

コンテナ基盤を利用した IoTプラットフォームサービス開発について

2021年10月29日

プラットフォームサービス本部
データプラットフォームサービス部 開発オペレーション部門
栗原 良尚

目次

- 1. 自己紹介**
- 2. NTT Communicationsの IoTサービスのご紹介**
- 3. IoT Connect Gatewayの利用例**
- 4. IoT Connect Gatewayの開発課題**
- 5. IoT Connect GatewayのCI/CDの取り組み**
- 6. まとめ/今後の取り組み**

自己紹介 & 取り扱い説明書

■所属 NTTコミュニケーションズ株式会社

■名前 栗原 良尚 (Yoshinao Kurihara)

■経歴

2001/4 高等専門学校（高専） 電気工学科

2008/4 大学院大学 情報科学研究科

2010/4 NTTコミュニケーションズ株式会社 入社
2010/7 インターネットマルチフィード 出向
JPNAP/MFEED(AS7521)運用業務
RFEED(AS45686)等の運用システム開発

2013/7 NTTコミュニケーションズ IAC
SDN/NFV担当

2014/7 NTTコミュニケーションズ IAC/技術開発部
テストベット担当
ネットワーク制御担当・OOL試験自動化PJ(PO)
ネットワーク・セキュリティ担当

2020/4 NTTコミュニケーションズ PS本部DPS部開発Op
IoT Connect Gateway開発担当



一応 1児の父やってます (2歳)



Power Elect
Silicon

Network
Security

Infrastructure
Cloud/Orchestrator

Application
SaaS/FaaS

だんだん上位レイヤに (本人の意識とは無関係)



20歳頃



23歳: 内定式 (社員証)



35歳: 今ココ

体重も順調に増加

自己紹介 & 取り扱い説明書

休日にはまること

ICGWを使ったIoTデバイス&センサ開発

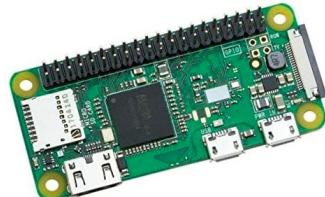
- ・開発プラットフォーム

- ・**RaspberryPi 4/zero/pico**

<https://jp.rs-online.com/web/generalDisplay.html?id=raspberrypi>



Pi4



Zero



Pico

- ・**M5 Stack/Stick**

<https://www.switch-science.com/catalog/6530/>



Stack Core2:



Stack Stick Plus:



Stack Core2 for AWS:



秋葉原を徘徊して色々電子パーツを
10万以上購入して嫁に怒られましたw

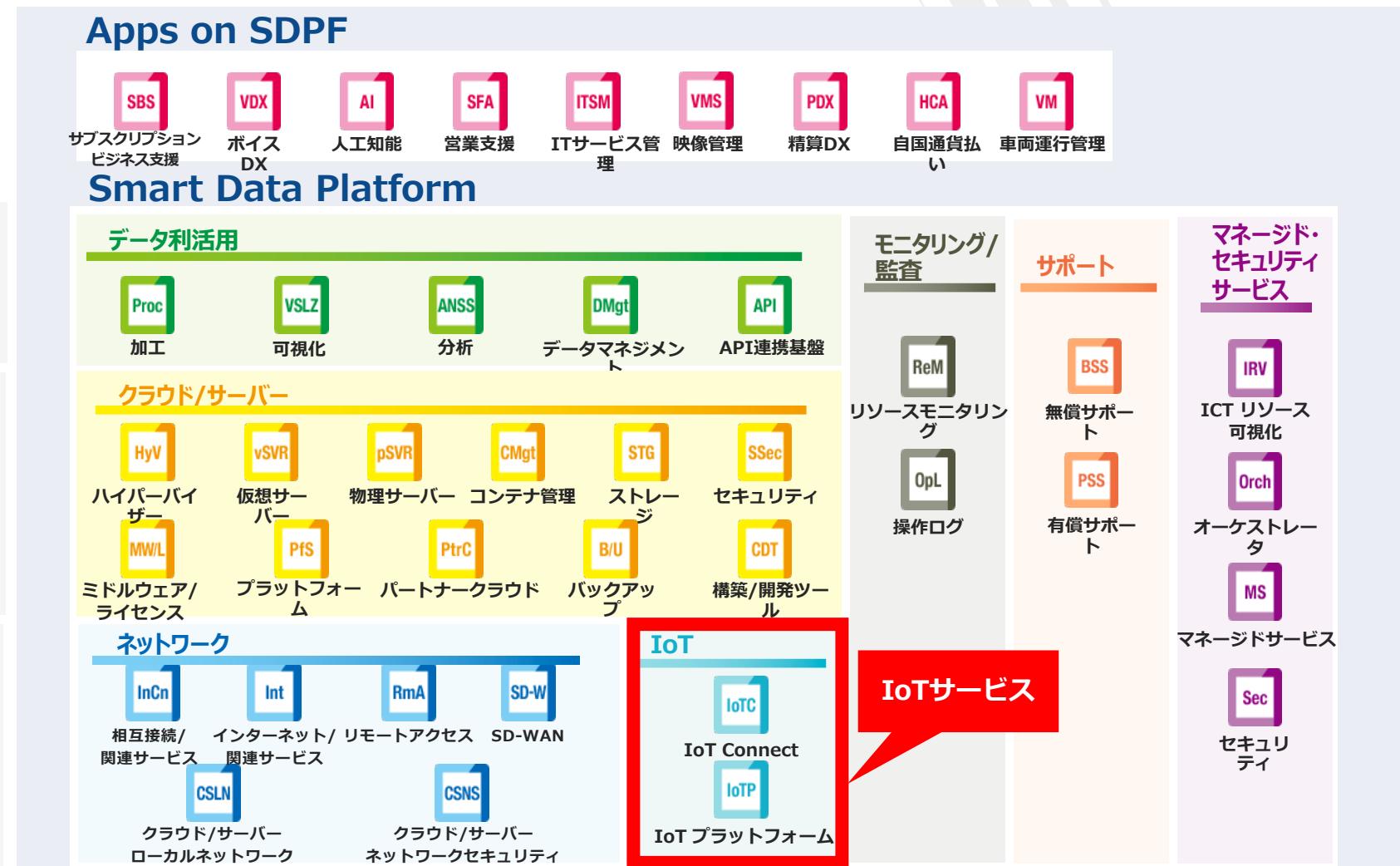
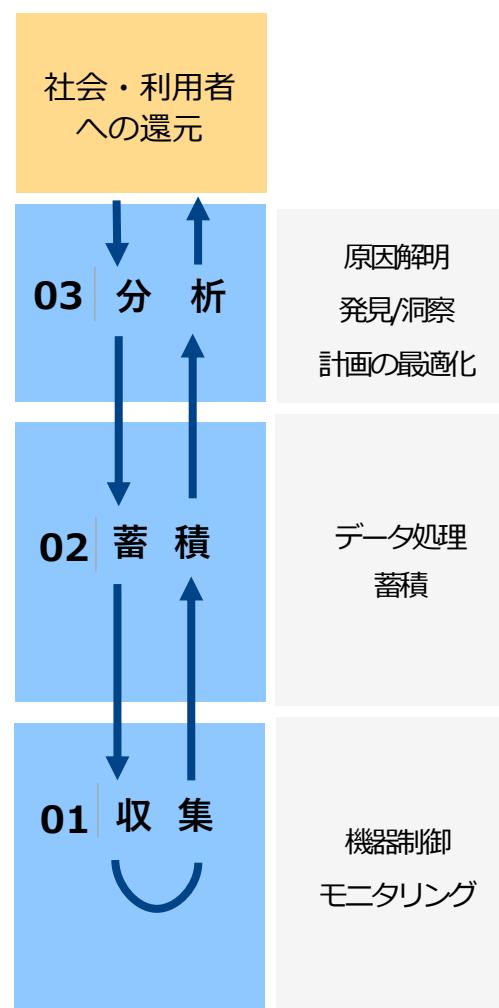
「IoT Connect Gatewayの利用者の立場に
立って使い勝手や課題試したい」

すいません、半分以上趣味ですw

NTT Communicationsの IoTサービスのご紹介

NTTコミュニケーションズのSmart Data Platform

Smart Data Platform(SDPF)はデータ利活用に必要な収集・蓄積・管理分析に関する機能をワンストップに提供し、お客様のDXを支援するプラットフォーム



