[AWS Black Belt Online Seminar]
AWS IoT Analytics Deep Dive

サービスカットシリーズ

Prototyping Solutions Architect
渡邉 聡
2019/10/30

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

自己紹介

渡邉 聡　(わたなべ さとし)

- プロトタイピング・ソリューションアーキテクト
- IoTの導入やPoC段階のご支援を主に担当

好きなAWSサービス

- AWS IoT サービス全般
  - AWS IoT Analytics
  - AWS IoT Core
  - AWS IoT Greengrass
- AWS Lambda

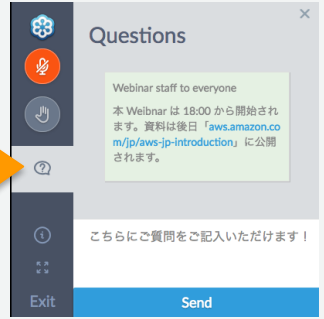
AWS Black Belt Online Seminar とは

「サービス別」「ソリューション別」「業種別」のそれぞれのテーマに分かれて、アマゾン ウェブ サービス ジャパン株式会社が主催するオンラインセミナーシリーズです。

質問を投げることができます！

- 書き込んだ質問は、主催者にしか見えません
- 今後のロードマップに関するご質問は
  お答えできませんのでご了承下さい

① 吹き出しをクリック
② 質問を入力
③ Sendをクリック



Twitter 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2019年10月30日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。最新の情報はAWS公式ウェブサイト(http://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が東京リージョンを使用する場合、別途消費税をご請求させていただきます。

- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at http://aws.amazon.com/agreement/. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

本日のアジェンダ

- IoTデータの特徴とその活用例
- AWS IoT Analyticsの詳細
- パイプライン構成パターンについて
- まとめ

本日のアジェンダ

- **IoTデータの特徴とその活用例**
- AWS IoT Analyticsの詳細
- パイプライン構成パターンについて
- まとめ

技術の進化と IoT サービスの進化

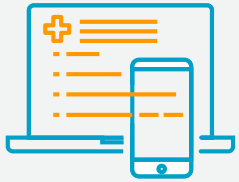
モニタリング	制御	最適化	自律化
• 製品の状態 • 製品の稼働・利用状況	• 製品機能の制御 • パーソナライズ	• 製品機能、性能の向上 • 予防診断、修理 • 新サービス	• 製品の自動運転 • 他製品やシステムとの自動連携 • 自己診断と修理・修復 • 製品の自動改良とパーソナライズ
センサー、CPU、メモリーなどの小型化・低コスト化	クラウド・ネットワークの進化	ビッグデータ・分析技術の進化	AI関連技術（機械学習・ディープラーニング等）の進化

aws

IoT データが生み出す価値

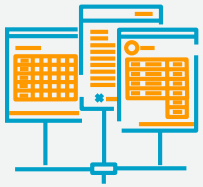
予知保全

ウェルネスと
健康ソリューション

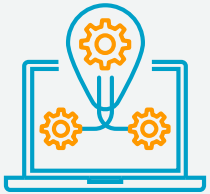


生産効率および
プロセスの最適化

スマートビルディング
スマートシティ



デバイスフリートの
管理



エネルギー効率の
監視

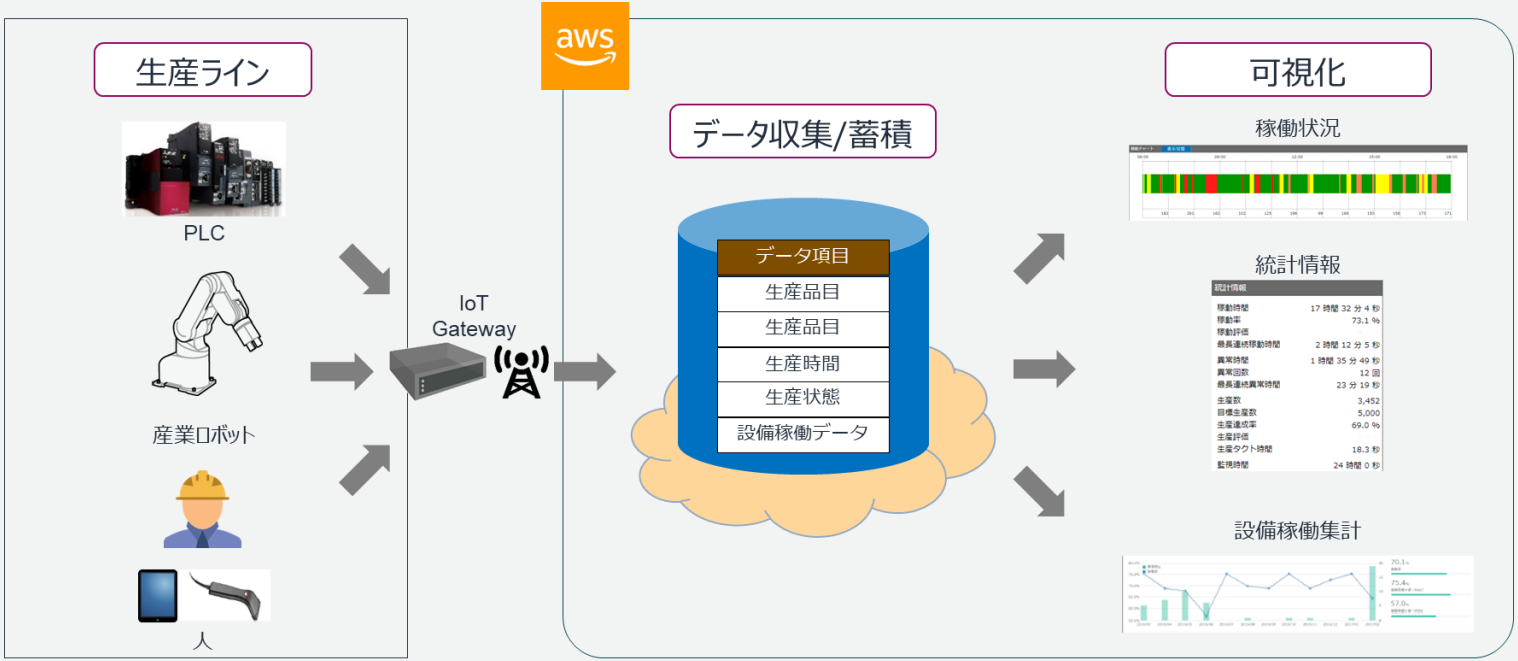
決済、保険、
Eコマース

製造設備の
安全管理

aws

活用例1: 生産状況の見える化

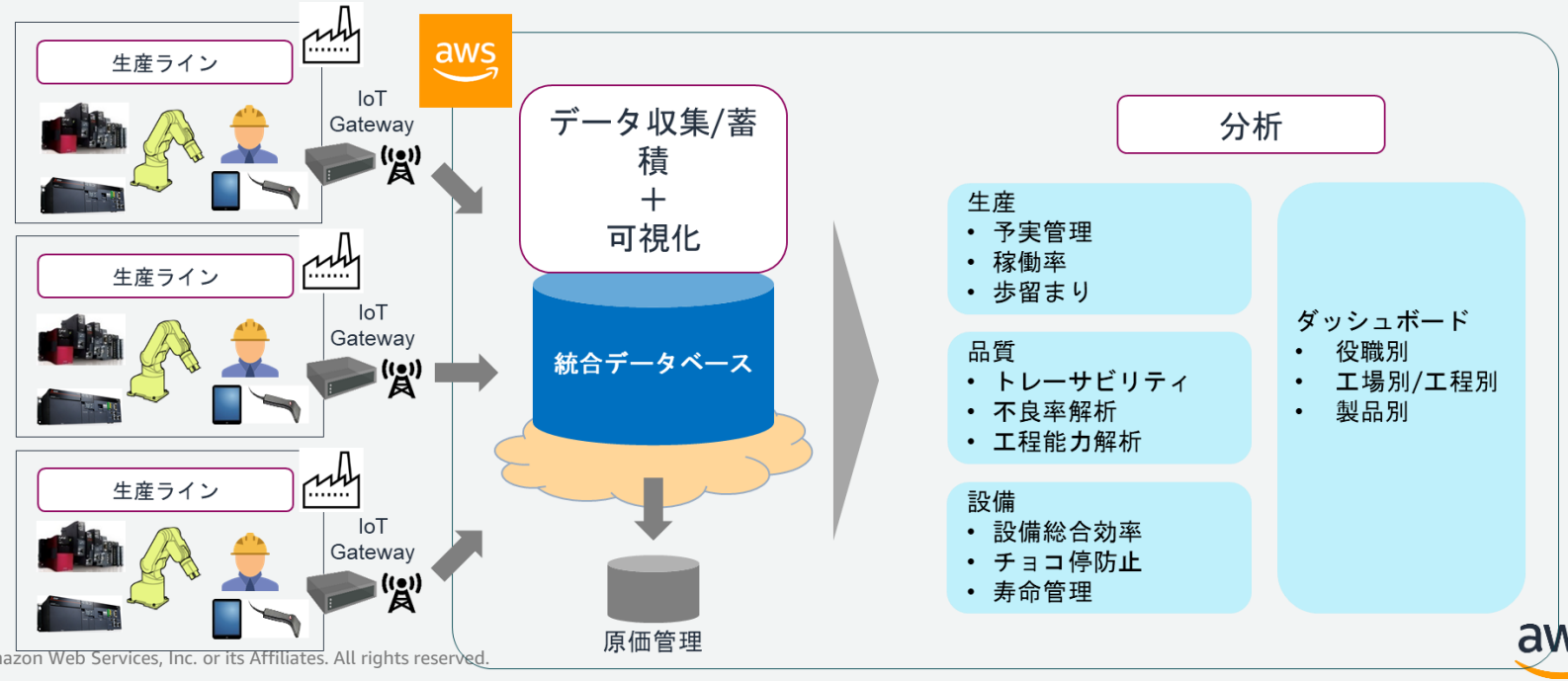
課題：簡単に早く設備の稼働状況を把握したい
方法：IoT Gatewayを経由してクラウド上にデータを収集してデータベースを構築し、設備の稼働状況の見える化を行う



活用例2: 分析による業務改善

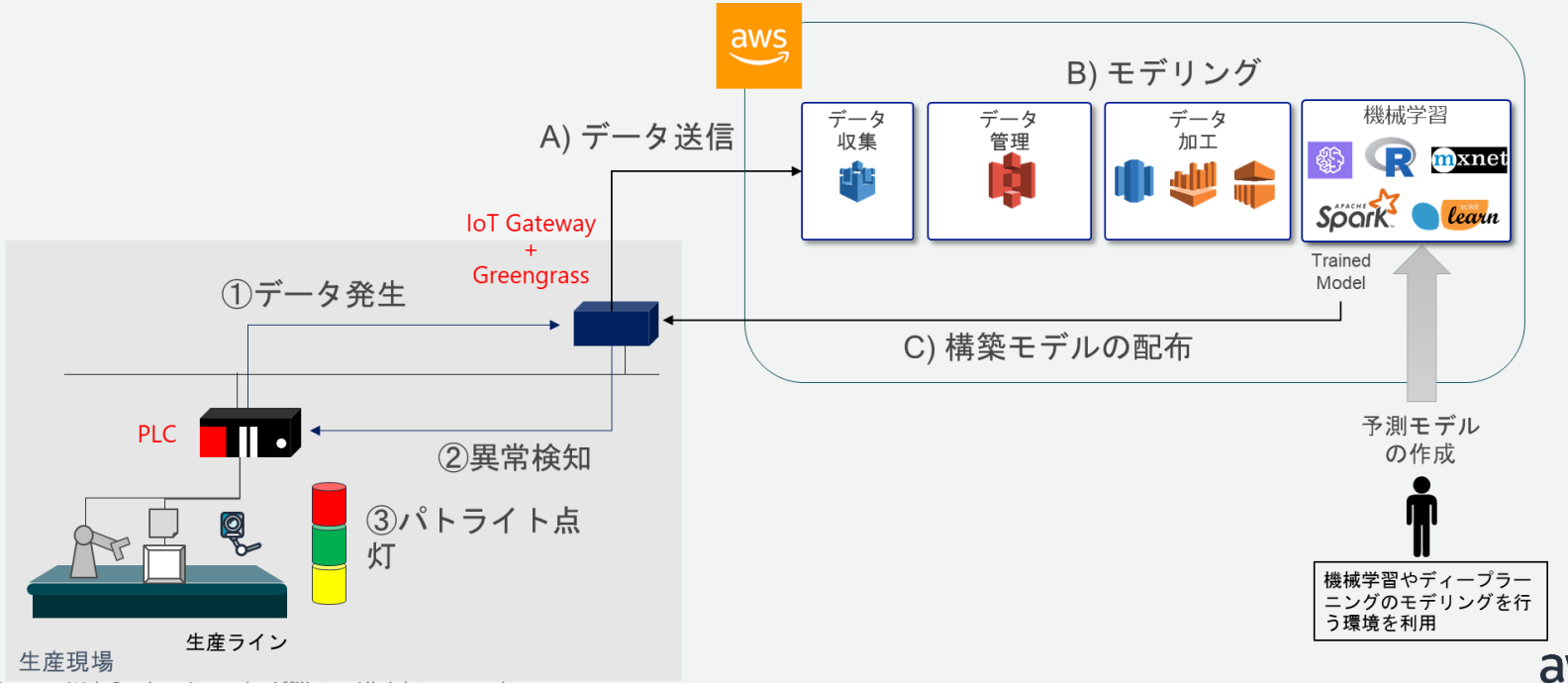
課題：設備稼働データを使って業務改善をしたいが、データが統合されておらず、分析のための集計作業に時間がかかる。

方法：各工場からデータを収集し、クラウド上に統合データベースを構築。DWHやBIツールを使ってデータ集計を自動化し、分析までの時間を短縮。複数の拠点のデータが集まっているので、ダッシュボードで役割別や工場別、製品別など、必要な情報を利用する事ができる。

