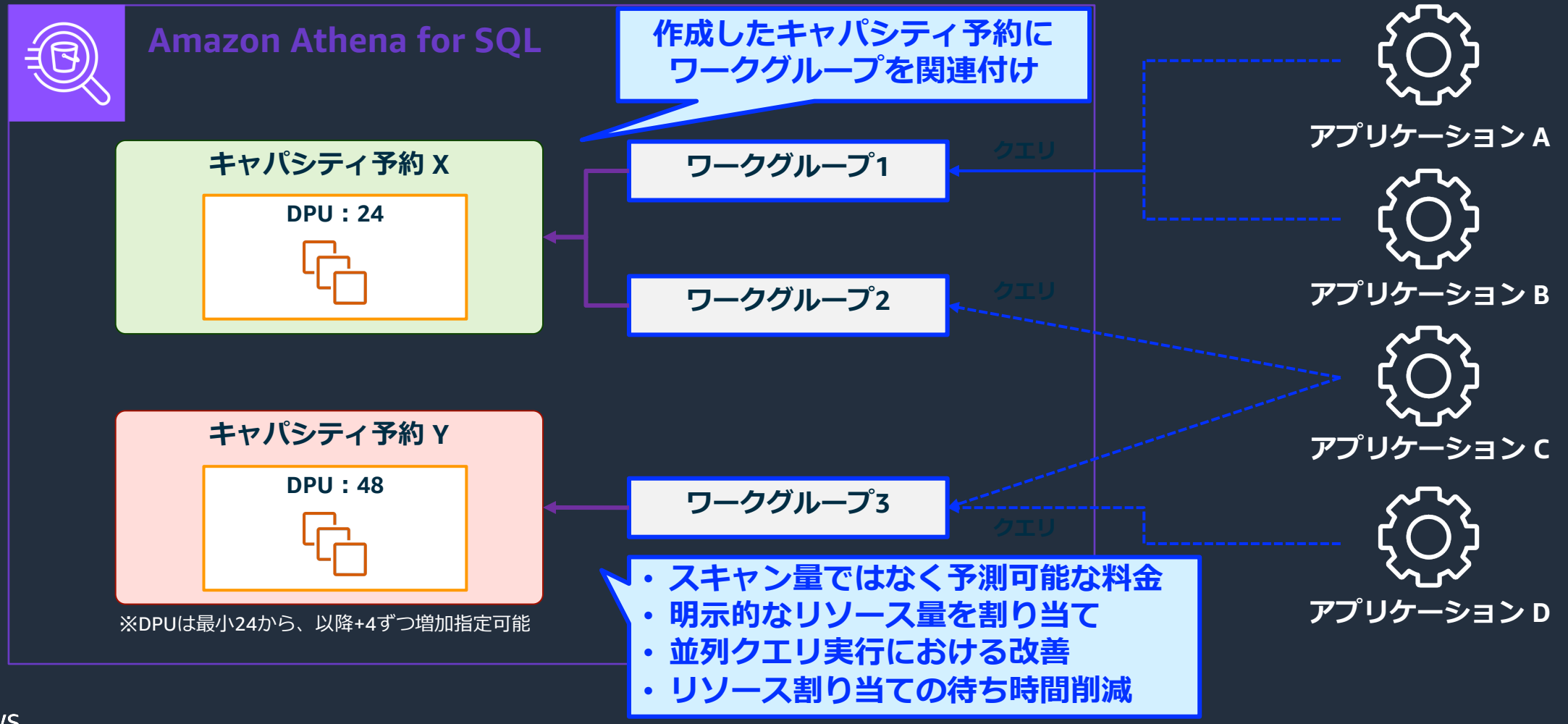
- WHERE で読み込み範囲を絞るときに、**頻繁に**使われるカラムをキーに指定
- 絞り込みの効果が高いものが適合
- ログデータの場合、日付が定番
- “year/month/day” と階層で指定

以下の Amazon S3 パスだけが読み込まれる

s3://athena-examples/action-log/year=2016/month=04/day=01/
s3://athena-examples/action-log/year=2016/month=04/day=02/
s3://athena-examples/action-log/year=2016/month=04/day=03/
…
s3://athena-examples/action-log/year=2016/month=06/day=30/

キャパシティ予約

- アカウント & リージョン毎にコンピュート能力を予約することが可能



DPU (Data Processing Unit)

- Athena SQLの キャパシティ予約 におけるコンピューティング能力の単位

並列クエリ数	DPU設定 初期参考値
10	40 DPU
20	96 DPU
30～	240～ DPU

クエリ種類	消費DPU数 参考値
DDL	4 DPU
DML	4～124 DPU

1 DPU

Athena SQL プロビジョニングDPU料金
→ $0.43/DPU時間 (東京リージョン)

