AWS

SUMMIT

最新 IoT デザインパターン

AWS IoT と AWS Greengrass を用いた構築パターン

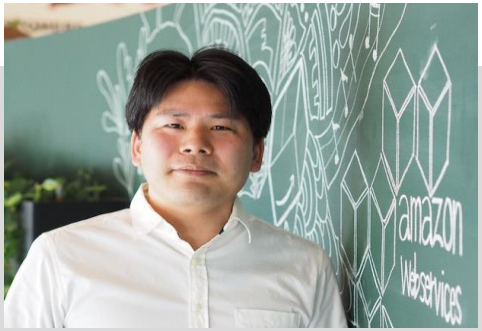
Amazon Web Services Japan
Soution Architect
Takashi KOYANAGAWA / Atushi FUKUI

2017/6/1

amazon web services

自己紹介

自己紹介



❖名前
- ❖ 小梁川 貴史（こやながわ　たかし)

❖所属
- ❖ 技術統括本部 IoT Solution Design Team
- ❖ ソリューション アーキテクト

❖前職
- ❖ 電機メーカー自社サービスの開発・運用
  元AWSユーザ

❖好きなAWSサービス
- ❖ AWS IoT
- ❖ AWS Lambda(python)
- ❖ Amazon Kinesis

❖名前
- ❖ 福井 厚（ふくい あつし）

❖所属
- ❖ 技術統括本部エンタープライズ ソリューション部
- ❖ ソリューション アーキテクト

❖前職
- ❖ エンタープライズ アプリケーション開発コンサルタント

❖好きなAWSサービス
- ❖ AWS Code シリーズ
- ❖ AWS IoT
- ❖ AWS Lambda（C#）

このセッションの対象者

- IoTに興味がある
- AWSのマネージドサービスを利用したことがある
- アーキテクチャの設計に関わる人材
    - アーキテクト、デベロッパー、マネージャ