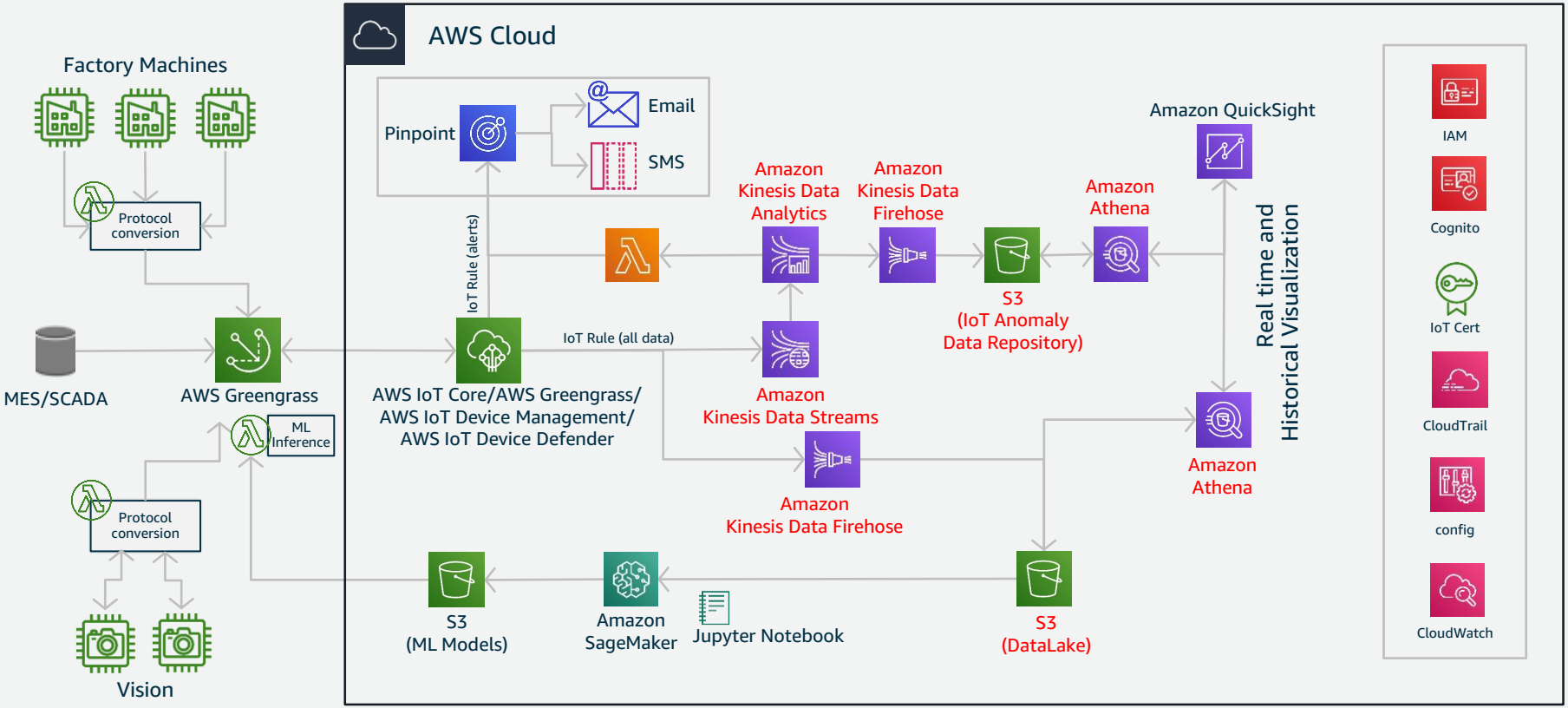


活用例3: 予知保全 / 品質予測

課題：稼働率向上のために予知保全を行いたい/品質予測により外部不良率を低減したい
方法：クラウド上のデータを使って機械学習により異常検知のモデルを構築し、
IoT Gateway で異常検知を行う。



