IoTプラットフォームを進化させるAWSの活用方法

IoT@Loft #20

Amazon Web Services Japan

概要

- デバイスの情報を集約・活用することで、エンドユーザーのみならず様々なステークホルダーがメリットを享受できます。
- このセッションでは、デバイスとデータの統合管理をおこなうIoTプラットフォームを構築することによって実現可能なユースケースの例や、構築するために解決すべき技術的課題、AWSを用いた解決方法について、事例を交えてご紹介します。

製品IoTにおける主なユースケース

遠隔操作

ブラウザやスマホからデバイスを遠隔操作

スマホで通知

デバイスの状態に変化があったときに、スマホなどに通知

機器間の連携

Echoデバイスからの操作に対応

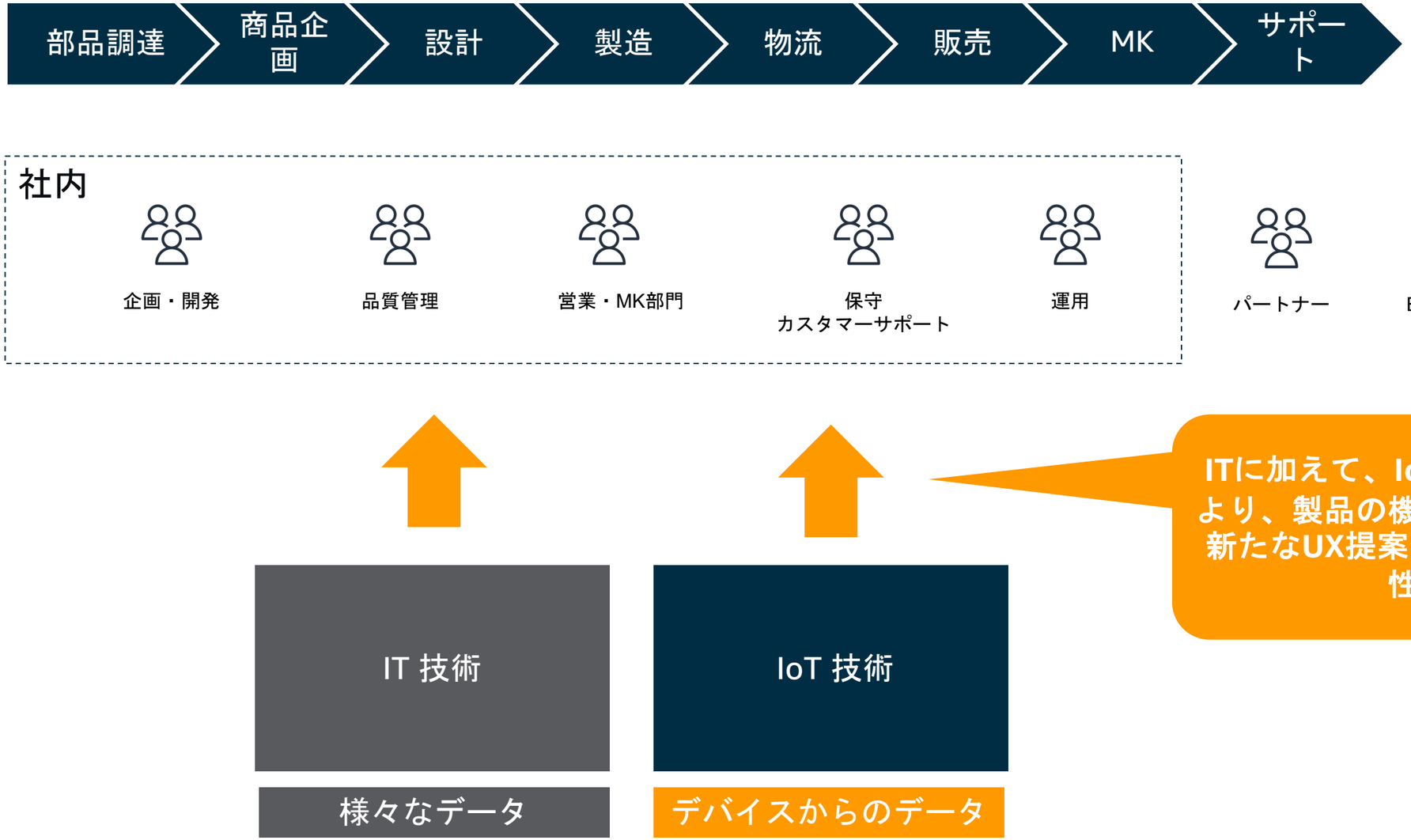
クラウド、デバイス、ユーザー間の処理の流れと要件

- 利用前
  - デバイス証明書のプロビジョニング
  - Wi-Fiセットアップ
- 利用中
  - スマホ操作でデバイス制御
  - デバイス状態をスマホで通知
  - 新機能のアップデート
  - 分析のためのデータ収集