- 4 vCPU
- 16 GB RAM

- クエリが行われていない間もプロビジョニングしたDPU(最小24)は料金が発生
- プロビジョニング可能なDPUはアカウント&リージョンごとに合計1,000まで (上限緩和不可)
- キャパシティ予約作成ごとに、AWSの承認対応(最大30分程度) が発生
- キャパシティ予約の作成が受理された場合、最低1時間分の料金が発生
- プロビジョニングしたDPUが単一のクエリに対して不足している場合、当該クエリはエラーを出力

クエリの最適化

1. **ORDER BY を最適化する**
   LIMIT 句をつけることで、ORDER BY の負荷を軽減
2. **JOIN を最適化する**
   結合の際には、大きなテーブルを左側に、小さなテーブルを右にする
3. **GROUP BY を最適化する**
   複数カラムを指定する場合には、カーディナリティ(カラム内のユニークな値の個数)の高いカラムを前に持ってくる
4. **LIKE 演算子を最適化する**
   クエリ内で複数の LIKE 演算子を使う場合には、RegEx におきかえた方が高速になる
5. **近似関数を使う**
   多少の誤差を許容できるなら、COUNT DISTINCT でなく APPROX_DISTINCT() を使用
6. **必要なカラムだけを読みこむ**
   できるだけ * を使わず、必要なカラムだけを指定して SELECT 文を実行

https://aws.amazon.com/jp/blogs/news/top-10-performance-tuning-tips-for-amazon-athena/

Partition Projection

- ユースケース
  - 多くのパーティションがあるテーブルに対するクエリ実行時間が長い場合
  - データに新しいパーティションが作成されたとき、
    定期的にパーティションをテーブルに追加している場合
  - 多くのパーティション化されたデータが S3 に保存されており、
    メタデータストアで管理するのが現実的ではない場合
- Projection 可能なパーティション構造
  - Partition Projection はパーティションが予測可能な場合に利用

Projection Type	パターン	例
整数	整数の連続シーケンス	[1, 2, 3, 4, ..., 1000] や [0500, 0550, ..., 2500] など
日付	日付/日時の 連続シーケンス	[20200101, 20200102, ..., 20201231]、 [1-1-2020 00:00:00, 1-1-2020 01:00:00, ..., 12-31-2020 23:00:00] など
列挙値	列挙値の有限セット	空港コードや AWS リージョンなど

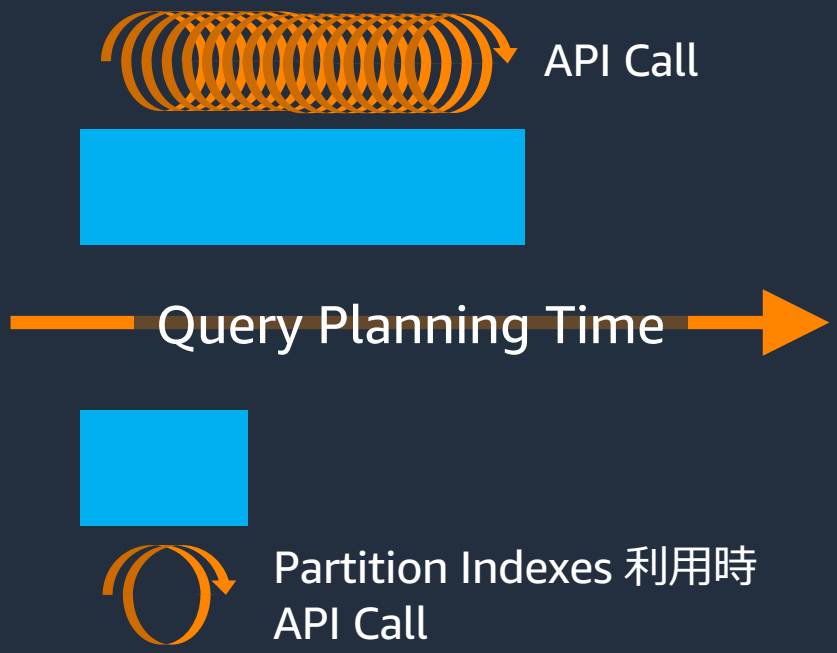
https://docs.aws.amazon.com/ja_jp/athena/latest/ug/partition-projection.html

AWS Glue Partition Indexes

- AWS Glue Data Catalog が提供する Partition Indexes を利用すことで、数十万のパーティションを持つテーブルのパーティションメタデータの取得、および、フィルタリングに必要な時間を短縮可能