予知保全を実現するアーキテクチャ例



IoTデータの特徴

- **ノイズを多く含むセンサーデータ**

  大きな誤差やメッセージの破損、誤認識が含まれる場合があるので事前にクリーンナップが必要となる

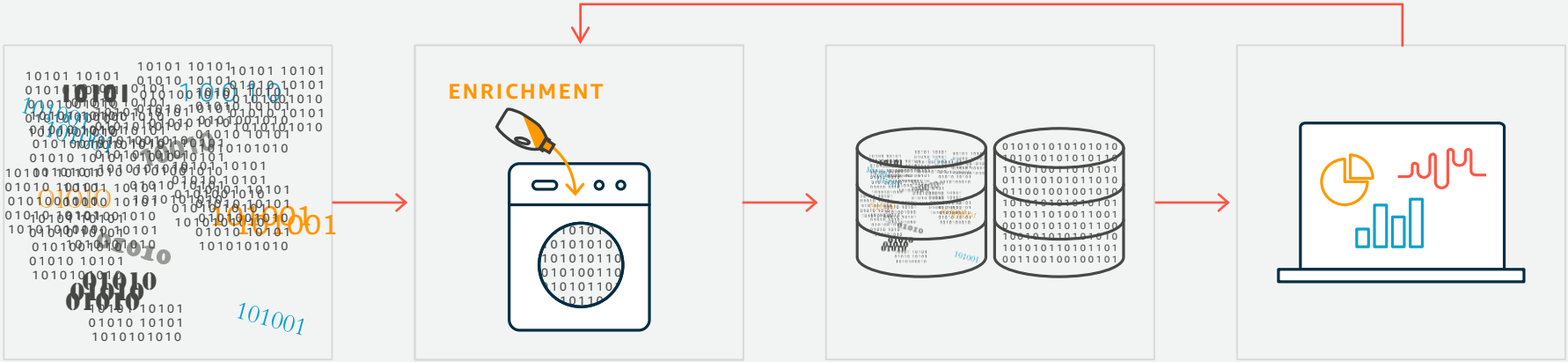
- **非構造データ**

  従来の分析BIツールは構造化されたデータを必要とする

- **データ拡張を必要とする**

  外部ソースのデータと合わせることで初めて意味が得られる場合も少なくない

IoTデータ処理の流れ



1. IoTデータはノイズが多く、正しい値とのギャップなどもあり、誤った測定値なども含まれる

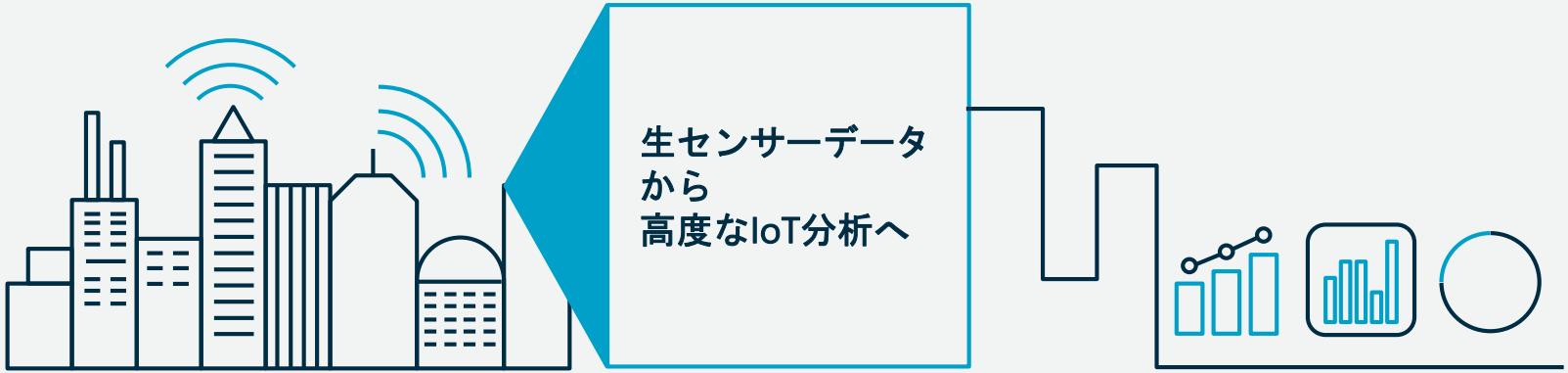
2. データのフィルタリング、処理、変換、および拡充

3. 生データと処理済みデータを保存する

4. アドホッククエリの実行や高度な分析、可視化

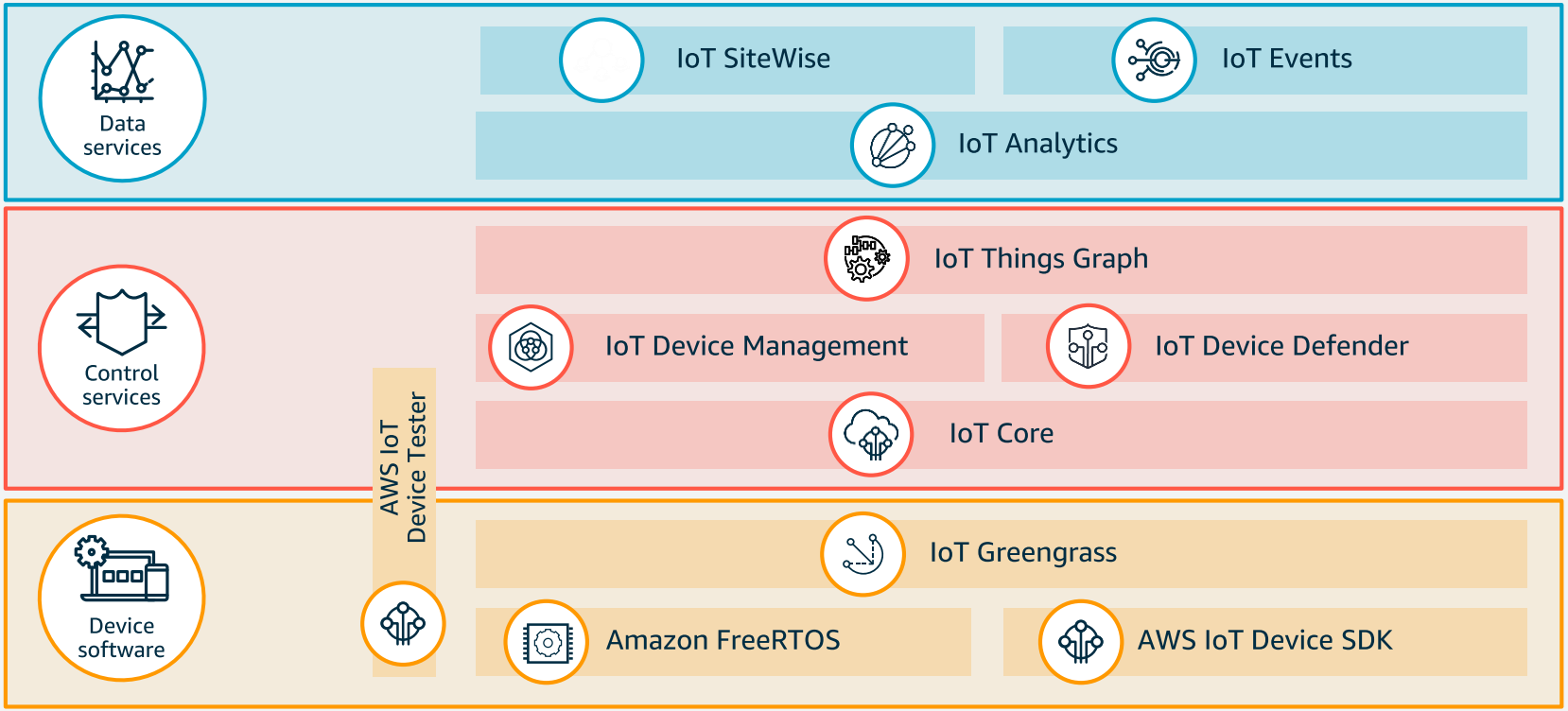
AWS IoT Analytics

AWS IoT Analyticsは、IoTデバイスデータを大規模に
収集、前処理、拡充、保存、分析、および視覚化する、完全マネージドなサービスです

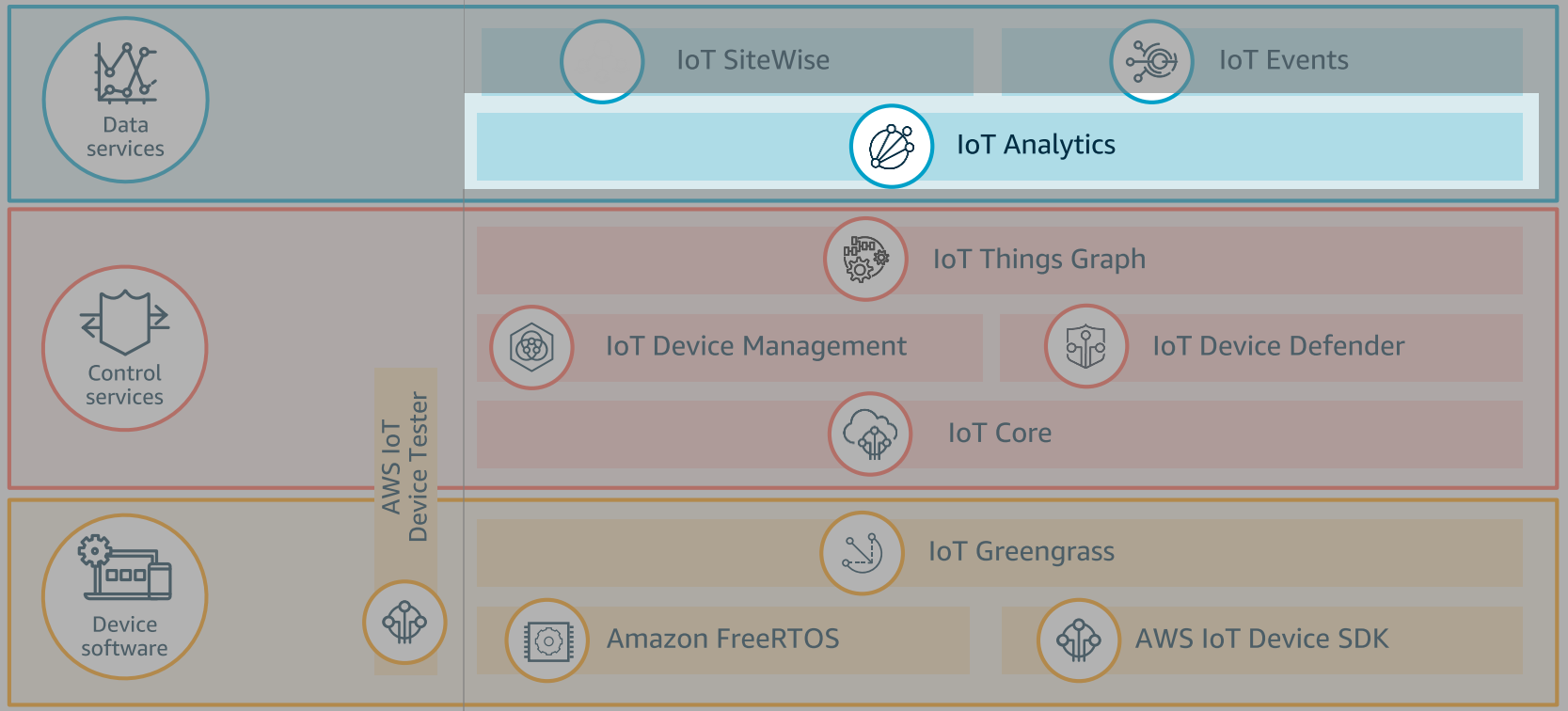


Data
services

AWS IoT アーキテクチャ



AWS IoT アーキテクチャ



本日のアジェンダ

- IoTデータの特徴とその活用例
- **AWS IoT Analyticsの詳細**
- パイプライン構成パターンについて
- まとめ

AWS IoT Analytics

AWS IoT Analyticsは、製造業やエンタープライズのために
IoTデータの処理、保存、分析、可視化を可能にするサービス

収集

保存および分析
したいデータ
のみを収集

処理

生データを
意味のあるデータに
変換

保存

分析のために
デバイスデータを
時系列データ
ストアに保存

解析

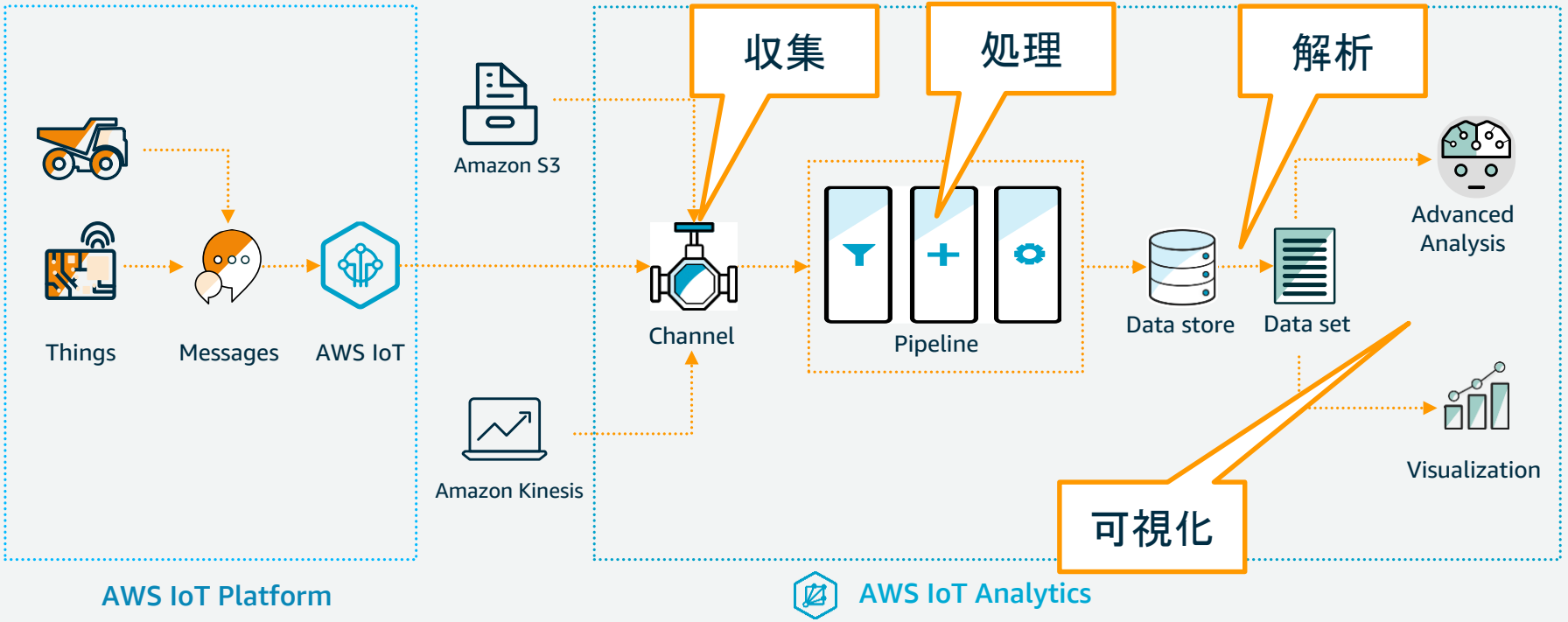
資産の健全性と
パフォーマンスに
対して洞察を得る

可視化

IoTデータを
すばやく可視化

Data
services

AWS IoT Analytics Overview

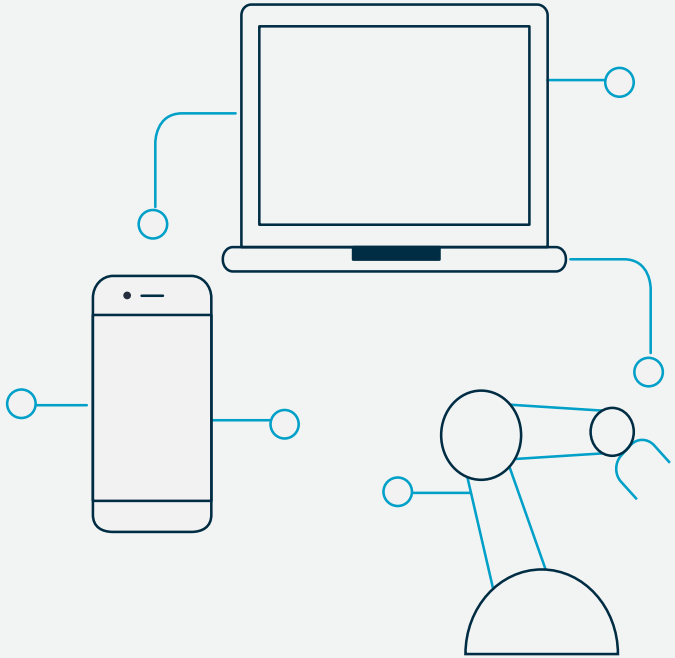


収集

保存・分析したいデータのみ収集

AWS IoT Core、Amazon S3、
Amazon Kinesis、またはその他の
ソースからのデータをAWS IoT Analyticsに
取り込むことが可能

様々な形式のデータを様々な頻度で受信。
AWS IoT Coreのルールアクションを介して
デバイスからメッセージを受信したり、
HTTPエンドポイントに対してデータをPUT
することでデータを収集



BatchPutMessage

IoT Core経由でメッセージを流す以外にBatchPutMessage APIを使ってHTTP API経由でデータを入力することが可能(S3, Kinesis経由の連携が実現可能)

Pandasを使ったCSV取り込み例

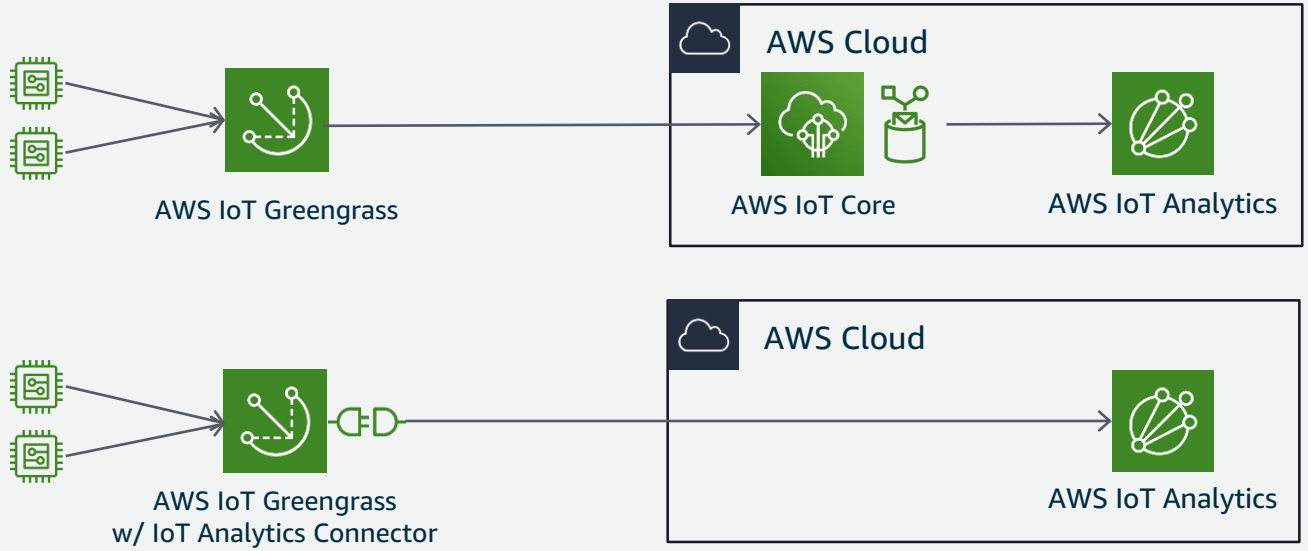
```
iota = boto3.client('iotanalytics')
reader = pd.read_csv('data.csv', chunksize=100)

for r in reader:
    messages = []
    for index, row in r.iterrows():
        messages.append({'messageId':str(uuid.uuid4()),'payload': row.to_json()})
    ret = iota.batch_put_message(channelName='sample_channel', messages=messages)
```

AWS IoT Greengrass IoT Analytics Connector

Greengrass側で発生したローカルIoTデータをIoT Analyticsに送るコネクタ

IoT Coreを経由せず、BatchPutMessage APIを使ったバッチ送信で効率よくデータ転送が行われる



取り込みデータ形式について

- JSON形式のメッセージだけでなく、バイナリ(Base64 エンコード済みメッセージ)なども取り込み可能
- ただし、JSON形式メッセージ以外の場合、Pipelineの最初でJSON形式への変換が必要
    - 変換にはPipelineでのLambda実行(Lambda Activity)を使用
- JSONメッセージの場合もフィールド名に制約があるので注意が必要
    - 英数とアンダースコアで構成され255文字を超えることができない
    - 英字もしくは単一のアンダースコアで始まる必要がある
    - 例：
      有効　{“temp_01”：29} または {“_temp_01”：29}
      無効　{“temp-01”：29}、{"01_temp"：29}または{"__temp_01"：29}

処理

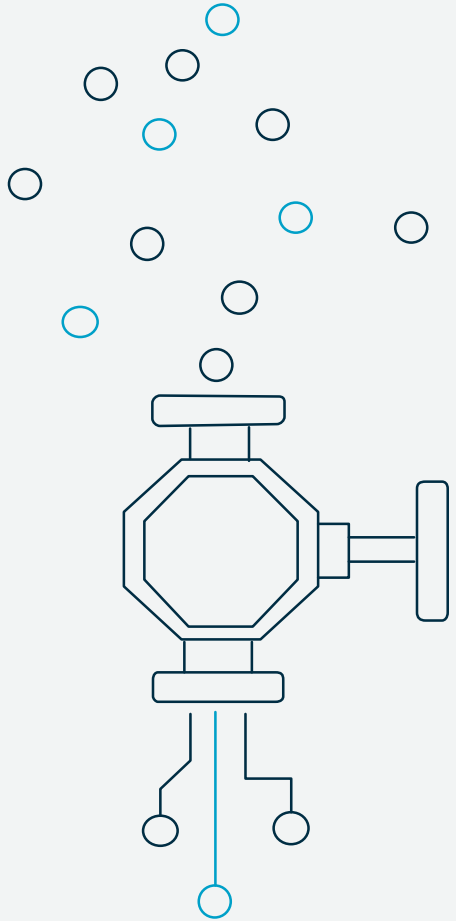
生データを意味のある情報に変換

欠落データが検出されたときにトリガーできるAWS Lambda関数を定義

定義した算術的または条件付き論理を使用してメッセージを変換

天気予報情報などの外部データソースを使用してデータを拡充

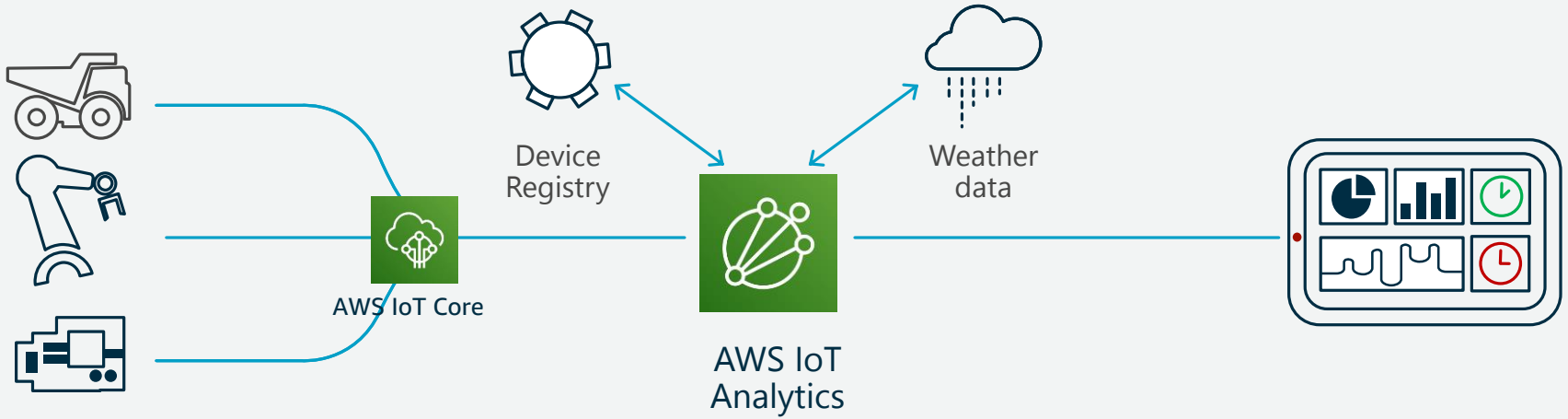
生データを再処理して新しいパイプラインを作成し、新たな変更を加えたり、または異なる方法でデータを処理して新規に保存



pipelineでのメッセージの変換・強化・フィルタ

- メッセージ属性の操作(削除、選択、追加)
- メッセージ属性の計算
- IoT Coreレジストリ情報でメッセージを強化
- IoT Coreシャドウ情報でメッセージを強化
- 条件付きでメッセージをフィルタ
- Lambda関数を使ったメッセージの変換