IoTプラットフォームをつくる1つの理由：

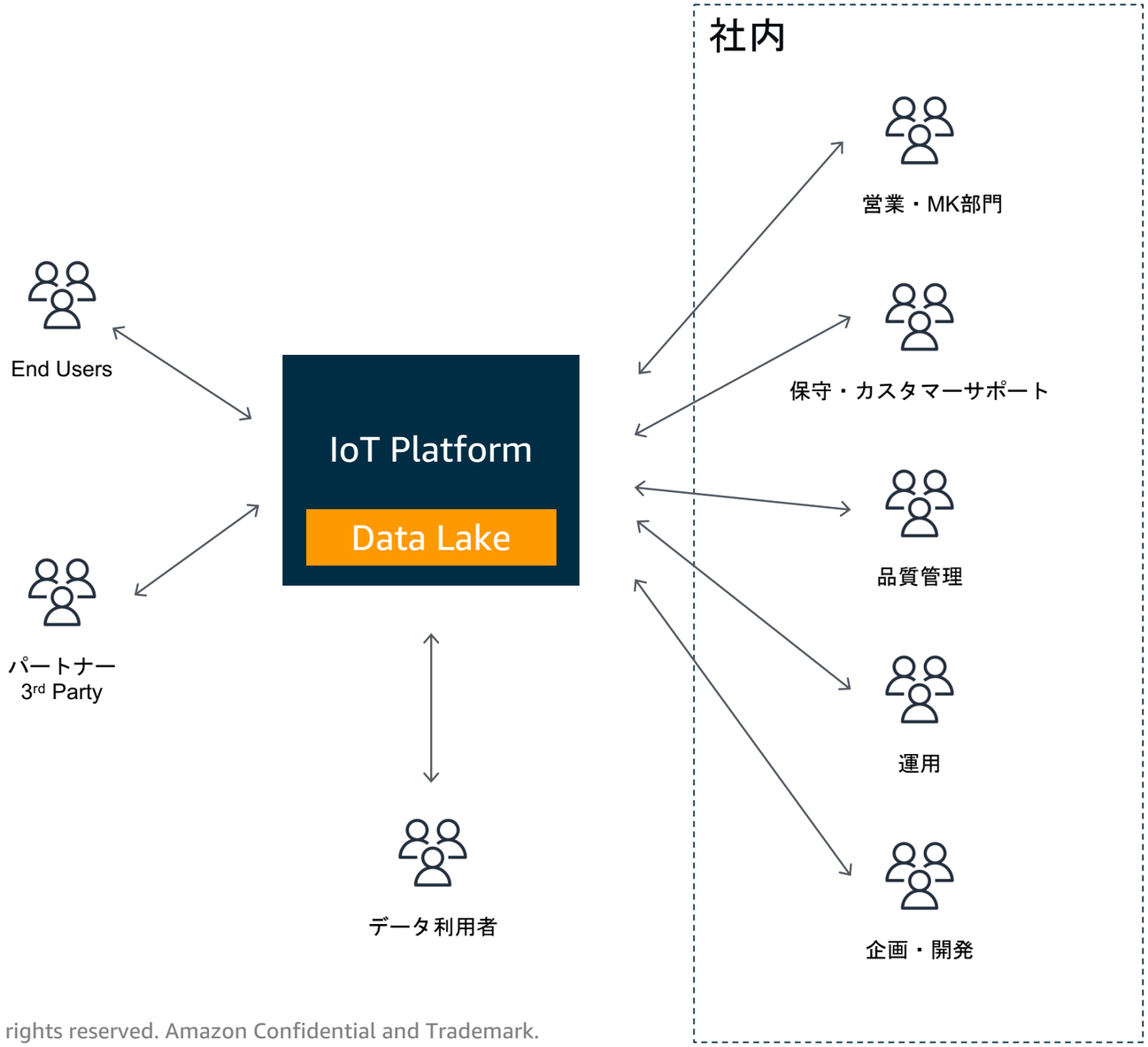
処理を共通化し、UX向上＆開発効率化

製品開発プロセスとIoTデータのさらなる活用



Smart Productにおける

IoTプラットフォームのユースケース



Smart Productにおける

IoTプラットフォームのユースケース

シンプルなデバイス管理
- 様々な製品を1つのユーザアカウントで管理
- 製品の登録フローがすべての製品で同じ

デバイス間連携
- 製品の状態変化を、製品のビープ音だけでなく、スマホやAlexaで通知し、見逃さない
- お風呂の湧き上がりをテレビで通知　など

UX向上に向けた機能の拡充
- 位置情報を利用したポータブルデバイス紛失時のトラッキング
- 製品利用状況に基づいた家族の見守り

IoTデータを集積、分析、処理する基盤を作ることで、これまで埋もれていたデータを活用したさまざまなユースケースの実現

End Users

IoT Platform

Data Lake

パートナーの製品との連携
- 自社製品とパートナー製品を連携させることでデバイス間連携など、新たな付加価値を提供

パートナー
3rd Party

データの購入
- プラットフォームから得られるデータを利用した新たなサービスの展開

データ利用者

社内

営業・MK部門

クロスセル・アップセル
- Amazonでの消耗品の自動発注（Dash Replenishmentサービスとの連携）
- ユーザーの利用傾向に応じたアクセサリのレコメンド

問い合わせ数の削減
- デバイスの状態やログデータを元に製品の故障前にアラートを通知

問い合わせ対応時間の短縮
- デバイスの症状をオペレータ確認可能にすることで、ユーザーへのヒアリング不要に

保守・カスタマーサポート

品質管理

不良解析
- 不良解析に、製品ログデータを使用
- 傾向不良の早期発見による製品回収リスクの低減

運用

モニタリング
- 接続状況をダッシュボードで確認
- セキュリティ攻撃の監視・監査、検出時のアラート
- 複数のモデルを1つのダッシュボードでモニタ

企画・開発

利用状況の分析
- デバイスの利用状況やバグレポートをもとにした新機能の追加、バグ修正の優先度付け

継続的なアップデート
- 今後出てくるであろう新たな要求に対して、クイックに機能追加し、製品サービスの価値を高いレベルで維持

IoTプラットフォームの設計

製品

- 制御コマンド
- 仕向け
- 設定情報
- 操作ログ
- FW ver
- 利用回数
- デバッグログ
- クラッシュログ
- 位置
- …

- 制御コマンド
- ファームウェア
- ML推論モデル
- 設定変更

IoT Platform

データの収集・分析

データ収集

データ処理・分析

データ保存
<Data Lake>

データ活用

デバイス制御

デバイス管理

リモート制御

社内

営業・MK部門

クロスセル・アップセル

保守・カスタマーサポート

デバイス設定状況やログを確認し、スムーズな応対

品質管理

ログを活用した不良解析
MLを活用した予兆保全

運用

クラウド・デバイスの状態監視、セキュリティ監視・監査

企画・開発

バグの修正や機能追加の優先度を決定する指標
自動アップデート

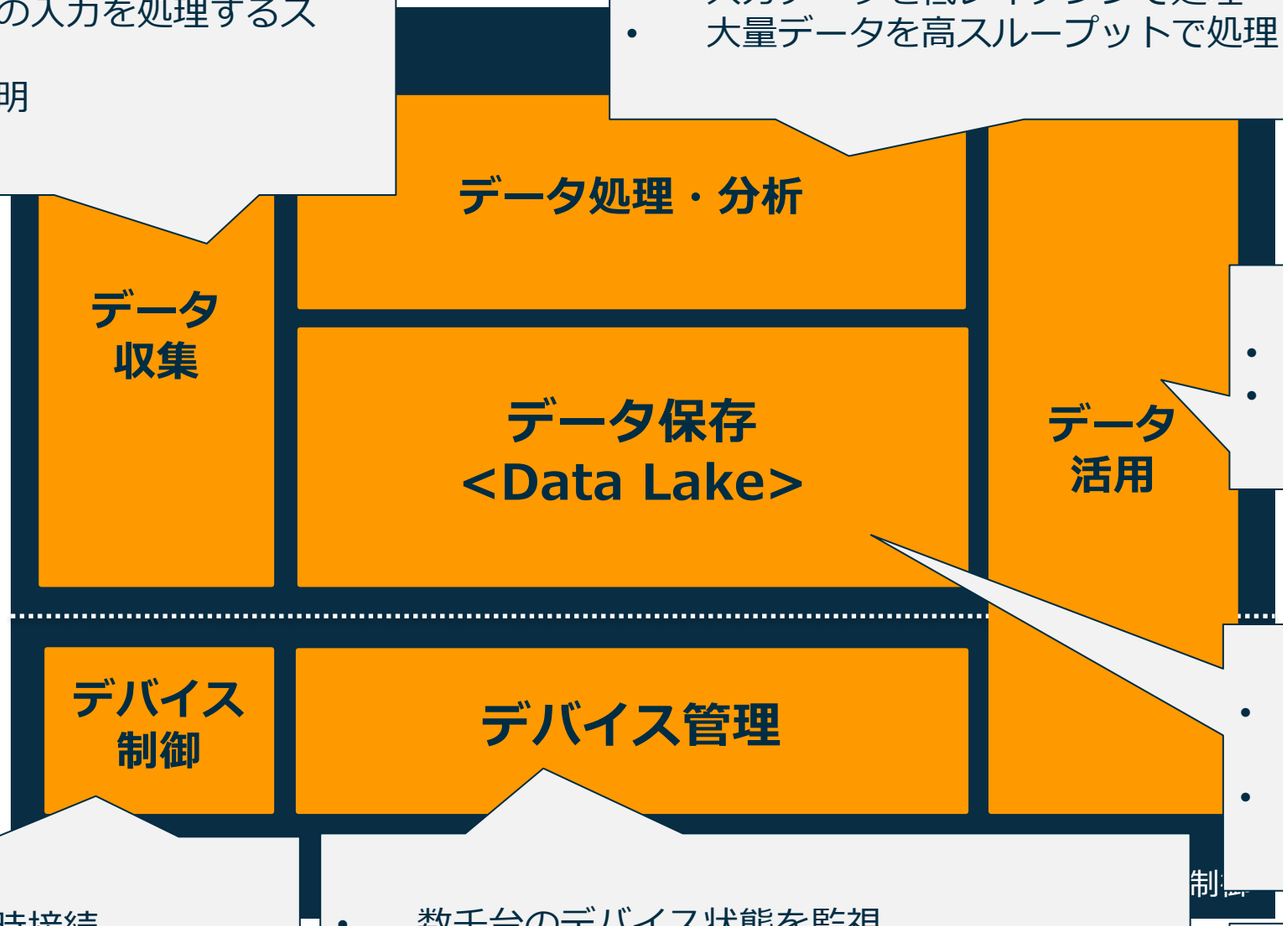
End User

感動体験、UX向上

IoTプラットフォームの設計課題

- 多数のデバイスからの入力を処理するスケーラビリティ
- デバイスの身元の証明

- 入力データを低レイテンシで処理
- 大量データを高スループットで処理



- リアルタイムで可視化、通知
- 数年のデータを可視化

- 過去数十年のデータを保存できるキャパシティ
- データのバックアップ

- 多数のデバイスを同時接続
- 低レイテンシ
- オフライン対応

- 数千台のデバイス状態を監視
- ファームウェア更新
- グループ管理

- 変わり続ける要求
- 品質
- 納期
- コスト

主な性能要件と実現方法

要件	AWS IoT 等を活用することによる実現
多数のデバイスからのアクセス	数百万台の同時接続可能なスケーラビリティ
デバイスの身元の証明	証明書による認証と認可、プロビジョニング
リアルタイム性	MQTTによる低レイテンシーでの双方向アクセス
不正アクセスなどの攻撃対策	デバイスの監視・監査
ストレージキャパシティ	S3との連携
ファームウェア更新・バッチ処理	IoTジョブの実行によるカナリアデプロイ
デバイスのモデル・バージョン管理	デバイスのグループ管理とクエリ
デバイスファームウェアの再利用性・移植性	Device SDKやFreeRTOSによるIFの統一
オフライン時の処理	デバイスの状態保持、状態差分の処理
コスト削減	使用した分だけの費用負担（従量課金）
変わり続ける要求	ビルディングブロックによる柔軟な機能追加・削除

IoT開発を行う上で直面する共通の課題をAWS が解決

IoTプラットフォームで利用するAWSサービス例

MCU

FreeRTOS

Embedded C SDK

MPU / Linux

AWS IoT Greengrass

Device SDKs
(Python, Javascript,
Java, etc…)

Echo

IoT Platform

データ収集

AWS IoT Core

データ処理・分析

Amazon Kinesis

AWS Lambda

Amazon SageMaker

データ保存

Amazon Simple Storage
Service (Amazon S3)

Amazon DynamoDB

デバイス制御

AWS IoT Core

デバイス管理

AWS IoT Device
Defender

AWS IoT Device
Management

データ活用

Amazon QuickSight

Amazon Connect

AWS Amplify

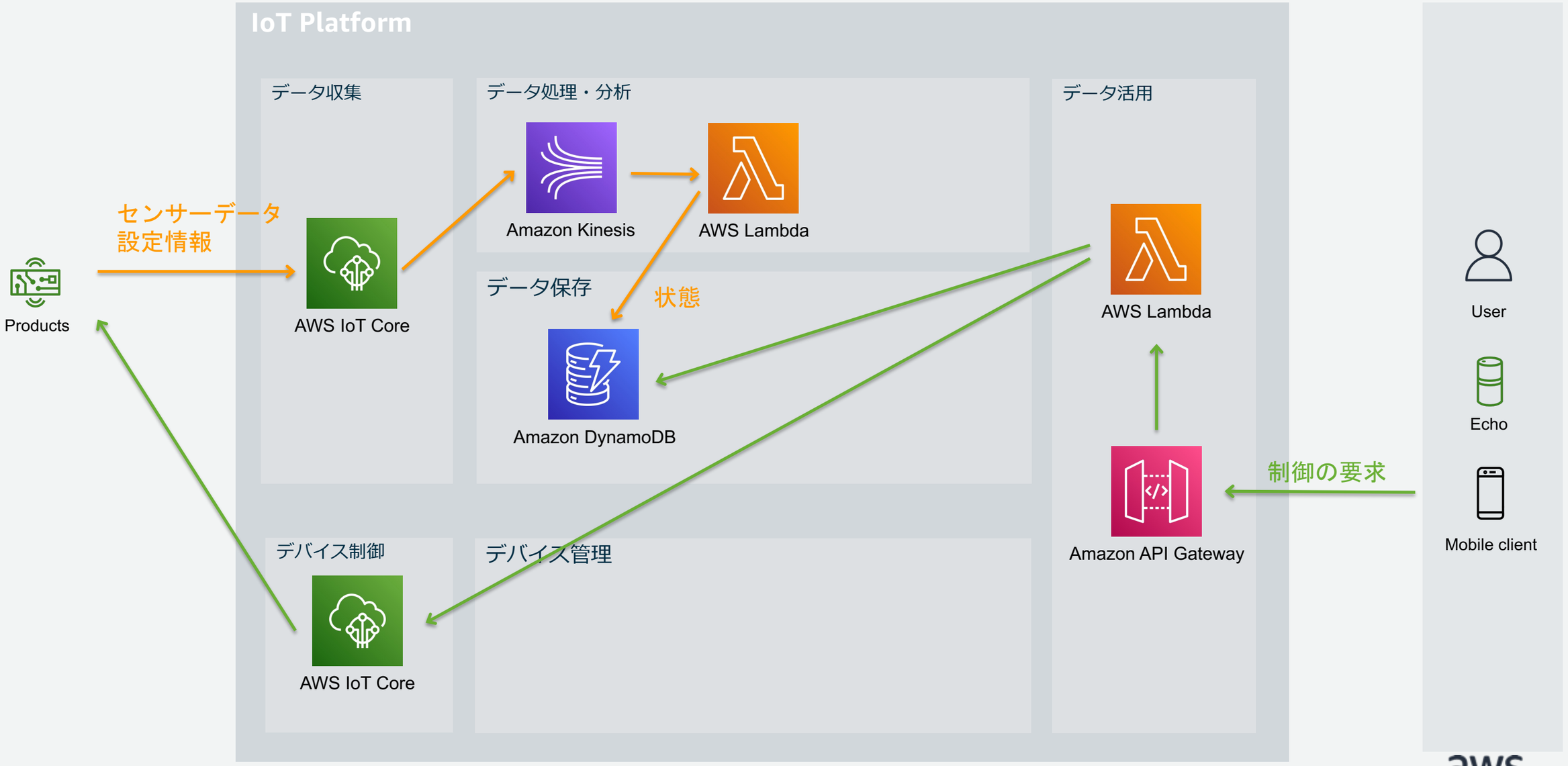
Amazon Simple Notification
Service (Amazon SNS)

Users

アーキテクチャの例

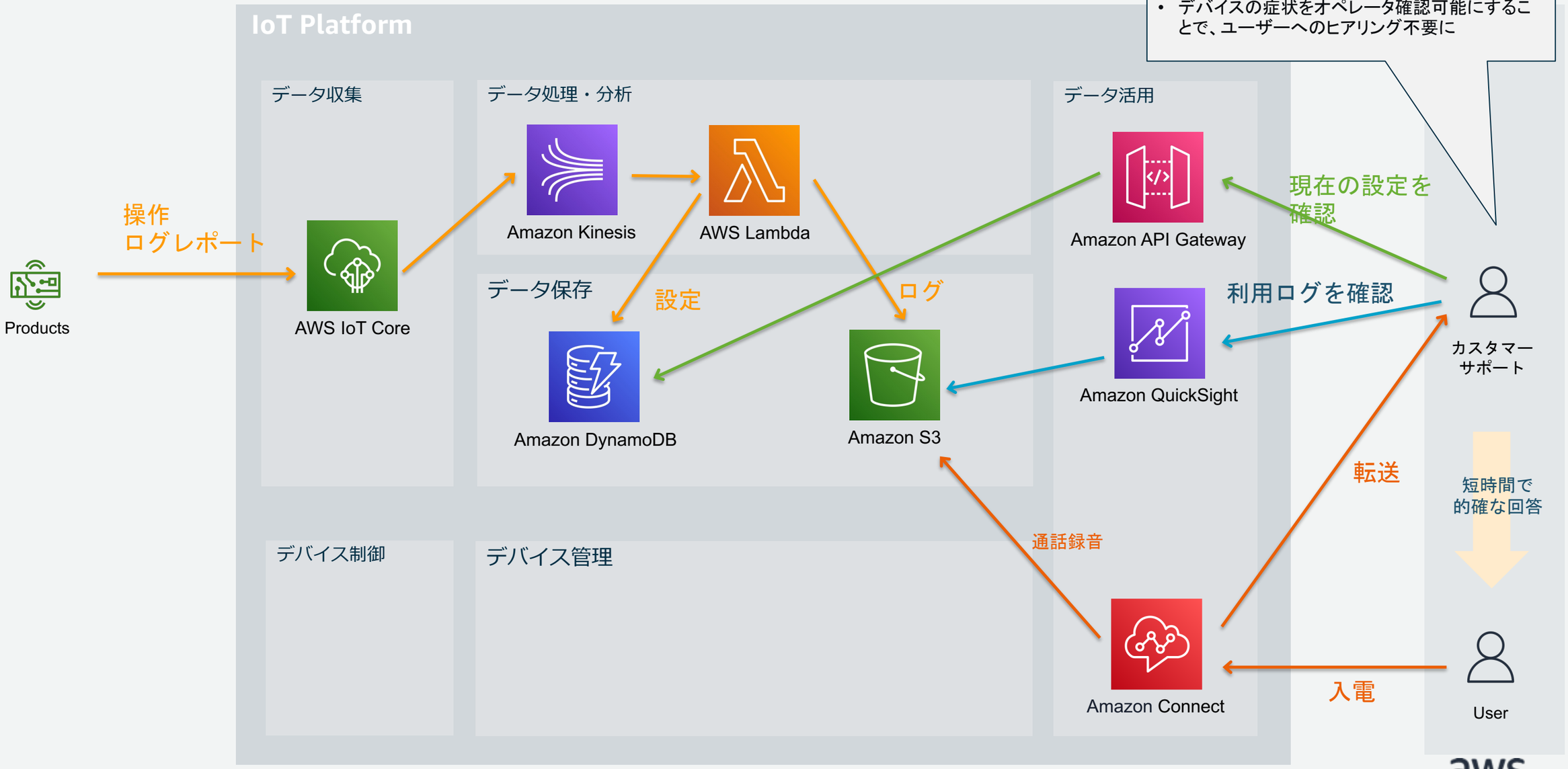
デバイスの遠隔制御

双方向通信を実現し、
スマートスピーカーやモバイルからの制御要求をデバイスに伝達



アーキテクチャの例

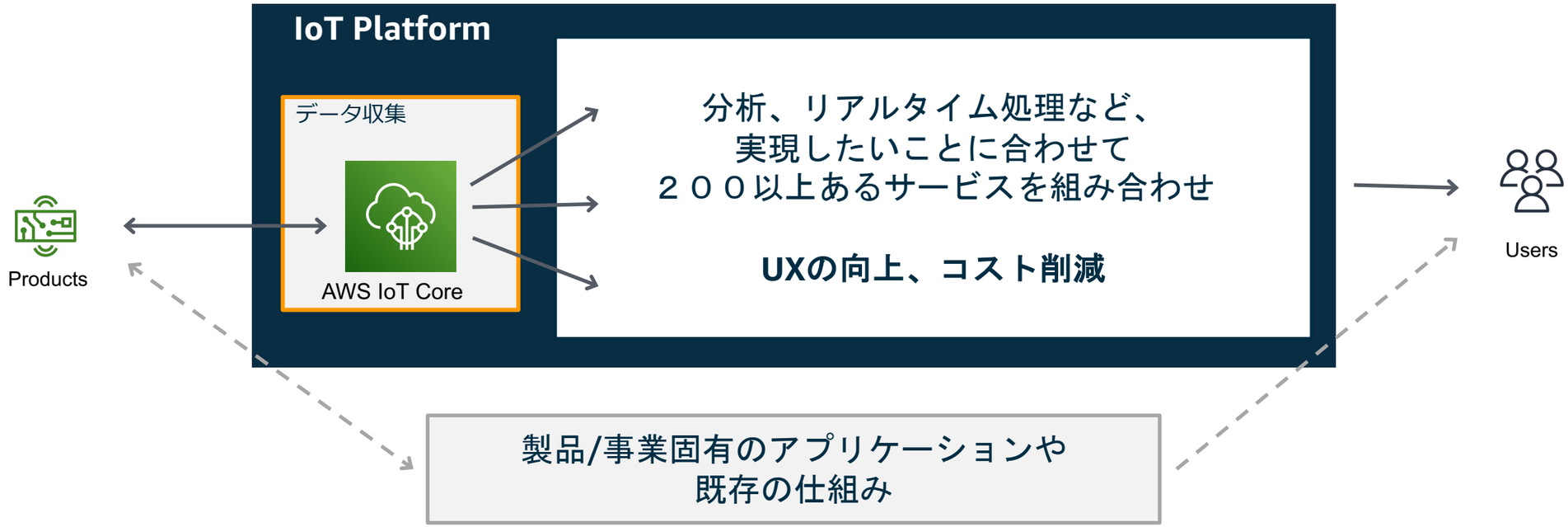
製品問い合わせ時のオペレーション効率化



※一部サービスの記載を省略しています

ユースケース実現に向けて

- 小さく、出来るところから試してみる
  - Building Blocks（必要なサービスを徐々に足していく）
  - 既存の仕組みを維持しながら試す
  - 効果を確認しながら機能を拡充



まとめ

- スマートプロダクトの統合管理による新たな顧客価値の創造とその実現方法
    - Smart Productの情報を集約・活用することによるUX改善の例
    - プラットフォーム構築するために解決すべき技術的課題
    - AWSを用いた解決方法