NTTコミュニケーションズが提供するIoTサービス群

- **IoT Connect Mobile (ICMS)**

複数のeSIMプロファイルの中から利用地域によってプロファイルを選択し、グローバルなIoTビジネスを可能にする
IoT向けモバイルデータ通信サービス

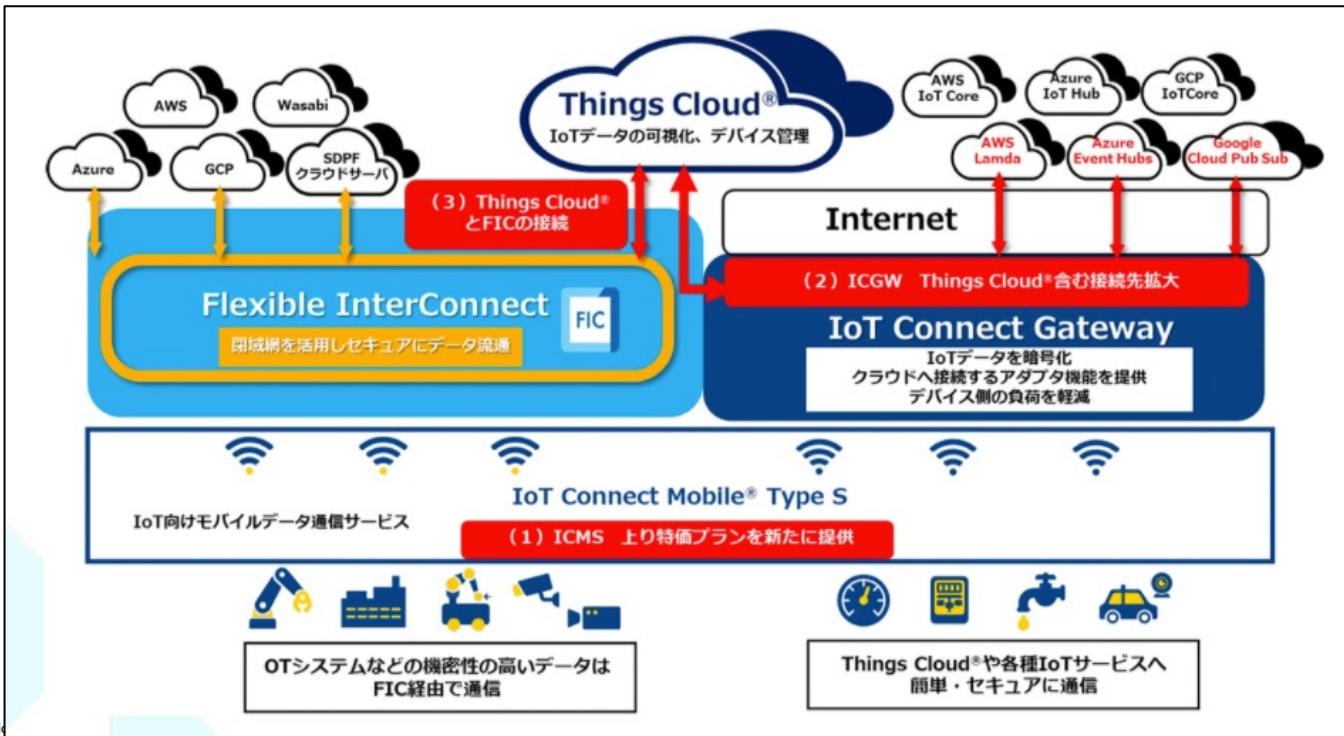
- **IoT Connect Gateway (ICGW)**

IoTデバイスの処理負荷、通信量、運用コストを削減し、各種クラウドサービスに簡単、セキュアに利用できる
IoT向けゲートウェイサービス

- **Things Cloud**

様々なセンサー/デバイス接続からのデータ収集、可視化、分析、管理などIoTに必要な機能が
パッケージ化されたIoTプラットフォーム

■ サービス組み合わせ概要

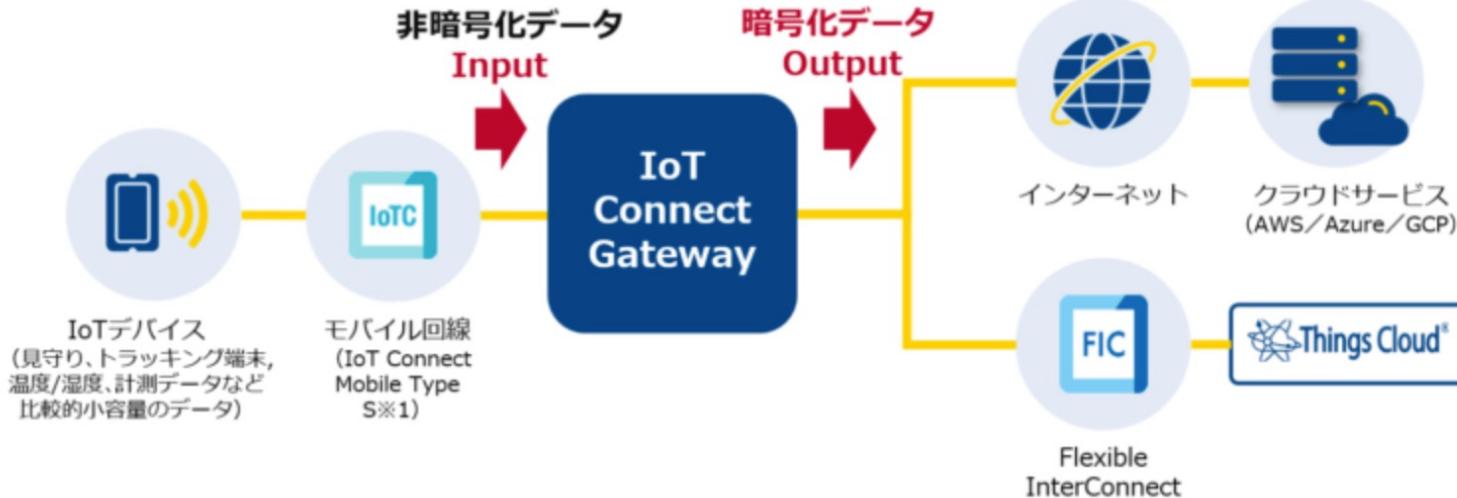


■ ICGW対応クラウドサービス一覧

メニュー	接続先クラウドサービス
スタンダード	Google Cloud IoT Core, AWS IoT Core, Azure IoT Hub, Things Cloud
汎用HTTP/HTTPSサーバ	
イベント	Google Cloud Pub/Sub, Azure Event Hubs
ファンクション	AWS Lambda

IoT Connect Gatewayとは

IoTデバイスの処理負荷やデータ量を気にすることなく、クラウド側のインターフェース仕様に合わせて簡単・セキュアな接続を実現するサービスです。「プロトコル変換機能」「クラウドアダプタ機能」の2つの機能をセットで提供

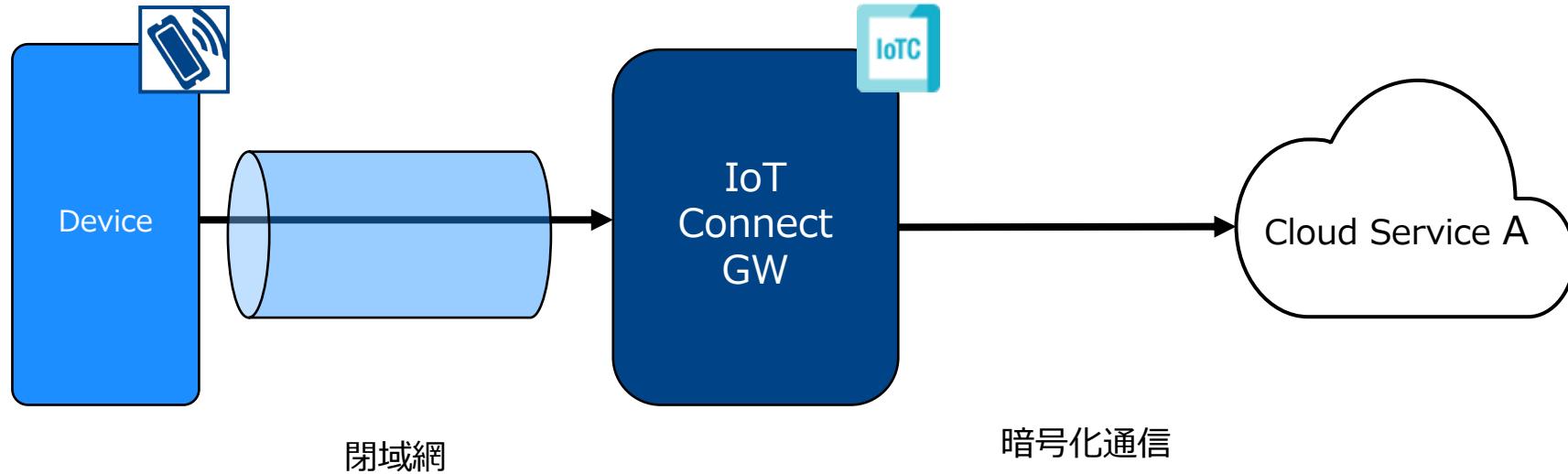


IoT Connect Gateway 対応メニュー／接続先サービス		
メニュー	①プロトコル変換機能	②クラウドアダプタ機能
スタンダード	HTTP→HTTPS MQTT→MQTT	Google Cloud IoT Core、AWS IoT Core、Azure IoT Hub Things Cloud
	HTTP→HTTPS	汎用HTTP/HTTPSサーバ
イベント	HTTP→HTTPS	Google Cloud Pub/Sub、Azure Event Hubs
ファンクション	HTTP→HTTPS	AWS Lambda

赤字 : 10/18リリース機能

IoT Connect Gatewayが解決する課題① ~通信量、端末オーバヘッドの削減~

IoT端末から直接暗号化をおこないクラウド側にデータ送信を行う場合、端末コスト、通信量、消費電力が課題になる場合がある



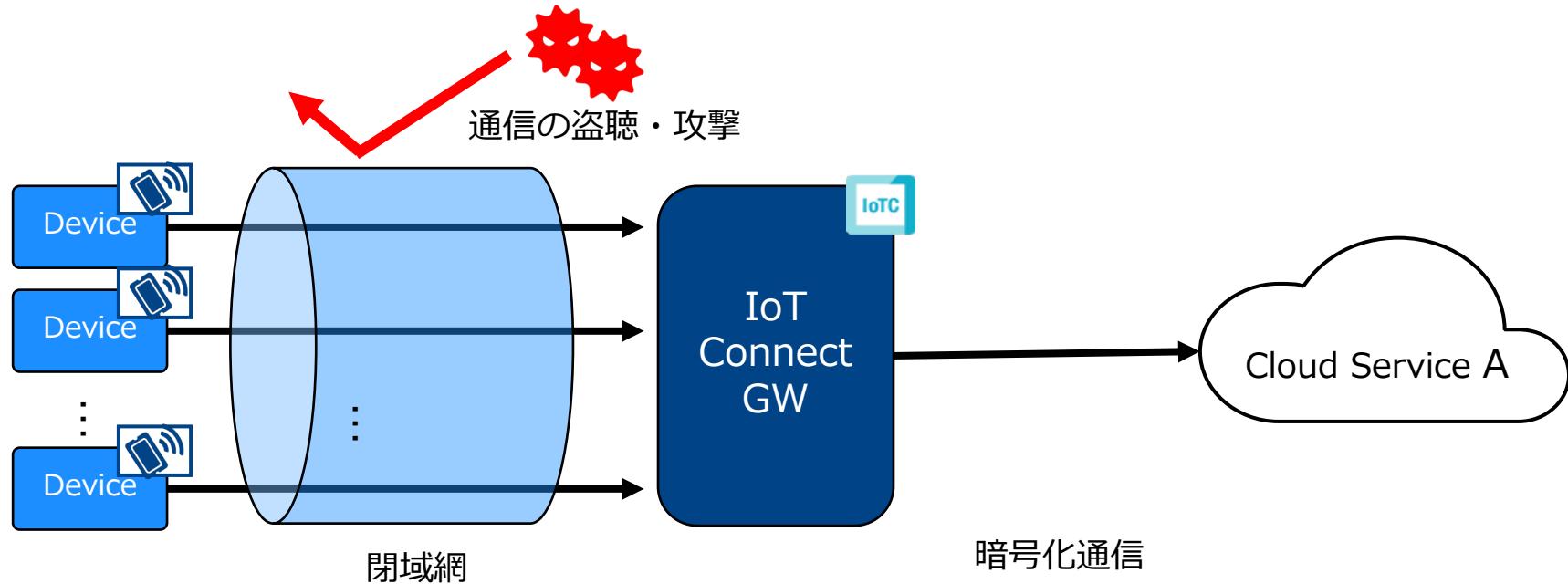
ICGWを使うことで、端末コスト、通信容量を削減可能に

- ・端末側で暗号化処理をICGWにオフロードすることで、端末コスト、通信量を削減を実現
- ・暗号化チップ、通信量も少なくなるので、デバイスの消費電力化も実現
- ・クラウドまでの通信はICGWが暗号化を行うことでセキュリティを担保

IoT Connect Gatewayが解決する課題② ~運用コスト、セキュリティ対策~

IoT端末はセキュリティの課題の検討が重要になるが、遠隔からのファームアップデートやセキュリティパッチ対策が困難な場合がある

- ・安価、大量の端末を様々な場所に配布するIoTサービスなど
- ・暗号化チップ、遠隔アップデート機能を追加した場合、端末コストが高価になってしまう



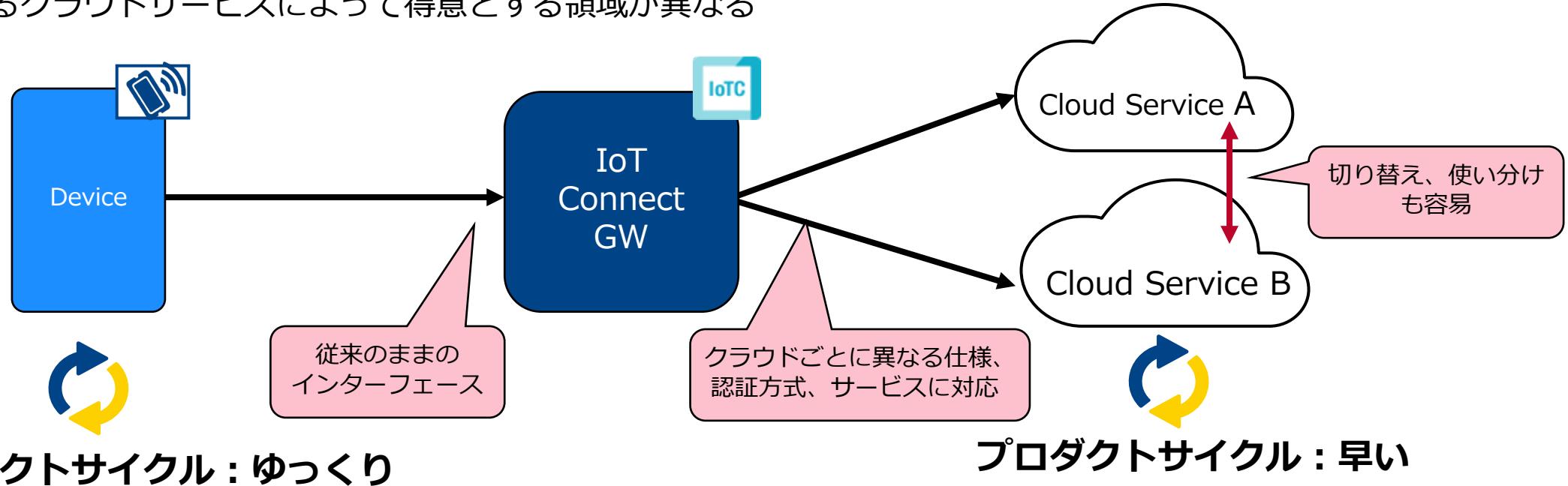
ICGWを使うことで、簡単にセキュリティ対策、運用コスト削減が可能

- ・クラウド側の認証方式を変更する場合でも、デバイスの鍵をすべて変更する必要なく、ICGWのポータルで一括して鍵変更対応が可能
- ・端末からICGWの間は閉域通信となっているため、外部からの盗聴、攻撃のリスクが少ない