データの拡充による強化



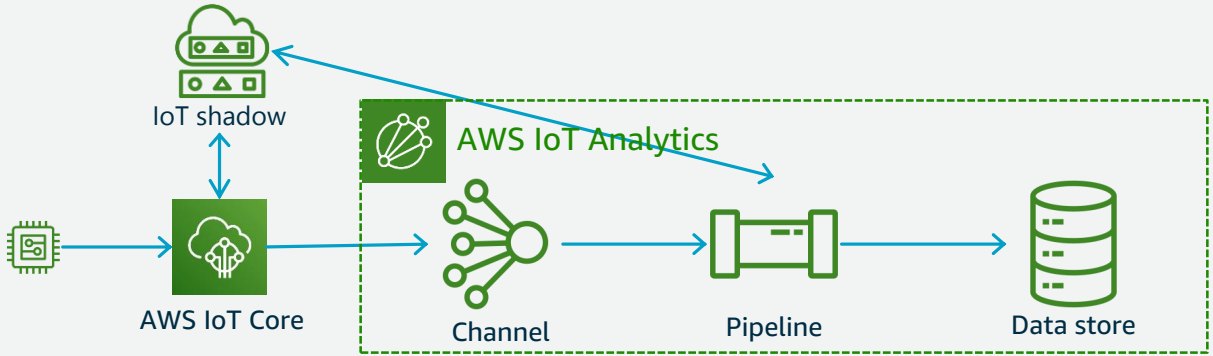
例：かんがいシステムの湿度センサーの配置された畑の位置を**デバイスレジストリから拡張**取得し、**その地点の降雨データ**でさらにデータを強化することで、水の使用を効率化する一方で、収穫を最大化する

IoT Coreレジストリ・シャドウ情報でメッセージを強化

- メッセージpayload中に含まれるThing名でマッピングしてメッセージを強化

  - **レジストリ情報で強化 (deviceRegistryEnrich)**
    - defaultClientId, thingName, thingId, ThingArn, ThingTypeName, attributes, version, billingGroupName をメッセージに追加

  - **シャドウで強化 (deviceShadowEnrich)**
    - desired, reported, delta, metadataをメッセージに追加



Lambda関数を使ったメッセージの変換・強化・フィルタ

- まとまったメッセージを受け取り属性操作、変換、強化・フィルタを行う
  - 例：
    - メッセージフォーマットをバイナリからJSONに変換する
    - 現時点の天気情報を追加する

- 1-1000でメッセージ上限を設定
  - 処理がタイムアウト(現在は最大15min)にならないようメッセージ上限を設定、検証が必要

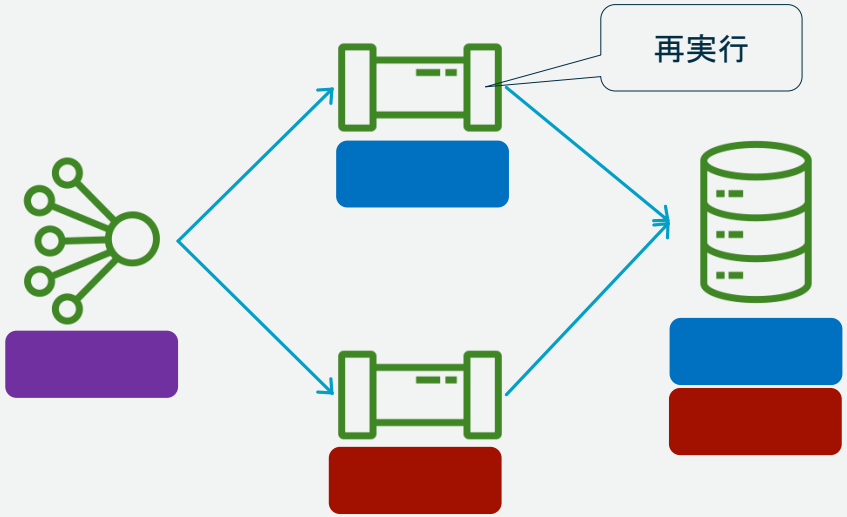
- エラーとなった場合はプログラムを修正してPipelineの再処理

AWS IoT Analytics から Lambda 関数を呼び出せるように、関数ポリシーを追加する必要あり

```
> aws lambda add-permission --function-name <lambda-function-name> --statement-id <your-statement> --principal iotanalytics.amazonaws.com --action lambda:InvokeFunction
```

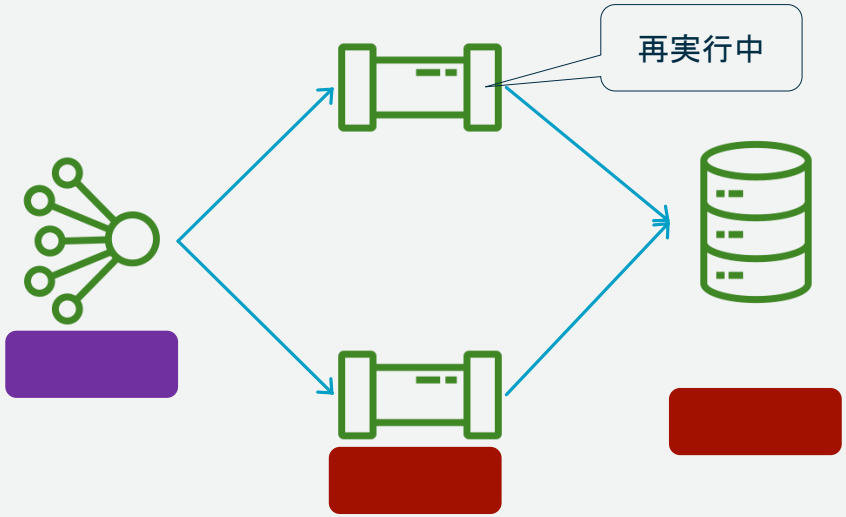
pipelineを指定してchannelデータ再処理

- pipelineを変更した際に、channelにあるrawデータからdatastoreを再処理できる
- rawデータの保存期間から再処理する任意の期間を決め実行



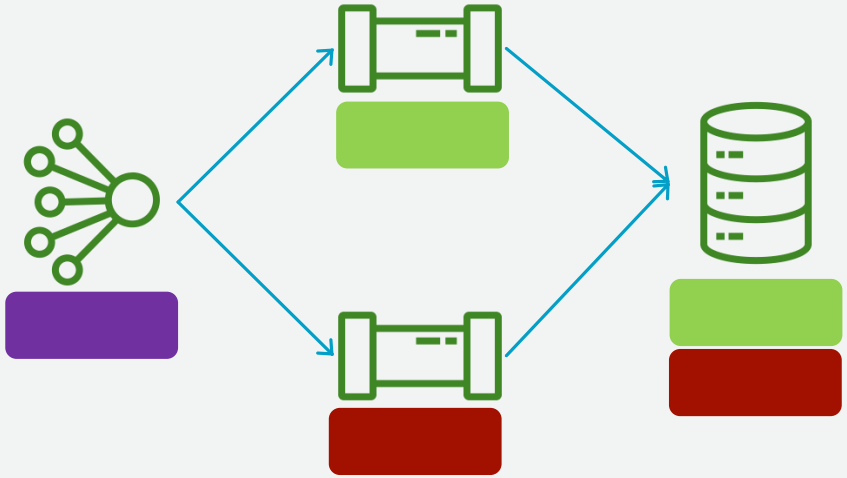
pipelineを指定してchannelデータ再処理

- pipelineを変更した際に、channelにあるrawデータからdatastoreを再処理できる
- rawデータの保存期間から再処理する任意の期間を決め実行



pipelineを指定してchannelデータ再処理

- pipelineを変更した際に、channelにあるrawデータからdatastoreを再処理できる
- rawデータの保存期間から再処理する任意の期間を決め実行

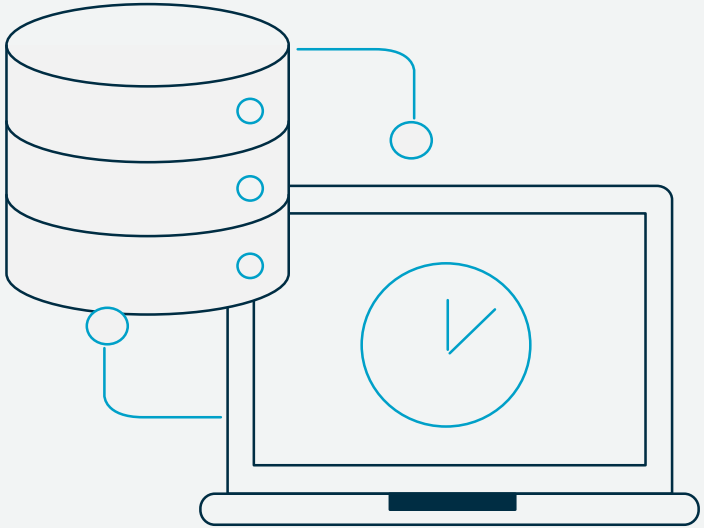


保存

分析のためにデバイスデータを時系列データストアに保存

データ保持ポリシーから、指定した期間でのデータ保存を指定
データを外部にエクスポート可能

処理済データの保存と同時に生データを自動的に保存しているのであとで再処理することも可能



ストレージタイプ

- **サービスマネージド型ストア**
  - IoT Analyticsのすべての機能を利用でき、設定した保存期間で自動でデータ運用
  - **制限**
    - 保存されたデータを直接参照することができない
- **顧客マネージド型S3バケット**
  - 任意のbucket、prefixの指定が可能
  - Channelで受信したRawデータはBase64エンコードされ、複数メッセージでまとめられgz圧縮された形で保存
  - DataStoreデータはJSON Lines形式でgz圧縮された形で保存
  - **制限**
    - IoT Analyticsと同じRegionである必要がある
    - Channel、DataStoreのサイズはS3コンソールで確認
    - データ保持期間はS3側で個別に設定が必要

解析

資産の健全性とパフォーマンスに対する深い洞察を得る

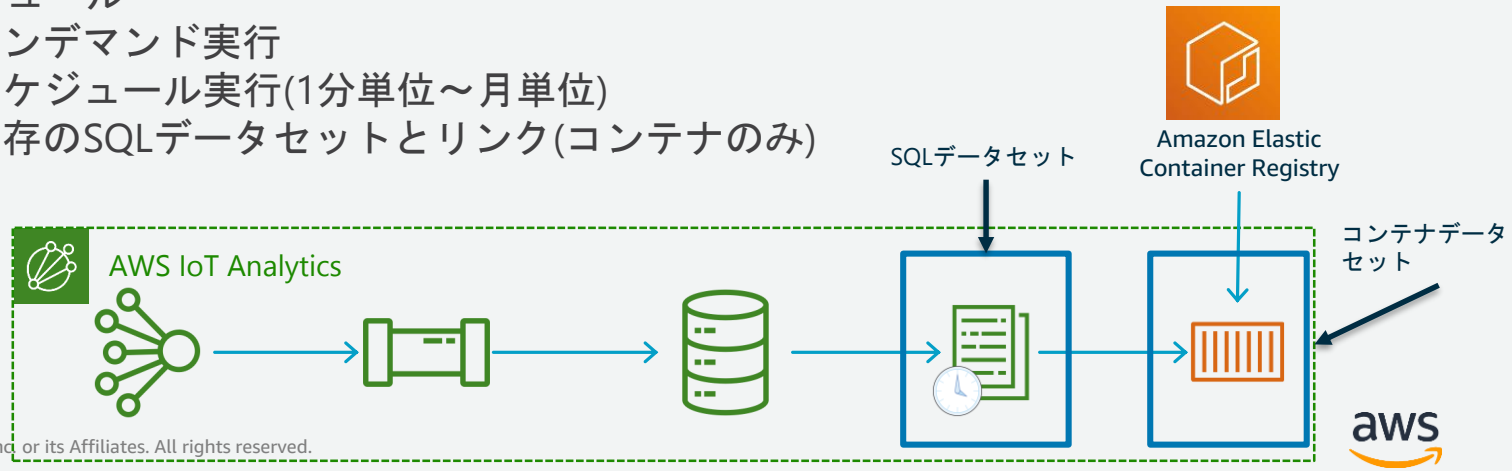
カスタマイズ可能なウィンドウで分析をスケジュール

継続的な洞察のために分析ワークフローを自動化

データに対してオンデマンド分析を実行可能

SQLデータセットとコンテナデータセット

- SQLデータセット
  - データストアのデータに対してSQL式を使用してCSVを出力
- コンテナデータセット
  - SQLクエリ結果を入力値とし任意のコンテナで処理を実行し結果出力
  - 出力ファイル形式は任意
- 実行スケジュール
  - オンデマンド実行
  - スケジュール実行(1分単位～月単位)
  - 既存のSQLデータセットとリンク(コンテナのみ)



SQLデータセット

- Amazon Athenaと同じSQLクエリ、関数、および演算子を使ってデータセット生成クエリを定義
- ただしJOIN句やWITH句はサポートされていない
    - 他のデータストアを結合するクエリや、サブクエリを外出しすることができない
- “__dt”でパーティションが日次(UTC)で切られているのでクエリに含むことでクエリパフォーマンスの維持とスキャンコストの削減を図る