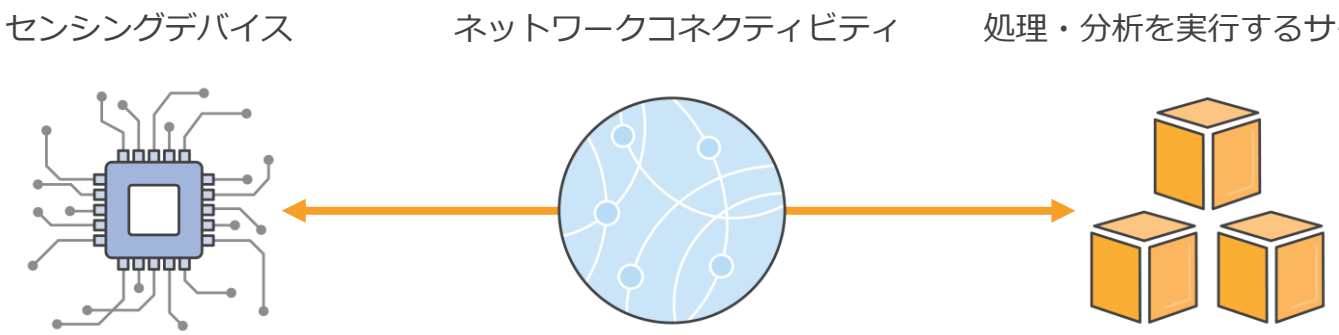
このセッションで持ち帰って頂きたいこと

- AWS IoTの機能を理解する
- AWS GreenGrassの機能を理解する
- AWS を利用したIoTデザインパターンを理解する

AWS における IoT Solutionのご紹介

amazon web services

IoT を構成する３要素



誰もがすぐに調達可能で
アイデアと実行能力があれば市場参入が容易
イノベーションにより市場に破壊が起きる

さまざまなIoTユースケース

製造

メンテナンス/異常検知

モニタリング

交通

車両センサー

ドライバーの安全

エネルギー

スマートメータ

メンテナンス/異常検知

リテール

動線把握

O2O

家電

スマート家電

オートメーション

ヘルスケア

医療機器管理

遠隔医療

農業

モニタリング

遠隔制御

IoTの代表的な要件

✓**モニタリング**

位置情報管理・状態監視・
実績把握・動線把握

✓**データ連携/モバイル**

保守作業・スマートデバイス・
企業連携・オープンデータ

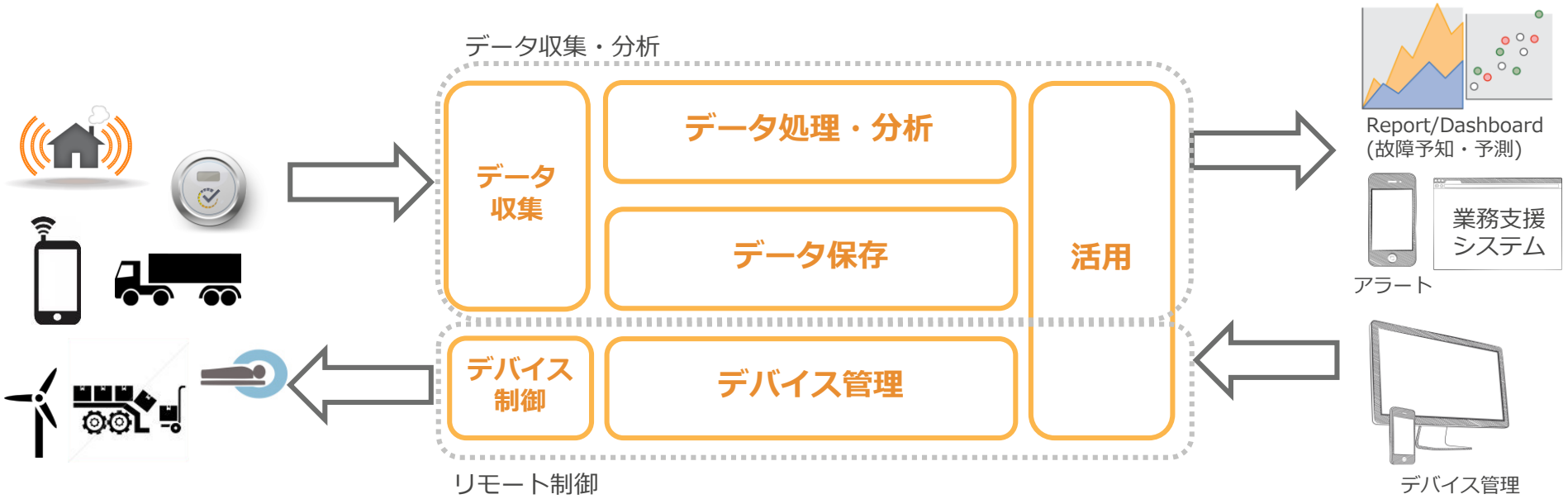
✓**予防予知保全**

異常検知・故障予測

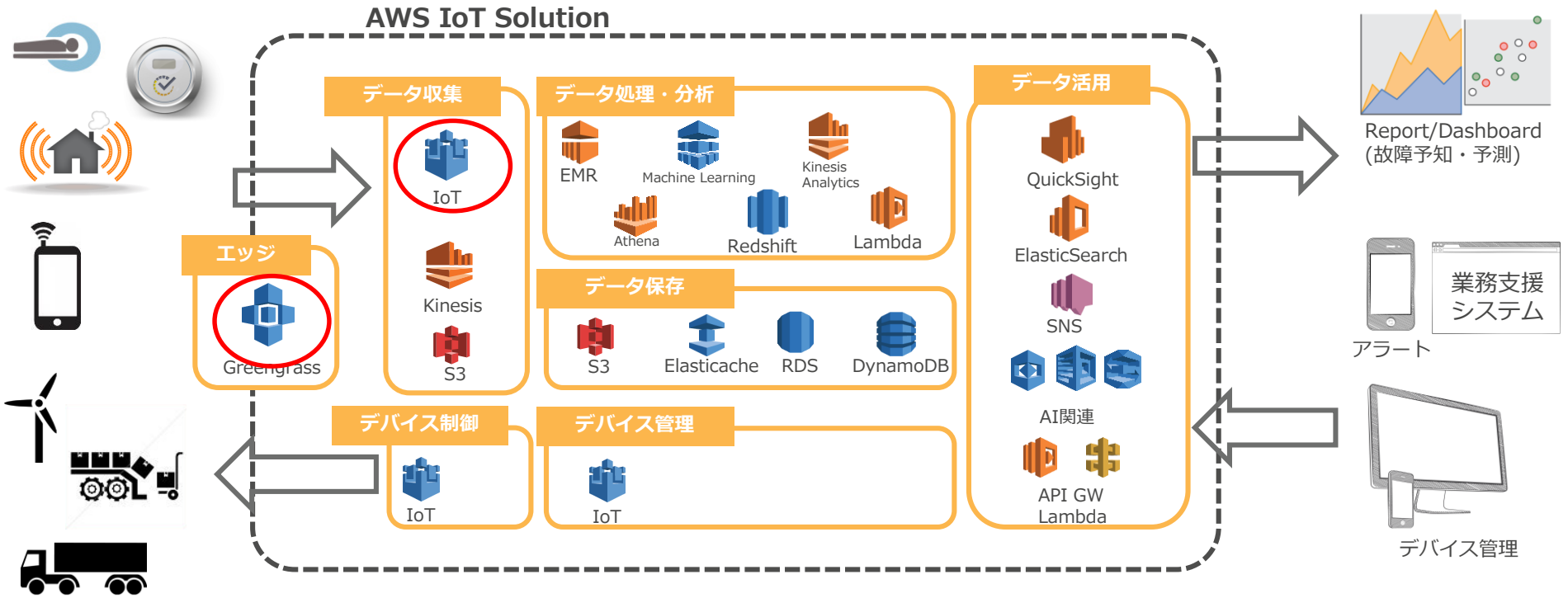
✓**遠隔制御**

機器運用・ファームアップ

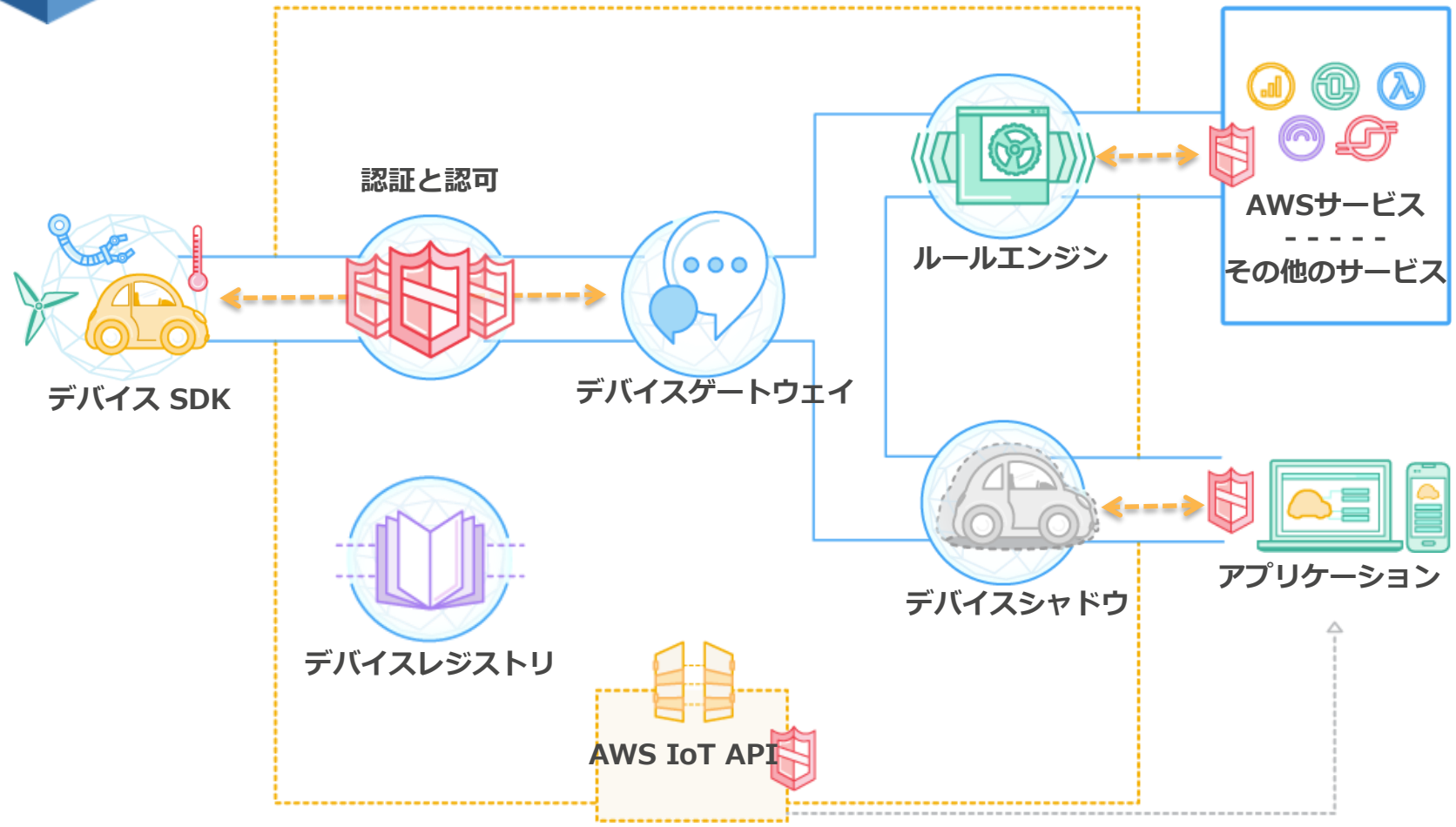
IoTプラットフォーム



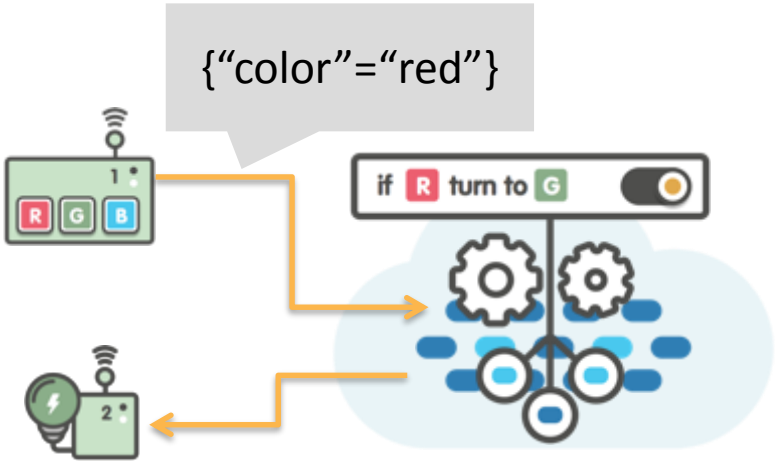
AWS IoT Solution



AWS IoT 全体構成


認証と認可
デバイス SDK
デバイスゲートウェイ
ルールエンジン
AWSサービス
その他のサービス
デバイスシャドウ
アプリケーション
デバイスレジストリ
AWS IoT API

ルールエンジン



SELECT Data FROM topic WHERE 条件

SELECT * FROM ‘things/thing-2/color’
WHERE color = ‘red’

■シンプル＆慣れた構文
・SQL文を使ったトピックのフィルタ
・オプションのWHERE句で条件を記述することが可能
・JSONサポート

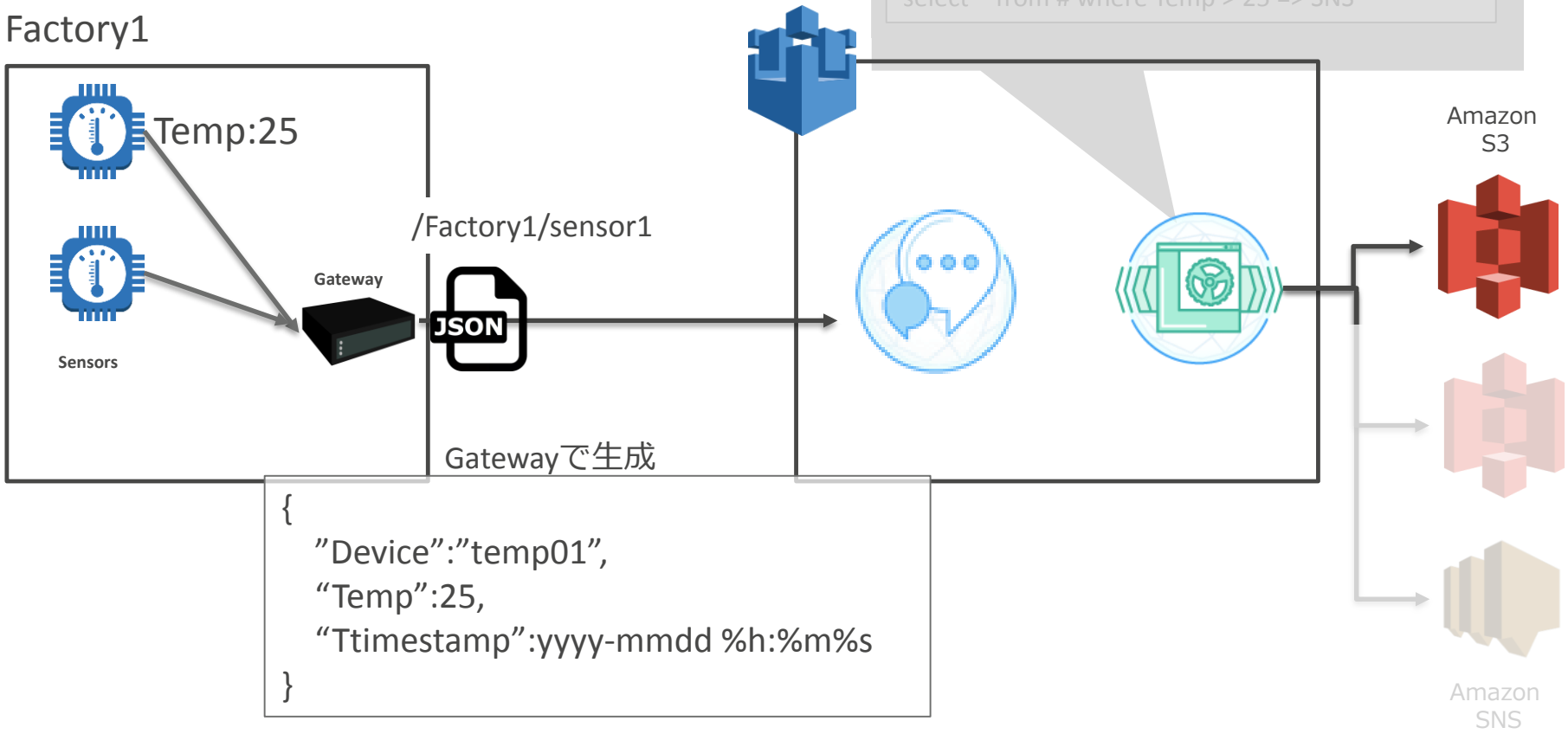
■メッセージ変換機能
・文字列操作 (正規表現サポート)
・算術計算
・コンテキストベースのヘルパー
・暗号
・UUID, Timestamp, 乱数など.

ルールエンジンの活用例

select * from /Factory1/sensor1 => S3(bucket1)

select * from /Factory1/sensor2 => S3(bucket2)

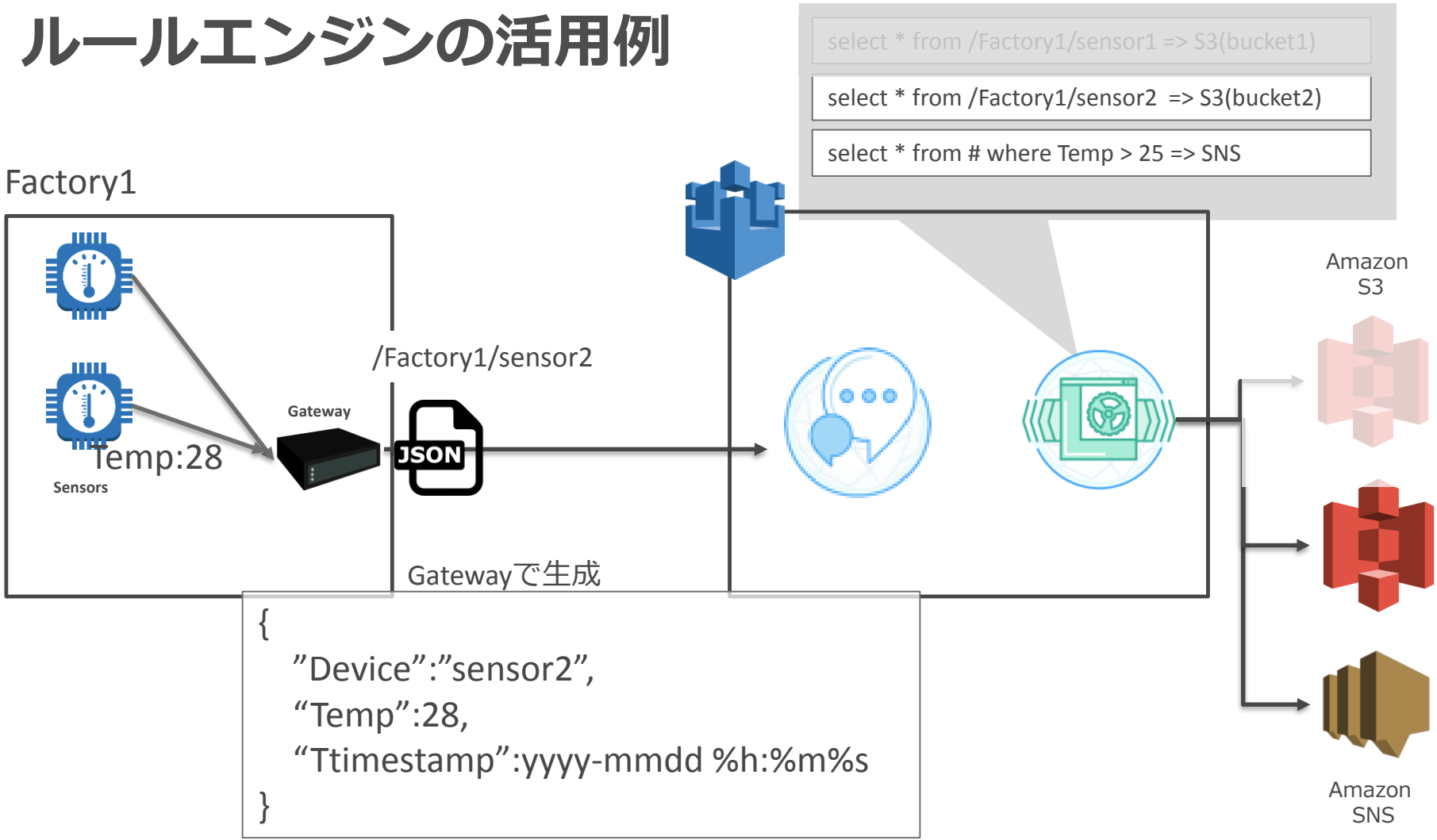
select * from # where Temp > 25 => SNS



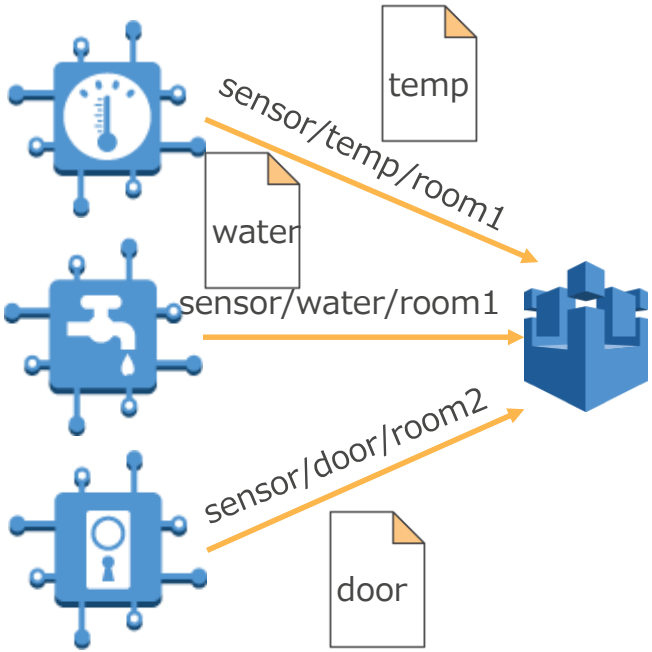
Gatewayで生成

```
{
   ”Device”:”temp01”,
   “Temp”:25,
   “Ttimestamp”:yyyy-mmdd %h:%m%s
}
```

ルールエンジンの活用例



データ収集のユースケース



sensor/#

ルールエンジンから、
以下に挙げるサービスとの連携
など*12*のアクションが定義可能

Amazon S3
Amazon DynamoDB
Amazon Kinesis
Amazon SNS
AWS Lambda
Amazon SQS
Amazon RDS
Amazon Redshift
Amazon Glacier
Amazon EC2

外部エンドポイント
(LambdaやSNS経由)

便利な機能がいろいろ

ルールエンジンと
アクション

SQLライクな構文でルールの設定
ルールに合致した場合、AWSの各種サービスとのインテグレーション

デバイスシャドー

デバイスが物理的に接続されてなくてもコマンドを伝達できる

デバイスとクラウドとの
相互認証

TLS1.2を用いた相互認証
証明書に対するポリシ設定も可能

デバイスレジストリ

多くのデバイスを管理
Key-Value形式でファームバージョンなどの属性情報も管理可能

IoTにおける
クラウドデザインパターン

IoTにおける
クラウドデザインパターン

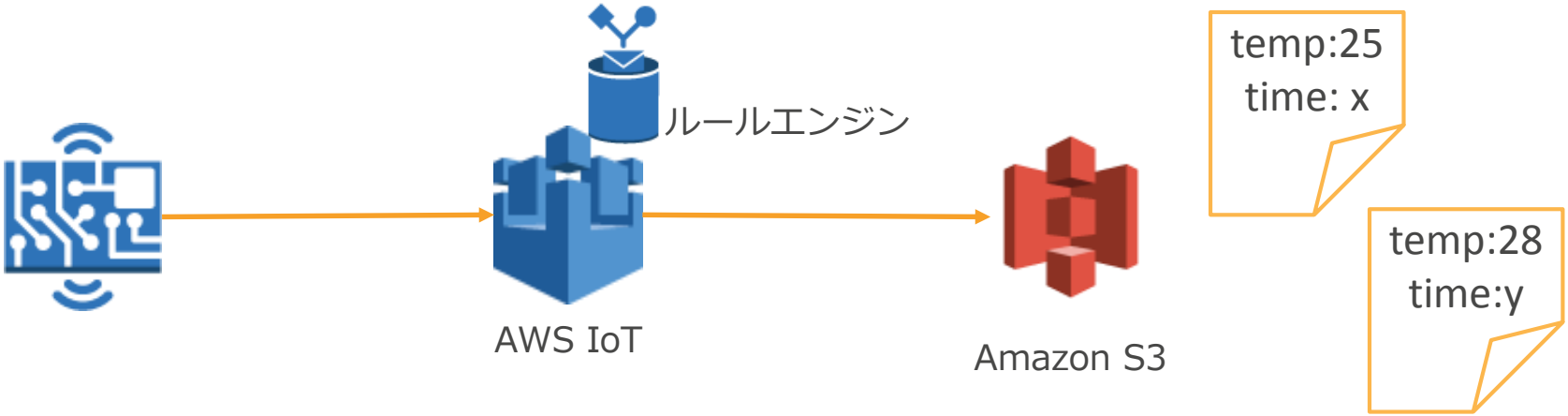
- データ収集・活用のユースケース
- shadow活用のユースケース
- その他

IoTにおける
クラウドデザインパターン

- データ収集・活用のユースケース
- shadow活用のユースケース
- その他

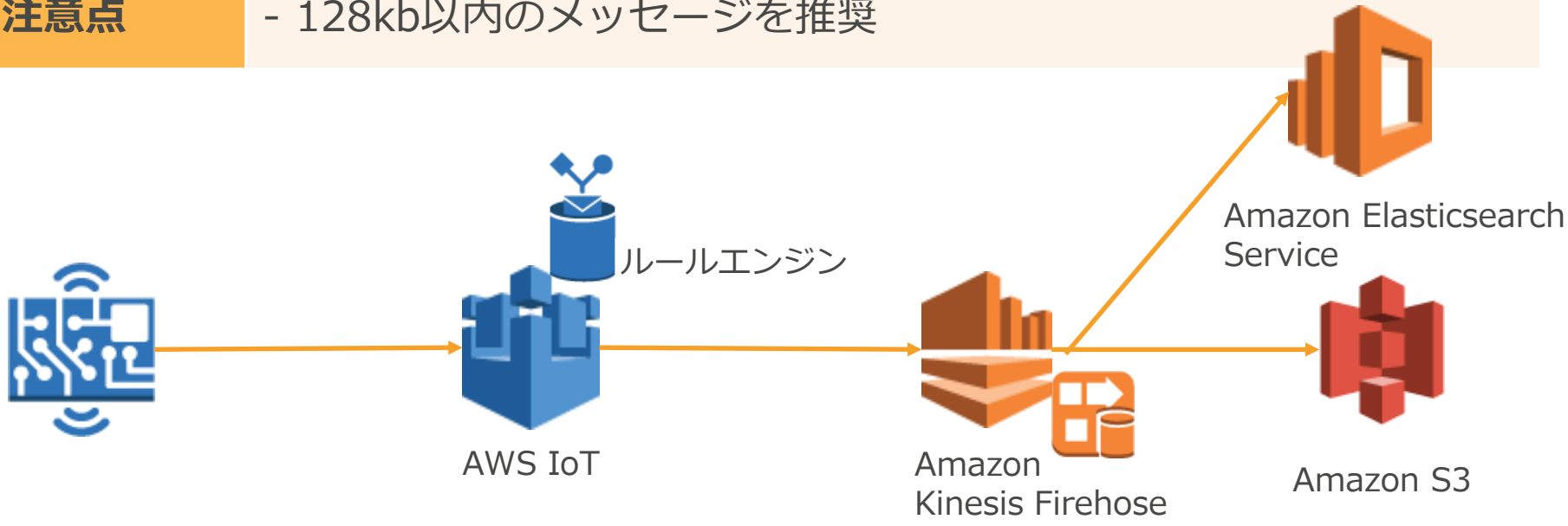
課題

ルールエンジンからS3へファイルを送信しているが、
1file/1dataの単位でデータ解析に使いづらい。



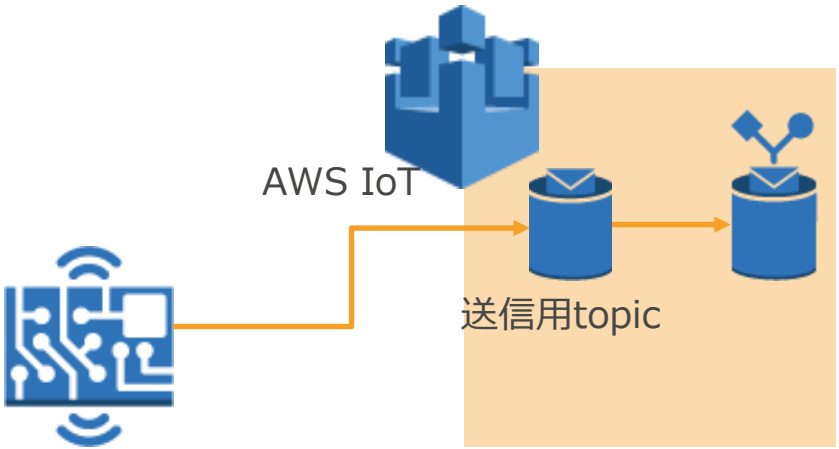
バッチ データ ストア パターン

利用用途	- センサーデータなどのバックアップ - 機械学習などのモデル作成に向けた準備
メリット	- メッセージを時間またはサイズ指定で1ファイルに複数データをかけるために今後データ活用がやりやすくなる
注意点	- 128kb以内のメッセージを推奨



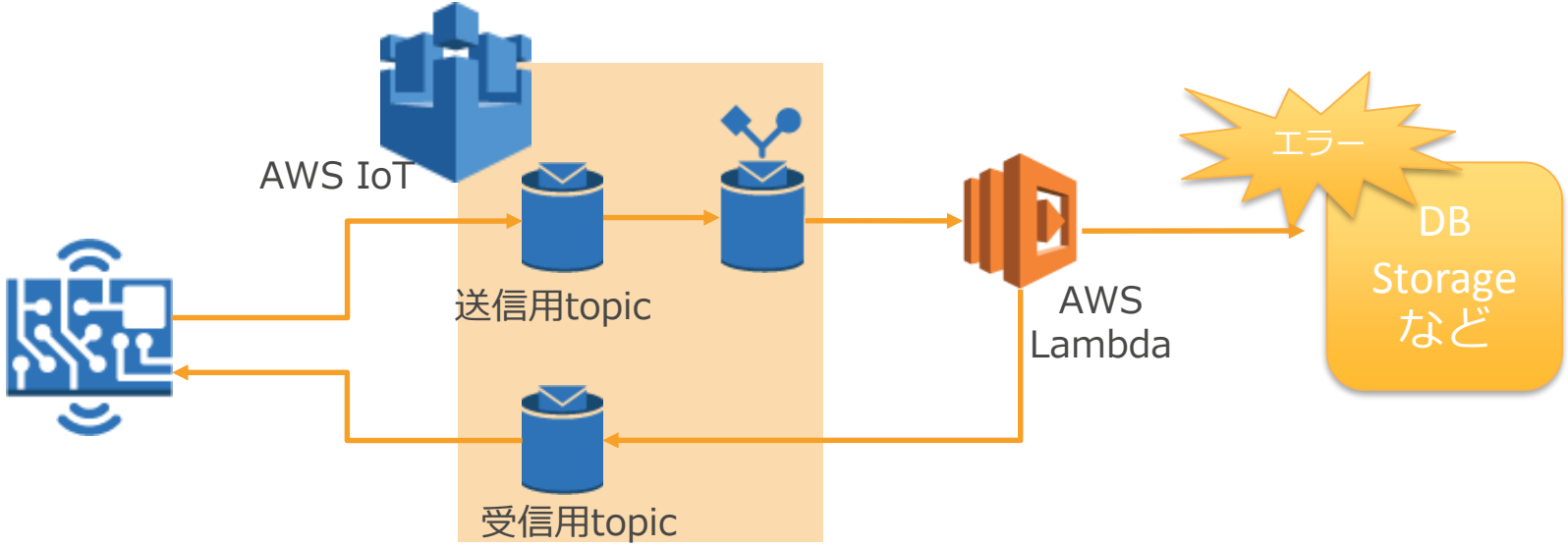
課題

backendシステムで正常終了したかや、エラーが発生したことを知りたい



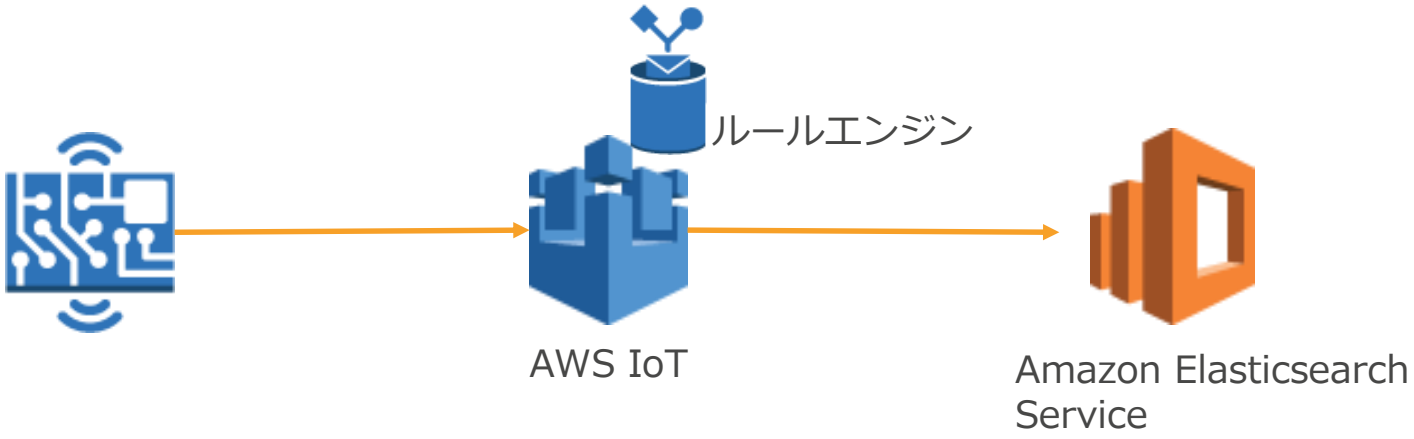
QoS2 エミュレート パターン

対象シーン	- データの到達通知 /エラー発生をデバイスで確実に受取たい
条件	- デバイス側で送信したデータに識別子を付与し、その識別子のデータ処理が正しく処理されたかを別トピックをサブスクライブして待ち受けることができる
注意点	- 受信を必ず返却するので必要なメッセージ数が2倍になる



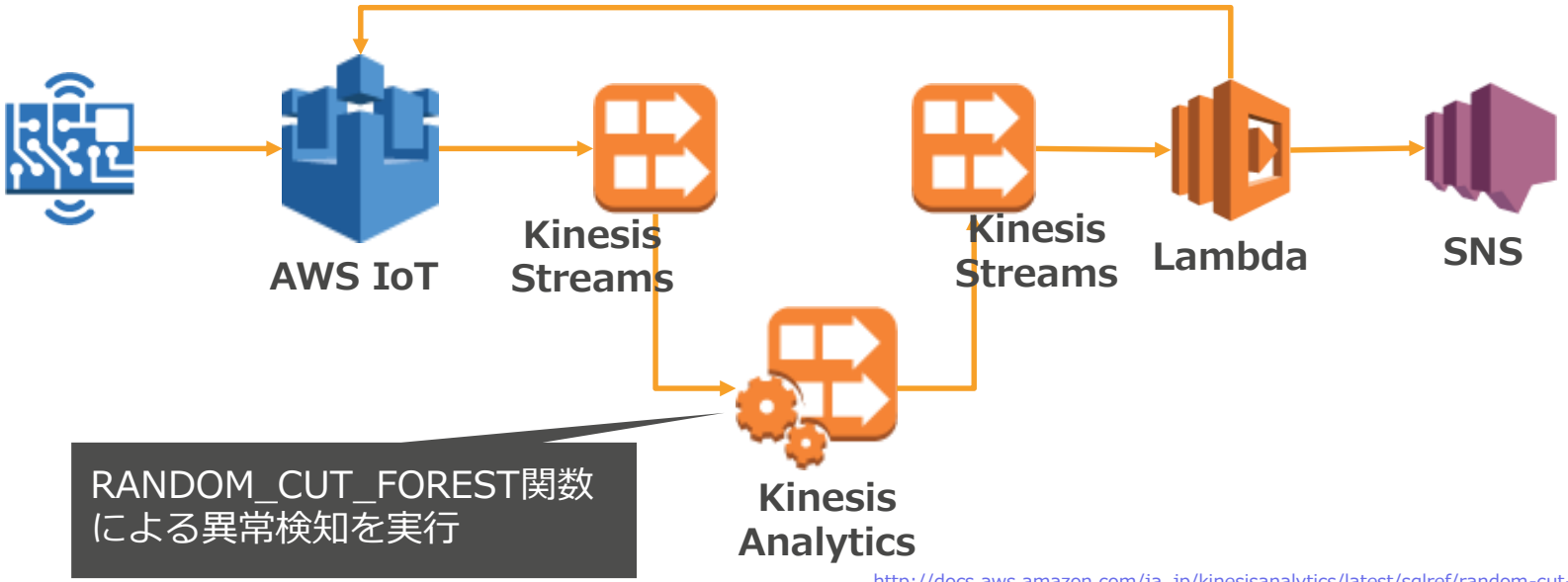
課題

ニアリアルタイムの処理/スライディングウィンドをマネージド・サービスで実現したい



リアルタイム異常検知パターン

対象シーン	- ニアリアルタイムでの異常検知、スライディングウィンドウを作成したい
条件	- Kinesis analyticsの利用
注意点	- 現在 Tokyo regionにkinesisn analyticsがローンチされていない



http://docs.aws.amazon.com/ja_jp/kinesisanalytics/latest/sqlref/random-cut-forest.html

IoTにおける
クラウドデザインパターン

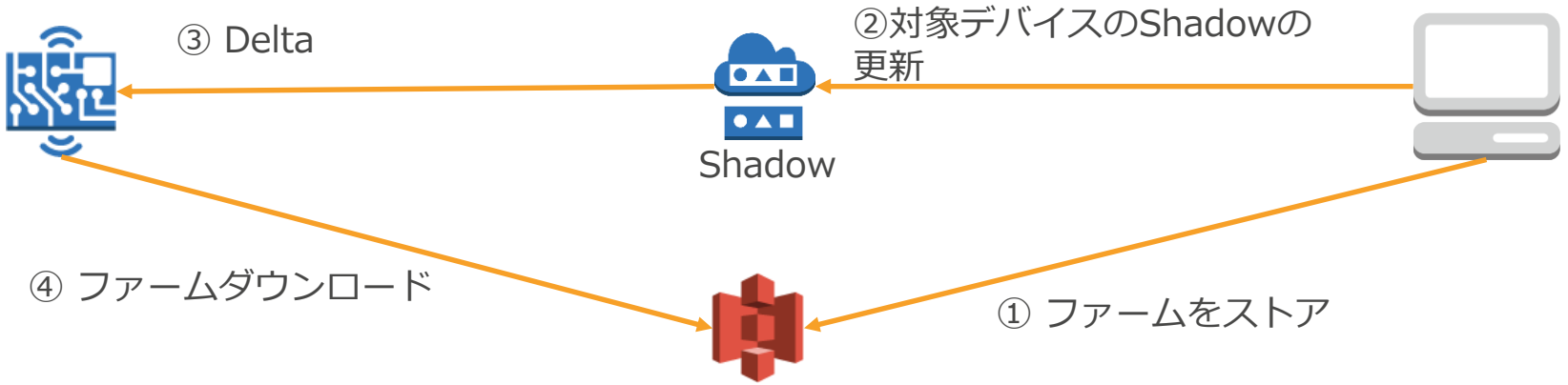
- データ収集・活用のユースケース
- shadow活用のユースケース(2)
- その他

課題

遠隔地にあるデバイス管理はコストが掛かる、または大量にあるデバイスのバージョン管理/update指示をしたい。

ファームアップパターン

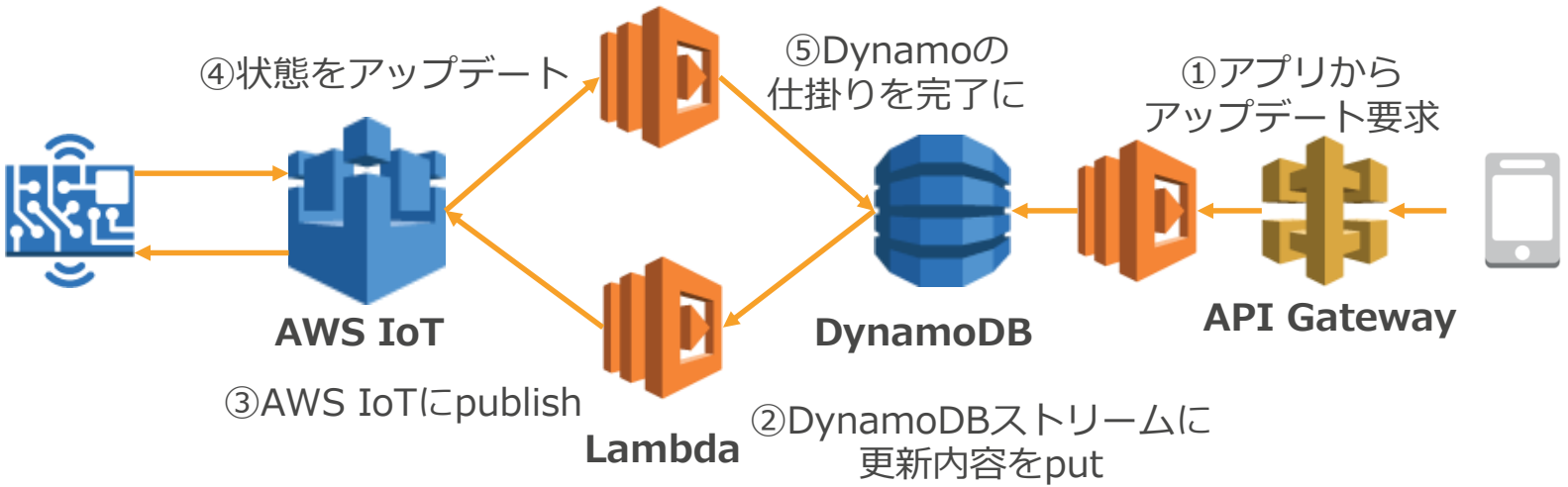
対象シーン	- 特定のデバイスに対してファームアップを指示
条件	- Device Registryにデバイスとファームバージョンを管理している - Device Shadowを使ってコマンドの実行が可能 - HTTPSでファームをダウンロード可能
注意点	- ファームバージョンアップのプログラムはお客様にて実装



課題

デバイスの一覧、状態を管理できるデータベースが欲しい。IoT shadowのthing statusコマンドにlistコマンドがない

インテリジェントシャドウパターン	
対象シーン	- Web/スマホアプリなどからデバイスをコントロールしたい
条件	- 特になし
注意点	- シャドウを利用することが必須(シャドウは8Kbの制約)



IoTにおける
クラウドデザインパターン

- データ収集・活用のユースケース
- shadow活用のユースケース
- その他

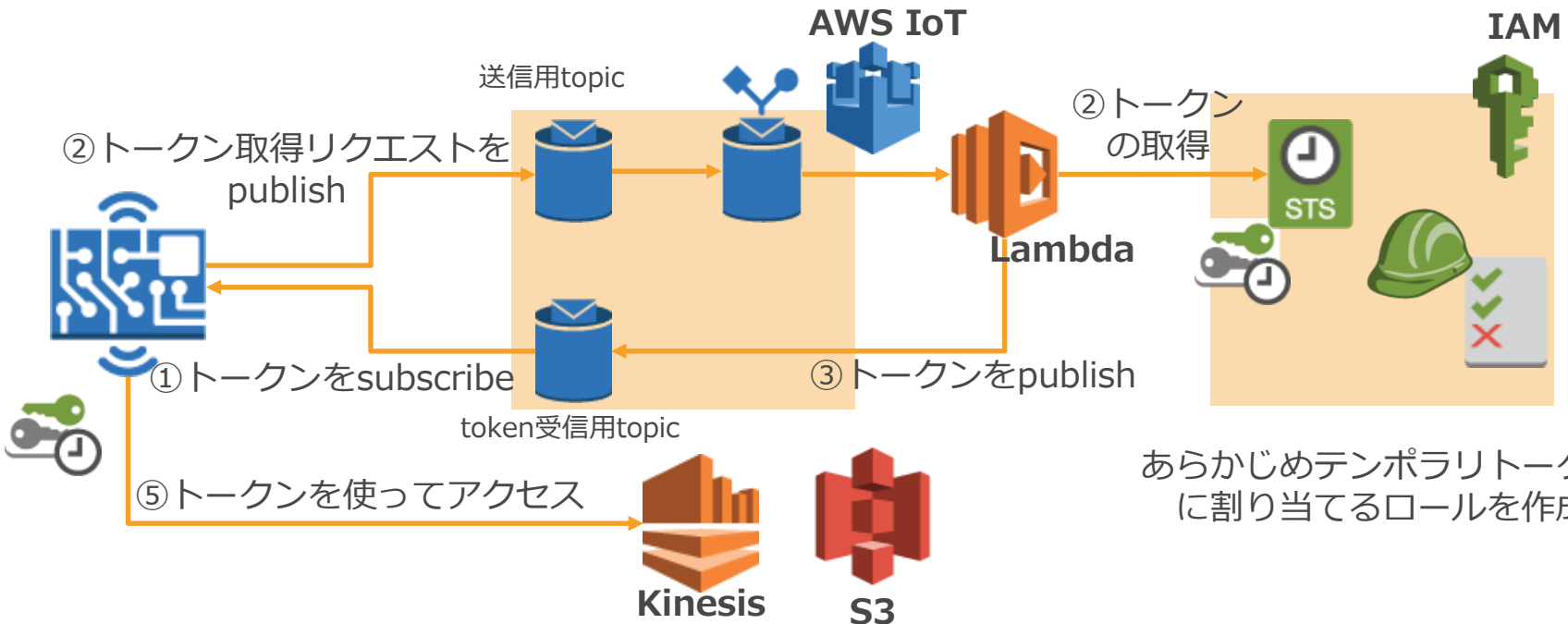
課題

AWS IoTではpayloadの観点から転送しにくい大きめのデータ転送をセキュアに送信したい。
(GatewayへThingにIAMは発行したくない)

- デバイスから写真や動画をS3へ
- ログファイルをkinesisへ

テンポラリトークン取得パターン

対象シーン	- AWS IoTの証明書を利用したセキュアな接続を利用してデバイスへAWSリソースに対してアクセス権を付与
条件	- 事前に利用するIAMが必要
注意点	- 特になし

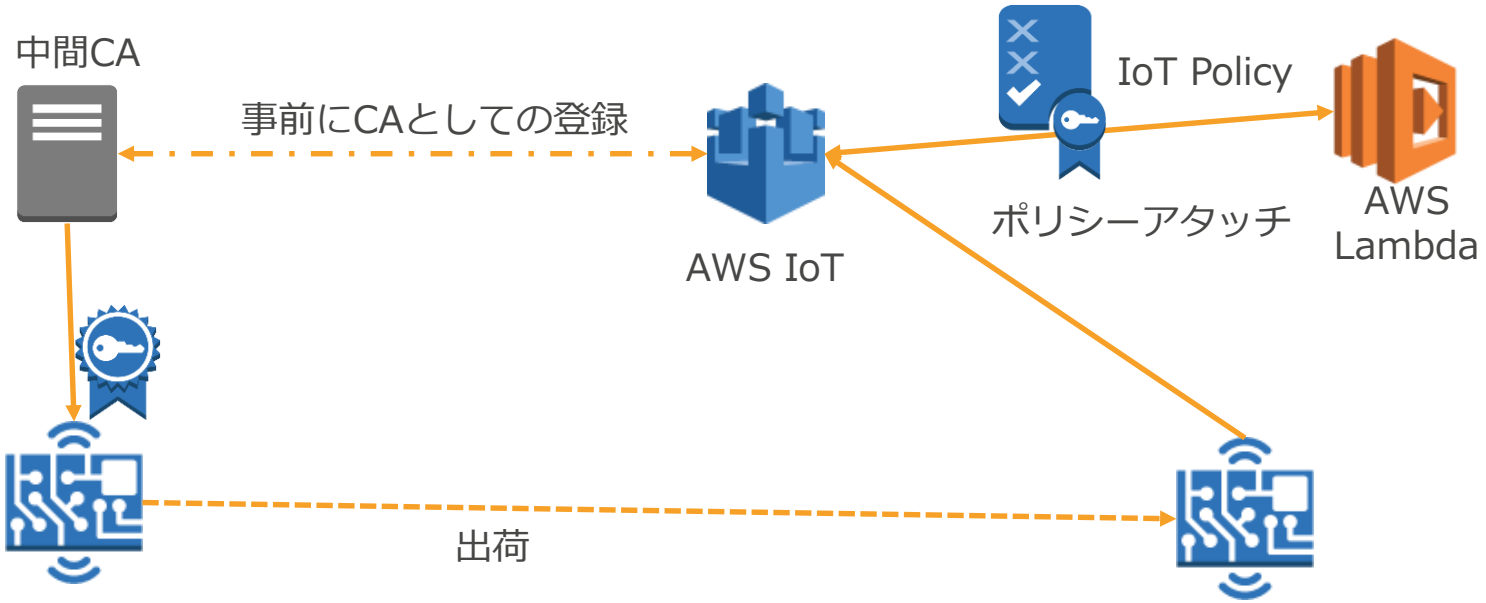


課題

いつどのくらい必要になるかわからない証明書を事前に作りたくない。デバイスが使われるときに証明書を有効化したい。

持ち込み証明書によるJust in time registrationパターン

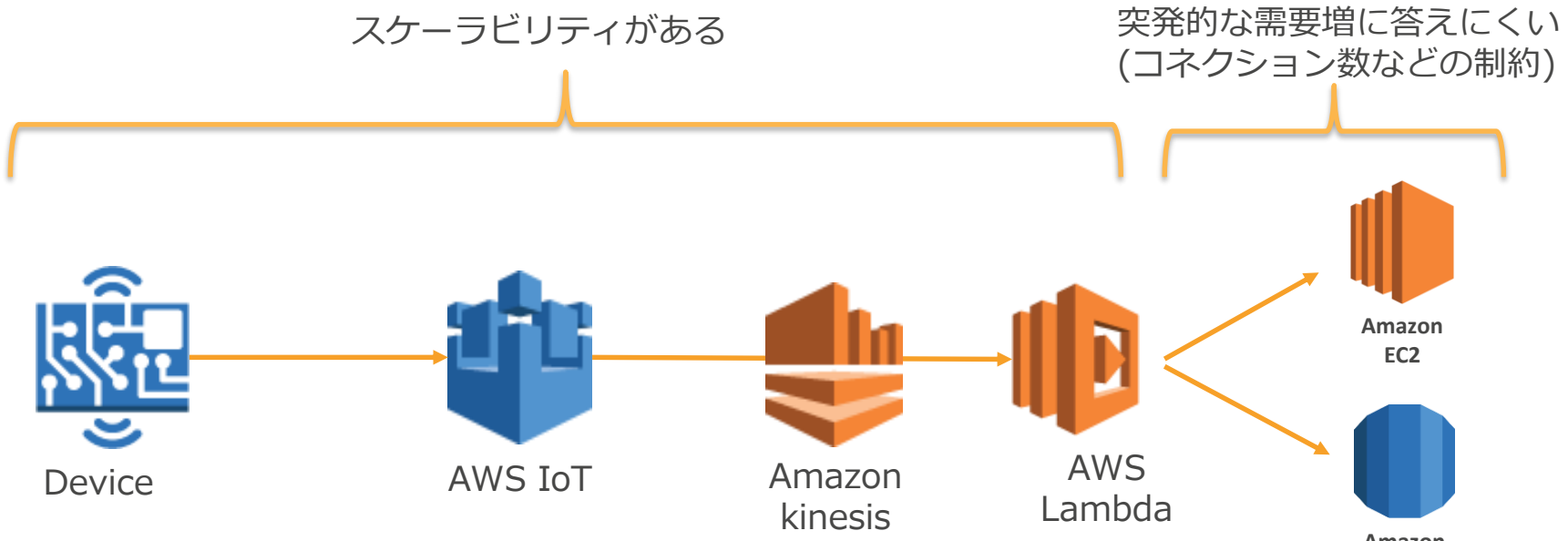
対象シーン	- デバイスを製造する段階でデバイス証明書を書き込む
条件	- 特になし
注意点	- CAの構築/運用が必要



AWS IoTにおけるアンチパターン

データ収集のアンチパターン

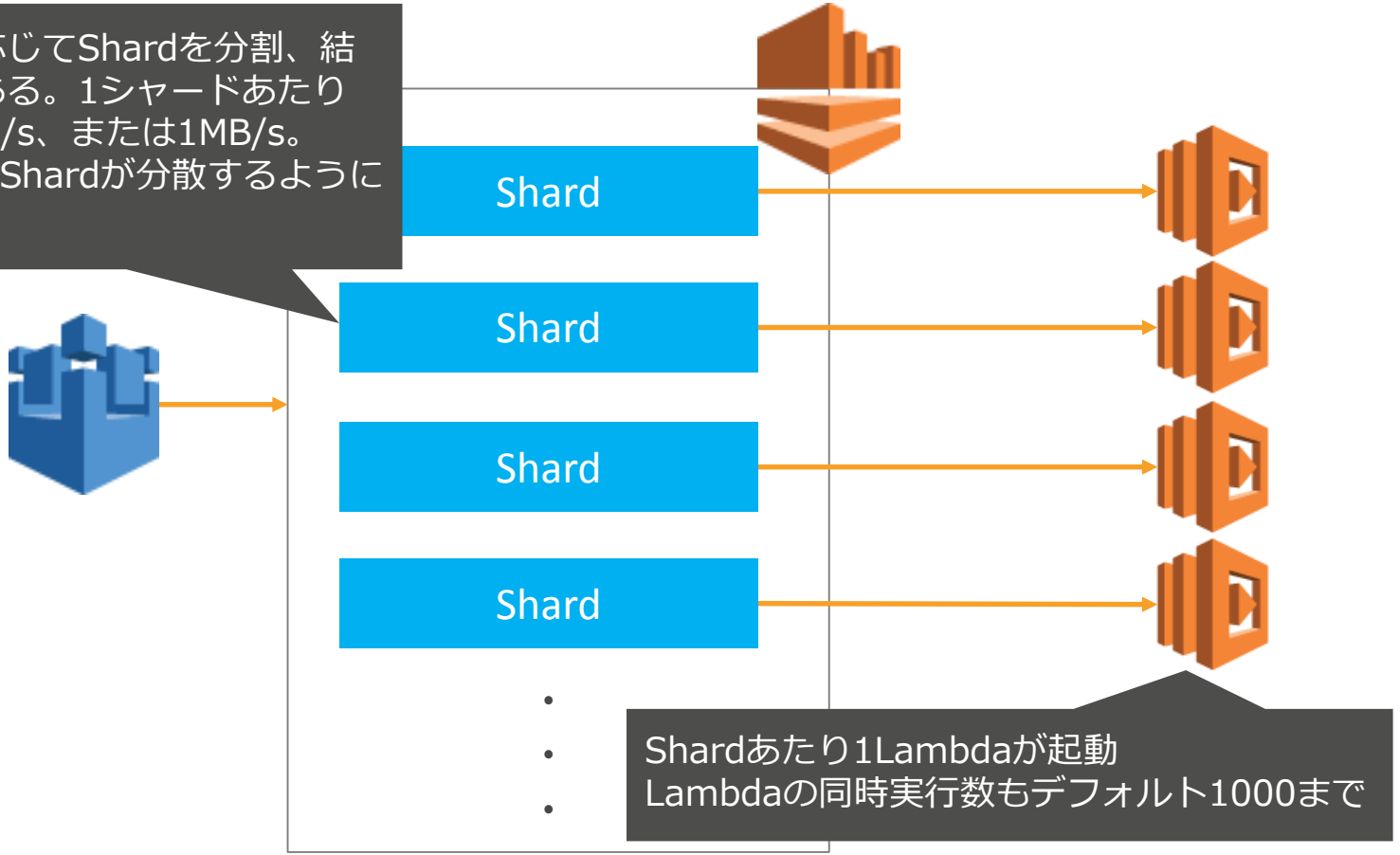
AWS IoTからS3、RDS、EC2へのサービスへの接続



- Kinesisを挟まないパターン: 同時イベント数＝Lambda同時起動数
- Kinesisを挟むパターン：Kinesis streamのshard数 = Lambda同時起動数

Kinesis利用上の注意

データ容量に応じてShardを分割、結合する必要がある。1シャードあたり1,000レコード/s、または1MB/s。Primary keyはShardが分散するように設計。



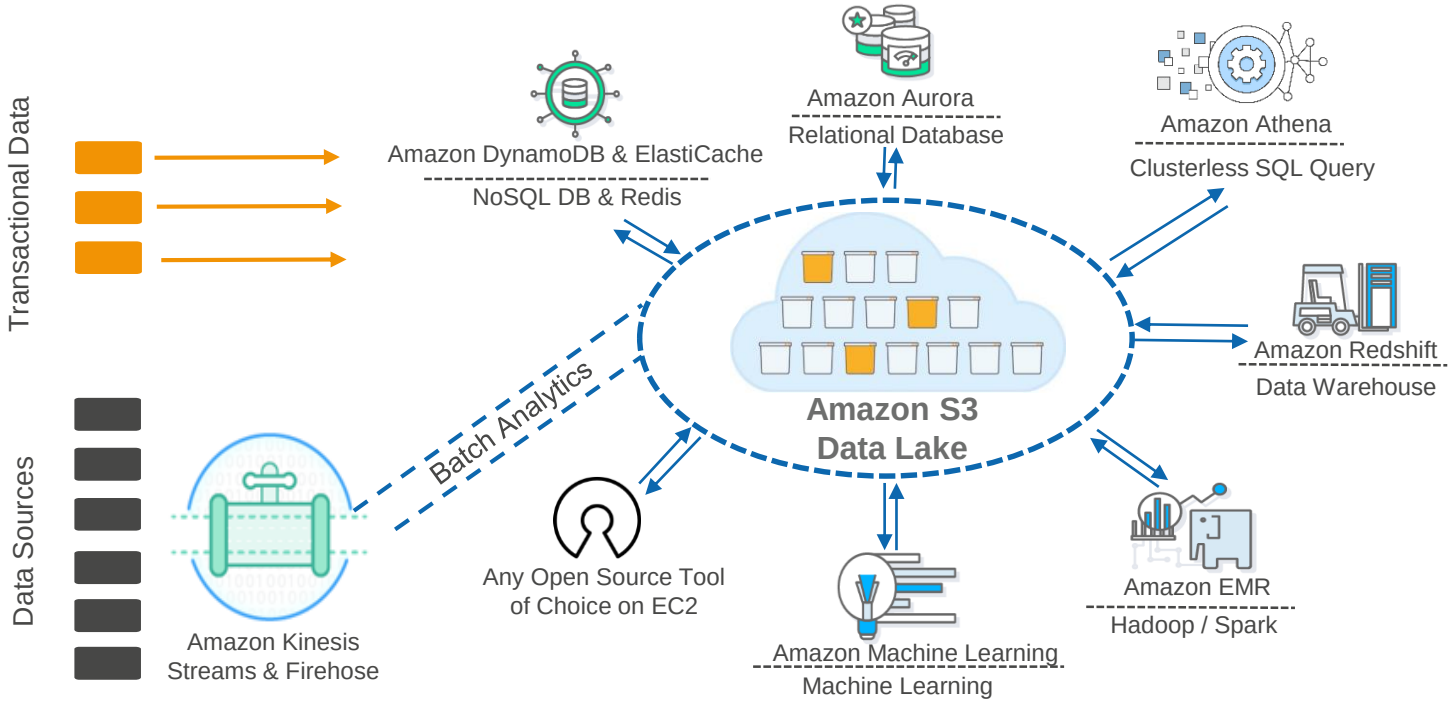
Shardあたり1Lambdaが起動
Lambdaの同時実行数もデフォルト1000まで

データ送受信(Ingest)



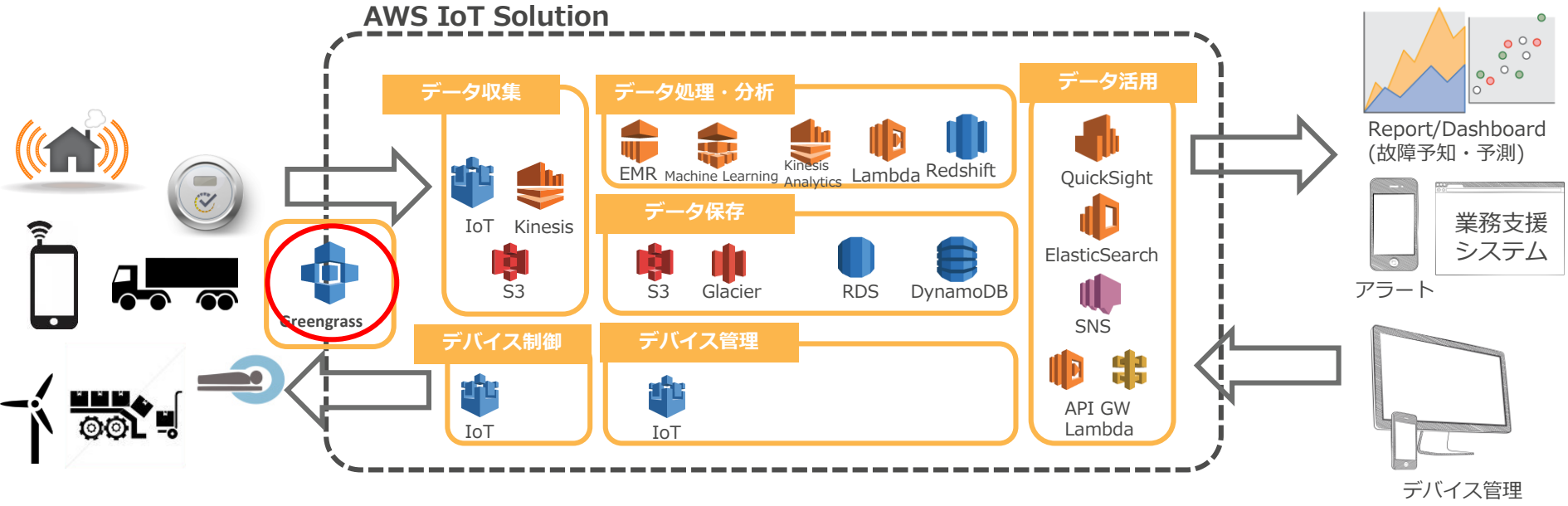
	AWS IoT	Amazon Kinesis	Amazon S3
ペイロード長	1メッセージあたり128KB	1レコードあたり1MB	1オブジェクトあたり5TB
プロトコル	MQTTS/HTTPS /Websocket	HTTPS	HTTPS
料金	512Byteを1メッセージとして $8/百万メッセージ	1シャード$0.0195/時 ペイロードユニットを25KB として$0.0215/百万ペイロード(Streams)	$0.025/GB(最初の50TB, 月)
認証	クライアント証明書 SigV4	SigV4	SigV4
使いどころ	ペイロードが小さく送信頻度も低い デバイスとクラウド間の双方向通信が必要 高いセキュリティを求められる	ペイロードが大きく、送信頻度が高い	メディアなどデータサイズが大きい場合、ファイル単位でデータを扱う場合

Amazon S3を中心としたデータレイクを構成

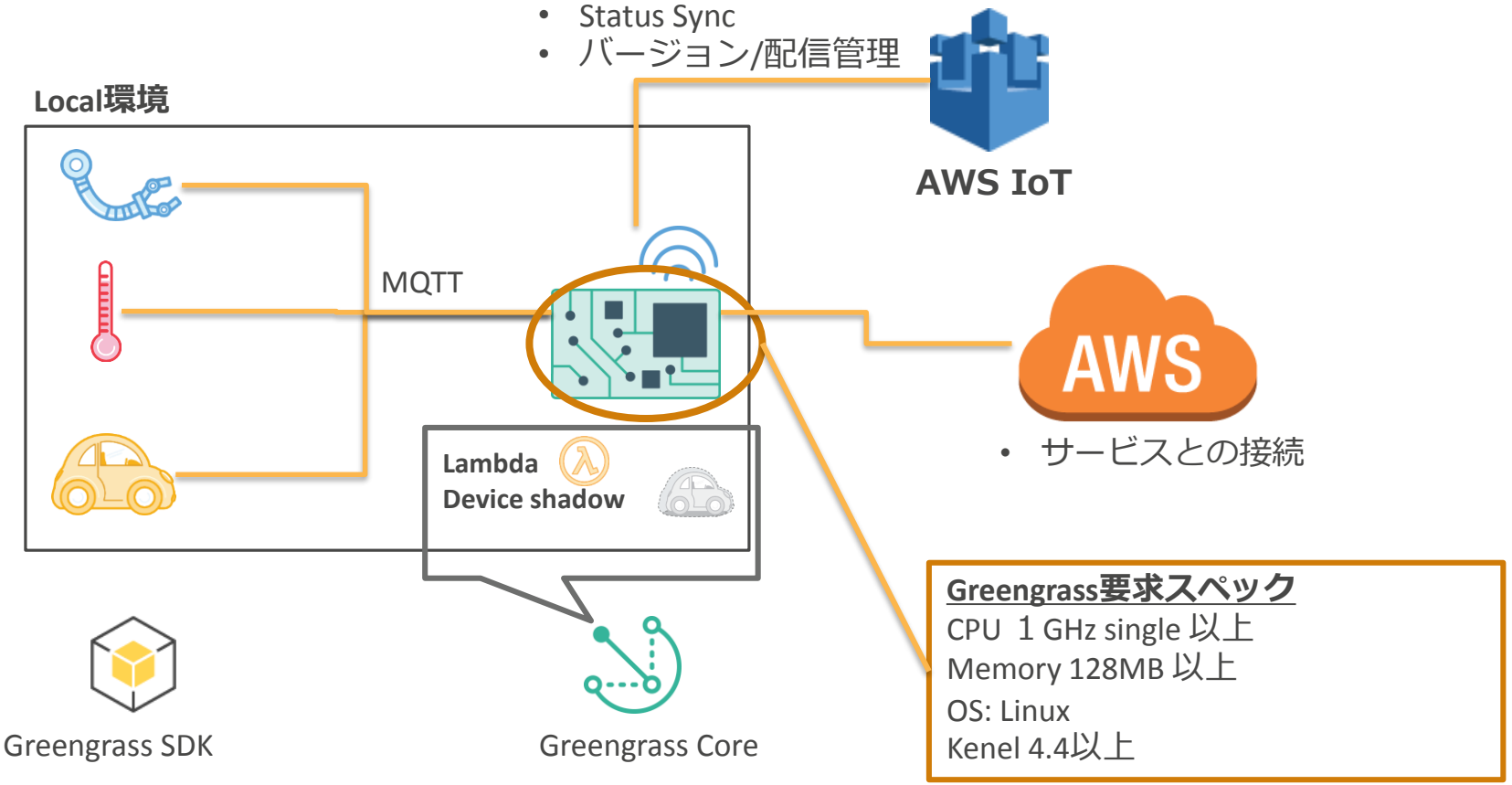


エッジ コンピューティング

AWS IoT Solution



AWS Greengrassのオーバービュー



AWS GreenGrass

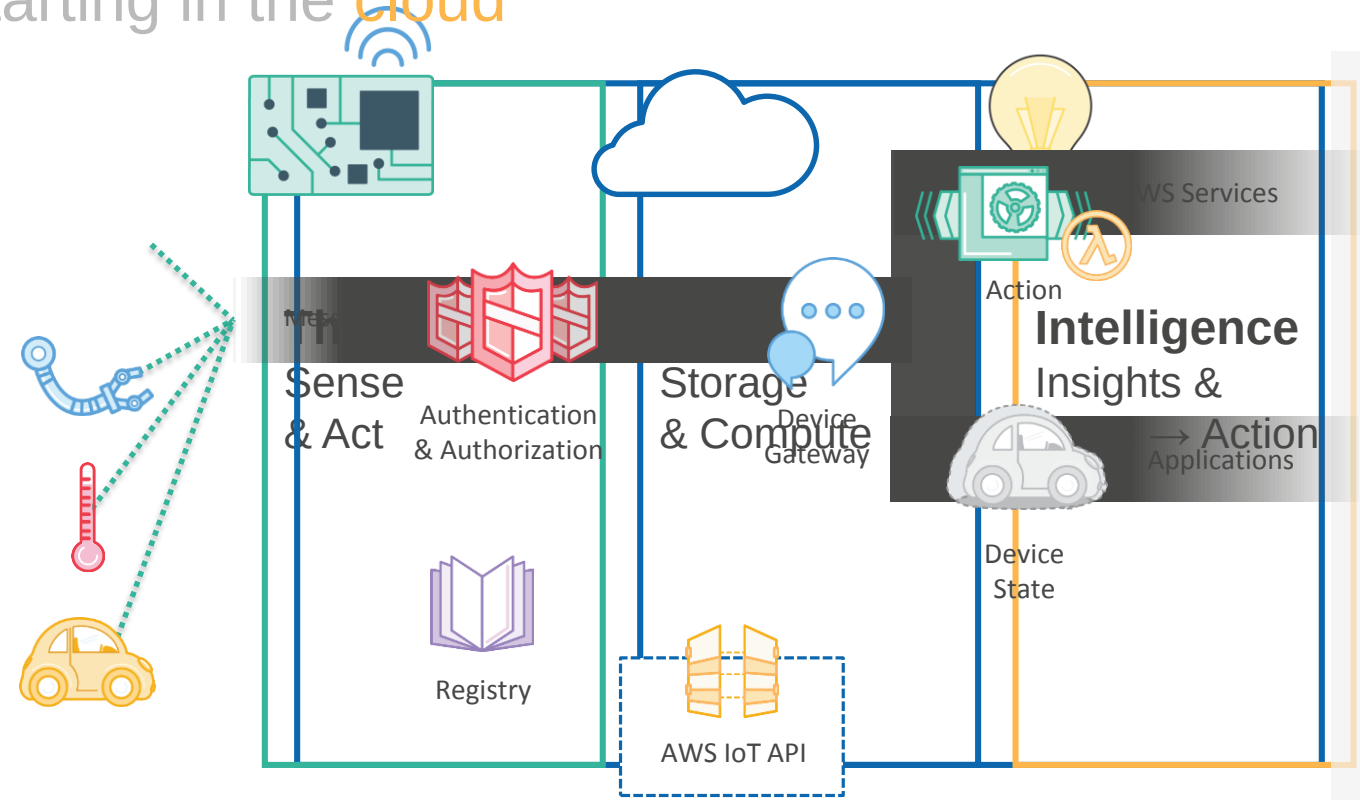
ローカル環境にある*Gateway*などの機器に、クラウドで開発したのモジュールを配布/管理可能かつ、ローカルで実行することにより、

- *Latency*の向上
- ネットワーク常時接続性が不要
- センシティブ情報の処理し、クラウドへデータ送信可能にできる

リリース当初の*Lambda*は*python2.7*をサポート予定

AWS IoT

Starting in the cloud

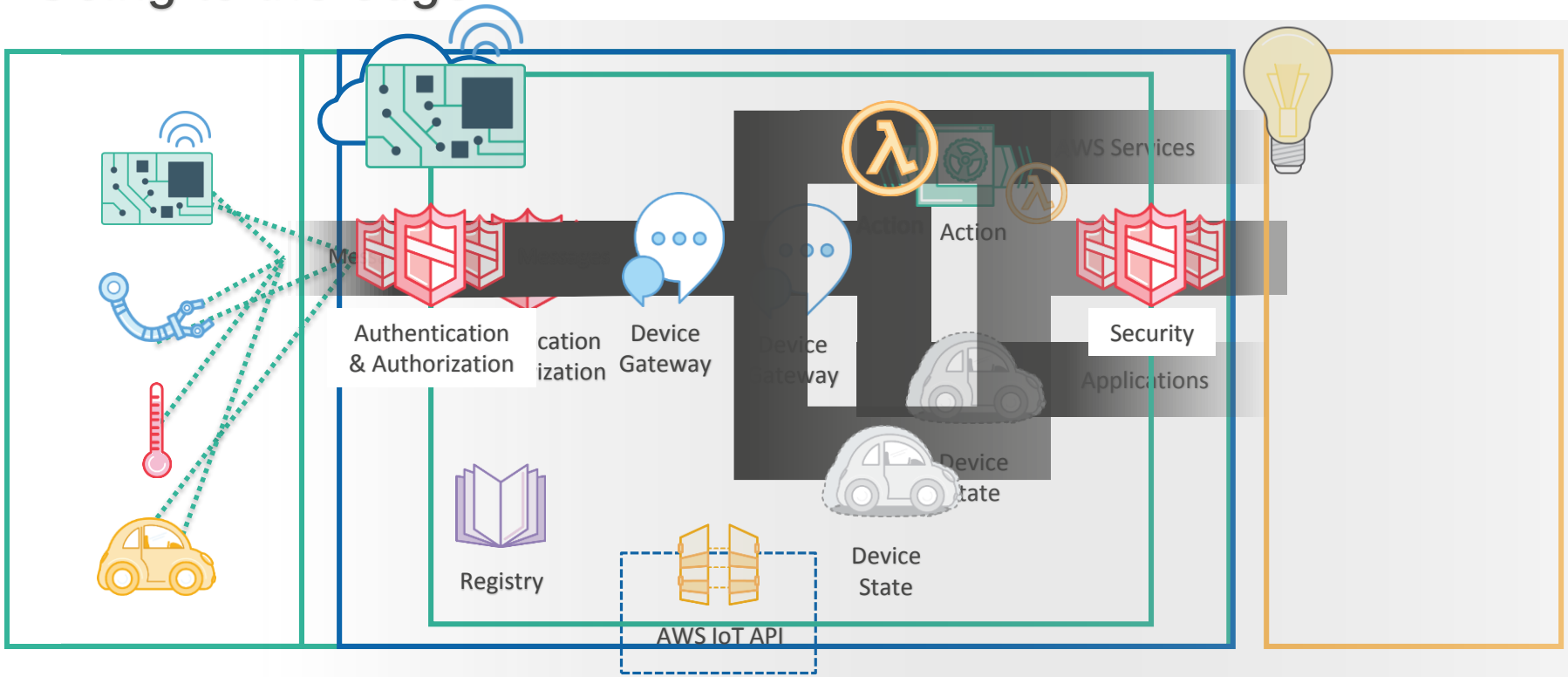


AWS Greengrass

AWS IoT

Going to the edge

Introducing AWS Greengrass



*Note: Greengrass is NOT Hardware (You bring your own)

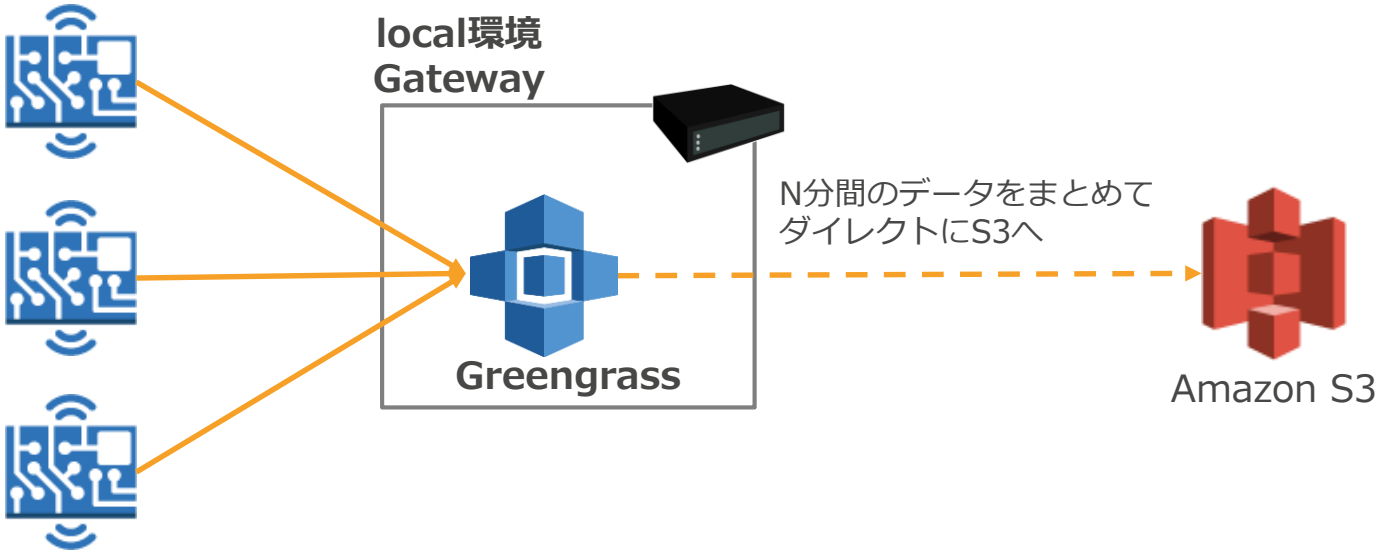
エッジ コンピューティング デザインパターン

課題

設置場所にNWがなく、SIMの電波もとどかない
Cloud経由のLatencyではユースケース上、遅い

データアグリゲーションパターン

対象シーン	- 通信の数を減らし、データをバルクでアップロードしたい - 常時ネットワークコネクティビティがない環境でのデータ収集
条件	- 特になし
注意点	- リアルタイム性が落ちる。リアルタイムの異常検知が必要な場合は Greengrass上とローカル環境で実装

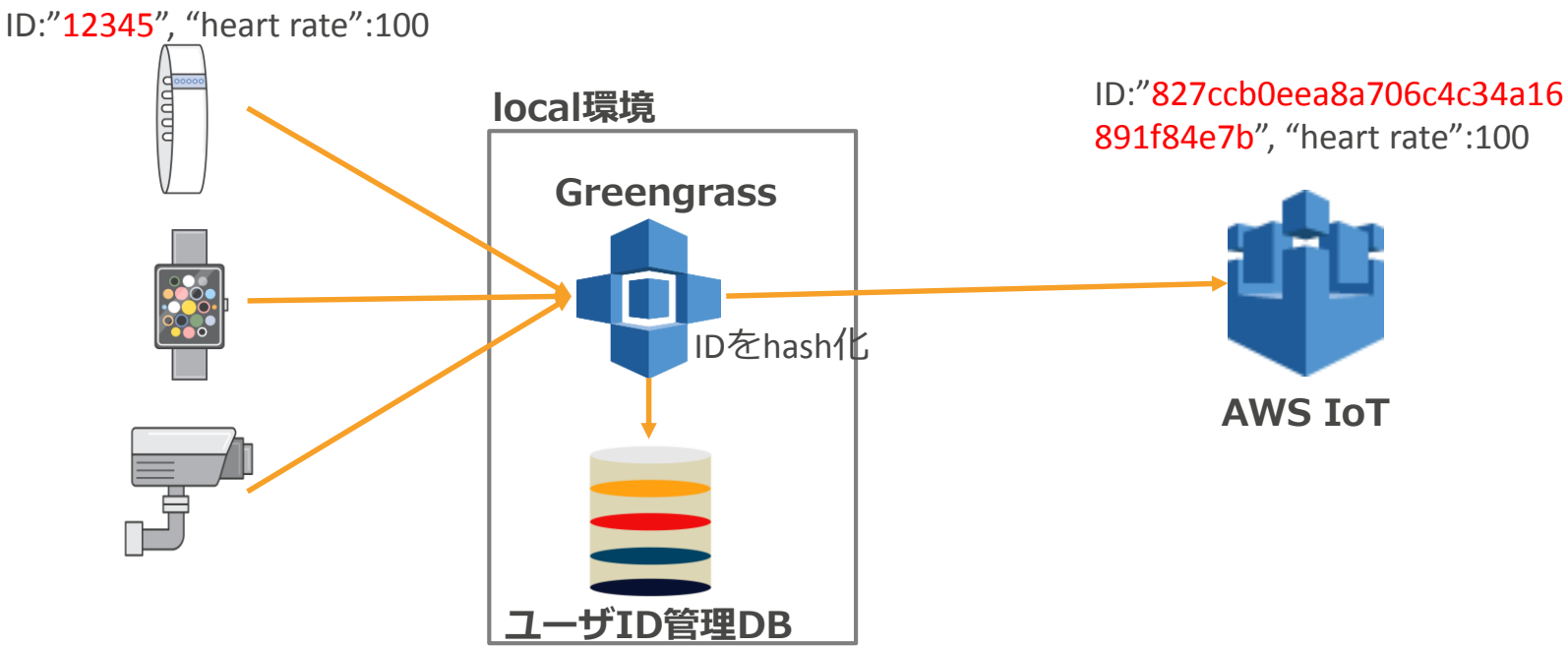


課題

データペイロードの一部にセンシティブ情報が含まれているのでクラウドを使いにくい。

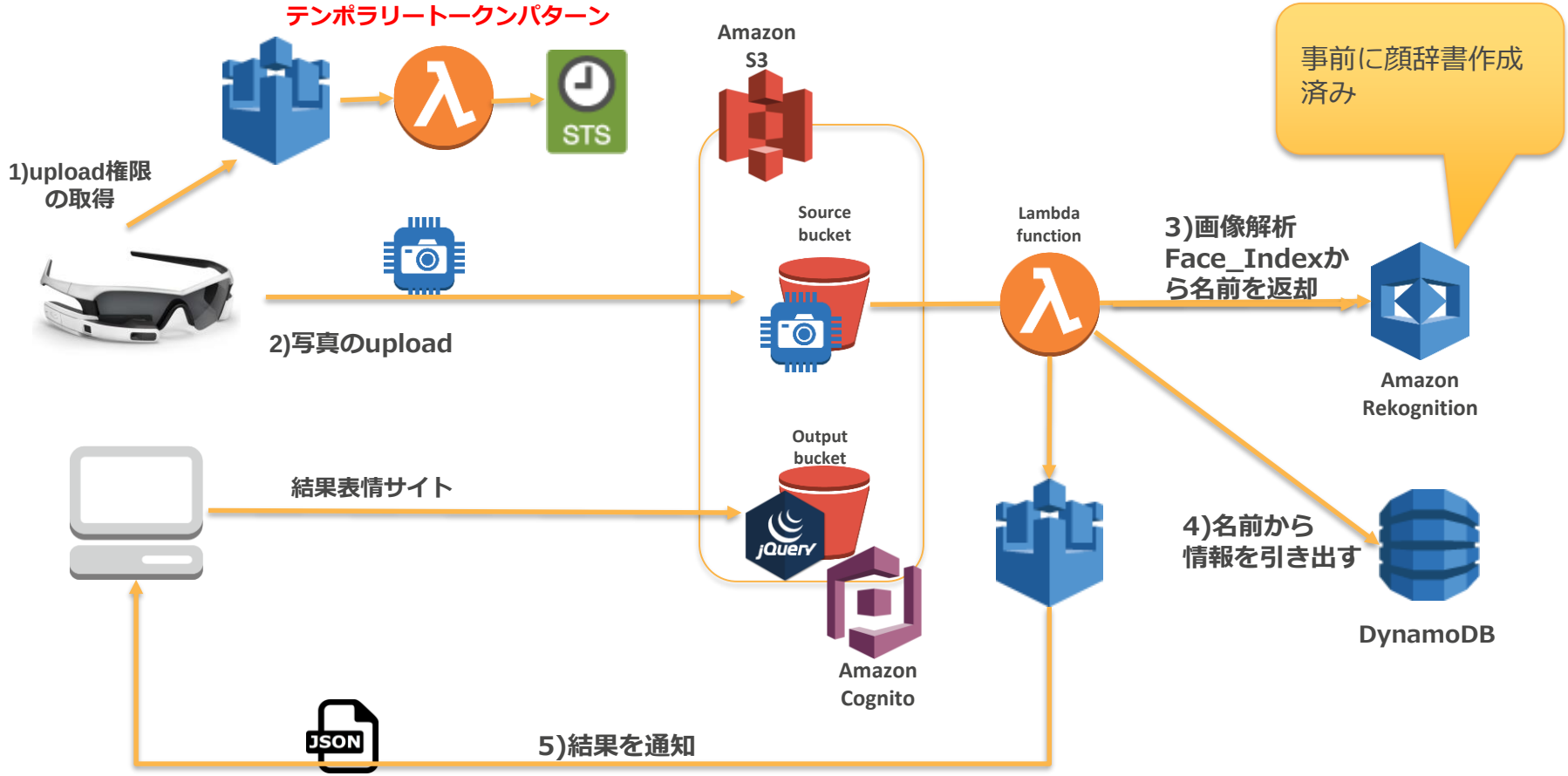
センシティブ情報マスキングパターン

対象シーン	- クラウドでのデータ活用がしたいがセンシティブ情報が含まれているためにクラウドへデータが送れないケース(社内規定などで)
条件	- 特になし
注意点	- maskingされた情報や削除した情報の管理はユーザサイドでの検討が必要



デザインパターンを使ったデモ

テンポラリートークンパターンでのデモ



まとめ

- AWS IoTの機能を理解する
  - ルールエンジン / デイバスシャドウ /認証機能を有効利用し、クイックなIoT環境の構築が可能
- AWS GreenGrassの機能を理解する
  - ネットワークコネクティビティがない、や、センシティブ情報を含むために利用しにくかったIoTユースケースに対応できるようになる
- AWS を利用したIoTデザインパターンを理解する
  - メリット、注意点を理解した上で利用をする

AWS

SUMMIT

Thank you

amazon web services