IoT Connect Gatewayが解決する課題③ ~新機能追加の柔軟性~

これまででは、IoTサービスを開発する場合、端末側からクラウドを通じてサービス設計を検討する必要があった

- ・端末、センサ側は頻繁な頻繁なアップデートが困難な場合が多い
- ・クラウド側は便利な新サービス、プラットフォームをスピーディーに機能拡充、変更が発生
- ・利用するクラウドサービスによって得意とする領域が異なる



ICGWが仲介することで、端末の開発コストをかけず新機能追加、選択の自由度を提供

- ・端末側とクラウドサービスのプロダクトサイクルや仕様差分を吸収
- ・IoTサービスによって最適なクラウドサービスが異なる場合でも柔軟に切り替え可能
(各クラウドに対応した開発コストの削減)

IoT Connect Gatewayの利用例

ICGWの設定の順序

Step1. SIM情報設定



Step2. Authentication設定



Step3. Group設定



Step 4. データ転送

使っているSIMに必要なオプションデータを付与

転送先クラウドでデバイスを識別するための情報
(IMSI, IMEI, MSISDN, DeviceName, SIM画面で設定した任意のパラメータ)

転送先クラウドごとに異なる認証データを登録

転送先サービスごとに登録するデータや認証方式は異なります

データ転送先の設定を登録します。

ICGWの待受PATHの設定や、転送先クラウドの設定などの
データ転送ポリシーを設定

実際にデータ転送を実施し、動作確認を行います。

IoT Connect Gateway設定画面

IoT Connect Gateway Dashboard Groups SIMs Authentications Settings Logged in as: [REDACTED] Select User Logout

Internal Probe for GCP

Internal Probe for AWS

Internal Probe for Azure

Internal Probe for TC

Test_Sumida_Group

ID 615a65af86568d88911ade14 Reset MQTT Session

Protocol Conversions Events Functions SIMs

Add Protocol Conversion ▾

ID	Name	Destination	Path	Edit	Delete
26066308-16e0-49b7-a9f3-3c7b917a4d3b	Internal Probe HTTP for GCP	https://cloudiotdevice.googleapis.com:443	/	Edit	Delete
6a99adc6-f10b-489b-b16f-7f7f41d721a2	Internal Probe MQTT for GCP	mqtt.googleapis.com:8883	N/A	Edit	Delete

IoT Connect Gateway設定画面

IoT Connect Gateway Dashboard Groups SIMs Authorizations Settings Logged in as: internal_probe@icgw.ntt.com Select User Logout

Update Protocol Conversion

Entry Point

Protocol Name: HTTP

Configuration Name: Internal Probe HTTP for GCP

Enabled:

Path: /

* Input "/" when using root path.

Destination

Type: Google IoT

Protocol: HTTPS

Port Number: 443

Credential Set: 22ba2c74-90aa-4867-b82c-2f9db1097fef (GCP IoT Core Cred)

Method: publishEvent

Subfolder:

Reset MQTT Session

Path

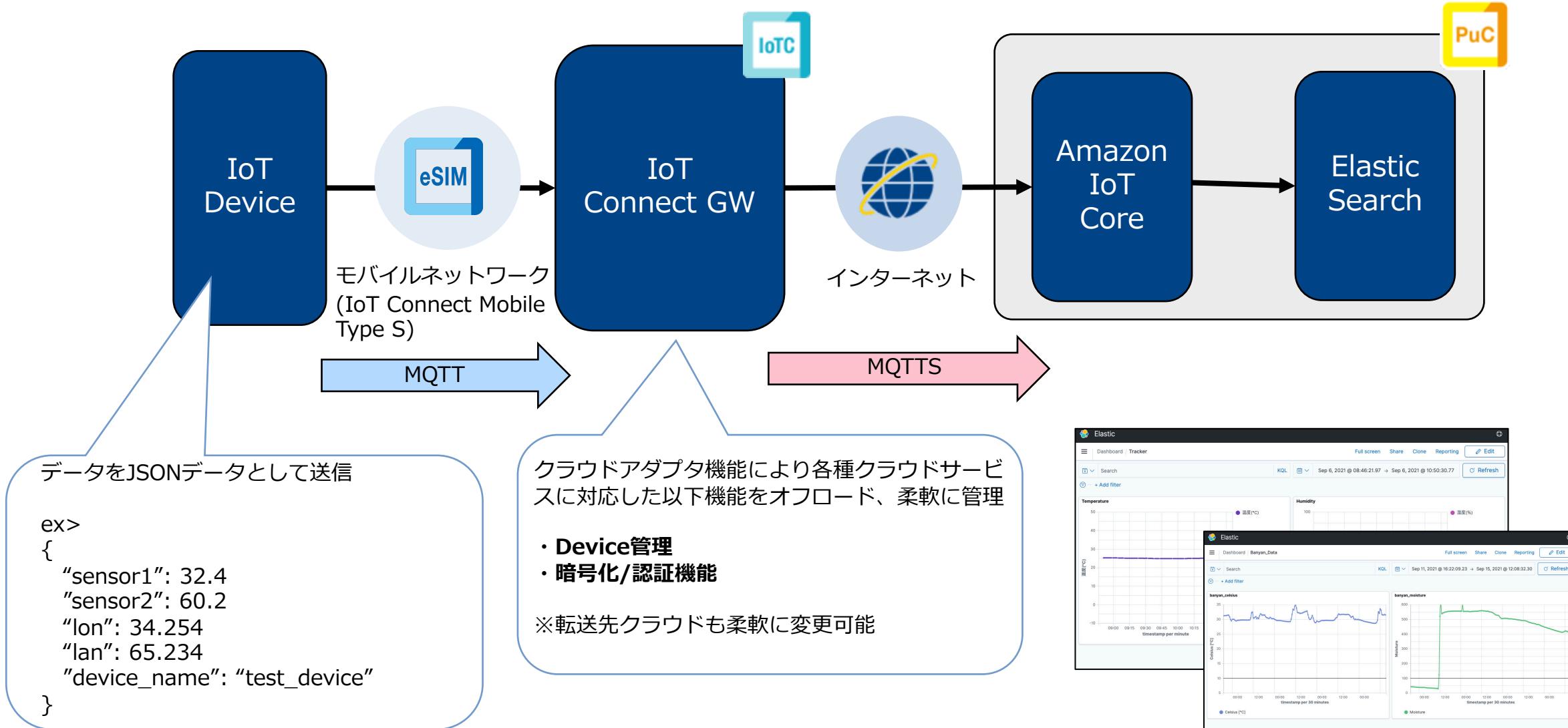
/	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
N/A	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>

« « 1 » »»

IoT Connect Mobile Type S × AWS IoT

デモ内容

IoT Deviceで取得した温湿度データを IoT Connect Gatewayで暗号化し、センサ情報をAWSで可視化し
IoT Connect Gatewayで管理することによって転送先クラウドや設定を柔軟に変更可能



IoT Connect Mobile Type S × AWS IoT

■今回のデモの設定概要

Protocol: MQTT

SaaS: AWS IoT Core

※ 他クラウドサービスへの切り替えもICGWの設定のSIMアサインを
変更するだけで、Device側の設定変更を必要とせず柔軟に切り替え可能

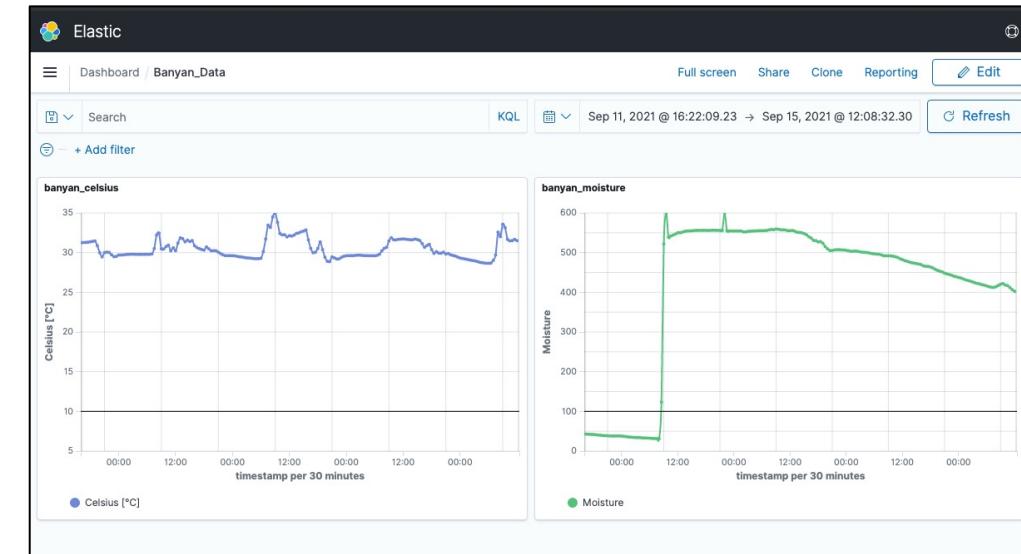
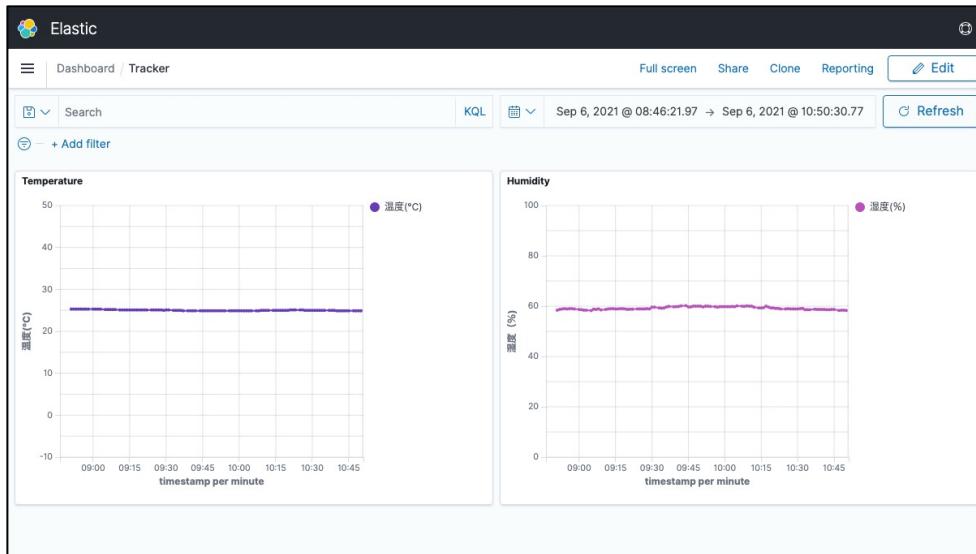
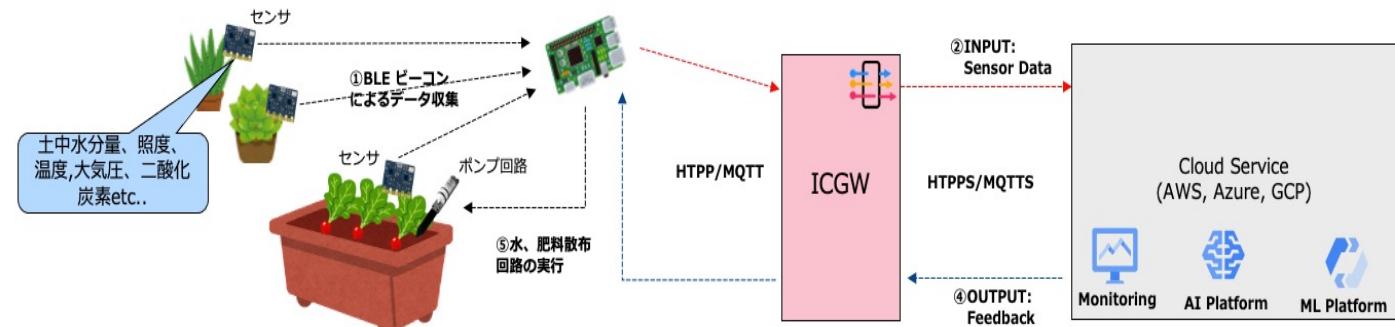
可視化: Elastic Search