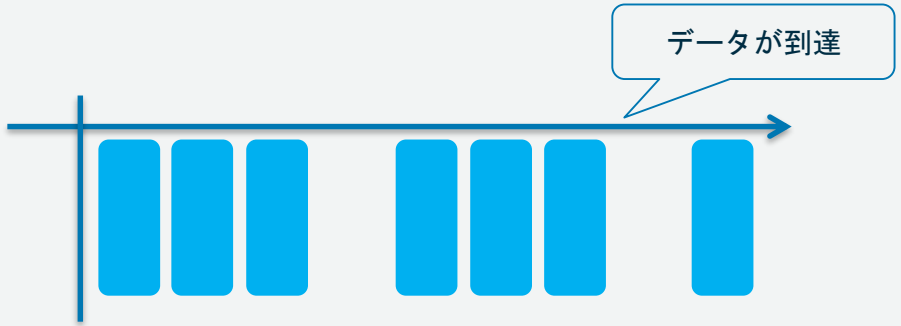
本日分のパーティションのみを指定してクエリする例

```
SELECT * FROM datastore WHERE __dt > current_date - interval '1' day
```


aws

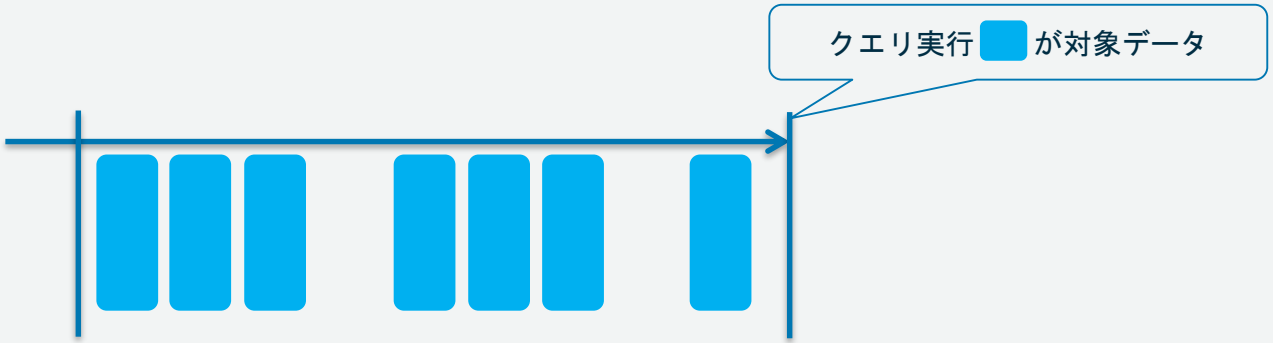
DeltaTime Window

- ウィンドウ単位で時系列データを処理し、データを漏れ・重複なく処理できる
- 定期的に到達した最新データのカウントや平均値を取りたい場合などに有効
- ウィンドウから漏れ、遅れて到達したデータをどこまで取り込むかデルタウィンドウのオフセットとして設定可能



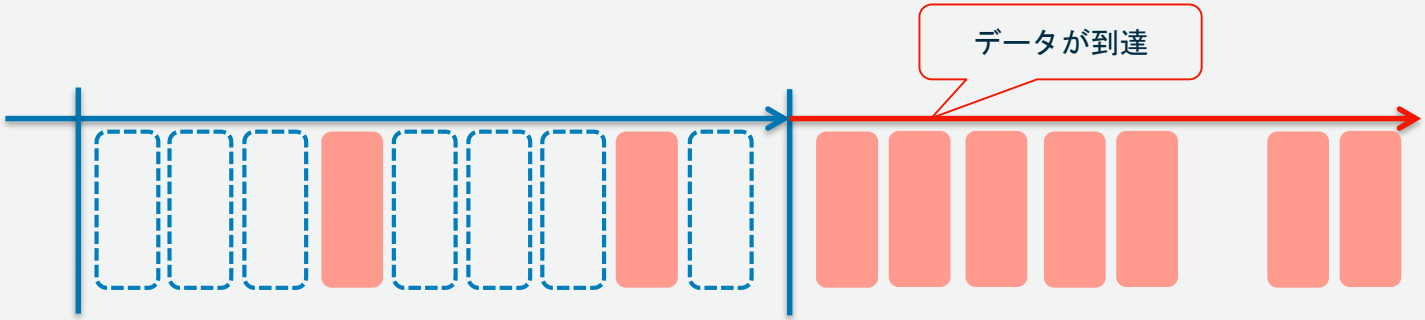
DeltaTime Window

- ウィンドウ単位で時系列データを処理し、データを漏れ・重複なく処理できる
- 定期的に到達した最新データのカウントや平均値を取りたい場合などに有効
- ウィンドウから漏れ、遅れて到達したデータをどこまで取り込むかデルタウィンドウのオフセットとして設定可能



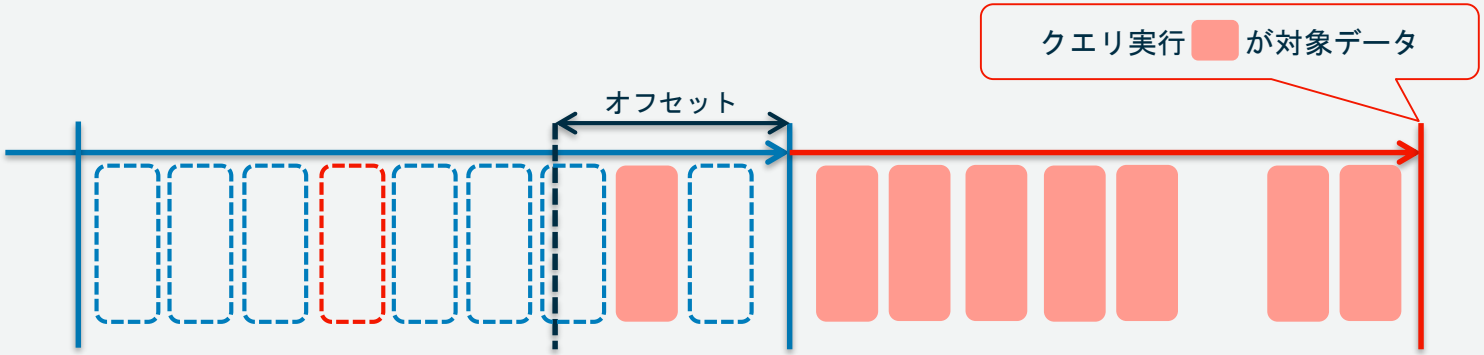
DeltaTime Window

- ウィンドウ単位で時系列データを処理し、データを漏れ・重複なく処理できる
- 定期的に到達した最新データのカウントや平均値を取りたい場合などに有効
- ウィンドウから漏れ、遅れて到達したデータをどこまで取り込むかデルタウィンドウのオフセットとして設定可能



DeltaTime Window

- ウィンドウ単位で時系列データを処理し、データを漏れ・重複なく処理できる
- 定期的に到達した最新データのカウントや平均値を取りたい場合などに有効
- ウィンドウから漏れ、遅れて到達したデータをどこまで取り込むかデルタウィンドウのオフセットとして設定可能



コンテナデータセット

- カスタムで実装されたコンテナやMatlab, Octave, R, Pythonなどサードパーティーツールで構築されたコンテナ、IoT Analytics提供のJupyter Notebooksで構築されたコンテナを用いて分析が可能

- パラメータタイプ
    - String: 任意の文字列、対象データセット名など
    - Double: 任意の数値
    - Output file: 出力ファイル名(プログラム側にはS3パスとして渡る)
    - Content version
        - 対象データセットのlatestバージョン(データセット名を指定)

- コンテナ内でのパラメータの受け取り方(Paramファイル)
    - '/opt/ml/input/data/iotanalytics/params' (JSON)の 'Variables' にパラメータが書かれる

対象データソースとするData SetやData Setのバージョンは Paramファイル経由で受け取る必要があるので、コンテナ内のプログラムに個別に実装する必要がある

コンテナプログラムサンプル (Python)

パラメータの読み込み例

```
import json

with open('/opt/ml/input/data/iotanalytics/params') as param_file:
    params = json.loads(param_file.read())

example_var = params['Variables'] ['example_var']
print(example_var)
```

出力の書き出し例

```
import boto3
import json
from urllib.parse import urlparse

ACCESS_CONTROL_LIST = 'bucket-owner-full-control'
data = 'output data'

with open('/opt/ml/input/data/iotanalytics/params') as param_file:
    params = json.loads(param_file.read())

custom_output = params['Variables'] ['custom_output']

bucket = urlparse(custom_output).netloc
key = urlparse(custom_output).path.lstrip('/')

s3_client = boto3.resource('s3')
s3_client .put_object(Bucket=bucket, Key=key, Body=data,  ACL= ACCESS_CONTROL_LIST)
```

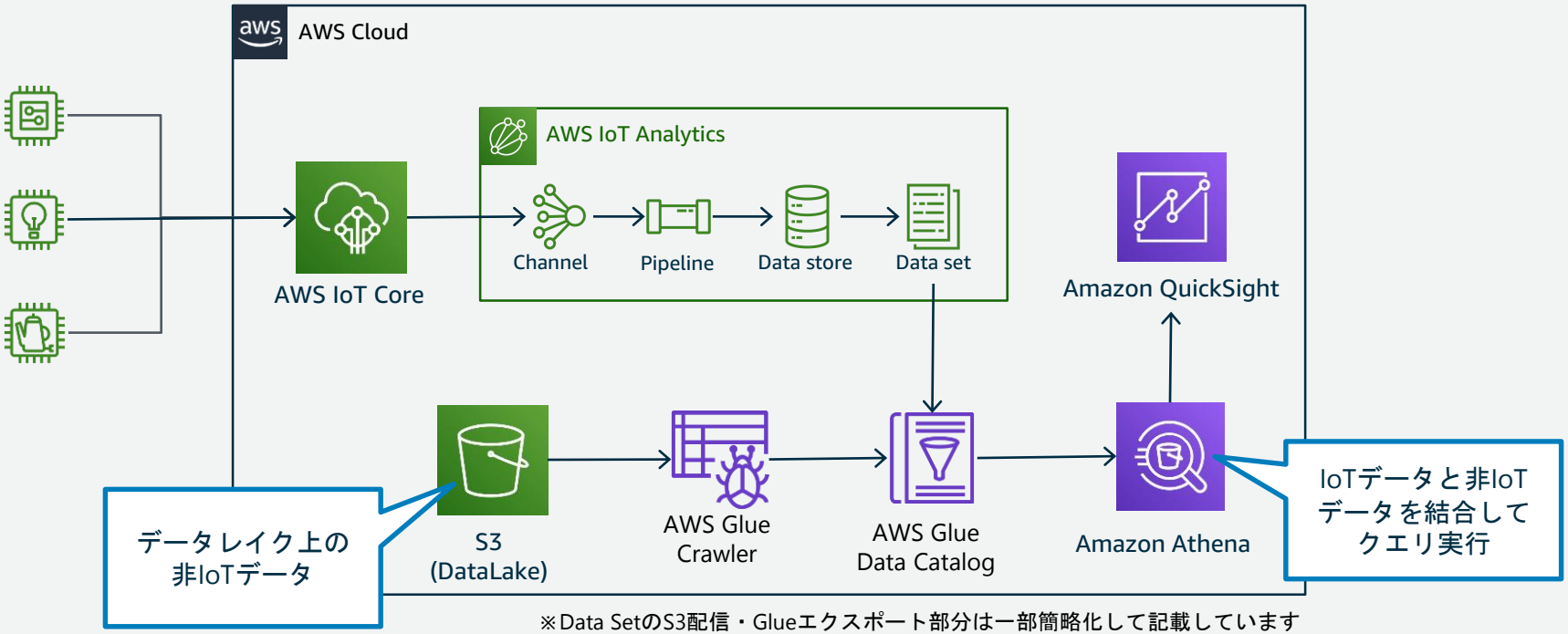
https://docs.aws.amazon.com/ja_jp/iotanalytics/latest/userguide/automate.html

配信ルール (S3アクション、IoT Eventsアクション)

- S3アクション
  - データセットの出力結果を任意のS3 Bucketに出力するよう設定可能
  - 同時にメタデータ(テーブル定義、ロケーション)のGlueエクスポートも設定可能
  - キー中に代入変数が使用可能
    - !{iotanalytics:scheduleTime} → クエリが実行された時間(Unix Time)
    - !{iotanalytics:versionId} → DatasetのコンテンツID (UUID)

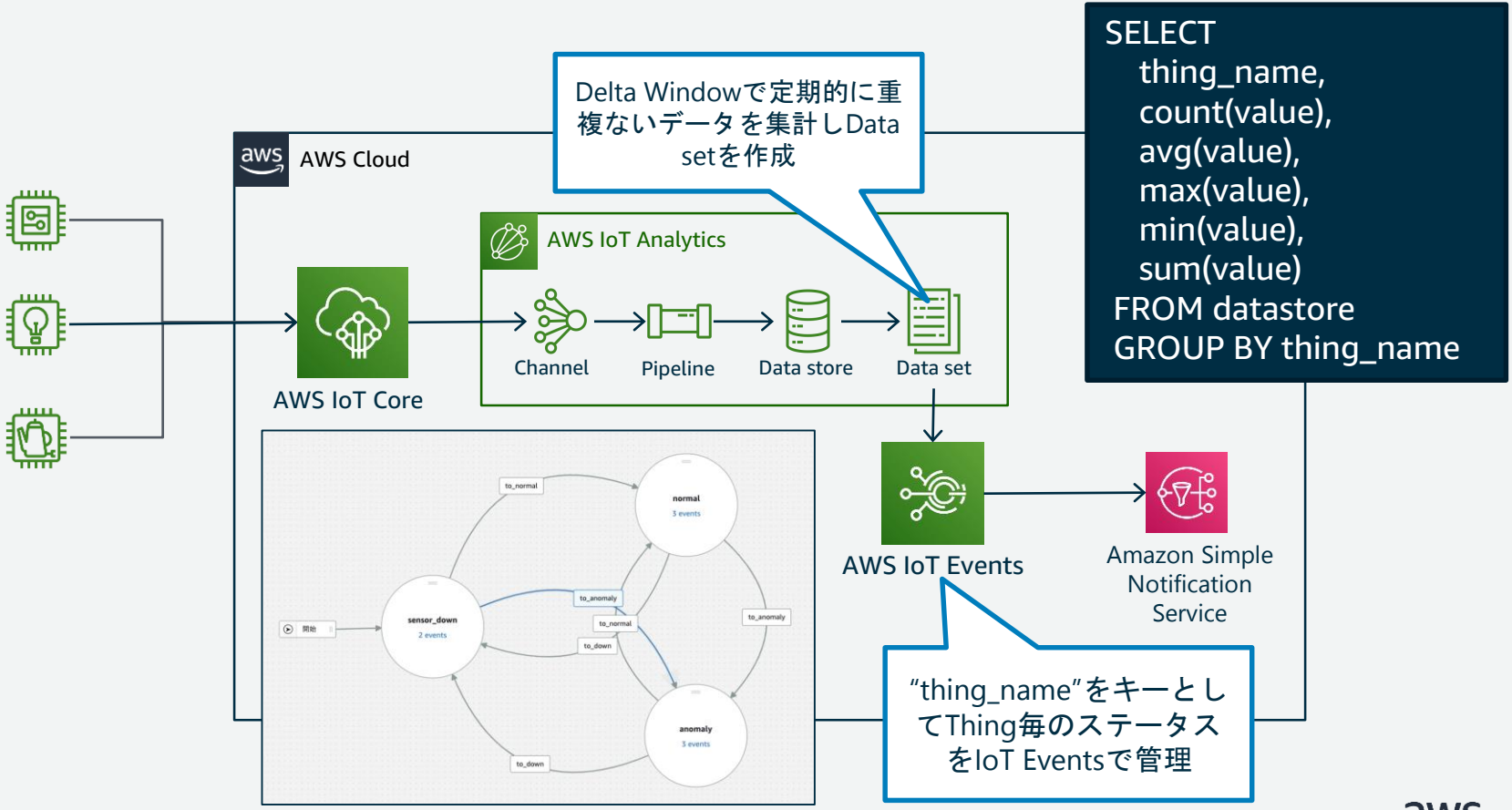
- IoT Eventsアクション
  - データセットの出力結果をIoT Eventsの入力として設定
  - 集計されたデータを評価し特定の状態になったらアラートを発生させる構成などが検討可能