- クエリ実行時間を短縮することが可能

- 複数サービスの機能で利用可能
  - Amazon Athena
  - Amazon EMR
  - Amazon Redshift Spectrum
  - AWS Glue ETL jobs



https://docs.aws.amazon.com/athena/latest/ug/glue-best-practices.html#glue-best-practices-partition-index

クォータ（制限事項）

- Athena SQL を利用する際には、以下の制約事項について考慮が必要

#	主なクォータ(アカウント&リージョンごと)	クォータ	緩和申請可否
1	実行/キューイング可能なDDL数	20	Yes
2	実行/キューイング可能なDML数	150(東京リージョン), 20(大阪リージョン)	Yes
3	クエリ文字数	262,144 bytes	No
4	データベース、テーブル、および列名	最大255 文字 (かつ小文字のみ=非CamelCase要)	No
5	DDLクエリタイムアウト	600 分	No
6	DMLクエリタイムアウト	30 分	Yes
7	Apache Spark DPU 同時割当数	160	No
8	ワークグループ数	1,000	No
9	ワークグループへの設定可能タグ数	50	No
10	アクセス可能なパーティション数	1,000,000 (1クエリあたり)	No
11	非Glueデータカタログのパーティション数	20,000 (1テーブルあたり)	Yes
12	最大のプロビジョニングDPU数	1,000	No

