



2-H1-1-10 / 2-H1-1-19

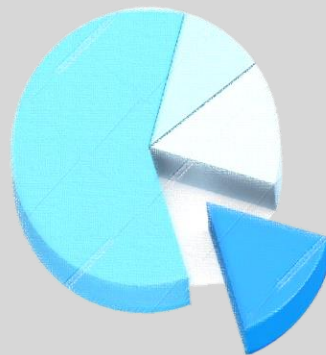
AWS の DataLake を使ったデータ分析入門

アマゾン ウェブ サービス ジャパン 株式会社 技術統括本部 ソリューションアーキテクト

上原 誠

本セッションのFeedbackをお願いします

お手元のサミットガイドブックの表紙に記載している『QRコード』からご回答ください。
もれなく素敵なAWSオリジナルグッズをプレゼントします。



プレゼントの引き換えは、パミール3F展示会場内アンケート確認エリア・受付エリアのいずれかにお越し下さい。

名前：上原 誠 (うえはら まこと)

所属：ソリューションアーキテクト

担当：メディア系、アドテクのお客様

最近よく触ってるサービス：AWS Glue



このセッションの対象者

- ビッグデータを活用したいエンジニア
- データ分析 / ML および周辺エコシステムに興味がある
- アーキテクチャの設計に関わるアーキテクト、デベロッパー、マネージャ

このセッションで持ち帰って頂きたいこと

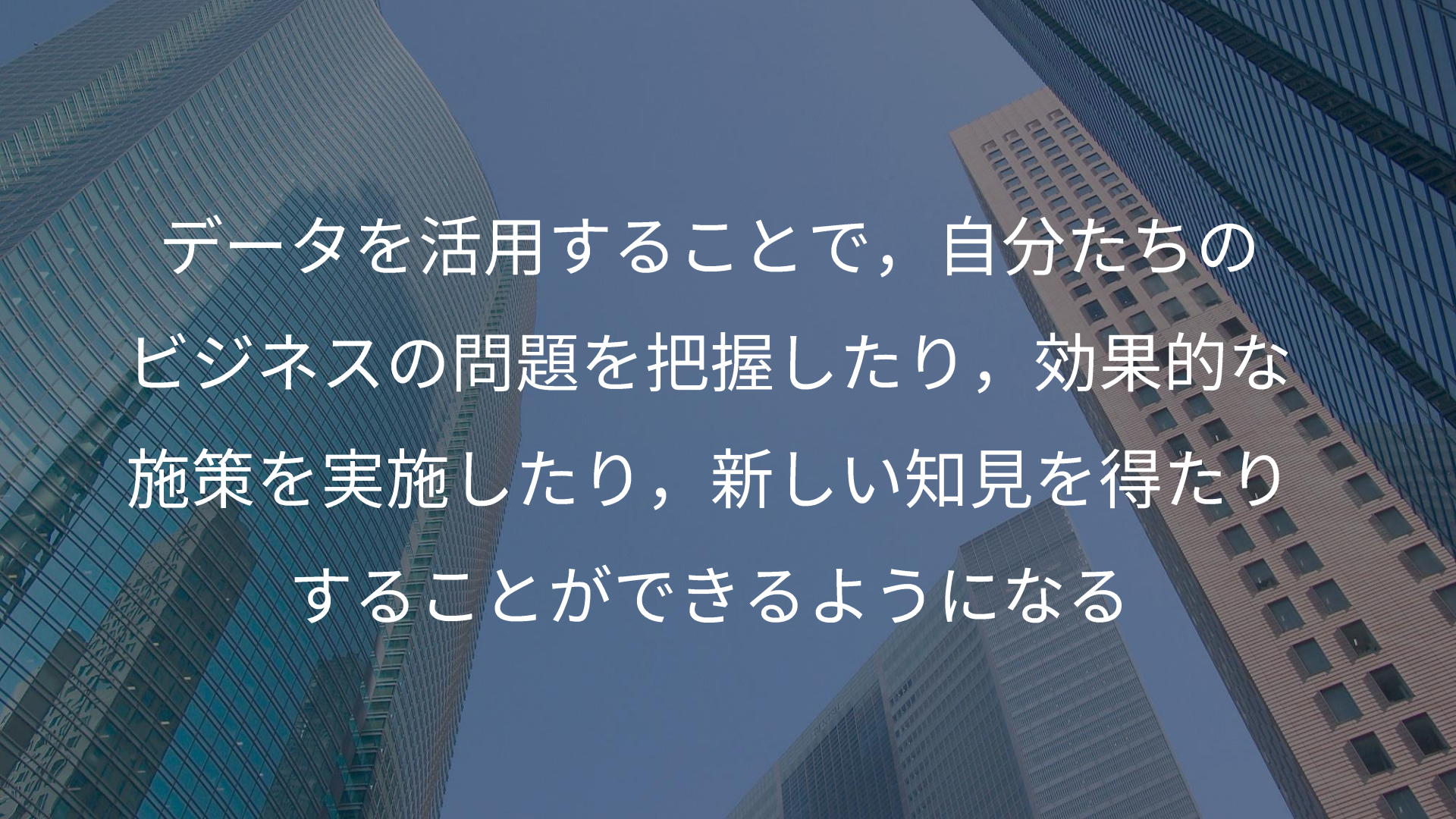
- データ活用の考え方
- データレイクの考え方
- データ活用のための試行錯誤を高速に行うことの必要性とその知識

アジェンダ

1. データ活用の流れ
2. 従来の RDB を中心としたデータ分析
3. データレイクとは
4. データレイクを構成する AWS サービス
5. AWS におけるデータ活用事例
6. まとめ

アジェンダ

1. データ活用の流れ
2. 従来の RDB を中心としたデータ分析
3. データレイクとは
4. データレイクを構成する AWS サービス
5. AWS におけるデータ活用事例
6. まとめ



データを活用することで，自分たちの
ビジネスの問題を把握したり，効果的な
施策を実施したり，新しい知見を得たり
することができるようになる

データ活用の流れ

データを貯める

データを貯める
活用するためのデータがなければ、なにもできない

データを可視化

貯めたデータを適切に加工整形処理して可視化
常に全ての必要なデータを見ることができるようにする

データ
分析 / ML

上記2つが回るようになってはじめて、高度な分析
を行いビジネス課題とゴールが明確にできる

ビッグデータ処理のパイプライン



データ分析において唯一の万能なツールはない



各フェーズで必要なツールやリソースを確保

- データ活用の際には、さまざまなツールが必要となる
 - データの取得と保存、加工整形処理を行うためのツール
 - 可視化- BIツール / 基礎集計のための SQL
 - ディープラーニングフレームワークや Hadoop クラスタ、Notebook などの分析ツール
- ツールやモデルによって必要なリソースも異なる
 - CPU / GPU / メモリ / IO
 - 複数インスタンス / クラスタ

高速に試行錯誤を行う

- データ活用は基本的に試行錯誤を伴う
 - データの量や中身の変化への継続的な対応
 - ビジネスの状況変化に伴う、さまざまな指標の定期的な見直し
 - 機械学習モデルの構築・改善サイクル
- この試行錯誤のサイクルをいかに高速に積み重ねるかが、
良い結果を導き出すために重要