IoTデータをデータレイク上の非IoTデータと統合する



※Data SetのS3配信・Glueエクスポート部分は一部簡略化して記載しています

IoT Eventsアクションを使った警報連携例



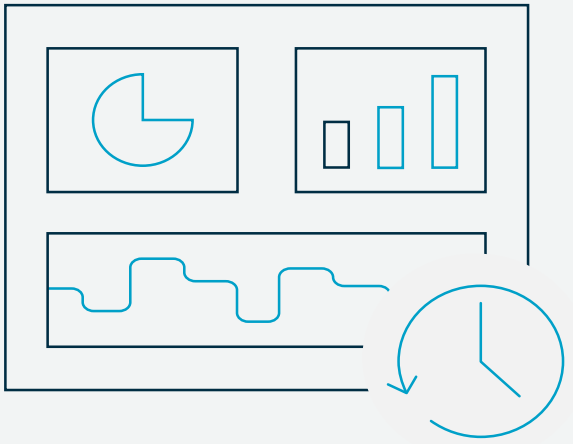
可視化

データセットをすばやく可視化

BIツール(QuickSight)と連携しデータを素早く可視化・分析

AWS IoT Analytics コンソール内で分析内容を可視化

分析ワークフローがトリガーされるたびに視覚化情報を自動的に更新

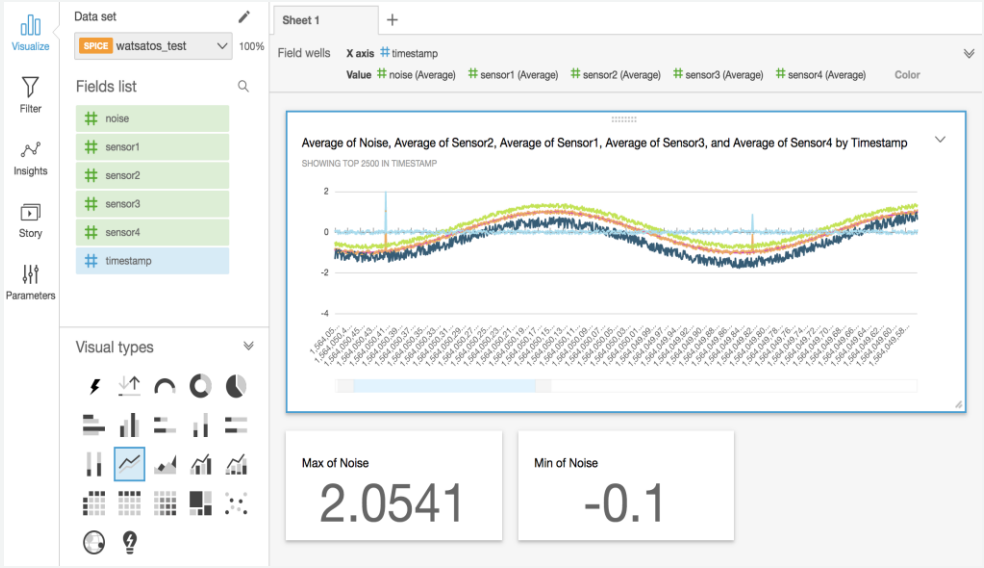


QuickSight

IoT AnalyticsのdatasetをSPICEに取り込んで分析を実行

定期的にデータを更新する場合は、datasetの作成スケジュールと合わせてSPICE取り込みのスケジュールを調整

QuickSight ML Insights(※)を利用することで時系列データからのForecastingや異常値検知などの機能も簡単に利用可能



※ ML Insightsは、Amazon QuickSight の エンタープライズエディションでのみ利用可能

SageMaker Jupyter Notebook Templates

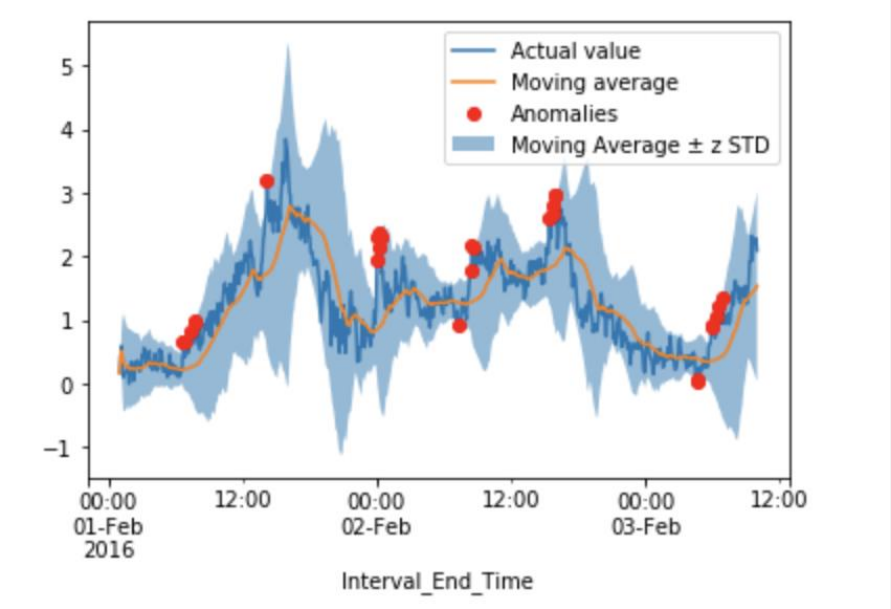
IoTデータ分析用テンプレートサンプルとして予めいくつかのシナリオに合わせたサンプルノートブックを準備

ノートブック内のいくつかのパラメータを変更するだけで自身のデータも適用可能

コンテキスト的な異常の検出 - Detecting Contextual Anomalies

Interval_End_Time	Wind_Velocity_Mtr_Sec
2016-02-01 01:00:00	0.17400
2016-02-01 01:05:00	0.44350
2016-02-01 01:10:00	0.59775
2016-02-01 01:15:00	0.24100
2016-02-01 01:20:00	0.09925

使用データ：
タイムスタンプと風速からなる時系列データ



時系列データ用に Poisson exponentially weighted moving average(PEWMA)モデルで測定された風速のコンテキスト異常を検出するサンプル

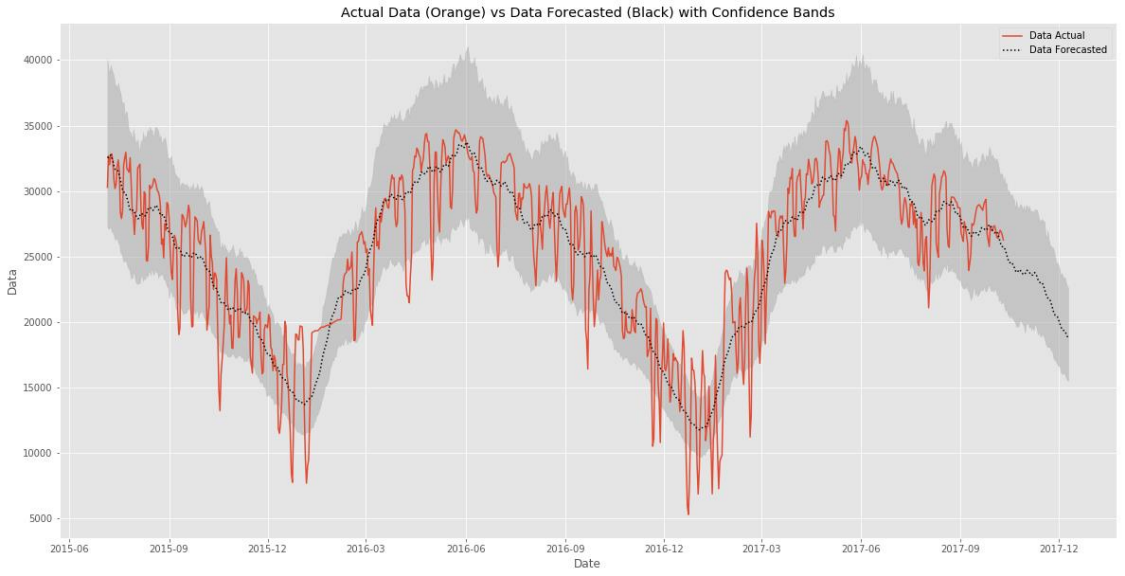
ソーラーパネルの出力予測 - Solar Panel Output Forecasting

Date	
2015-07-06	30286
2015-07-07	32675
2015-07-08	32014
2015-07-09	32715
2015-07-10	32836

使用データ：
タイムスタンプとソーラパネルの発電量からなる時系列データ

シンプルな単一の時系列データからオープンソース予測ライブラリであるProphetを使用して、ソーラーパネルの未来の電力出力を予測するサンプル

時系列データからのForecastingを行いたい場合は容易に流用もできる



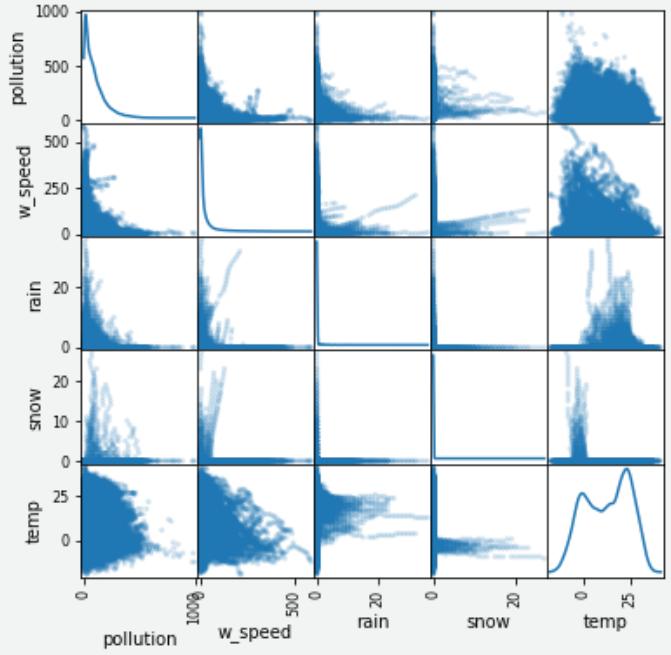
スマートシティの大気汚染予測 - Smart City Air Quality Forecasting

year_month_day_hour	pollution	dew	temp	pressure	w_dir	w_speed	snow	rain
2010-01-02 00:00:00	129.0	-16	-4.0	1020.0	SE	1.79	0	0
2010-01-02 01:00:00	148.0	-15	-4.0	1020.0	SE	2.68	0	0
2010-01-02 02:00:00	159.0	-11	-5.0	1021.0	SE	3.57	0	0
2010-01-02 03:00:00	181.0	-7	-5.0	1022.0	SE	5.36	1	0
2010-01-02 04:00:00	138.0	-7	-5.0	1022.0	SE	6.25	2	0
2010-01-02 05:00:00	109.0	-7	-6.0	1022.0	SE	7.14	3	0
2010-01-02 06:00:00	105.0	-7	-6.0	1023.0	SE	8.93	4	0
2010-01-02 07:00:00	124.0	-7	-5.0	1024.0	SE	10.72	0	0
2010-01-02 08:00:00	120.0	-8	-6.0	1024.0	SE	12.51	0	0
2010-01-02 09:00:00	132.0	-7	-5.0	1025.0	SE	14.30	0	0

使用データ：
環境センサ(温度、気圧、風向、風速、雨、雪)、
および大気汚染濃度(PM2.5)の1時間毎のデータ

MXNet (LSTM) を用いてN次元のベクトルデータ(環境センサの値)から値(汚染濃度)の予測を行うサンプル

このサンプルでは値の相関があまりないのか、良い結果は出ていない

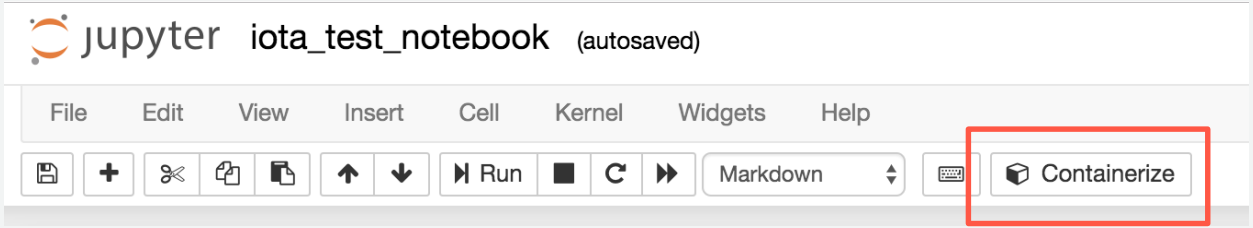


SageMakerコンテナでの分析定期実行

- IoT Analyticsから作成したノートブックのメニューに追加されている"Containerize"ボタンを押下することで分析用コンテナが作成できる
    - コンテナ化することで定期的に分析を実行するよう構成できる