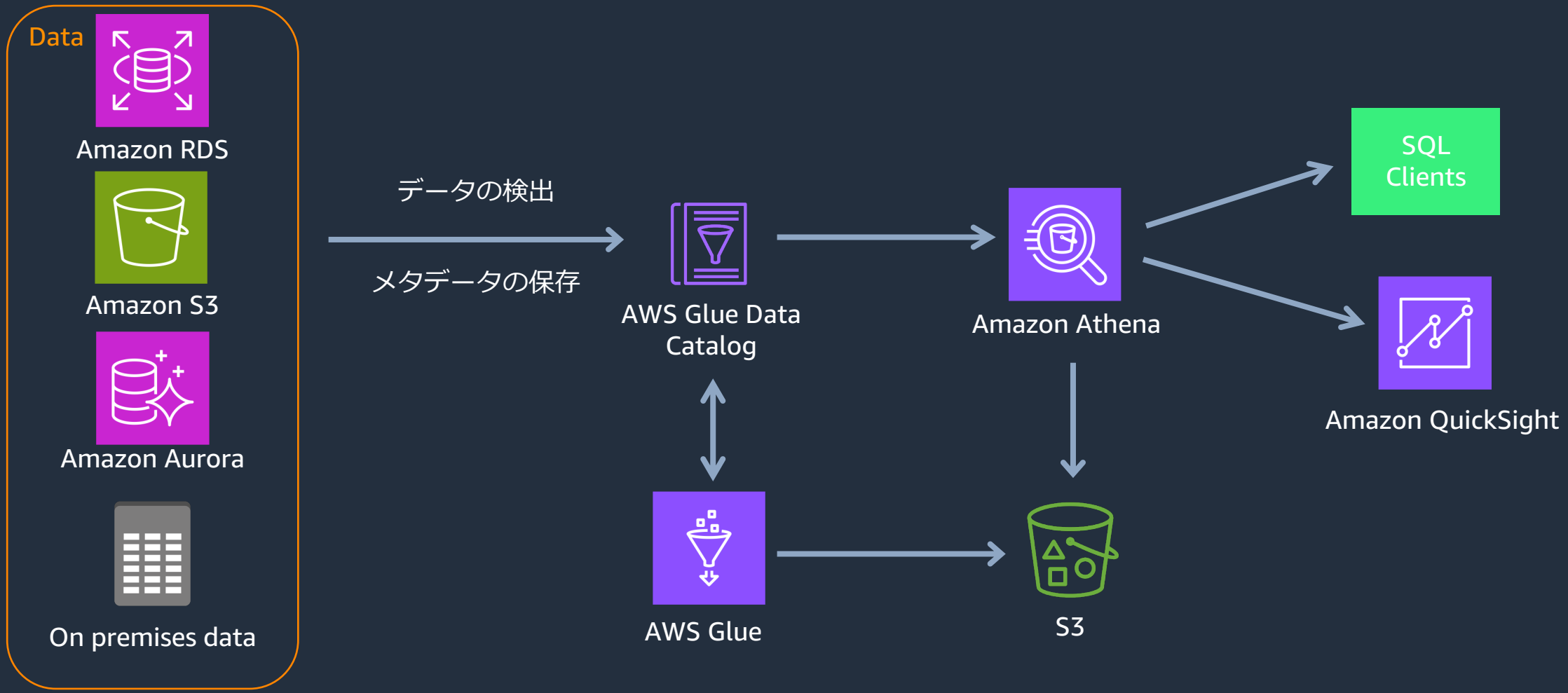
ユースケース

アナリストによるアドホックな分析

- アナリストが分析をするときの典型的なプロセス
  - 手元から Amazon S3 に CSV フォーマットのデータをアップロード
  - 分析しやすくするため CT AS で Parquet ファイルに変換
  - 定番の変換処理をまとめて、VIEW として登録
  - 地理空間データの分析のような、さまざまな処理を行う
  - 結果を Amazon QuickSight やその他 3rd party BI ツールから可視化

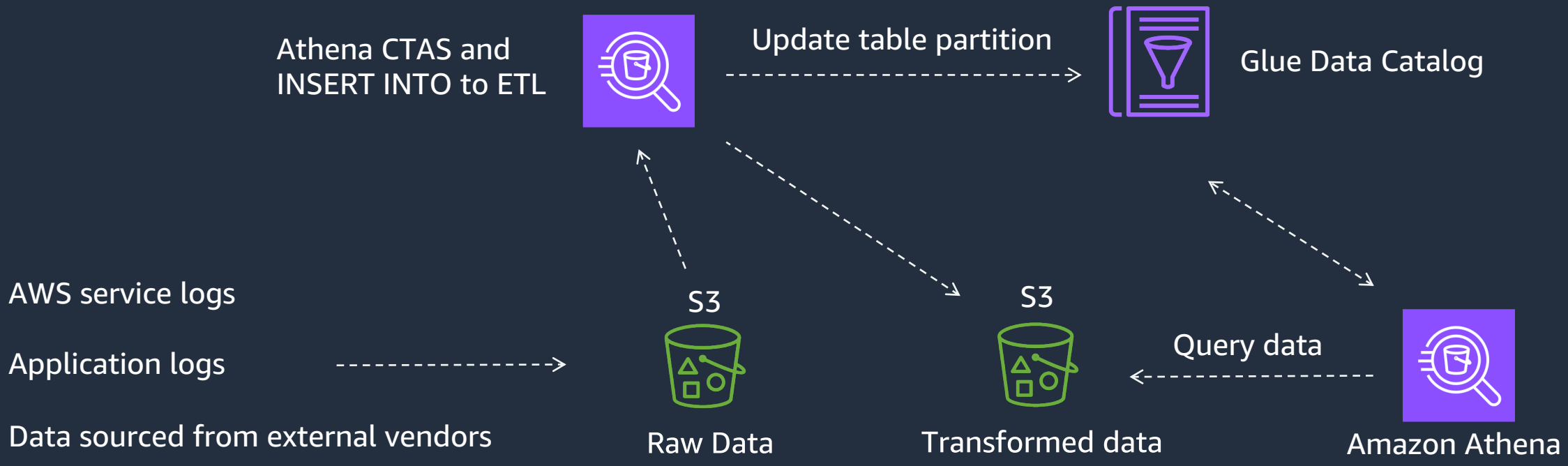
データ探索

- さまざまなデータソースにデータを保持している場合に有効



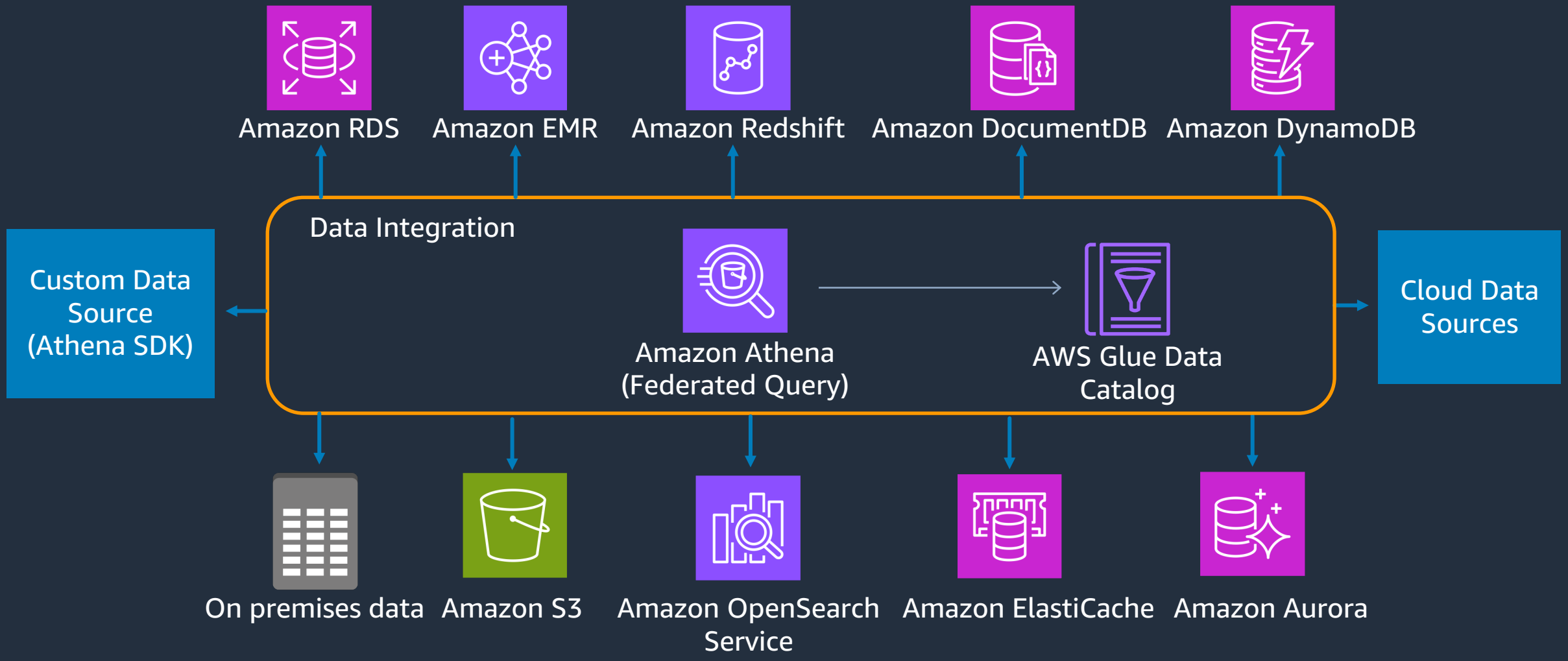
ETL 処理を Amazon Athena のみで実施

- 典型的なバッチ処理の流れ
  - 連携先サービスなどから、継続的にデータが送られてくる
  - CTAS / INSERT INTO で Parquet に変換してパーティションを追加
  - 変換後のデータに対して、Athena からクエリを実行



データ統合

- ETL処理を実行せずに、直接参照することも可能



Athena Pricing

Athena SQLの利用料金

- Athena SQLでは、スキャンデータ量と使用したDPU量に応じた費用が発生

① スキャンデータサイズ

5.00 USD/1TB
※クエリ結果データサイズではない
※10MB未満のスキャン時は
10MBに切り上げ
※DDLは料金発生の対象外

+

② プロビジョニングしたDPU

0.43 USD/DPU-時

※最低1時間、以降1分単位
※キャパシティ予約毎の
最小DPUは24

+

③その他 サービス料金

Amazon S3

+

AWS Lambda

+

その他連携サービス

⋮

https://aws.amazon.com/jp/athena/pricing/

まとめ

まとめ

- Amazon Athena SQL を使用することで、S3 を中心に様々なデータストアにアドホックでインタラクティブな分析が可能

- 様々なデータストアに分析を行えるだけでなく、
  - データ整形のパイプラインを簡素化
  - 機械学習での活用
  - 複数のデータソースの統合等

様々なユースケースに対応可能

AWS Black Belt Online Seminar とは

- 「サービス別」「ソリューション別」「業種別」などのテーマに分け、アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナーシリーズです
- AWS の技術担当者が、 AWS の各サービスやソリューションについてテーマごとに動画を公開します
- 以下の URL より、過去のセミナー含めた資料などをダウンロードすることができます
  - https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/
  - https://www.youtube.com/playlist?list=PLzWGOASvSx6FIwIC2X1nObr1KcMCBBlqY

ご感想は X (Twitter) へ！ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では資料作成時点のサービス内容および価格についてご説明しています。AWS のサービスは常にアップデートを続けているため、最新の情報は AWS 公式ウェブサイト (https://aws.amazon.com/) にてご確認ください
- 資料作成には十分注意しておりますが、資料内の価格と AWS 公式ウェブサイト記載の価格に相違があった場合、AWS 公式ウェブサイトの価格を優先とさせていただきます
- 価格は税抜表記となっています。日本居住者のお客様には別途消費税をご請求させていただきます
- 技術的な内容に関しましては、有料の AWS サポート窓口へお問い合わせください
- 料金面でのお問い合わせに関しましては、カスタマーサポート窓口へお問い合わせください (マネジメントコンソールへのログインが必要です)

Thank you!