データ分析をうまく進めるには
適切なツールやリソースを確保し

試行錯誤のサイクルを高速に実施する必要がある

アジェンダ

1. データ活用の流れ

2. 従来の RDB を中心としたデータ分析

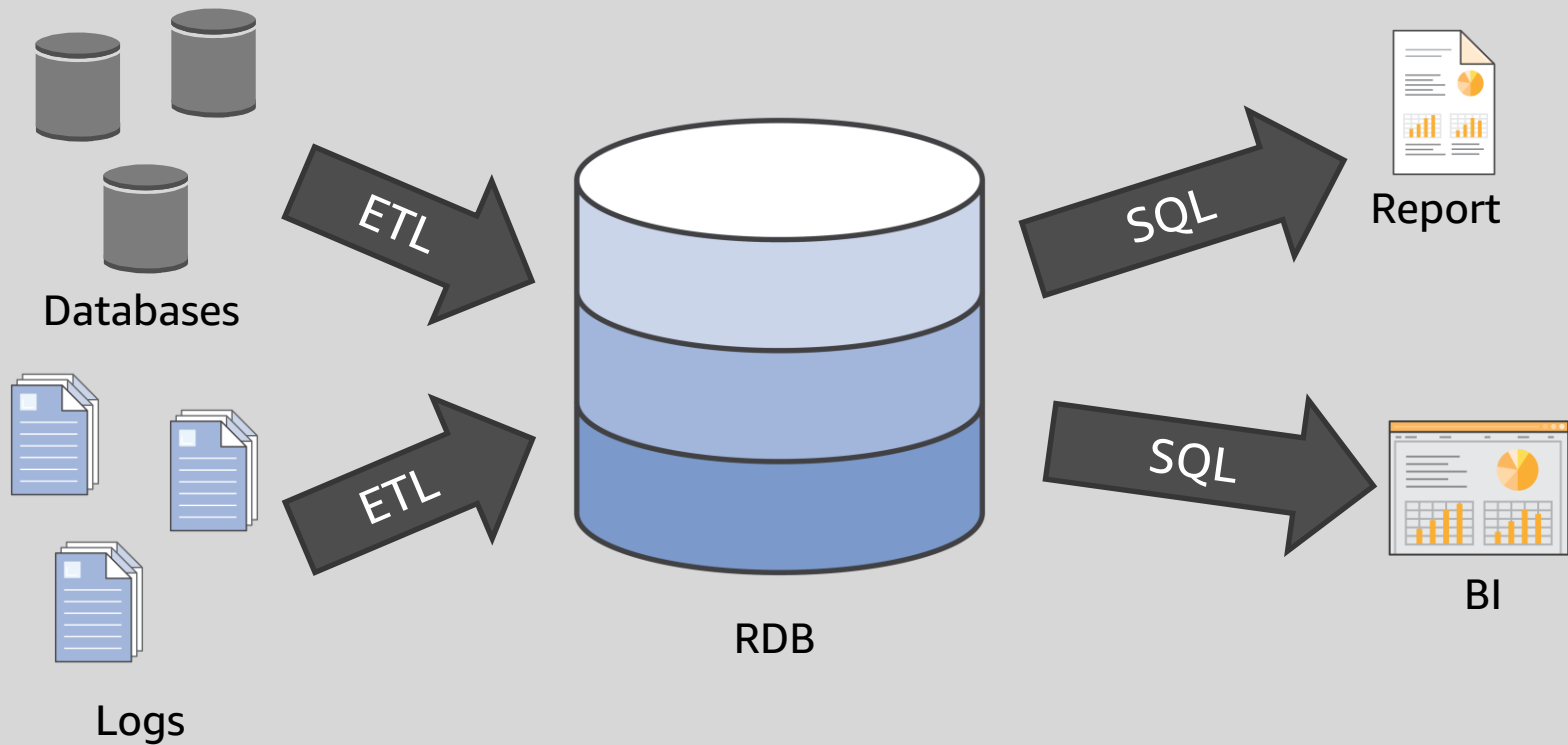
3. データレイクとは

4. データレイクを構成する AWS サービス

5. AWS におけるデータ活用事例

6. まとめ

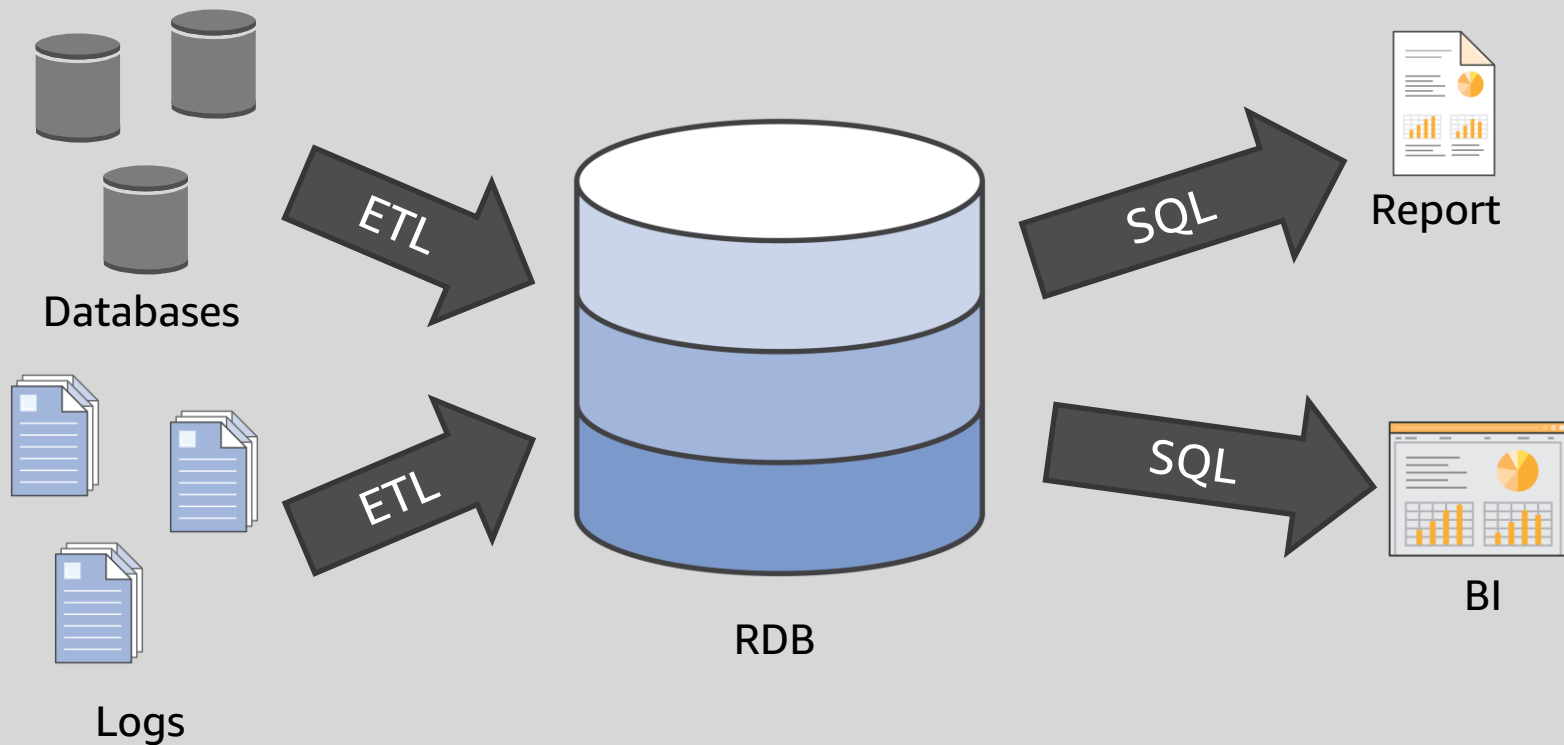
従来の RDB を中心としたデータ分析



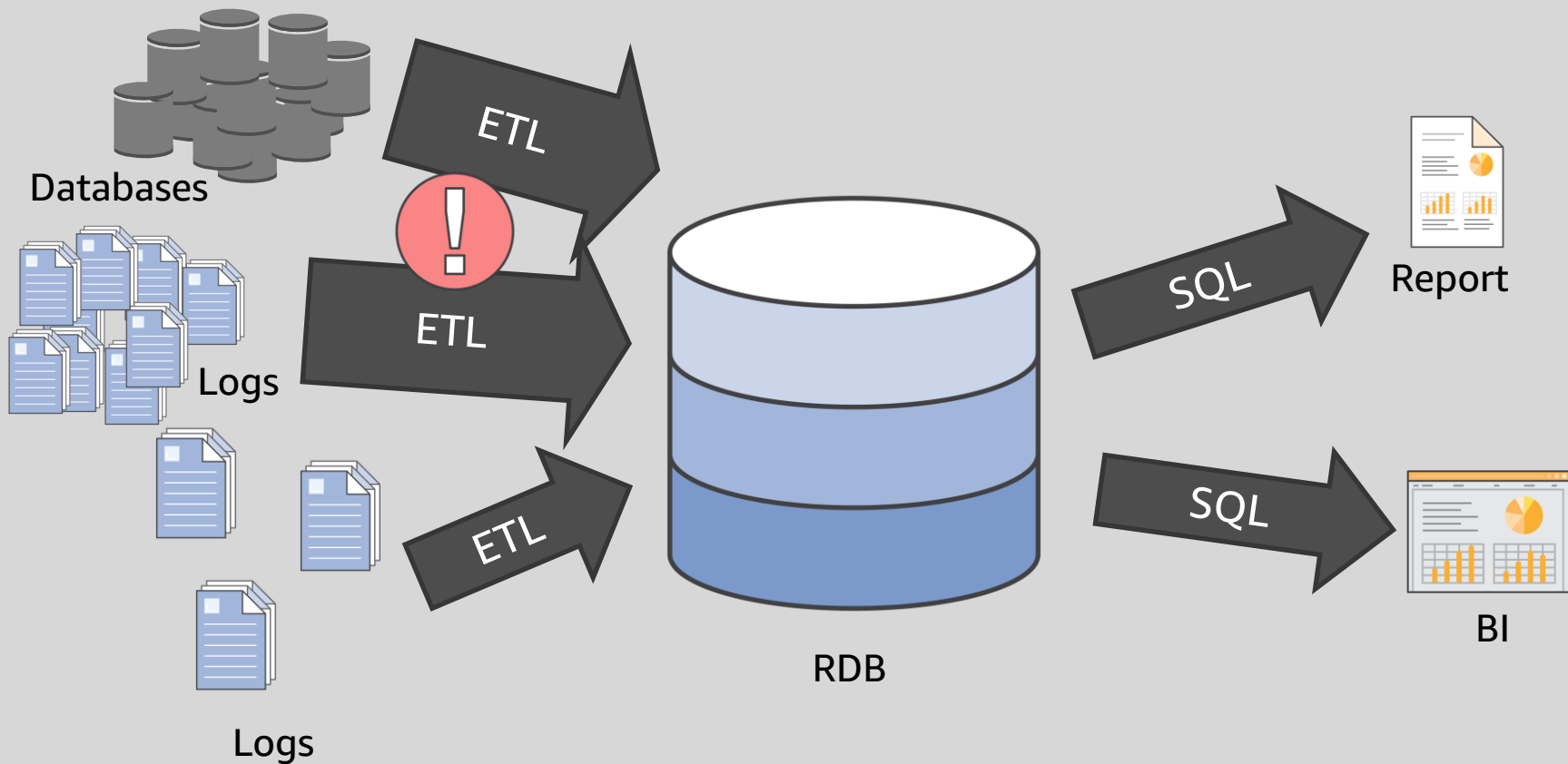
RDB を利用する利点

- スキーマが定義されている
- データの型も強制できる
- アクセスするツールが簡単、エコシステムが安定
- トランザクション

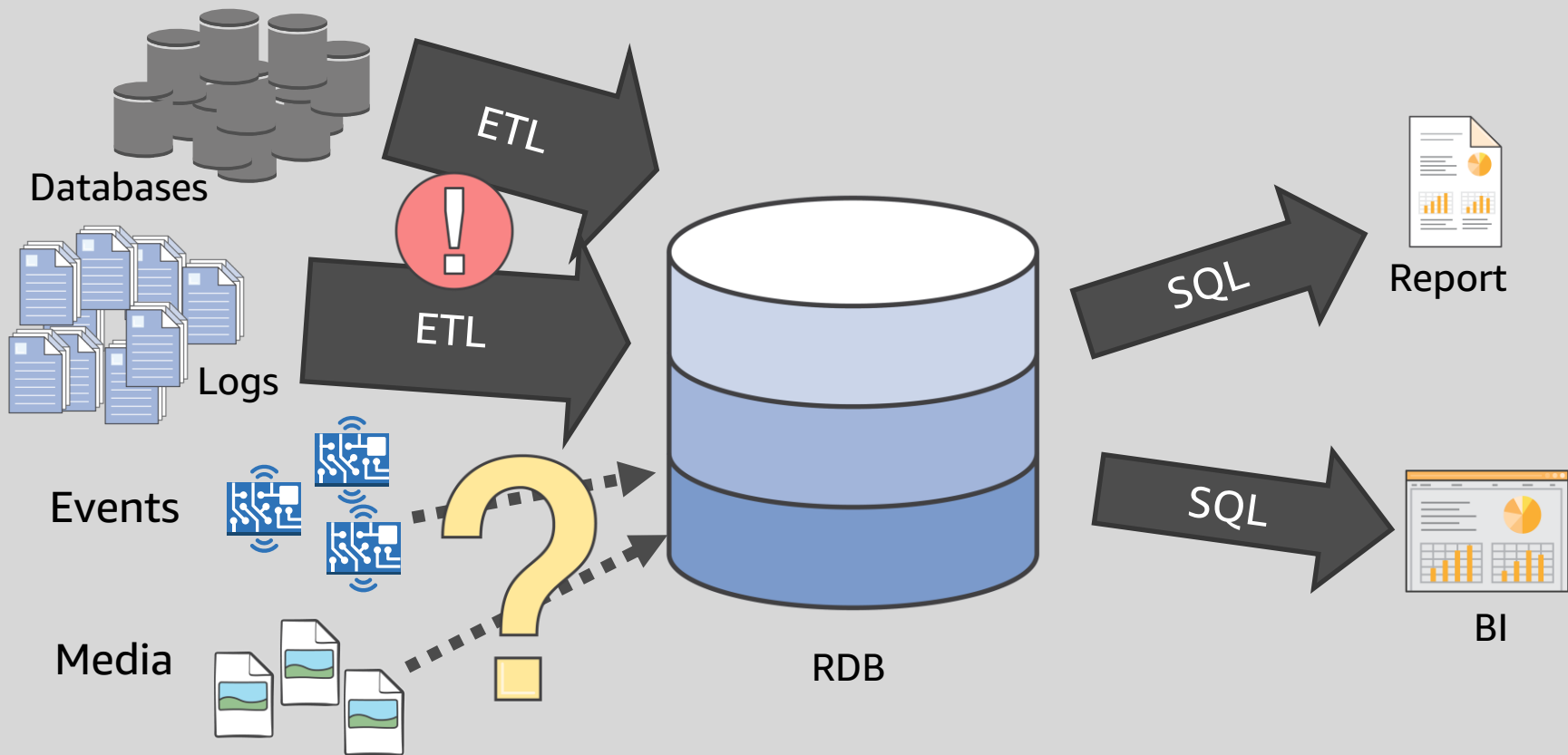
課題1: 入ってくるデータの量と質の変化



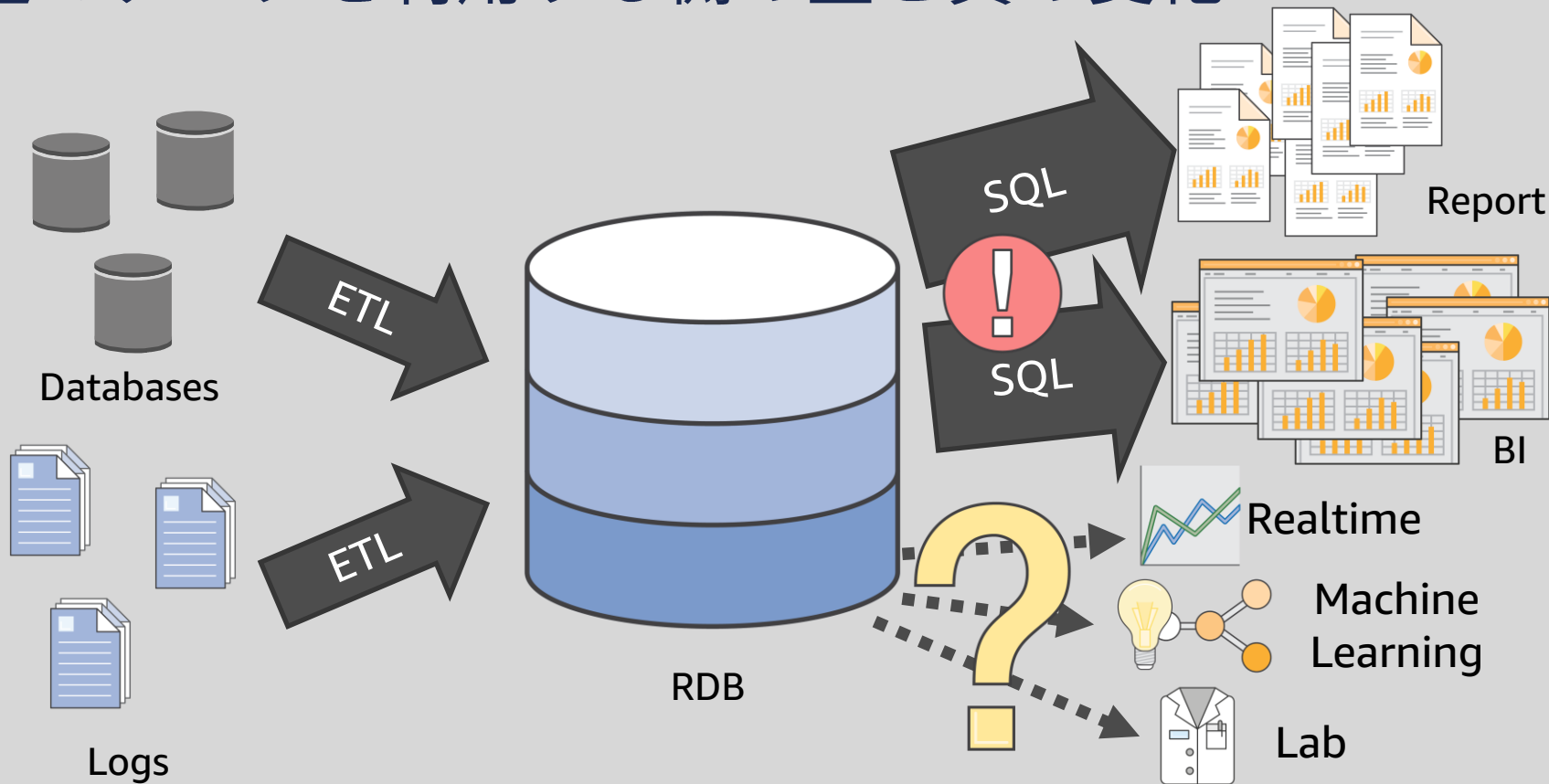
課題1: 入ってくるデータの量と質の変化



課題1: 入ってくるデータの量と質の変化



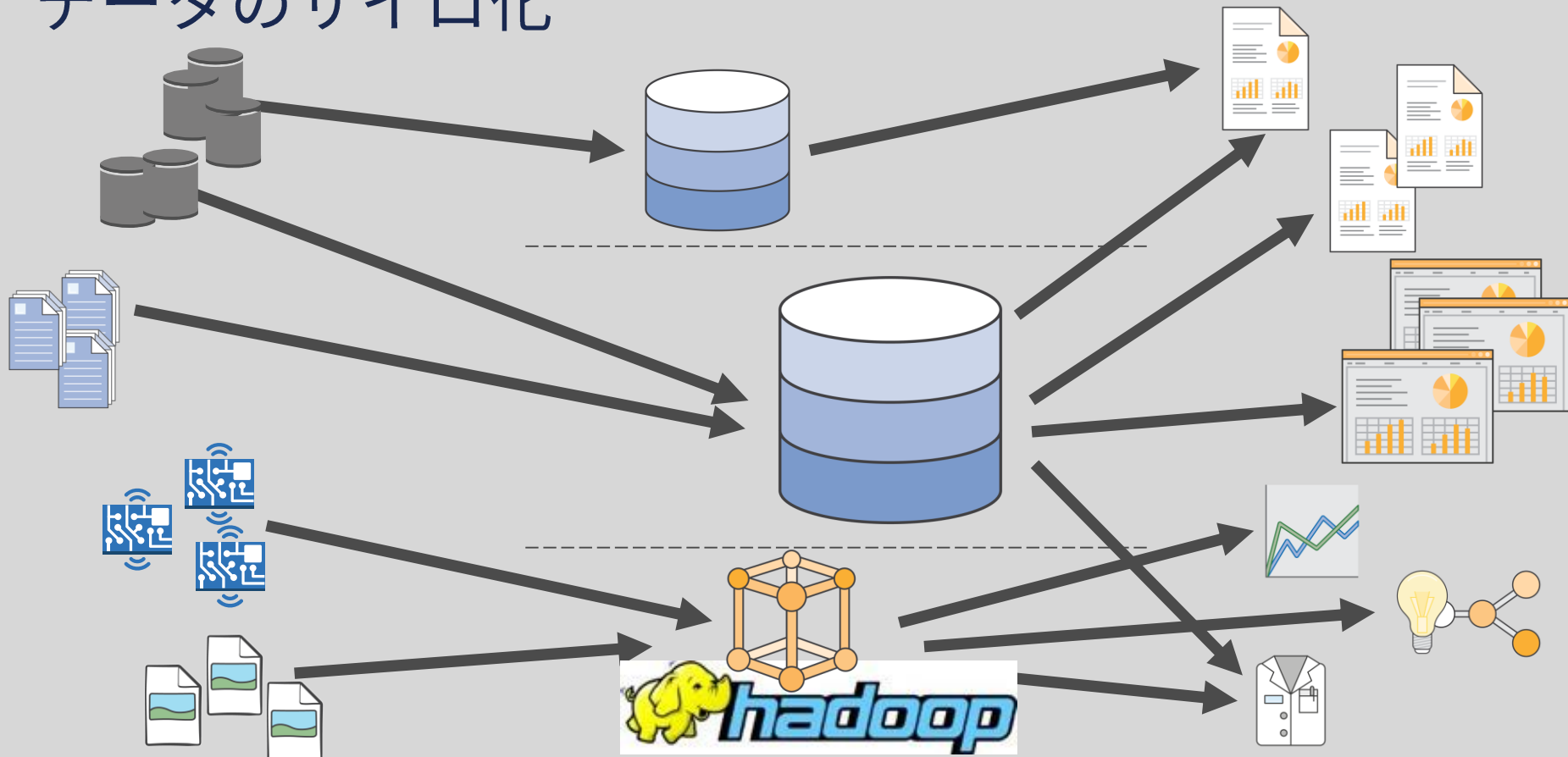
課題2: データを利用する側の量と質の変化



RDB の課題

- スケーラビリティに限界がある
- 非構造化データに対応しづらい
- リアルタイム分析や機械学習などの新たな分析要件に対応しづらい、またはできない

データのサイロ化



データのサイロ化の課題

- スケーラビリティの課題は解消しない
 - 要求が10倍になったら、サイロ数が10倍になる??
- どこにどのデータがあるか分からない
 - 途中で場所が変わったら更にカオスに
- サイロをまたいだ分析が困難
 - 同じデータを複数サイロに重複して持つと、無駄や同期が課題に

アジェンダ

1. データ活用の流れ
2. 従来の RDB を中心としたデータ分析
3. データレイクとは
4. データレイクを構成する AWS サービス
5. AWS におけるデータ活用事例
6. まとめ

データレイクとは

James Dixon's Blog

James Dixon's thoughts
on source and open source

Pentaho, Hadoop, and Data Lakes

with 15 comments

Earlier this week, at Hadoop World in New York, Pentaho announced availability of our first Hadoop release.

As part of the initial research into the Hadoop arena I talked to many companies that use Hadoop. Several common attributes and themes emerged from these meetings:

- 80-90% of companies are dealing with structured or semi-structured data (not unstructured).
- The source of the data is typically a single application or system.
- The data is typically sub-transactional or non-transactional.
- There are some known questions to ask of the data.
- There are many unknown questions that will arise in the future.
- There are multiple user communities that have questions of the data.
- The data is of a scale or daily volume such that it won't fit technically and/or economically into an RDBMS.

In the past the standard way to handle reporting and analysis of this data was to identify the most interesting attributes, and to aggregate these into a data mart. There are several problems with this approach:

- Only a subset of the attributes are examined, so only pre-determined questions can be answered.
- The data is aggregated so visibility into the lowest levels is lost

Based on the requirements above and the problems of the traditional solutions we have created a concept called the Data Lake to describe an optimal solution.

If you think of a datamart as a store of bottled water – cleansed and packaged and structured for easy consumption – the data lake is a large body of water in a more natural state. The contents of the data lake stream in from a source to fill the lake, and various users of the lake can come to examine, dive in, or take samples.

For more information on this concept you can watch a presentation on it here: [Pentaho's Big Data Architecture](#)

<https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/>

増え続けるデータに対して多くの企業での課題

Hadoop 初期の研究で、Hadoop を使用する多くの企業で共通の課題

- 構造化データと非構造化データの混在
- 分析パターンの把握と整理の難しさ
- ビッグデータへの対処

データレイク

どのような分析を行いたいかを事前に完璧に把握することはできない

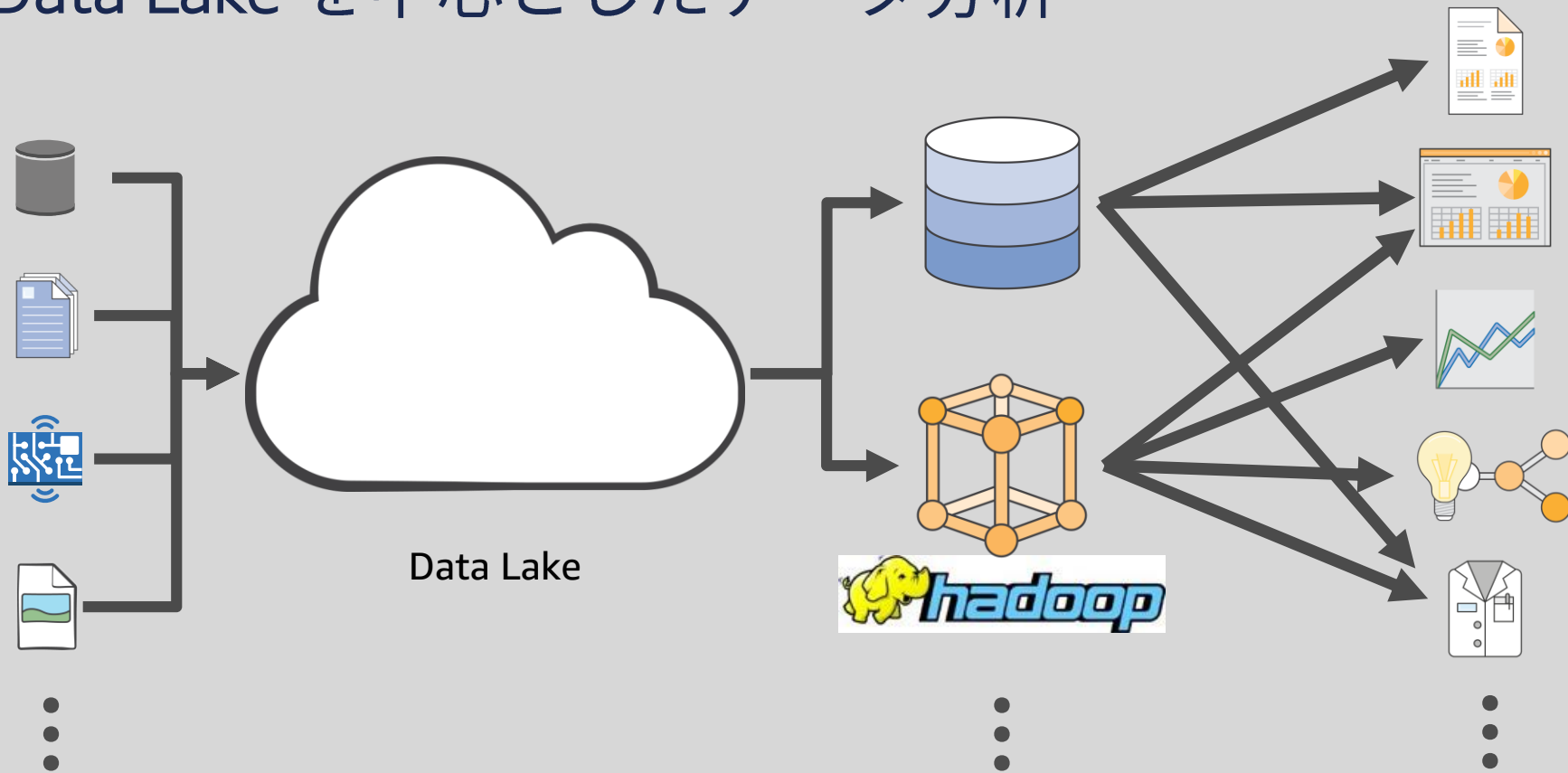
データレイクは、どんな要望がでてきても対処できるように

全てのデータを生のフォーマットで安全に保存し

データ活用時に必要な構造で取り出すという考え方

データレイクは未来の不確実性に対処する術

Data Lake を中心としたデータ分析

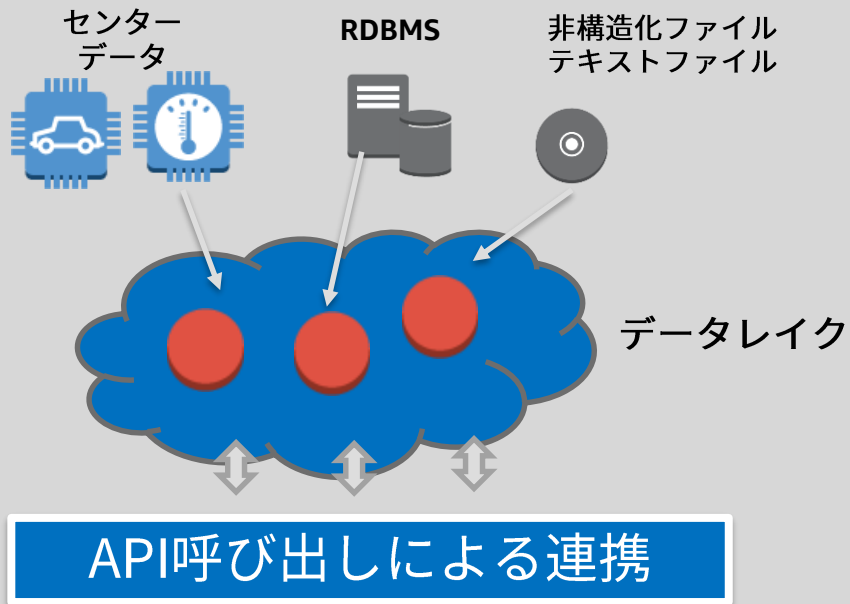


データレイクのメリット

- ストレージと計算処理の分離
 - それぞれ独立してスケールできるので最適化しやすい
- Single Source of Truth (SSOT)
 - データレイクにあるものを正とすれば良い
- 様々な input/output 手法に対応
 - in/out が独立、ETL も独立できるので、後からの拡張がスムーズ

データレイクとは (ここまでのまとめ)

- 多様なデータを一箇所に保存
- データを失わない
- サイズ制限からの開放
- 決められた方法 (API) ですぐにアクセスできる
- システム全体のハブ



アジェンダ

1. データ活用の流れ
2. 従来の RDB を中心としたデータ分析
3. データレイクとは
4. データレイクを構成する AWS サービス
5. AWS におけるデータ活用事例
6. まとめ

データ処理のパイプライン



クラウドで変わるデータ分析： データは加工せず全期間を残す

これまで：

1. ディスクの容量に上限がある
2. データはサマリーだけ、もしくは期間限定で保存
3. 処理できる内容は固定的

インフラ管理者の仕事：
ストレージを溢れさせず、時間内に処理が終るようにサイズや処理内容を調整する

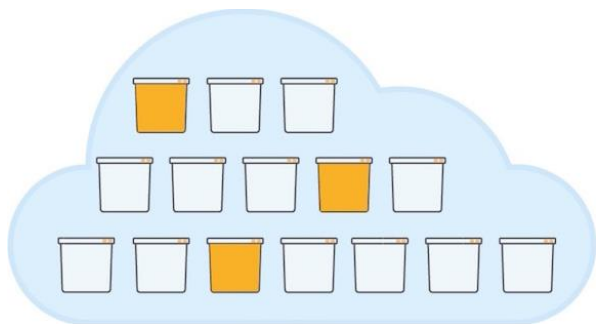
On AWS：

1. 安価・上限無しのストレージ
2. オリジナルデータを全て残す
3. 処理対象・処理内容はビジネスに合わせて変わる

インフラ管理者の仕事：
データを活用して新しい課題に素早く対応できるインフラを用意する。個別リクエストへの対応

Amazon S3

高い耐久性と可用性を持つスケーラブルなオブジェクトストレージ



- 容量無制限
- 99.999999999%の耐久性
- 利用した分だけの課金
- API操作可能 多くの AWS サービスと連携

AWSのデータレイク = Amazon S3



AWSのデータレイク = Amazon S3

様々なアクセスに対する S3 のセキュリティ



- IAM を使ったきめ細かなアクセス制御



- CloudTrail を使った S3 操作ログの記録と監査

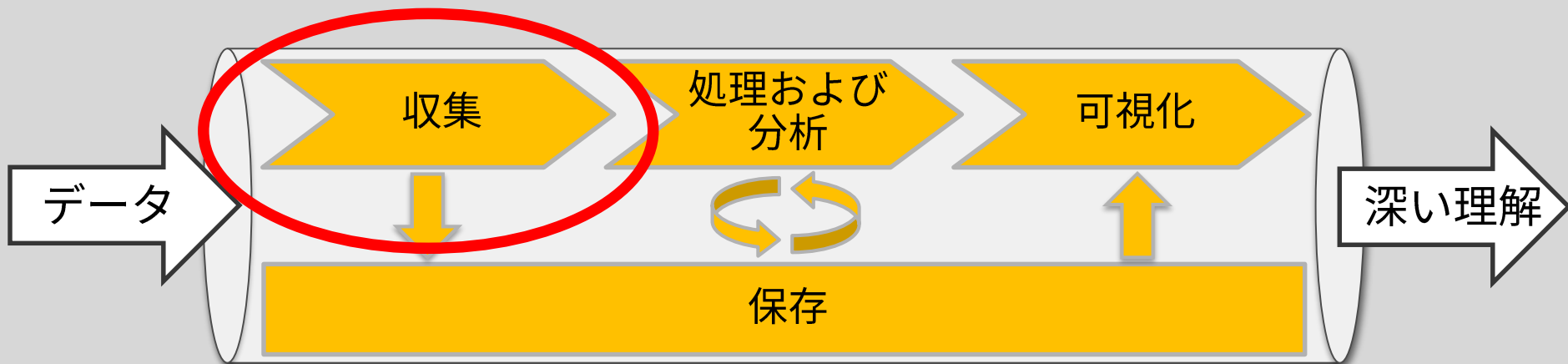


- S3 へのアクセスログの記録と監査

- バケットポリシーによるアクセス制御

- オブジェクトの暗号化 (CSE, SSE)

データ処理のパイプライン



要件に合わせた収集のツールやサービスを使う

OSS on EC2

Fluentd
Logstash
Flume



Amazon Kinesis



リアルタイムな
ストリーミング
処理のための安全
なストレージ

AWS Snowball



物理デバイスで
オンプレミスの
ペタバイト規模
のデータをS3に
セキュアに転送

AWS IoT



MQTTやHTTPSに
よる数十万規模
のデバイスから
のセキュアな
データ収集

広く使われる OSS



OVERVIEW

Plugins

List of Plugins By Category

Find plugins by category ([Find all listed plugins here](#))

[Amazon Web Services](#) / [Big Data](#) / [Filter](#) / [Google Cloud Platform](#) / [Internet of T](#)

AMAZON WEB SERVICES

Certified	Download	Name	Author	About	Version
	13150	amazon_sns	Tatsuhiko Miyagawa	Fluent output plugin to send to Amazon SNS	0.0.6

The screenshot shows the Elastic Logstash Reference documentation page for Input plugins. The page title is "Input plugins" and it includes a description: "An input plugin enables a specific source of events to be read by Logstash. The following input plugins are available below. For a list of Elastic supported plugins, please consult the [Support Matrix](#)." Below the description is a table listing input plugins:

Plugin	Description	Github repository
beats	Receives events from the Elastic Beats framework	logstash-input-beats
cloudwatch	Pulls events from the Amazon Web Services CloudWatch API	logstash-input-cloudwatch

Additional elements in the screenshot include the Elastic logo, navigation links (Products, Cloud, Services, Customers, Learn), a search bar, and a sidebar with "Getting Started Videos" (Starting Elasticsearch, Introduction to Kibana, Logstash Starter Guide) and a version selector for Logstash Reference (6.2 (current)).

25

[kinesis](#)

Amazon Web Services

Fluentd output plugin that sends events to Amazon Kinesis.

08

[kinesis-aggregation](#)

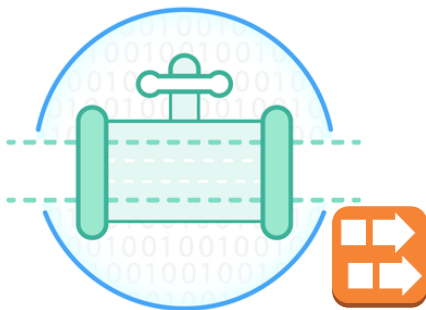
Atlassian

Fluentd output plugin that sends KPL style aggregated events to Amazon Kinesis.



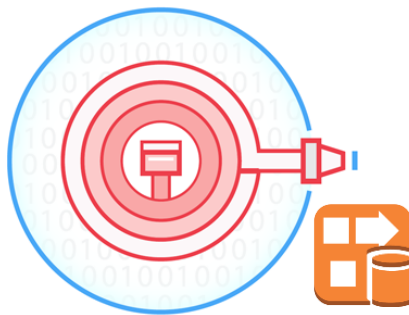
Amazon Kinesis

ストリームデータを収集・処理・配信するためのフルマネージドサービス群



Kinesis Data Streams

ストリームデータを
処理・分析するための
データを格納



Kinesis Data Firehose

ストリームデータを
S3, Redshift,
Amazon ES, Splunk に簡
単にロード

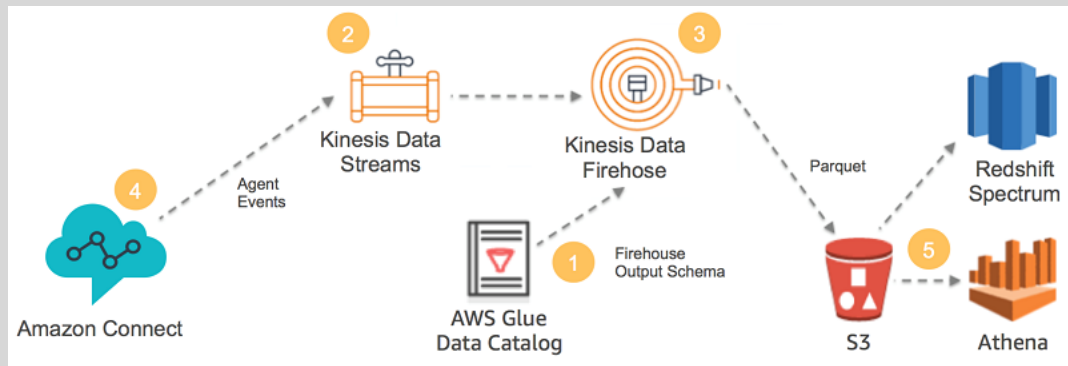


Kinesis Data Analytics

ストリーミングデータを
標準的な SQL クエリで
簡単に分析

Kinesis Data Firehose から AWS Athena へ連携

2018.5.10 GA



Convert record format

Data in Apache Parquet or Apache ORC format is typically more efficient to query than JSON. Kinesis Data Firehose can convert your JSON-formatted source records using a schema from a table defined in [AWS Glue](#). For records that aren't in JSON format, create a Lambda function that converts them to JSON in the **Transform** source records with [AWS Lambda](#) section above.

Record format conversion Disabled Enabled 1

If record format conversion is enabled, Firehose can deliver data to Amazon S3 only. Record format conversion will be configured using the OpenX JSON SerDe. For other options use the [AWS CLI](#).

Output format* Apache Parquet Apache ORC 2

Specify a schema for source records. Kinesis Data Firehose references table definitions stored in AWS Glue. Choose an AWS Glue table to specify a schema for your source records. You can [manually create a new table in AWS Glue](#) or [add a crawler in AWS Glue](#) to create a new table using a schema from an existing JSON object in S3. [Learn more](#)

AWS Glue region* US East (N. Virginia) 3

AWS Glue database* default 4

[View database-011 in AWS Glue](#)

AWS Glue table* kfhconnectblog

AWS Glue table version* Latest

<https://aws.amazon.com/blogs/big-data/analyzing-apache-parquet-optimized-data-using-amazon-kinesis-data-firehose-amazon-athena-and-amazon-redshift/>

要件に合わせたツールでデータレイクにデータ収集

オープンソース 物理的なデータ転送 IoTデバイスログ DBマイグレーション

Fluentd
Logstash
Flume

OSS



Snowball



AWS IoT



AWS DMS



Amazon Kinesis



Amazon S3



Amazon
Elasticsearch
Service

データ処理のパイプライン



要件に合わせた分析・処理サービスを使う

Amazon Redshift



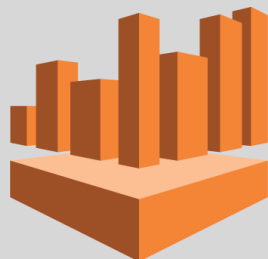
高速DWHとしての機能+S3上への全データへのアクセス

Amazon EMR



自由度が高く、高いスケーラビリティを誇るプログラム実行基盤

Amazon Athena



S3上のデータにアクセス。サーバ管理や準備不要でアドホックなクエリに強い

AWS Glue

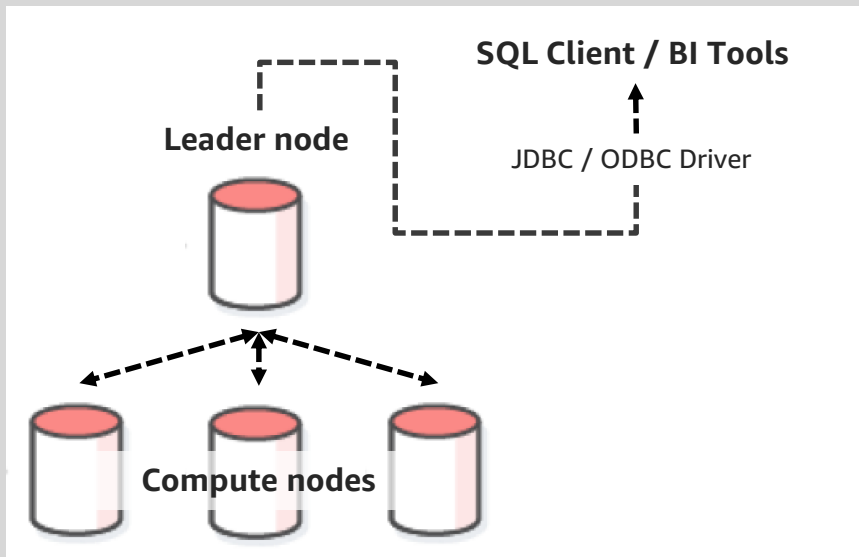


サーバーレスなETL処理と共通のスキーマ情報のクローリングと管理



Amazon Redshift

フルマネージドでスケーラブルなデータウェアハウスサービス

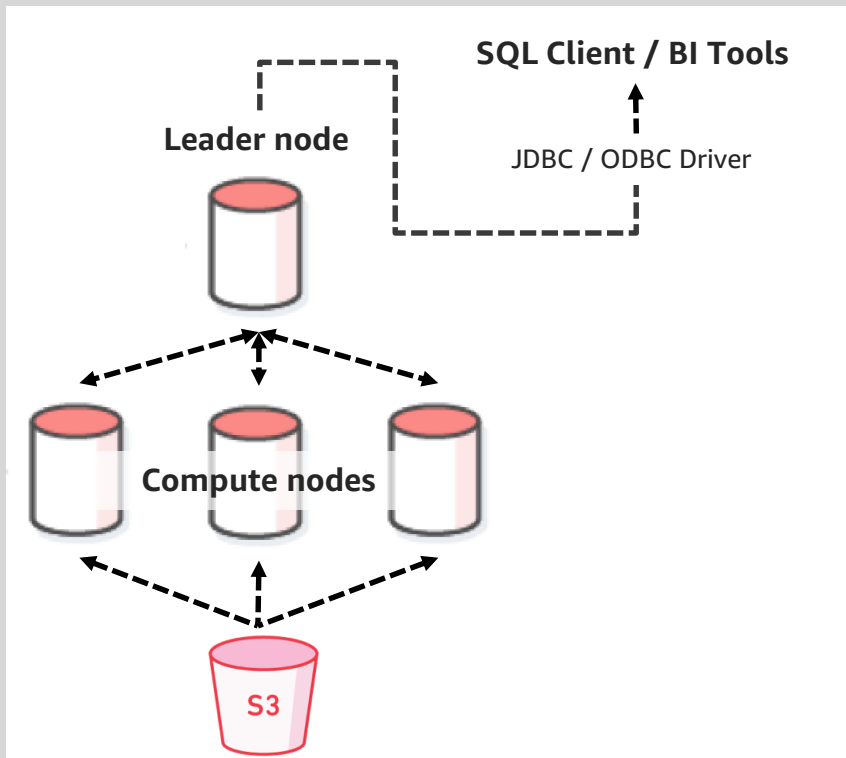


- MPP(Massively Parallel Processing) アーキテクチャとカラムナofデータ格納により、スケーラブルで高速なクエリが実行可能
- データストアを最大 2PB まで拡張
- JDBC/ODBC 経由でさまざまな BI ツールと連携



Amazon Redshift Spectrum

Redshift から S3 上のデータに直接クエリできる拡張機能



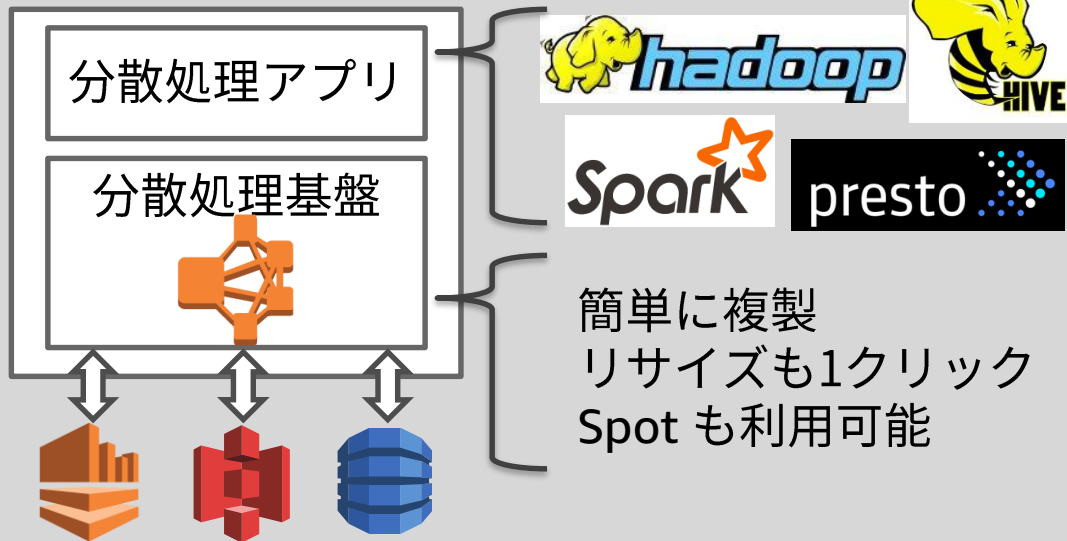
- Redshift クラスタから、直接 S3 上のデータに対してクエリを実行
- Redshift 内のデータと JOIN することも可能
- 利用頻度の低いコールドデータを S3 に保存して Spectrum 経由でアクセスし、ホットデータのみ Redshift 内にロードしておく
- 複数の Redshift クラスタから同じ S3 上のデータを読み込む



Amazon EMR

大規模データ処理を Hadoop/Spark などの分散処理フレームワークを使って効率的に処理

Amazon EMR クラスタ



簡単に複製
リサイズも1クリック
Spot も利用可能

AWS 上の分散処理サービス

- 簡単かつ安全に Big Data を処理
- 多数のアプリケーションサポート

簡単スタート

- 数クリックでセットアップ完了
- 分散処理アプリも簡単セットアップ

低コスト

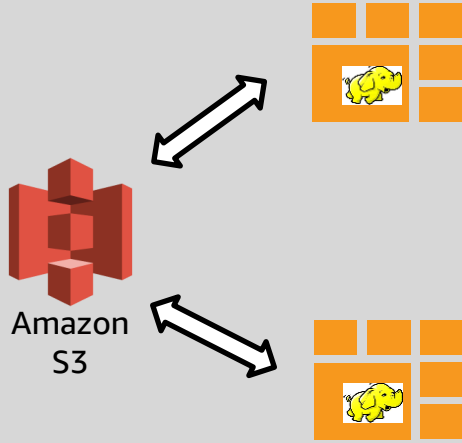
- ハードウェアへの投資不要
- 従量課金制
- 処理の完了後、クラスタ削除
- Spot インスタンスの活用

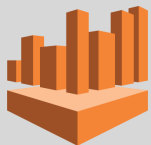


EMRFS: S3 を HDFS の様に扱う

“s3://” と指定するだけでHDFSと同様にS3にアクセス

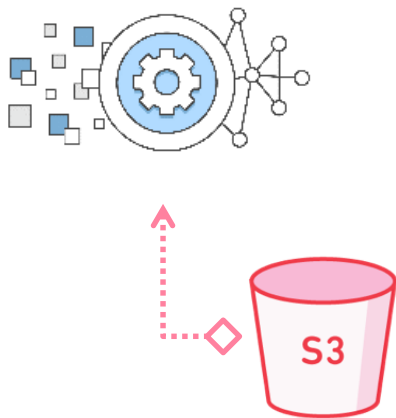
- 計算資源とストレージを分離できる
 - コスト面でもメリット大
- クラスターの削除が可能
 - クラスターを消してもデータをロスしない
- 複数クラスター間でデータ共有が簡単
- データは耐久性の高い S3 に配置





Amazon Athena

サーバーレスのインタラクティブなクエリサービス



- サーバーレス検索サービス
- S3 のデータに対して直接クエリできる
- Presto ベースで標準 SQL が実行可能
- 実行したクエリの容量ぶんの従量課金
- スキャンされたデータ 1TBあたり 5\$



Amazon Athena

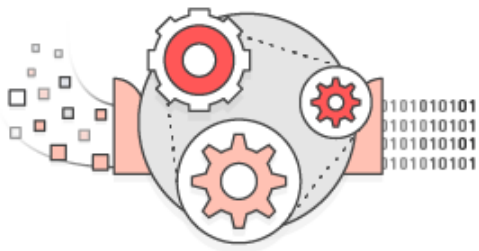
S3 に保存したファイルに直接クエリー

The screenshot shows the Amazon Athena console interface. At the top, there are tabs for 'New query 1' through 'New query 5'. The first tab is active, showing a SQL query: `SELECT * FROM "sampledb"."elb_logs" limit 10;`. Below the query editor, there are buttons for 'Run query', 'Save as', 'Format query', and 'New query'. The 'Run query' button is highlighted with a red box. Below the buttons, the console displays the results of the query in a table format. The table has columns for 'request_timestamp', 'elb_name', 'request_ip', 'request_port', 'backend_ip', 'backend_port', and 'rec'. Three rows of data are visible, each with a row number (1, 2, 3) in the first column.

	request_timestamp	elb_name	request_ip	request_port	backend_ip	backend_port	rec
1	2015-01-01T08:00:00.516940Z	elb_demo_009	240.136.98.149	25858	172.51.67.62	8888	9.9
2	2015-01-01T08:00:00.902953Z	elb_demo_008	244.46.184.108	27758	172.31.168.31	443	6.3
3	2015-01-01T08:00:01.206255Z	elb_demo_008	240.120.203.212	26378	172.37.170.107	8888	0.0

- クエリエディタにSQLを記述しクエリーを実行
- コンソールに結果が表示
- 過去のクエリー結果は History からダウンロード可能
- クエリー結果は S3 に自動保存

データソースの把握・準備・データ格納を簡単で確実に



- フルマネージドのETL サービス + データカタログ
- GUI 上で作成した ETL 処理フローから **PySpark** または **Scala** コードが生成され必要に応じて編集可能
- データカタログは、**Athena, Redshift Spectrum, EMR 上の Spark, Hive** からも利用可能



Glue : GUI での ETL コード雛形生成

Add job: schema

Job properties: DeleteMe, Data source (city_baltimore), Data target (canonical), Schema, Review

Column name	Data type	Map to target	Column name	Data type
crimedate	string	crimedate	crimedate	string
crimetime	string	crime_time	crime_time	string
crimecode	string	crimecode	crimecode	string
location	string	location	location	string
description	string	description	description	string
inside/outside	string	-	weapon	string
weapon	string	weapon	district	string
post	bigint	-	neighborhood	string
district	string	district	total incidents	long
neighborhood	string	neighborhood		
location 1	string	-		
premise	string	-		
total incidents	bigint	total incidents		

Services Resource Groups

Action Generate diagram

Insert template at cursor

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7
8 ## @params: [JOB_NAME]
9 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
10
11 sc = SparkContext()
12 glueContext = GlueContext(sc)
13 job = Job(glueContext)
14 job.init(args['JOB_NAME'], args)
15 ## @type: DataSource
16 ## @args: [database = 'nytaxianalysis', table_name = 'city_baltimore', transformation_ctx = '']
17 ## @return: DataSource
18 ## @inputs: []
19
```

Database Name nytaxianalysis
Table Name city_baltimore

Transform Name ApplyMapping

Transform Name SelectFields

Database Name incidents
Table Name canonical

Logs Schema Statistics

- GUIでデータソース、ターゲット、列のマッピングを設定することで雛形が生成される
- CSV→Parquetのフォーマット変換だけといった処理であればコード作成無しで実現



Glue : ETL コードの作成

ジョブ = ETL処理を実行する単位

- PySpark, Scala で記述
- Extract (抽出) や、Load (取り込み) は抽象化されているため、主に **Transform (変換)** を既述する

■ サンプルスクリプト集

<https://github.com/aws-labs/aws-glue-samples>

(※同じサイトにFAQもあり、こちらも必読)

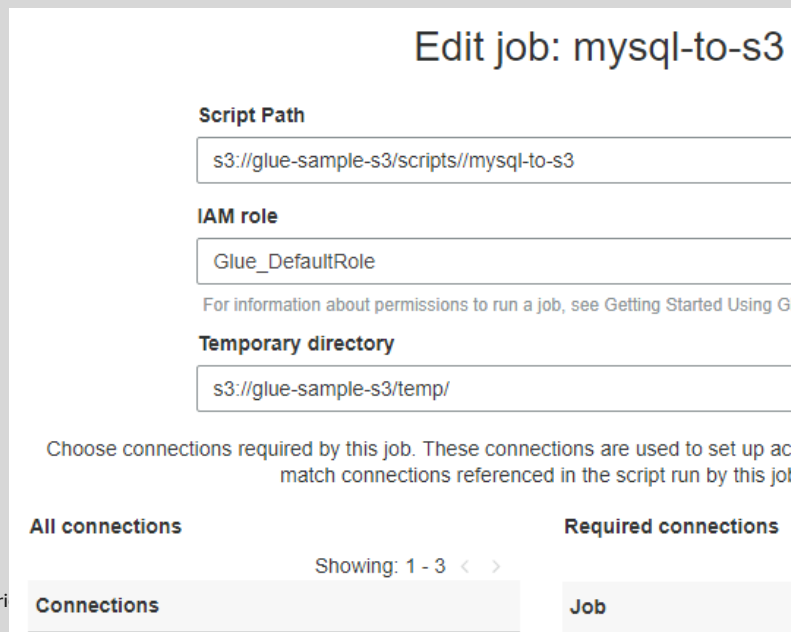
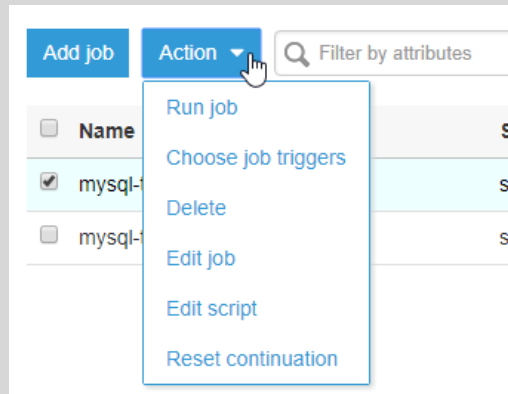
The screenshot displays the AWS Glue console interface. On the left, a diagram shows the job configuration: a source database 'sakila' with table 'sakila_film' is connected to a transform 'ApplyMapping', which is then connected to a target database 'pgsql-db3'. On the right, the PySpark code for the job is shown, including imports for GlueContext, SparkContext, and ApplyMapping, and the job execution logic.

```
1 import sys
2 from aws glue .utils import getResolvedOptions
3 from pyspark .context import SparkContext
4 from aws glue .context import GlueContext
5 from aws glue .job import Job
6 from aws glue .transforms import ApplyMapping
7
8 ## @params: [JOB_NAME]
9 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
10
11 sc = SparkContext()
12 glueContext = GlueContext(sc)
13 job = Job(glueContext)
14 job.init(args['JOB_NAME'], args)
15 ## @type: DataSource
16 ## @args: [database = "sakila", table_name = "sa
17 ## @return: datasource0
18 ## @inputs: []
19 datasource0 = glueContext.create_dynamic_frame.f
20 ## @type: ApplyMapping
21 ## @args: [mapping = [{"special_features", "stri
22 ## @return: applymapping1
23 ## @inputs: [frame = datasource0]
24 applymapping1 = ApplyMapping.apply(frame = datas
25 ## @type: DataSink
26 ## @args: [catalog_connection = "pgsql-db3", con
27 ## @return: datasink2
28 ## @inputs: [frame = applymapping1]
29 datasink2 = glueContext.write_dynamic_frame.from
30 job.commit()
```



Glue : ジョブの定義と実行

- 作成したコードを読み込んで実行
 - IAM ロールで権限を設定
- ジョブの実行開始方法
 - API コール (手動)
 - トリガー (スケジュール実行可能)
- リトライ制限の指定や、パラメータを渡すことが可能
- 実行ログは CloudWatch Logs に出力

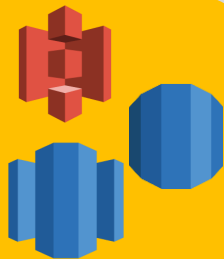




データカタログで AWS 上のメタデータを集中管

理

クローラーで
メタデータ収集



Glue data
catalog

Uses

Uses

Uses

Uses

ETL (変換)



AWS Glue

アドホック分析



Amazo Athena

データウェアハウス



Redshift Spectrum

分析業務



EMR (Hive/Spark)

メタデータ（定義）の例

テーブルの
主要情報

テーブルの
プロパティ

テーブル
スキーマ

The screenshot displays the AWS Glue console interface for a table named 'ny_pub'. The left sidebar contains navigation options like 'Data catalog', 'Databases', 'Tables', 'Connections', 'Crawlers', 'Classifiers', 'ETL', 'Jobs', 'Triggers', 'Dev endpoints', 'Tutorials', 'Add a crawler', and 'Explore table'. The main content area shows the table's details, including its name, description, database, classification, location, connection, and last updated date. Below this, there are sections for 'Properties' and 'Schema'. The 'Properties' section lists various crawler-related settings. The 'Schema' section shows a table with 12 columns, including 'vendorid', 'pickup_datetime', 'dropoff_datetime', 'ratecode', 'passenger_count', 'trip_distance', 'fare_amount', 'total_amount', 'payment_type', 'year', 'month', and 'type'. The 'type' column is marked as a partition key.

Table Information:

- Name: ny_pub
- Description: nrtaxianalysis
- Database: nrtaxianalysis
- Classification: parquet
- Location: s3://[redacted]/canonical/NY-Pub/
- Connection: No
- Deprecated: No
- Last updated: Tue Jul 25 09:49:20 GMT-400 2017

Properties:

- sizeKey: 32032431007
- objectCount: 19
- UPDATED_BY_CRAWLER: RidesCrawler
- CrawlerSchemaSerializerVersion: 1.0
- recordCount: 2870781820
- averageRecordSize: 11
- CrawlerSchemaDeserializerVersion: 1.0
- compressionType: none
- typeOfData: file

Schema:

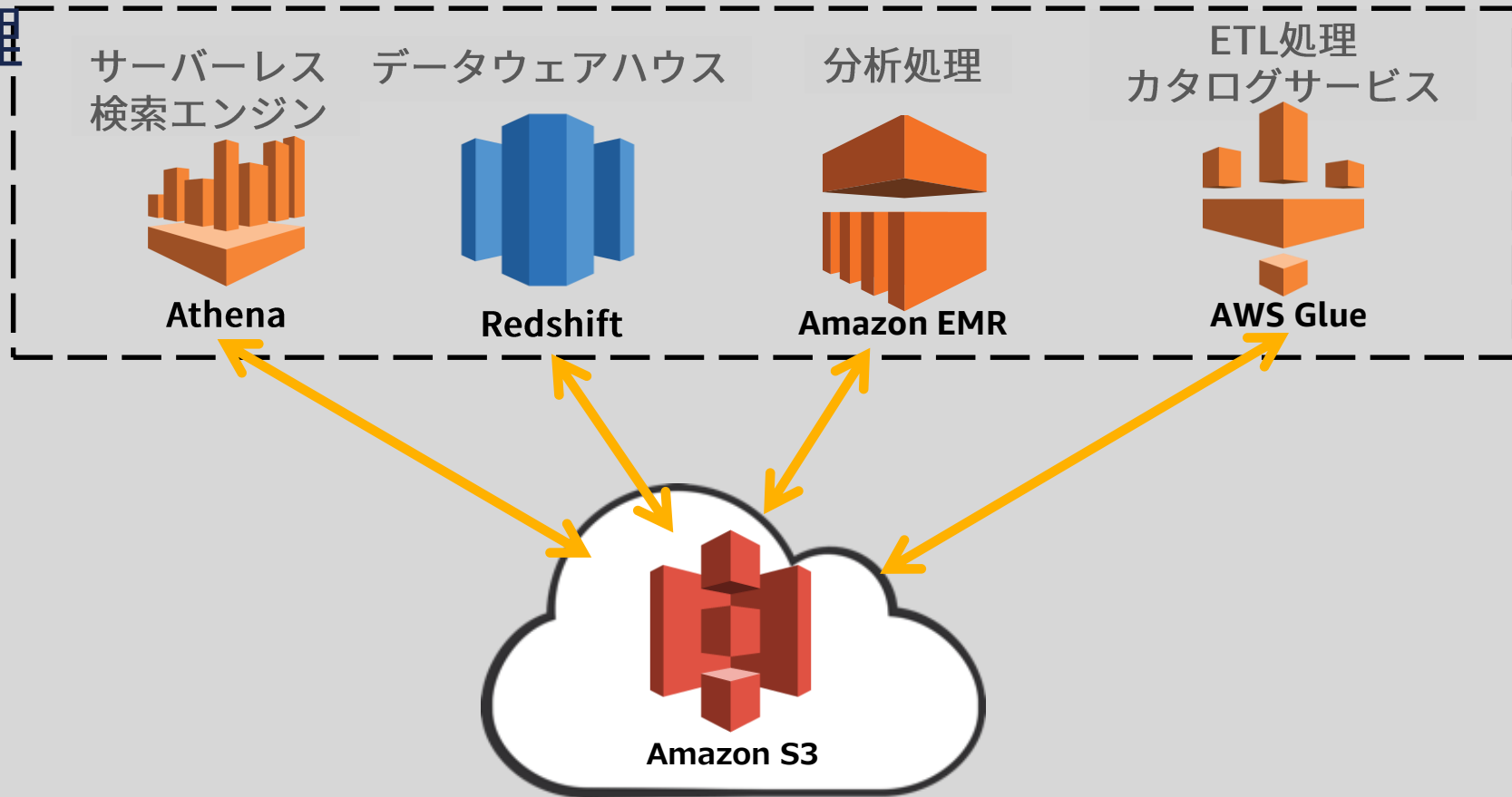
	Column name	Data type	Key
1	vendorid	string	
2	pickup_datetime	bigint	
3	dropoff_datetime	bigint	
4	ratecode	int	
5	passenger_count	int	
6	trip_distance	double	
7	fare_amount	double	
8	total_amount	double	
9	payment_type	int	
10	year	string	Partition (0)
11	month	string	Partition (1)
12	type	string	Partition (2)

テーブル
バージョン

テーブル
パーティション

要件に合わせたサービスでデータレイクを分析処理

理



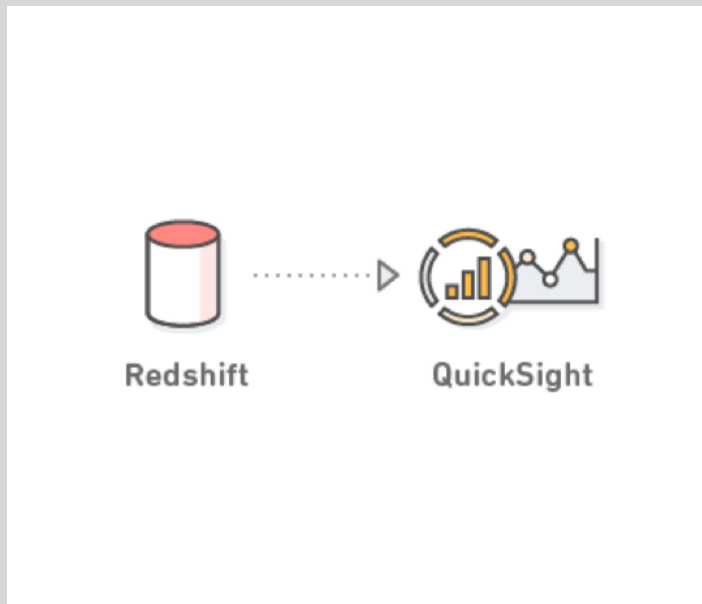
データ処理のパイプライン





Amazon QuickSight

フルマネージドな BI サービス



- 1 ユーザあたり \$9/月からの安価な費用
- サーバレスでデータ分析
- ハイパフォーマンスなデータ処理エンジン
SPICE を持ち、高速に分析が可能
- Redshift, RDS, S3, Athena, Salesforce, ローカル
ファイルなどさまざまなソースに接続可能
- AWS のデータソースをすばやく可視化して、
組織で共有

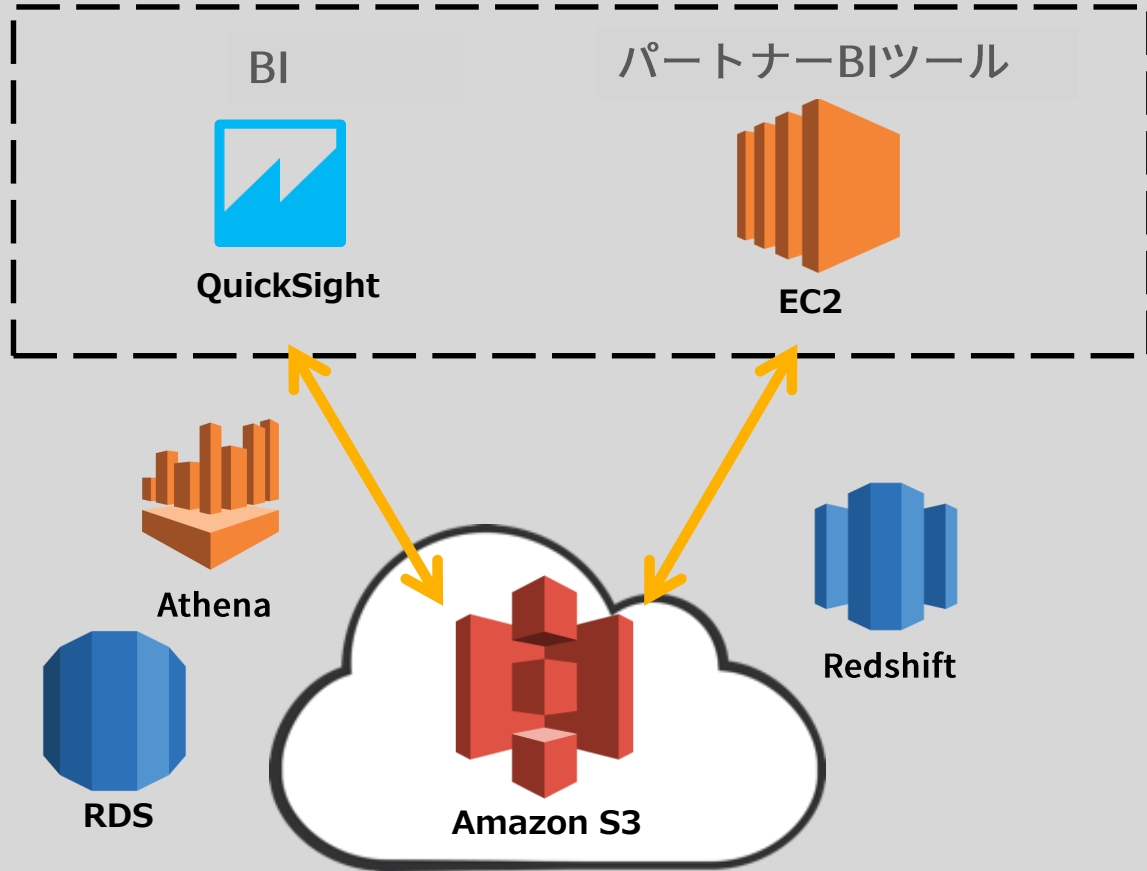
パートナー様ソリューション

EC2+BIツール

- 多彩なパートナーソリューション・OSSをEC2上で活用



要件に合わせたツールでデータレイクを可視化



データ処理のパイプライン

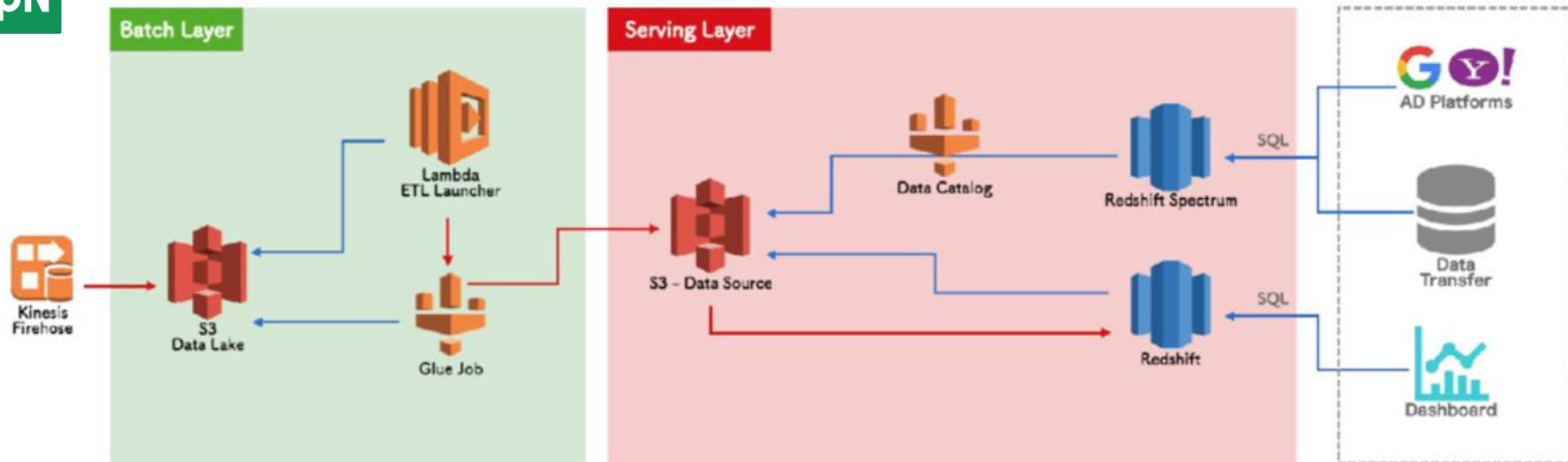


アジェンダ

1. データ活用の流れ
2. 従来の RDB を中心としたデータ分析
3. データレイクとは
4. データレイクを構成する AWS サービス
5. AWS におけるデータ活用事例
6. まとめ

事例：primeNumber 様による DMP (Data Management Platform)

pN



https://speakerdeck.com/hiro_koba_jp/aws-etlji-ri-aws-gluehuo-yong-shi-li-at-primenumber

アジェンダ

1. データ活用の流れ
2. 従来の RDB を中心としたデータ分析
3. データレイクとは
4. データレイクを構成する AWS サービス
5. AWS におけるデータ活用事例
6. まとめ

まとめ

- データは加工せずデータレイクに全て残す
- 全てを叶える万能なツールは存在しない
- AWS の様々なサービスを活用し、ストレージと計算処理の分離することで、データ活用に必要なツールやリソースの柔軟な確保を行い、高速な試行錯誤のサイクルを回すことができる

Thank you !