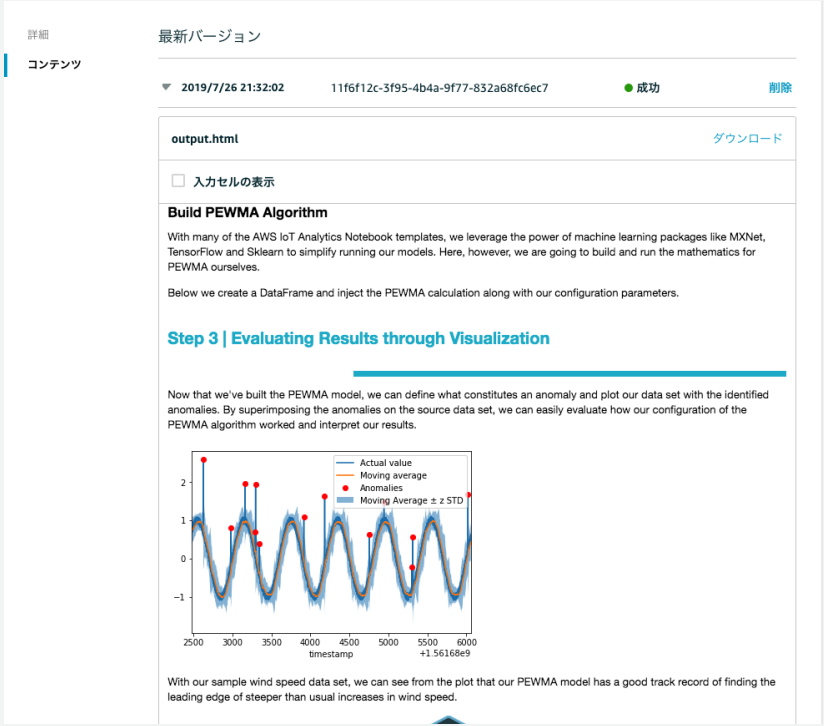
- コンテナで異常を検出した場合には通知するような実装を行うなどして異常検出の自動化を図ることも可能

- カスタムコンテナと使用できるパラメータは同一
- パラメータでGlobal変数の値を設定可能



SageMakerコンテナでの可視化・分析結果表示

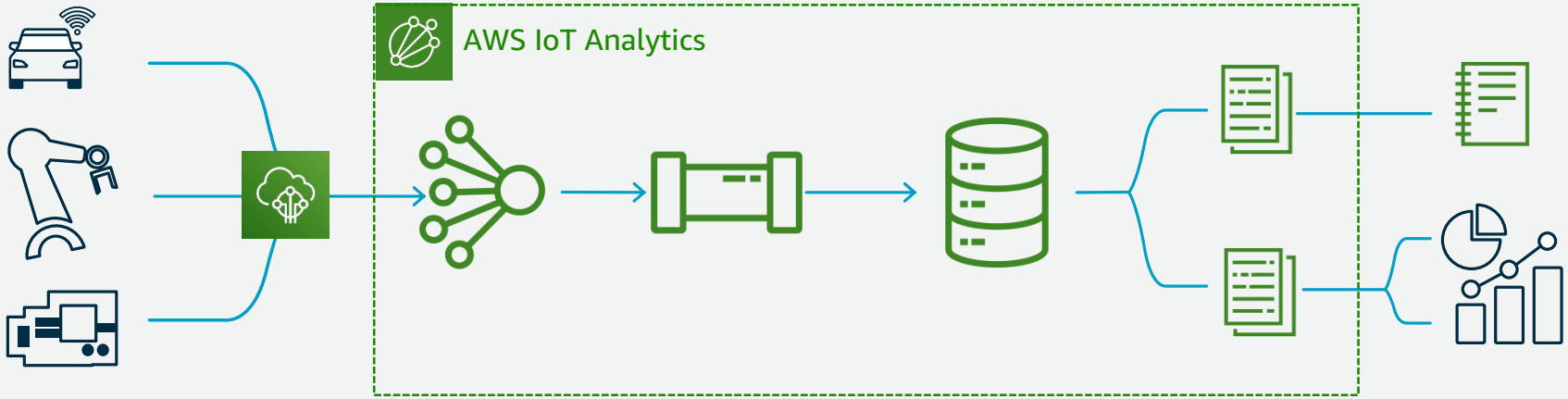
- SQLデータセットとリンクしてコンテナを定期実行
- SageMakerのアウトプットHTMLをコンテンツビューで確認可能



本日のアジェンダ

- IoTデータの特徴とその活用例
- AWS IoT Analyticsの詳細
- **パイプライン構成パターンについて**
- まとめ

パイプラインを構成して複数の分析を実行する



例：センサーデバイスすべての情報を可視化しつつ、特定の条件を満たしたデータを用いてモデルを作成する

パイプライン構成パターン

- 1 channel + 1 pipeline + 1 datastore
- N channel + N pipleline + 1 datastore
- 1 channel + N pipeline + N datastore
- 1 channel + N pipeline + 1 datastore

aws

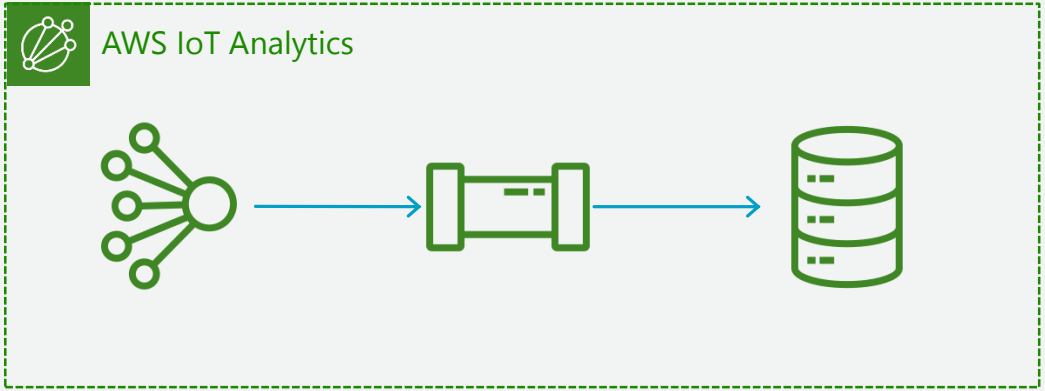
Type1: 1 channel + 1 pipeline + 1 datastore

Pros

- 単一スキーマのメッセージ処理に適す
- 全てのメッセージが単一のデータストアに格納されるためデータセットの作成が簡単
- 全ての属性がデータストアに正規化されて格納されるためデータ分析の柔軟性が増す
- メッセージ処理が一回なので費用対効果が高い

Cons

- 異なるスキーマを持つメッセージは処理できない
- パイプライン処理が複雑になる可能性がある



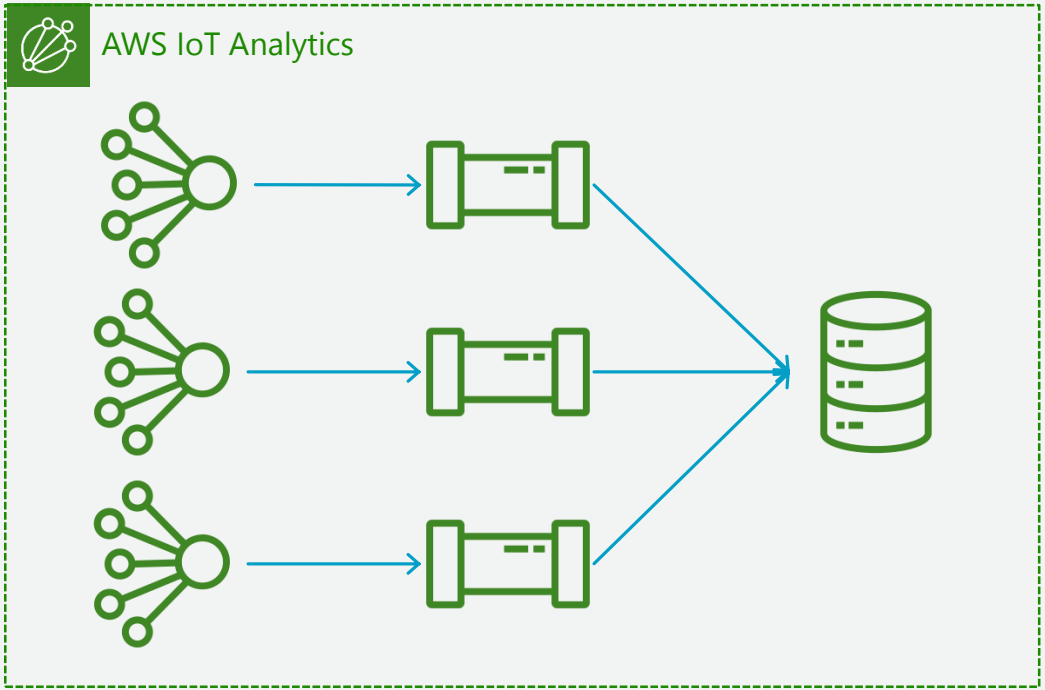
Type2: N channel + N pipeline + 1 datastore

Pros

- マルチスキーマメッセージはType 1データフローよりもはるかに処理が簡単
- パイプライン機能は、特定の種類の変換に対してのみ適用できる
- データセットは1つのデータストアから簡単に生成できるため、複数のデータストアを結合することを心配する必要はない
- メッセージは1回だけパイプラインを通過するため、チャネルレベルでの複製コストを節約できる

Cons

- データストアはパイプラインが増えるにつれ急激に増加する可能性がある
- データセットを作成するにあたりより多くのデータをスキャンするクエリが必要になり、データセット作成コストが増加する可能性がある



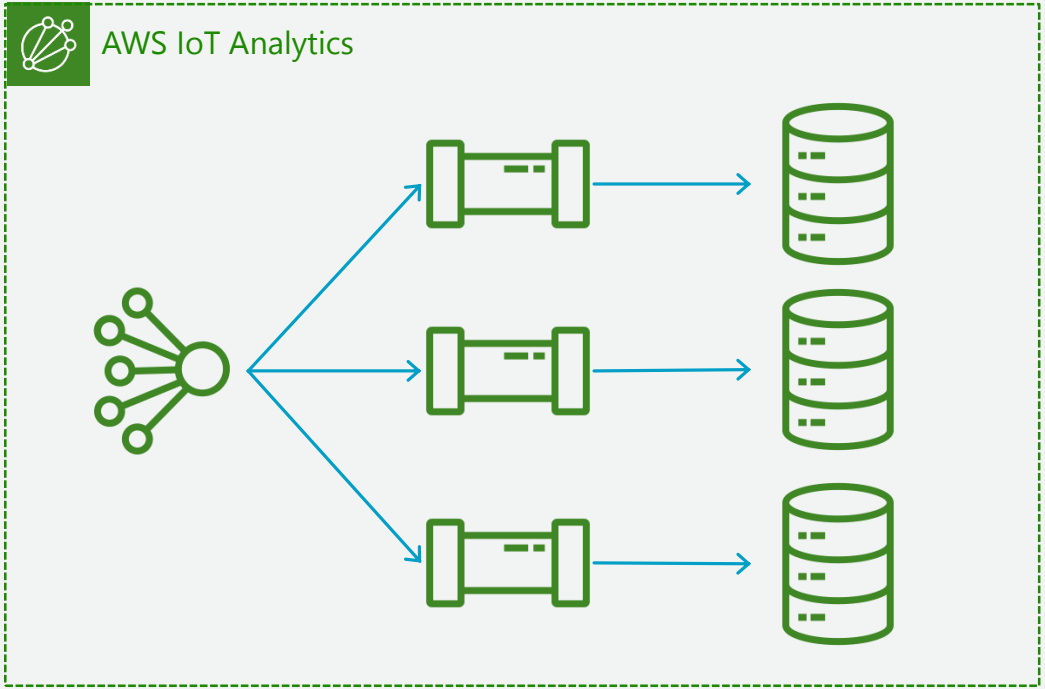
Type3: 1 channel + N pipeline + N datastore

Pros

- 同じメッセージに対して複数の独立した変換を実行できる
- パイプライン内のメッセージをフィルタリングすることで、個々のデータストアはすべてのIoTメッセージのサブセットのみを格納するように設計できる
- パイプラインがメッセージを除外するように設定されている場合、データセットを生成するためのクエリで大量のデータをスキャンする必要がない

Cons

- 複数のパイプライン間でのメッセージの複製によりコストが増加する可能性がある (代替ソリューションとして、各チャネルがIoTルールエンジンから特定のタイプのメッセージを受信するように設定された、複数のチャネルを作成すること。後述)



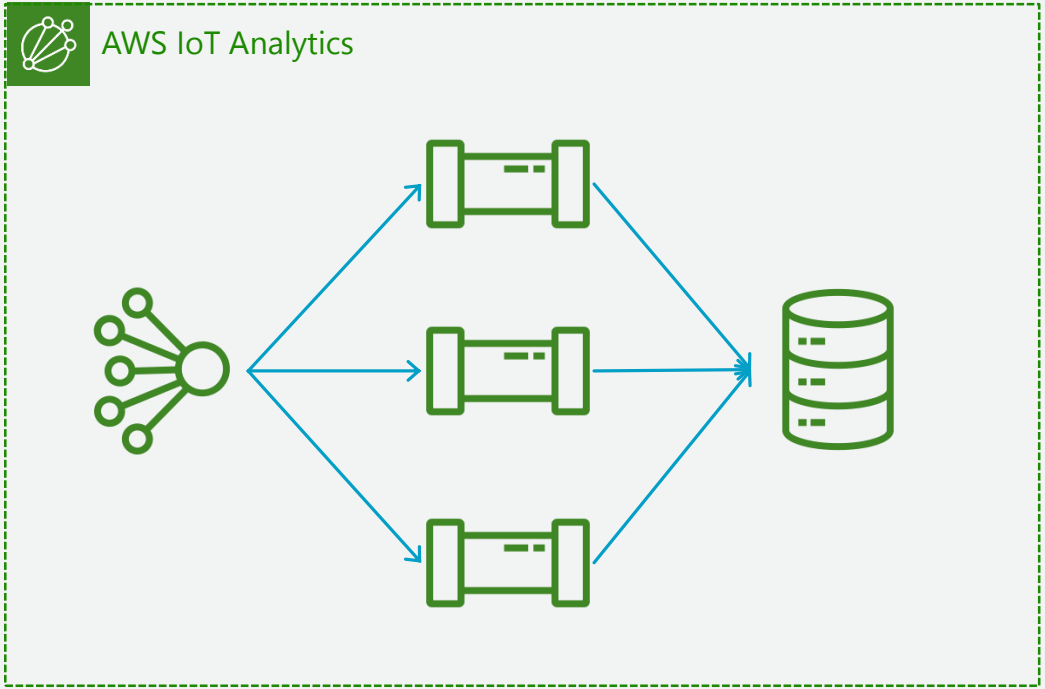
Type4: 1 channel + N pipeline + 1 datastore