※AWS IoT CoreのAction機能を利用し、
データ加工(Lambda)、Mail/Slack通知など
他のAWSのSaaSサービスとの連携が容易

■利用想定イメージ

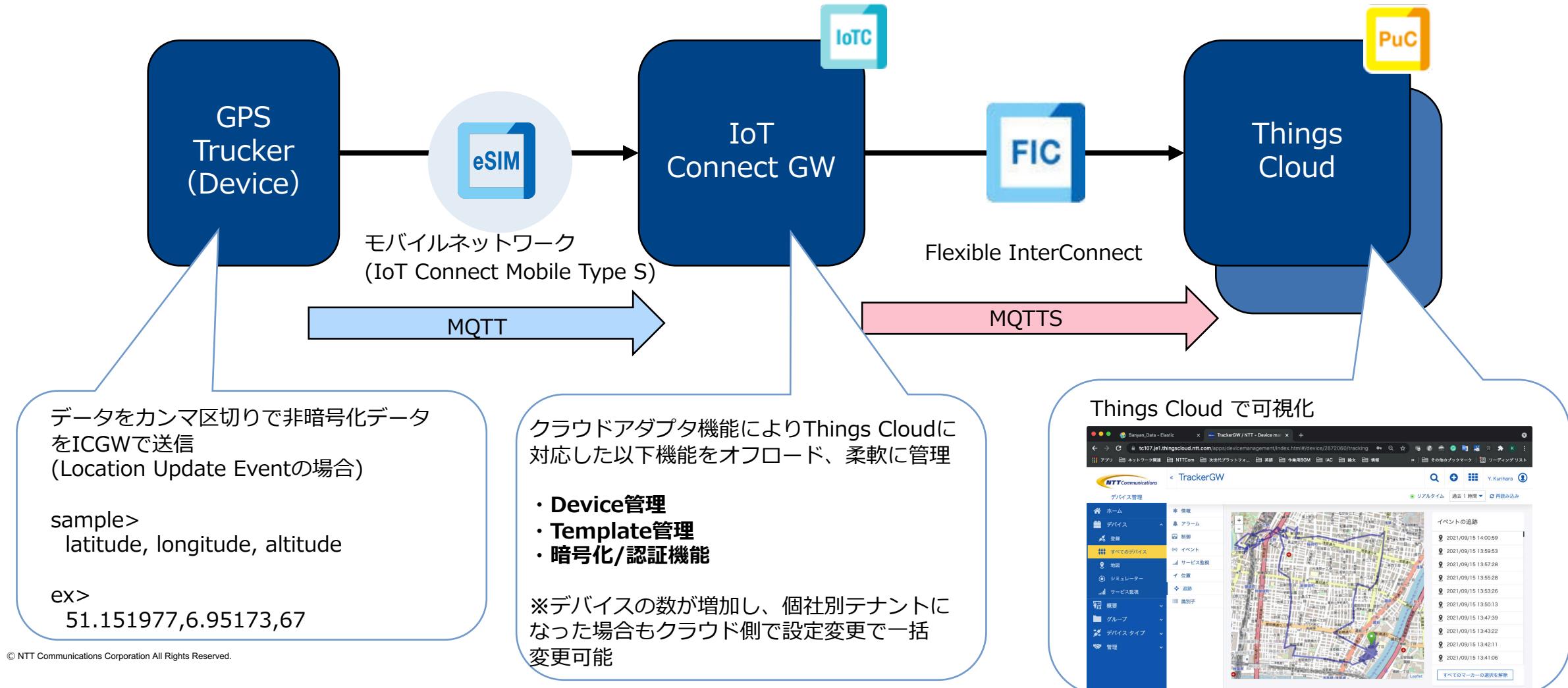
・ IoT農業(観葉植物 or 二十日大根) with ICGW

ICGW経由で、センシングデータ（土中温度、水分、大気圧、二酸化炭素 etc.）をクラウドにあげて水やりや肥料の散布を
センシングデータの解析の結果によってコントロール、フィードバックをディープラーニングなどと組み合わせたらおもしろそうなどの妄想中



IoT Connect Gateway x Things Cloud

GPS Trackerで取得したGPS情報をIoT Connect Gatewayで暗号化、Things Cloud側の対応したテンプレートに変換し、Things Cloudで可視化を行うための設定投入



IoT Connect Gateway x Things Cloud

GPS TrackerからGPS Location情報をHTTPでICGW経由で送信することで、モジュールの暗号化を必要とせず消費電力、デバイスの開発コストを削減可能。Things Cloudで可視化を実現

NTT Communications

デバイス管理

TrackerGW

リアルタイム カスタム 2021-09-23 2021-09-23 再読み込み

検索 フィルター ユーザー

デバイス管理

ホーム デバイス 登録 すべてのデバイス 地図 シミュレーター サービス監視

概要 グループ デバイス タイプ 管理

情報 アラーム 制御 イベント サービス監視 位置 追跡 識別子

地図

イベントの追跡

2021/09/23 23:58:58
2021/09/23 23:57:51
2021/09/23 23:56:45
2021/09/23 23:55:39
2021/09/23 23:54:21
2021/09/23 23:53:07
2021/09/23 23:51:56
2021/09/23 23:50:30
2021/09/23 23:49:19
2021/09/23 23:48:09

すべてのマーカーの選択を解除

※上記は9/23に千葉まで車で移動（祖母の家）した際のGPSデータ

IoT Connect Gatewayの開発課題

開発開始当初の課題

IoT Connect Gatewayの開発は、NTTComの従来ネットワークサービスとまったく違った要件

①市中製品は存在しない

ベンダ製品ではないため、内製開発が必要

ソフトウェアバグがお客様通信にダイレクトに影響、サービス品質を高める仕組みが必要

②トラフィックやユーザの増減によって柔軟な拡張/縮退が必要

モバイルサービスのため回線増加、需要トラフィックの増加が予測が困難

③頻繁なリリースサイクルに耐える設計

接続するクラウドサービスのサービス拡充が頻繁、

それらに追従してサービス開発の継続やリリースが必要

④短納期かつ小規模開発チーム

アプリケーションサービス開発チームを立ち上げたばかりで数人の開発チーム..

つまり…ネットワークサービスでありながら

- ・トラフィック需要に応じた柔軟な拡張/縮退、新機能の追加が必要
- ・開発/リリースサイクル/オペレーションコストの効率化、安定稼働の両立が必要

では、どうしたか？

DPSDevOps
22

よろしい、ならばクラウドサービスで開発しよう！！

■ 基本設計ポリシー

① 開発スコープ、注力ポイントを絞る

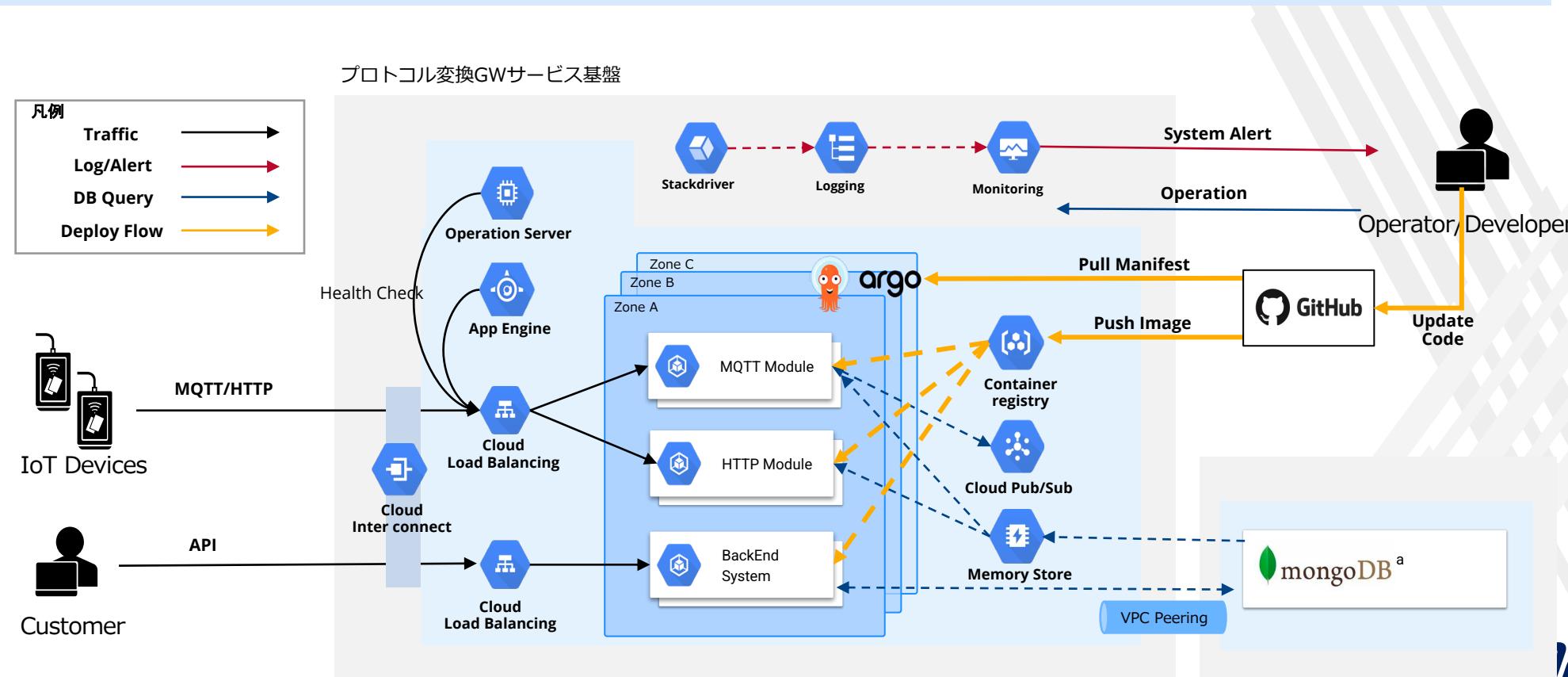
⇒ マネージド・サービスを利用することで開発スピードの効率化、オペレーションコストの削減

② アプリケーションをコンテナ化し、ステートレス設計

⇒ 耐障害性の向上、スケールイン/スケールアウトを実現

③ ダウンタイムなしのアップデート&リリース

⇒ ローリングアップデート、ブルー・グリーンデプロイ



トラフィック需要に応じたスケールアウト/スケールイン

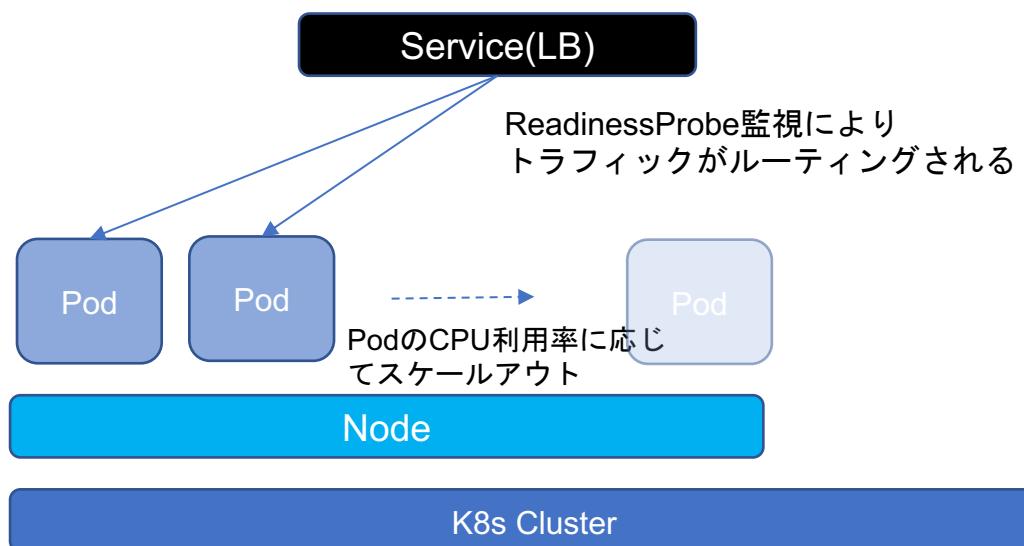
各種モジュールはコンテナ化を行うことで、オートスケールアウト機能により、柔軟にユーザの増加、
トラフィック増加に対応

各種機能モジュールをコンテナ化することで新規機能の追加を容易にする

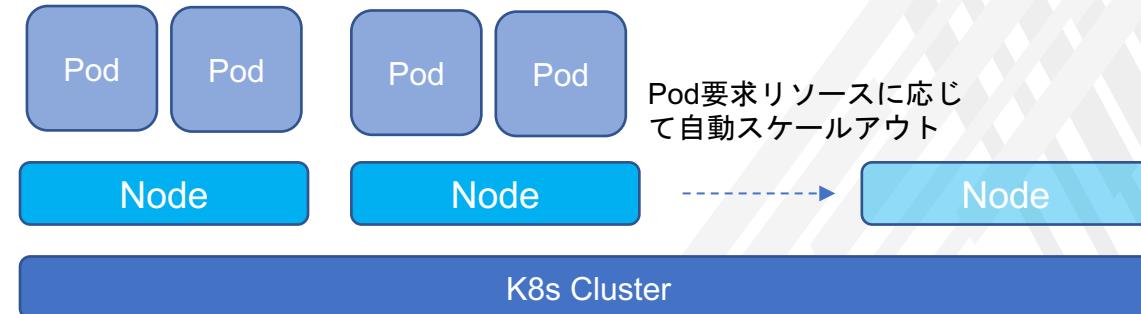
コンテナスケールアウト/スケールアップ ポリシー

1. 水平Podスケールアウト : HPA (自動)
2. Nodeスケールアウト: Cluster Autoscaler (自動)
3. Node インスタンス スケールアップ (メンテナンス作業)
4. クラスタ増設/サイト増設 (メンテナンス)

1. 水平Podスケールアウト(HPA)



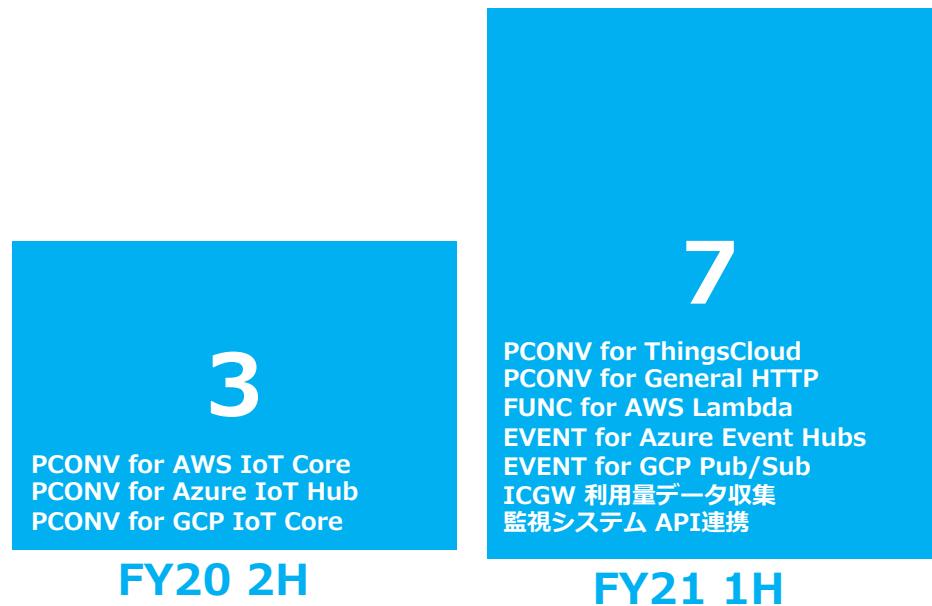
2. Nodeスケールアウト: Cluster Autoscaler



新機能リリース/リリースサイクル短縮の挑戦

ICGWでは、2021/4/8サービスリリース以降、新機能リリースを実施し機能拡充を実施
今後もさらなる新機能追加を予定

ICGW新機能リリース



ICGWモジュールリリース



Kubernetes Upgrades

- ✓ PROD環境 Kubernetes Upgrade 20回(2021/4~)
NODE_POOL Upgrade 10回 x2node pools

IoT Connect GatewayのCI/CDの取り組み

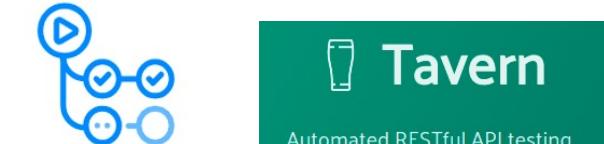
ICGWで導入したDevOpsのための取り組み

1. Infrastructure as Code

～Terraformによるインフラ構成管理～

- ①環境手配の遅れリカバリ
- ②Prod/Stg環境差分撲滅
- ③バグ再現しません問題回避
- ④開通遅延、モバイル網の構築トラブル回避

ICGW基盤ではGitHubのCode&設定
すべての基点として位置づけられている



GitHub Actions



2. Test as Code

～CI/CDによる検証稼働削減、デグレ対策～

- ①Unit Test/Integration Testの自動化
- ②API Testの自動化
- ③E2Eテストの自動化

3. GitOps

～Argo + Helmを使ったコンテナサービスリリース～

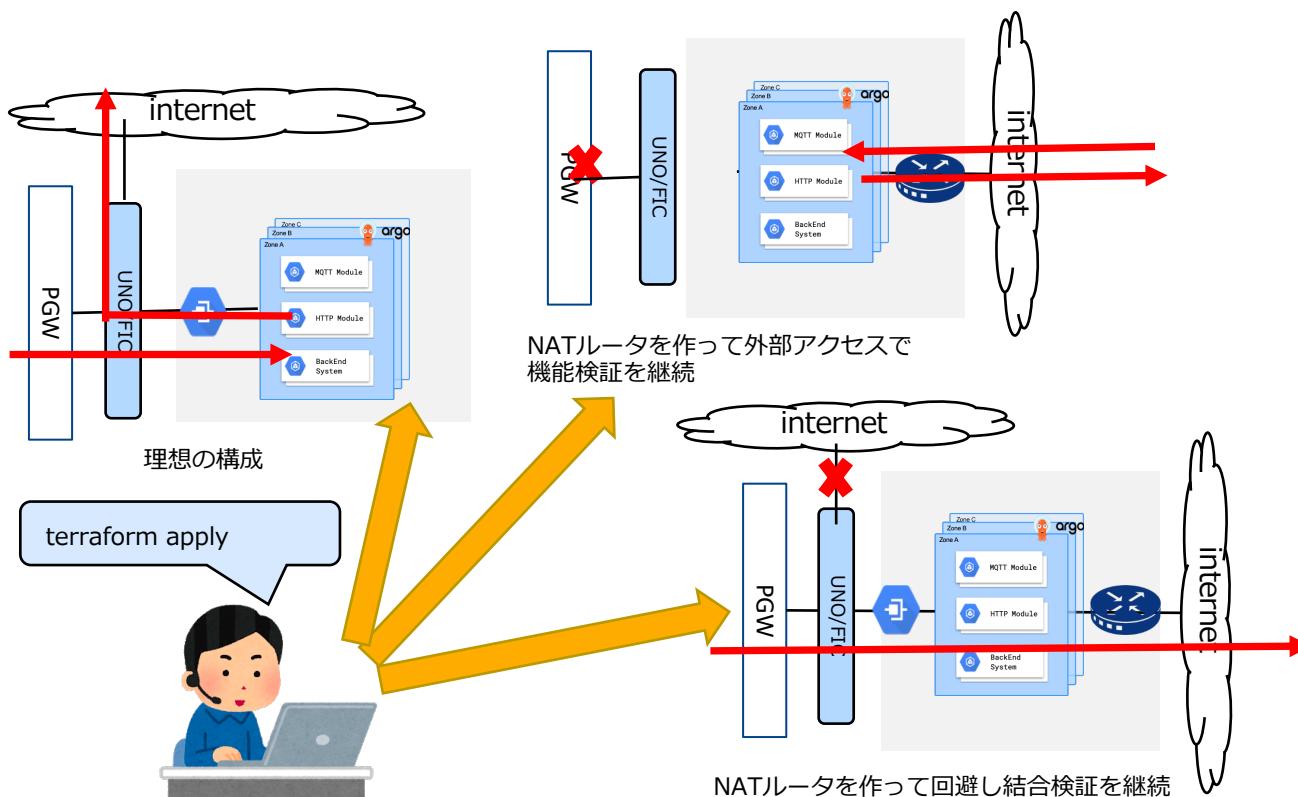
- ①コンテナイメージ構築
- ②Gitでのデプロイ制御
- ③ローリングアップデート&リリース
- ④Probeによるお客様影響の確認

1. Infrastructure as Code

～Terraformによるインフラ構成管理～

■ Terraform化しておいてよかったこと

- ①環境手配の遅れリカバリ（1ヶ月）
- ②Prod/Stg環境統一
- ③環境差分でバグ再現しません問題の回避
- ④モバイル網の開通遅延、構築トラブルによる検証遅延回避（3週間）



ICGW基盤の構築をCode(Terraform)で定義することで構築、廃止、
設定変更が1コマンドで実現可能
手動構築の時間短縮、設定差分の抑制が可能！

© NTT Communications Corporation All Rights Reserved.

The terminal screenshot shows the Terraform directory structure and some configuration code. Red boxes highlight specific parts of the code and directory structure, with callouts explaining their purpose.

- FIC接続用のterraformモジュール (Module for FIC connection):** A red box highlights the "flic" module under the "regional" directory.
- NTTCOM Dev用の個別パラメータ (Individual parameters for NTTCOM Dev):** A red box highlights the "common.tfvars" file in the "develop" directory under "nttcom".
- NTTCOM Prod用の個別パラメータ (Individual parameters for NTTCOM Prod):** A red box highlights the "common.tfvars" file in the "production" directory under "nttcom".
- NTTCOM Stg用の個別パラメータ (Individual parameters for NTTCOM Stg):** A red box highlights the "common.tfvars" file in the "staging" directory under "nttcom".
- PNL Dev用の個別パラメータ (Individual parameters for PNL Dev):** A red box highlights the "common.tfvars" file in the "develop" directory under "pn1".
- 環境毎の個別パラメータを定義しterraform plan prod (確認) terraform apply prod (構築) (Define individual parameters per environment and run terraform plan prod (check) terraform apply prod (build)): A callout points to the "common.tfvars" files in the "staging" and "develop" directories.**
- create_nat = true, create_interconnect = falseに変更しterraform apply コマンドを実行することで、GCPにNATルータが構築され、FIC接続用のInterconnect, Routerが廃止される (Change to create_nat = true, create_interconnect = false and run the terraform apply command to build a NAT router in GCP and decommission the Interconnect and Router for FIC connection): A callout points to the "common.tfvars" file in the "staging" directory, specifically highlighting the "create_nat" and "create_interconnect" settings.**

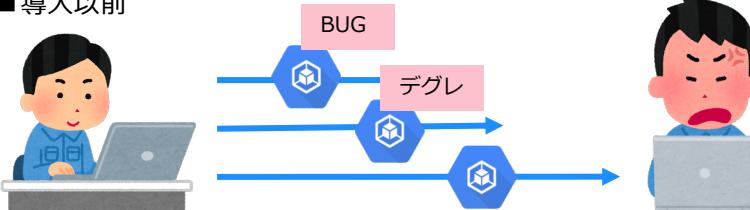
2. Test as Code

～CI/CDによる検証稼働削減、デグレ対策～

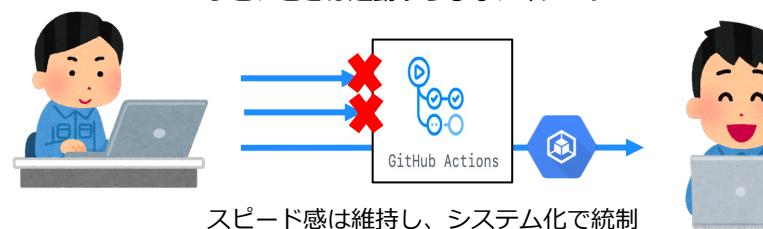
①Unit Test/Integration Test

- ・ 頻発するデグレ&頻繁なイメージ変更
(3times/week)
に対抗するためにUnitTest/IntegrartionTestの導入

■導入以前



■導入後



Git RepositoryにCodeをPushすることでテスト&イメージビルドが実行
イメージアップロードする環境はgit tagで制御
v1.0.1 ⇒ 開発環境(dev)のみ
v1.0.1-stg ⇒ 開発環境(dev) + リリース前検証環境(stg)

GitHubActionをトリガに
UnitTest/IntegrartionTestが実行

docker buildでコンテナイメージを構築

NTTCom(Stg)のGCRにコンテナイメージのアップロード

GKE ManifestのImage Versionを変更してcommit

②API Test Automation (E2E)

- ・ 数あるAPI Test Toolから我々がTavernを選択した理由
 - ・ チームメンバがみんなコードを書くのが好き&得意なわけじゃない
 - ・ DevOpsチームは1日にしてならず、得意でない人の心理障壁を削減
 - ・ JSON Test定義ファイル(Postmanなど)はGit管理しにくい
 - ・ 冗長になる&差分管理がしにくい

Tavern for API testing : Why Tavern?

Tavern is a **pytest** plugin, command-line tool and API testing tools Python library for automated testing of APIs.

- Simple, concise and flexible YAML-based syntax.
- It's very simple to get started
- Can create highly customizable for complex tests.

```
test_name: Get some fake data from the JSON placeholder API

stages:
  - name: Make sure we have the right ID
    request:
      url: https://jsonplaceholder.typicode.com/posts/1
      method: GET
    response:
      status_code: 200
      json:
        id: 1
```

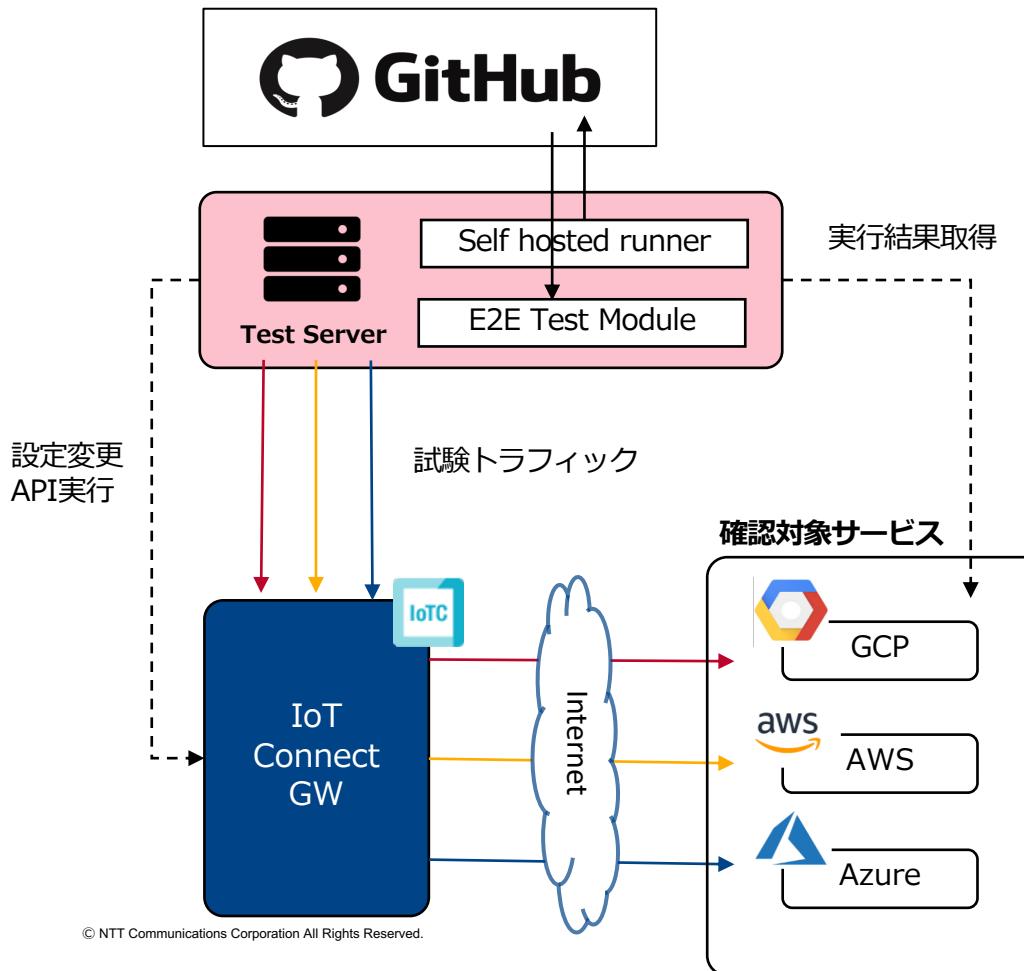
2. Test as Code

～CI/CDによる検証稼働削減、デグレ対策



③E2E Test

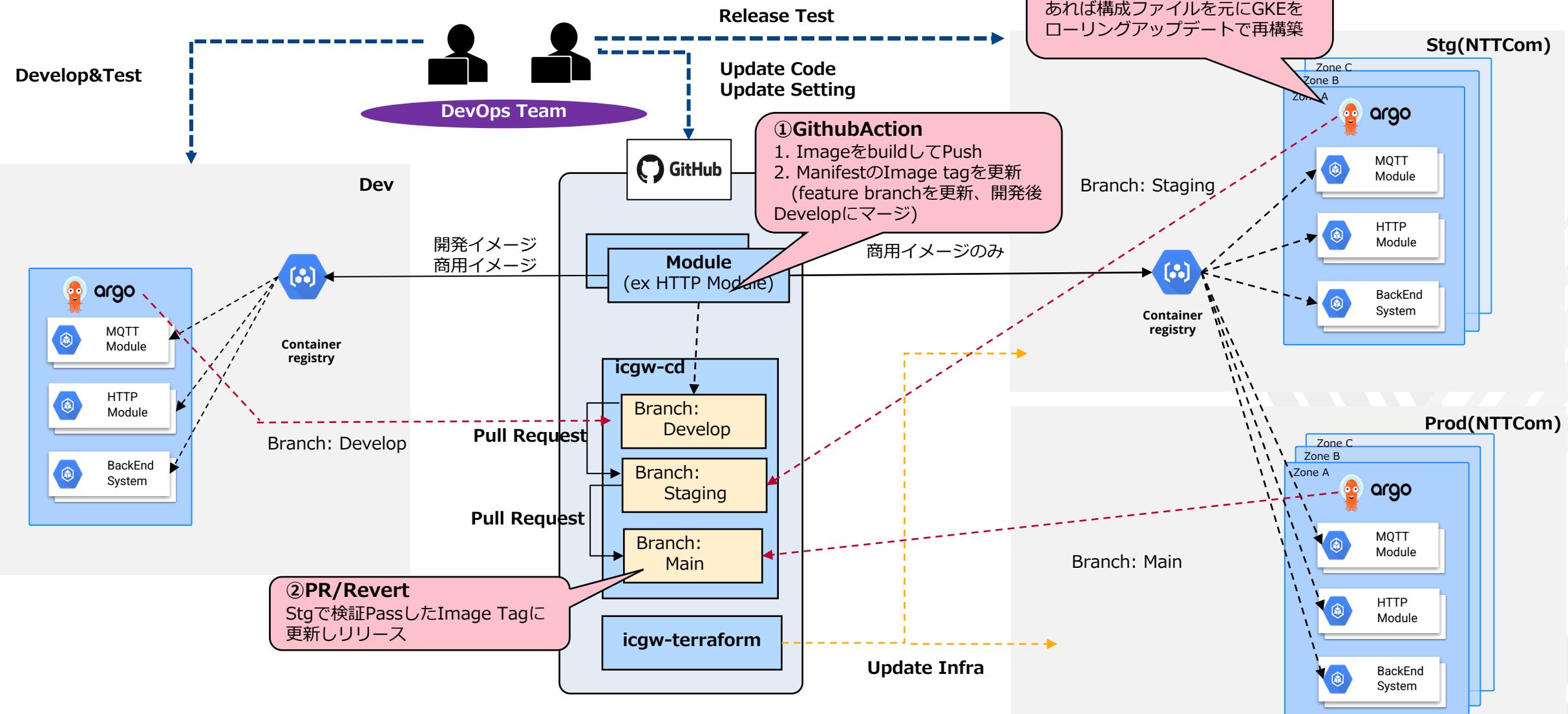
ICGWの提供しているサービスモジュールの設定変更を実施しながら、網羅的にE2Eテストを実行
Github Actionから実行により、手動試験作業稼働の削減



The screenshot shows the GitHub Actions interface for the repository "pconv / tester-script". The "Actions" tab is selected. In the center, there is a section titled "E2E TESTS" with a sub-section "e2e.yaml". Below this, a table lists 129 workflow runs, each with a status of "E2E TESTS" and a note that they were "Manually run by mahoko-tamura". On the right side of the screen, there is a configuration sidebar for running the workflow. It includes fields for "Use workflow from" (set to "Branch: main"), "Specify protocol to test (http or mqtt or adopter). Empty lets all tests run.", "Specify pytest marker expression.*" (set to "aws or azure or gcp or gen"), "Run last failed tests (yes/no) ? *" (set to "no"), and "Specify pytest options.". At the bottom right of the sidebar is a green "Run workflow" button.

3. ICGWで実現したGitOps

~ Argo + Helmを使ったコンテナリリース ~

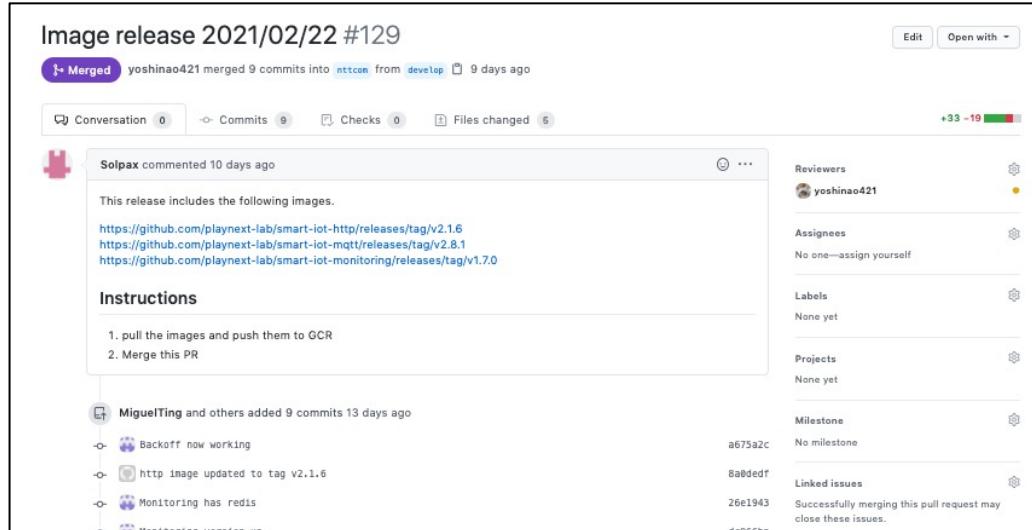


すべてのコンテナのイメージ構築、構成管理はGitで集中管理しデプロイまで自動化！！
ロールバック(Git Revert)も容易になり、構築ミスも時間も削減

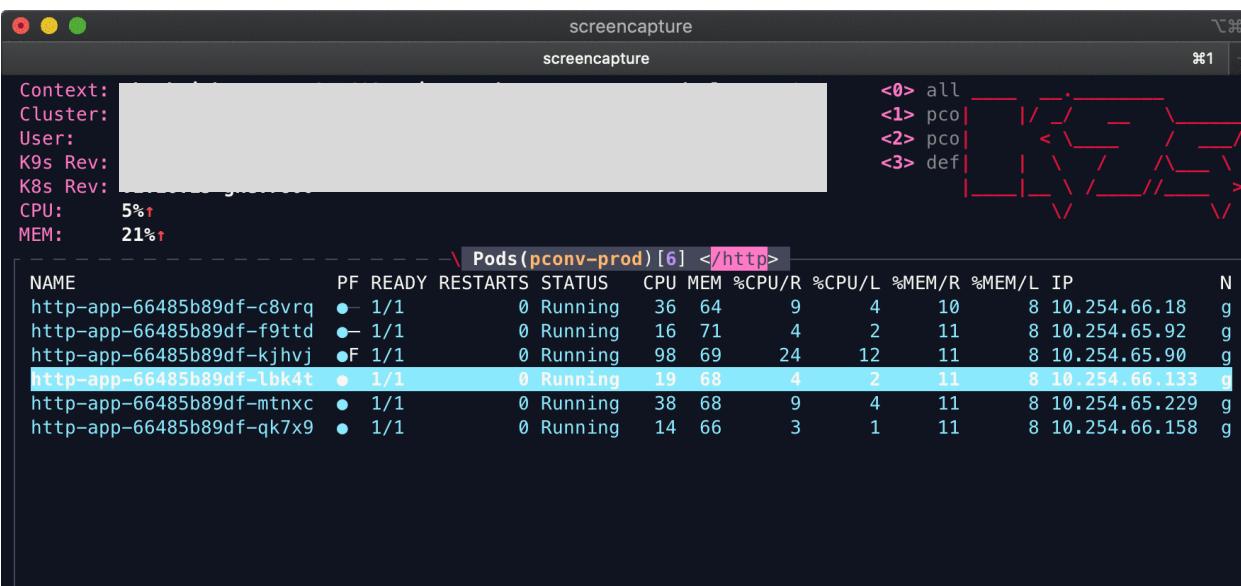
3. ICGWで実現したGitOps

～Argo + Helmを使ったコンテナリリース

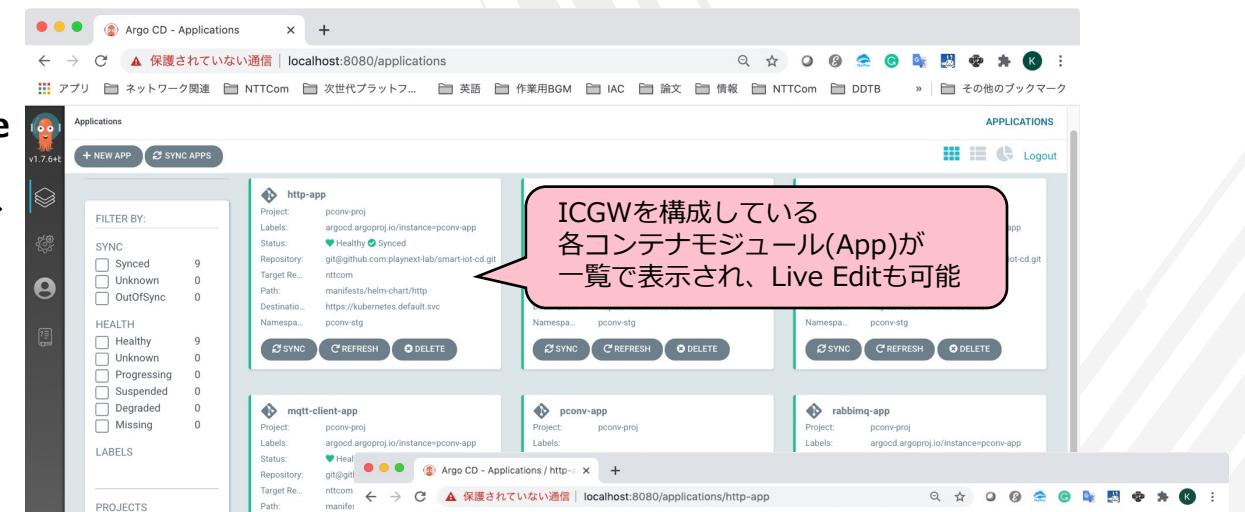
①PRによるイメージリリース



③ローリングアップデートによるダウンタイムなしリリース



②ArgoがGit差分を検知し、Helmによって環境毎のパラメータを反映しk8sにコンテナを構築



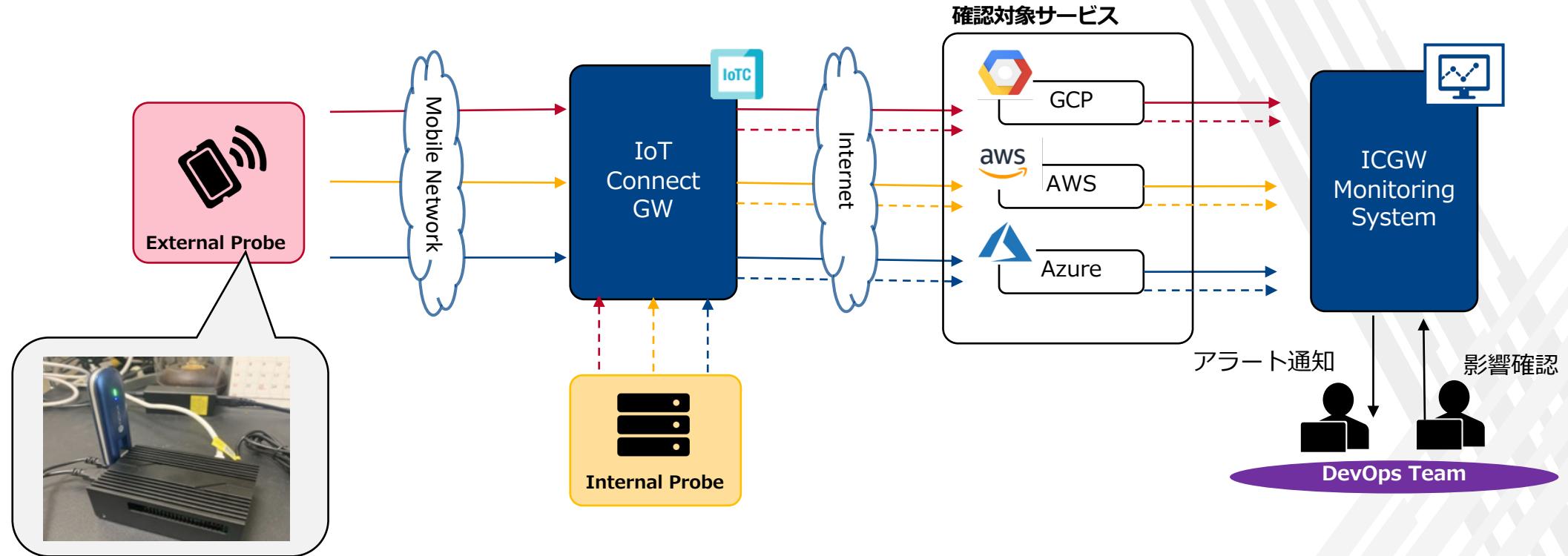
HTTPモジュール(App)の構成要素
(LB,Deployment,HPA…etc)
とステータスが一覧可能、編集可能

Deploy

3. Probeによる定常監視

Stg/Prod環境で各クラウドサービスに対する通信を常時実施することで、可視化、異常検知する仕組みを導入

- 複数のサービスの組み合わせによる複雑な切り分けや検証が簡単に
- 検証稼働削減を実現し、より開発に注力できる環境を実現



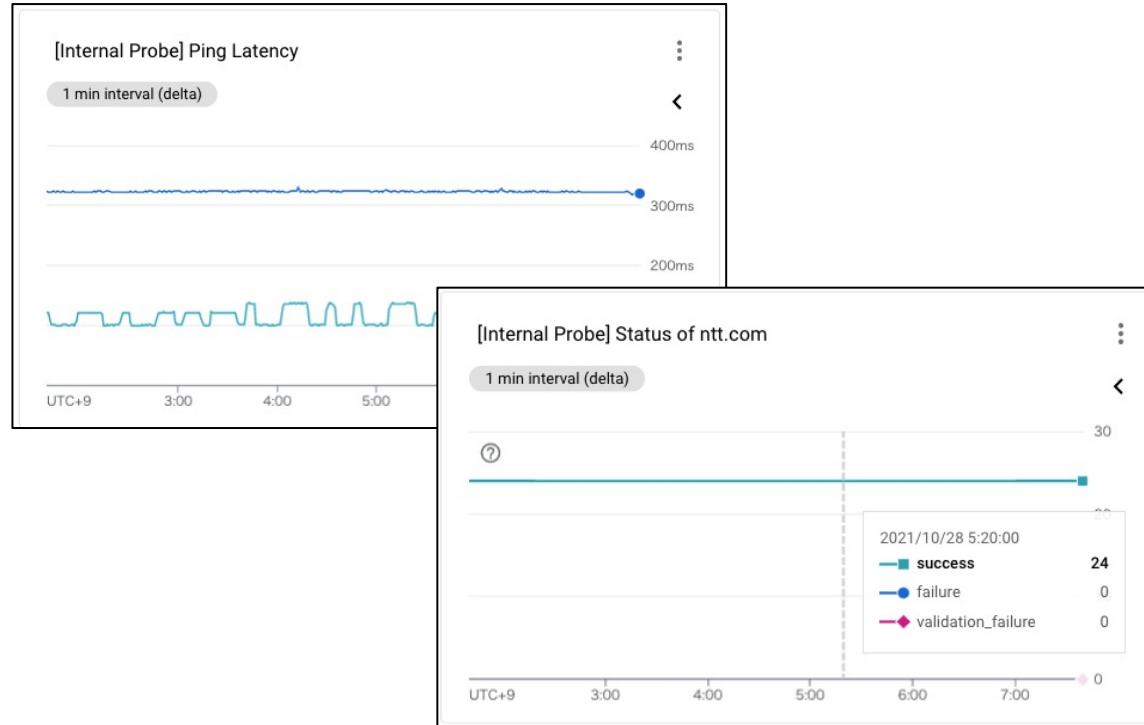
Internal Probe: ICGW基盤から各クラウドサービスの正常性を監視

External Probe: 端末から各クラウドサービスの正常性を監視

Probeによる定常監視

■ ICGW基盤のネットワーク監視

転送先クラウドサービスまでのネットワーク遅延などの問題がないか、ICGW基盤のネットワークが正常かを確認可能



■ 転送メッセージ遅延監視

各クラウドサービスに対するメッセージの遅延や欠損がないか確認可能



正常性試験、長期安定試験などの検証の稼働を大幅削減！
開発と並行して、お客さま影響を確認可能

まとめ/今後の取り組み

まとめ

- ・**基盤のコンテナ化**により、需要によって自動的にスケールアウト/インを実現、ダウンタイムなしのサービスリリース、新機能追加リリースを実現
- ・試験自動化やCI/CDの取り組みにより、新機能リリースと安定性の両立を実現でき、リリースサイクルの短縮

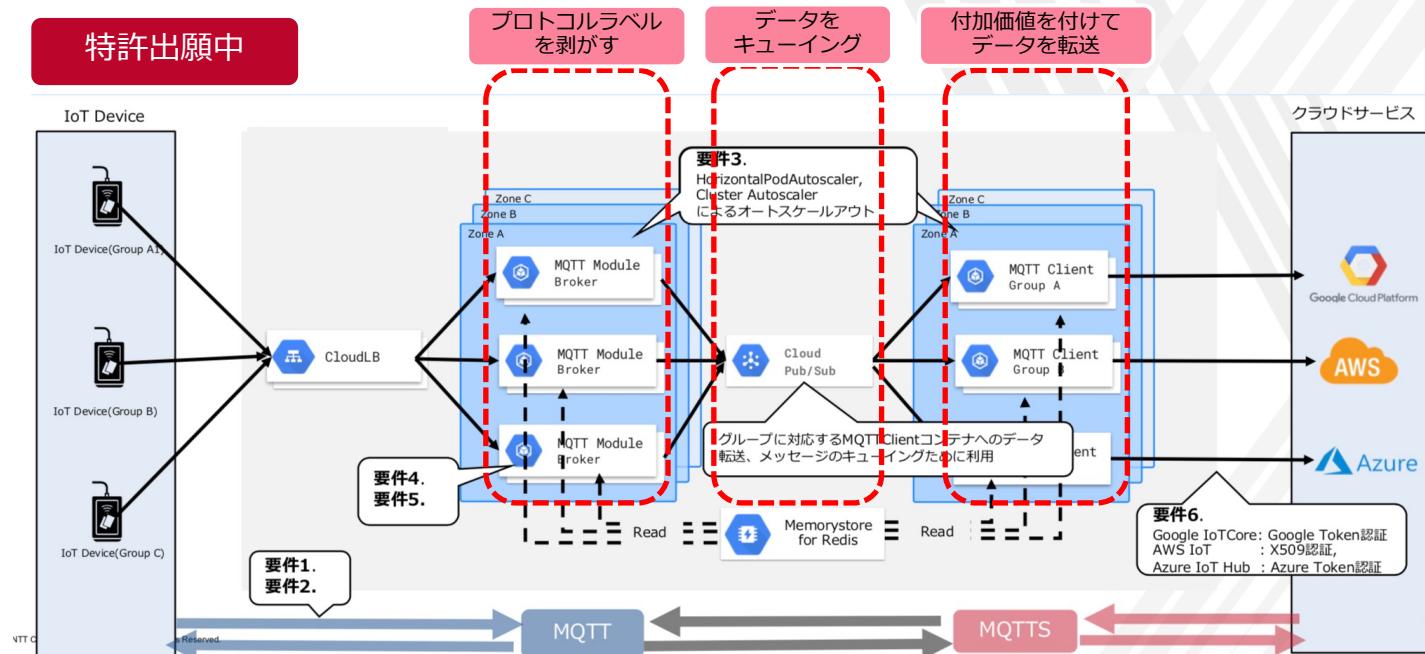
■ SDN Japan2014で登壇資料（栗原）より抜粋

https://onic.jp/archive/2014/document_2014/30_session2_Kurihara.pdf



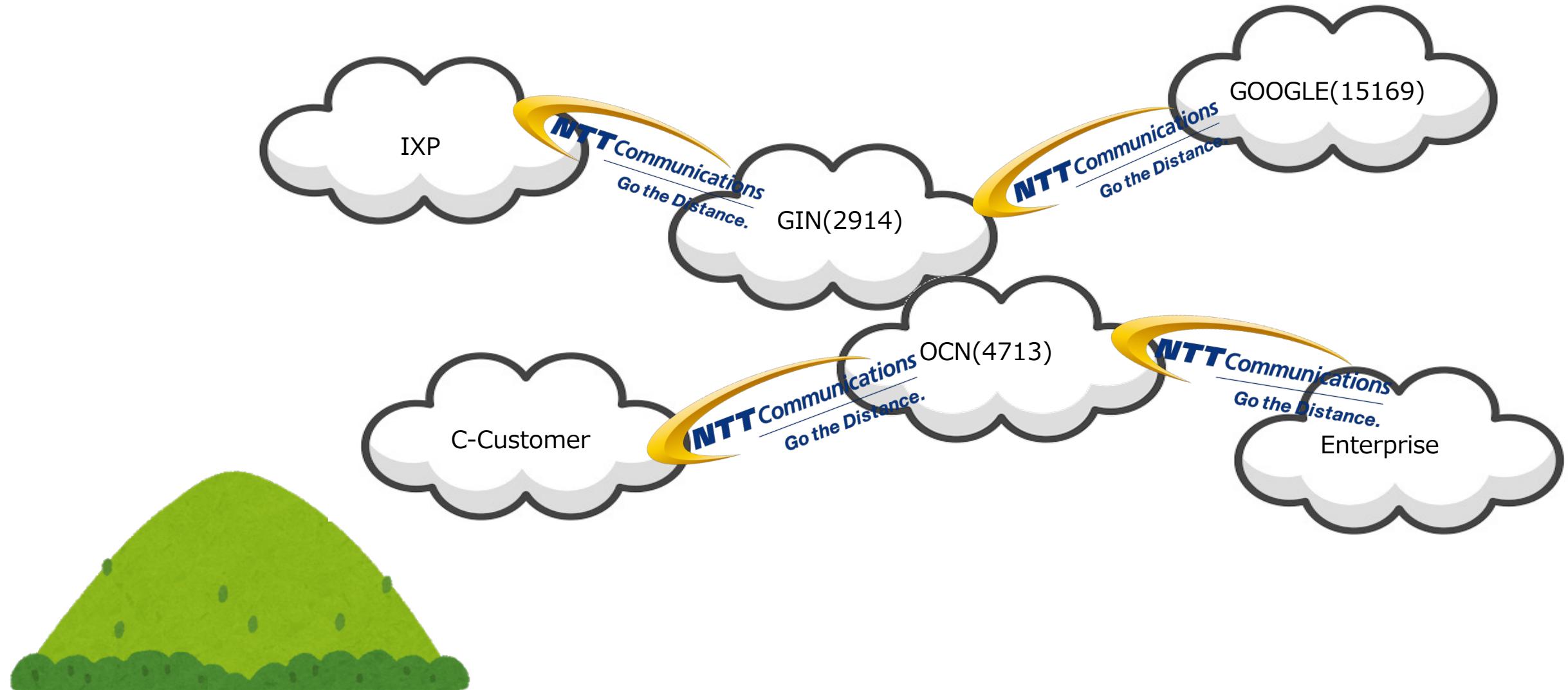
今後の展開と野望

- ・ICGWのアーキテクチャはコンテナを追加することで拡張可能な仕組みを実現しているので、様々なプロトコル変換機能、各種付加価値追加を実施していく予定
- ・Local 5GなどのICMS以外の回線サービスの対応や、IoT用途以外(ex サービスマッシュのハブ)としての機能拡張の実現へ



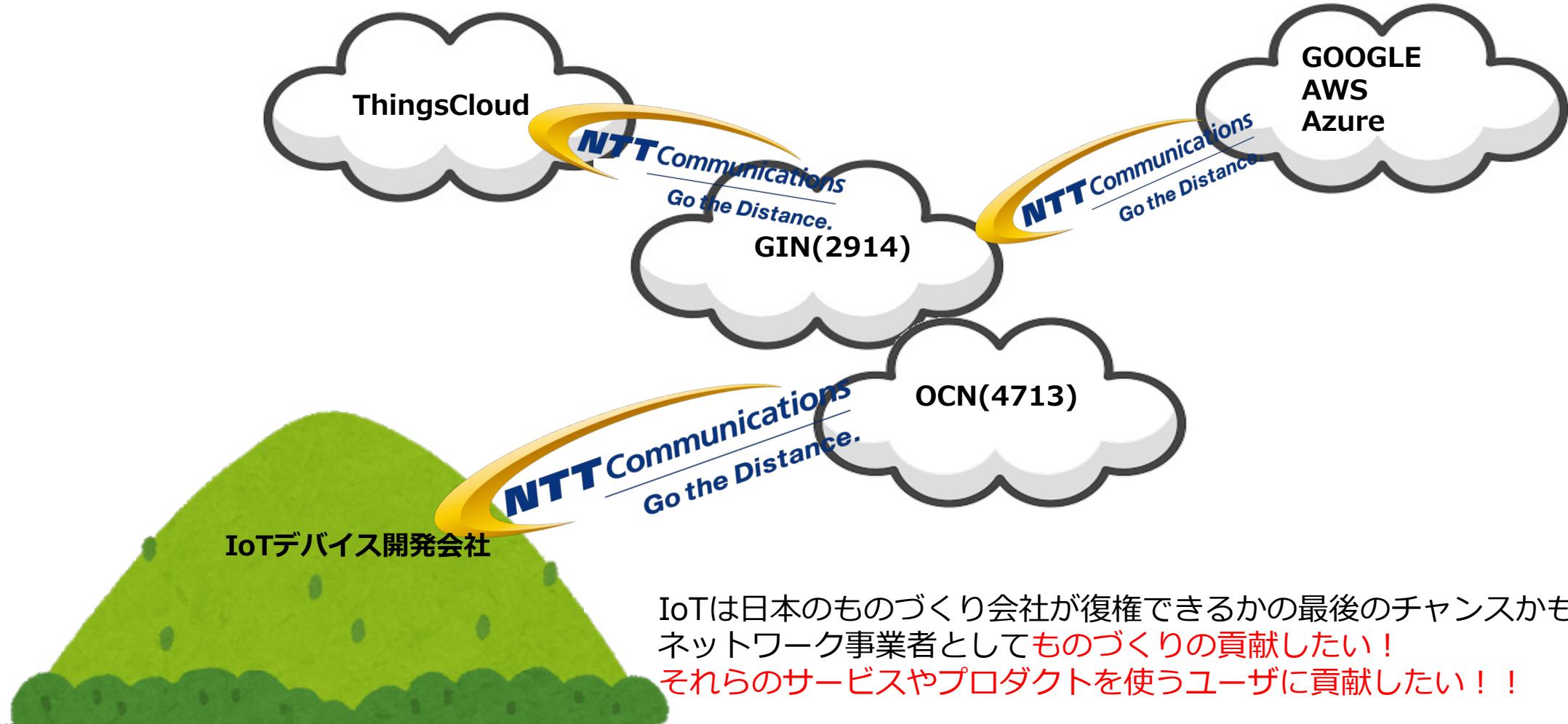
最後に

今までのNTTComはネットワークやクラウドの会社でしたが….



最後に

IoT Connect Gatewayはこれまで好きだった電子回路の世界と
大学院から夢中になったインターネット、クラウドの世界をつなぐサービス



ご清聴ありがとうございました

Re-connect X™