Pros

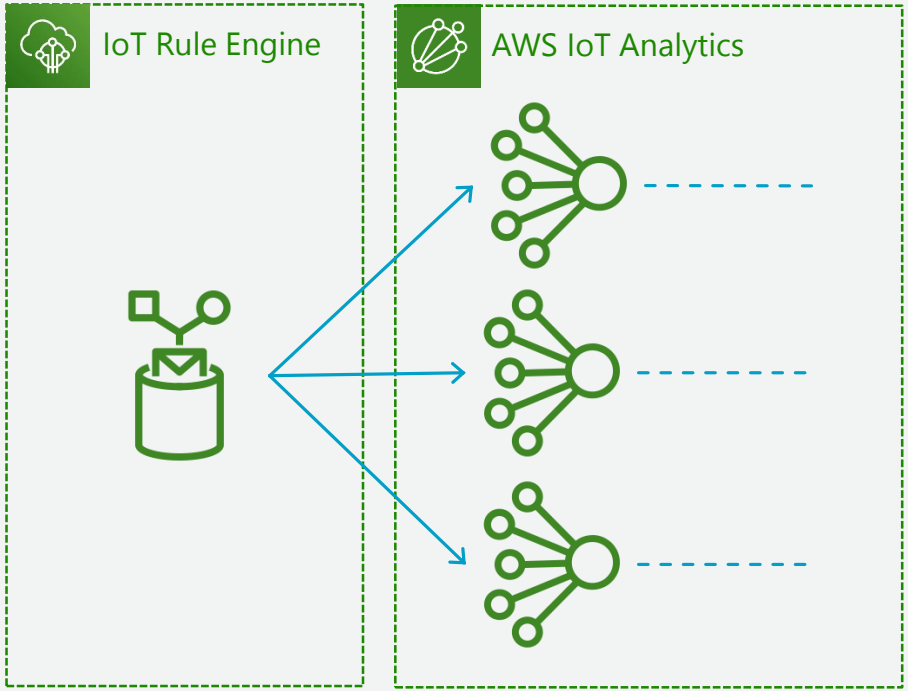
- 同じメッセージに対して複数の独立した変換を実行する
- 変換されたメッセージは1つのデータストアにまとめられるため、異なるデータセット作成要件に答えやすい

Cons

- 複数のパイプライン間でメッセージを複製することによるコストの増加 (Type3と同様)
- パイプラインでのメッセージ処理を間違えるとデータストアに蓄積されたメッセージが意図せず複数生まれてしまう

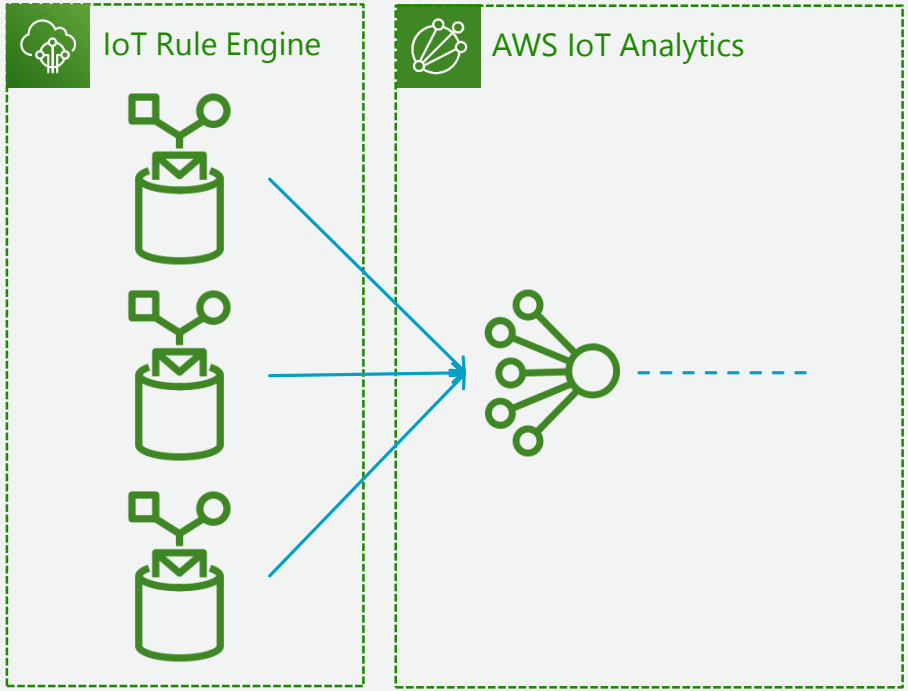


AWS IoTルールエンジンとAWS IoT Analyticsの接続 ①



1つのAWS IoTルールでメッセージを複数のチャネルに複製

AWS IoTルールエンジンとAWS IoT Analyticsの接続 ②

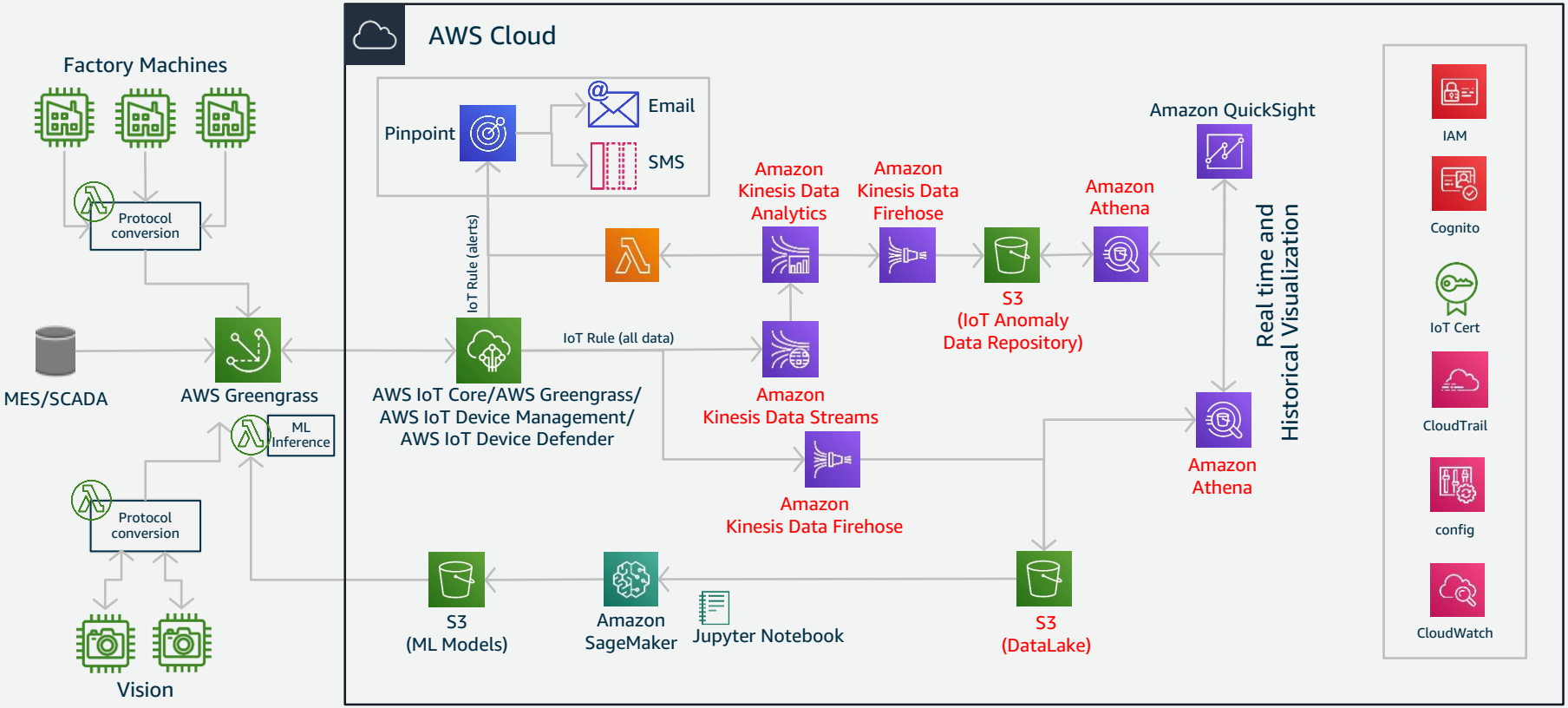


複数のルールでメッセージを同じチャネルに転送

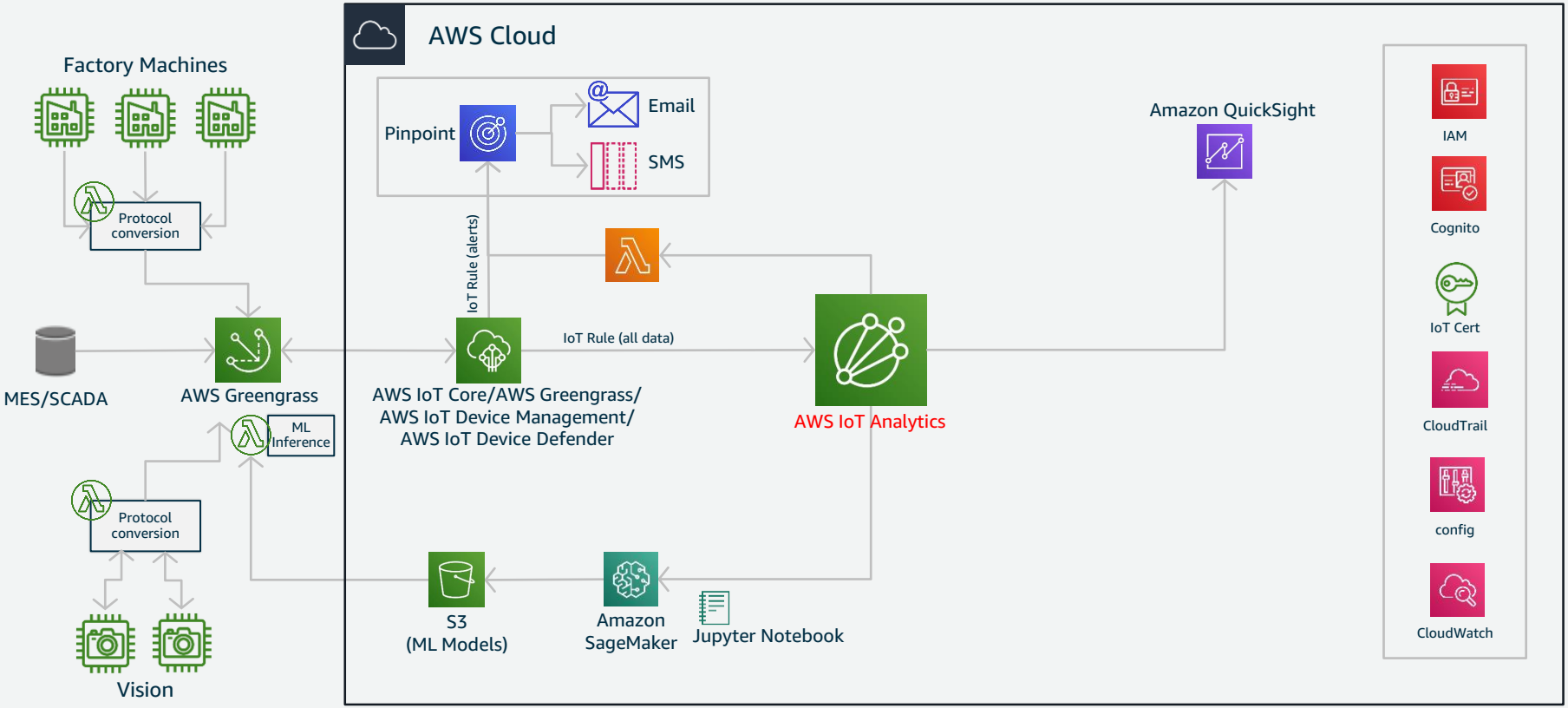
本日のアジェンダ

- IoTデータの特徴とその活用例
- AWS IoT Analyticsの詳細
- パイプライン構成パターンについて
- まとめ

予知保全を実現するアーキテクチャ例



予知保全を実現するアーキテクチャ例



まとめ

- IoTデータの収集、蓄積、解析を行うための機能が一つのサービスにまとまっており、分析、可視化するまでの構成が**容易に準備できる**
- S3配信機能による**既存データレイクとの連携**や、パイプラインでの**Lambda利用による他のサービスと連携**したデータ分析も実現できる
- QuickSightとの連携による時系列データ分析
    - **QuickSight ML Insights**を使うと時系列データに対する予測や異常検知も容易に実現可能
- SQLによる分析やJupyter Notebooksによる分析のほか、カスタムコンテナを使った**独自の分析を導入**できる
    - IoT分析を学んだり、自身のデータに適用可能なテンプレートも用意されている
- マルチスキーマメッセージを扱う場合は**パイプライン構成とIoT Coreのルールエンジン**をうまく活用しコストを最小限におさえる構成を工夫する

参考資料

- AWS IoT Analytics documentation
  https://docs.aws.amazon.com/iotanalytics/
- IoTデータをデータレイクと統合する (Blog)
  https://aws.amazon.com/jp/blogs/news/integrating-iot-data-with-your-data-lake-with-new-aws-iot-analytics-features/
- Designing dataflows for multi-schema messages in AWS IoT Analytics (Blog)
  https://aws.amazon.com/jp/blogs/iot/designing-dataflows-for-multi-schema-messages-in-aws-iot-analytics/
- Containerize your IOT application with AWS IOT Analytics (Blog)
  https://aws.amazon.com/jp/blogs/iot/containerize-your-iot-application-with-aws-iot-analytics/

AWS の日本語資料の場所「AWS 資料」で検索

https://amzn.to/JPArchive

AWS Well-Architected 個別技術相談会

毎週”W-A個別技術相談会”を実施中

- AWSのソリューションアーキテクト(SA)に対策などを相談することも可能
- 申込みはイベント告知サイトから

(https://aws.amazon.com/jp/about-aws/events/)

AWS イベント で[検索]

ご視聴ありがとうございました

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive