KubernetesでネットワークコントローラKuesta(ケスタ)」をつくってみたが、登りたい山はまだいくつもある

2024年1月19日

NTTコミュニケーションズ株式会社
イノベーションセンター
坂井 立晟

© NTT Communications Corporation All Rights Reserved.

自己紹介スライド

- NTTコミュニケーションズ
- 開発エンジニア

- 坂井 立晟（さかい たつき）
- 略歴
  - 2021年 NTTコミュニケーションズ入社
  - 伝送NWオーケストレーター開発
  - 社内検証NWの設定自動化ツール開発
  - 現在: Kuesta(ケスタ)開発
- メールアドレス: tatsuki.sakai@ntt.com

© NTT Communications Corporation All Rights Reserved.

Kuesta(ケスタ)概要

© NTT Communications Corporation All Rights Reserved.

Cuesta（意味: 岩石層が交互に重なり合っている緩やかな丘陵）の頭文字をkにして『Kuesta』

商標に類似の名前があったので名称は見直し予定、、、

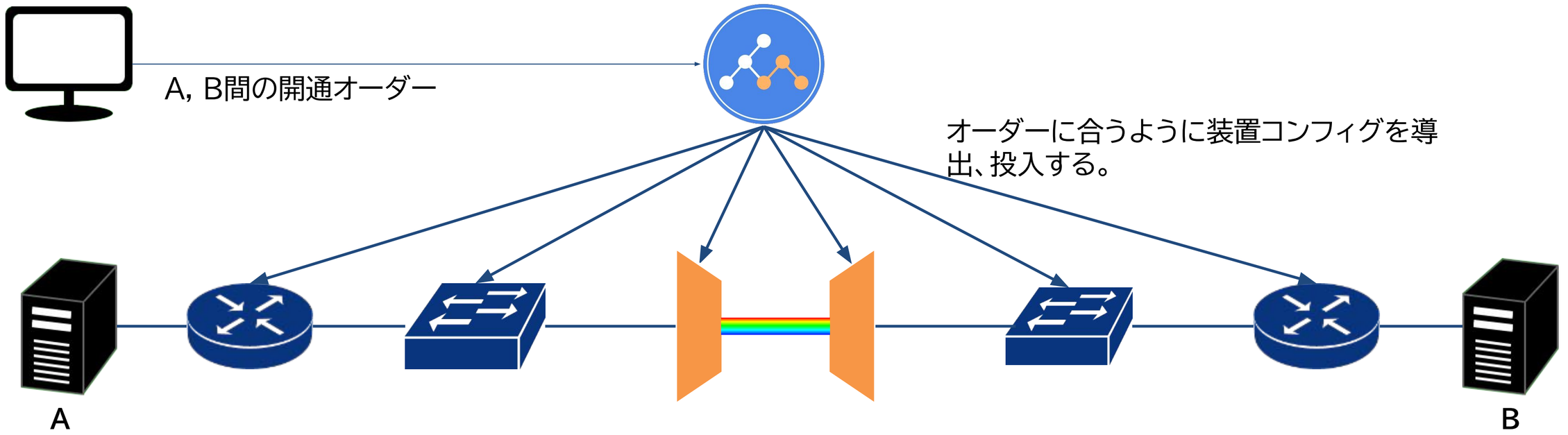
Kuesta（ケスタ）

Github: https://github.com/nttcom/kuesta
Community site: https://nttcom.github.io/kuesta-website/community

© NTT Communications Corporation All Rights Reserved.

Kuesta(ケスタ)とは

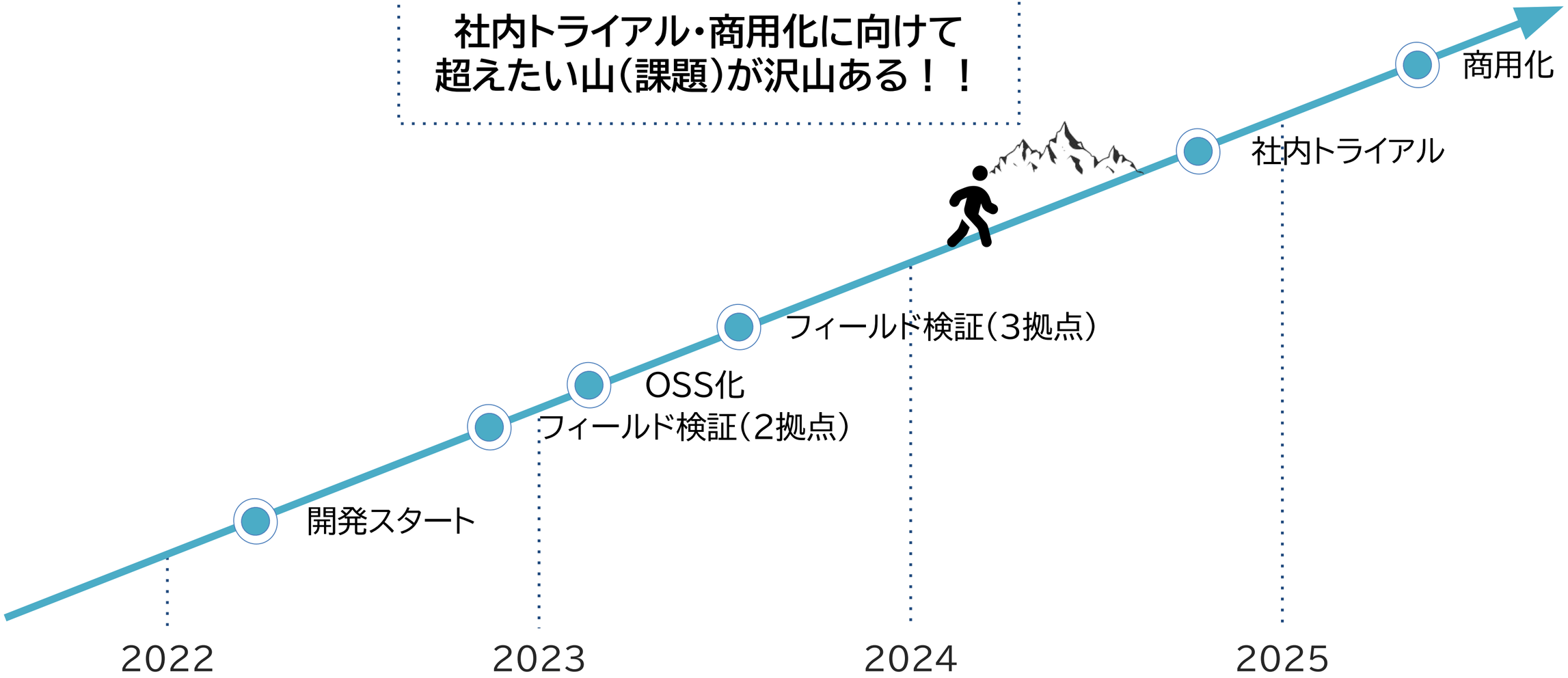
- K8s、GitOps等のクラウドネイティブ技術を活用した宣言的NWコントローラー
  - ルーター/スイッチ/伝送装置を制御対象
  - マルチベンダに対応



© NTT Communications Corporation All Rights Reserved.

活動線表

社内トライアル・商用化に向けて
超えたい山（課題）が沢山ある！！



© NTT Communications Corporation All Rights Reserved.

開発背景

© NTT Communications Corporation All Rights Reserved.

なぜNWコントローラーを開発しているのか

- 所属チームでは色々な社内案件で市販のNWコントローラーを使っている。
- その中で以下課題を感じていた。
  - ライセンスが結構高い。
  - 必要な技術スキルが多くてエンジニアを集めづらい ＋ 育てるのが大変。

自分達で必要なNWコントローラーを開発したい！！

→ 新NWコントローラーを内製開発

© NTT Communications Corporation All Rights Reserved.

NW装置へのコンフィグ投入自動化は難しい

- **ベンダ・NOSによって仕様やオペレーションが異なる**
  - サービスや装置ごとに自動化ツールに差異が生じる、もしくは手動での操作が入る

  → バラバラのツール類、手動での操作と自動化部分の混在によって管理が余計困難になる。

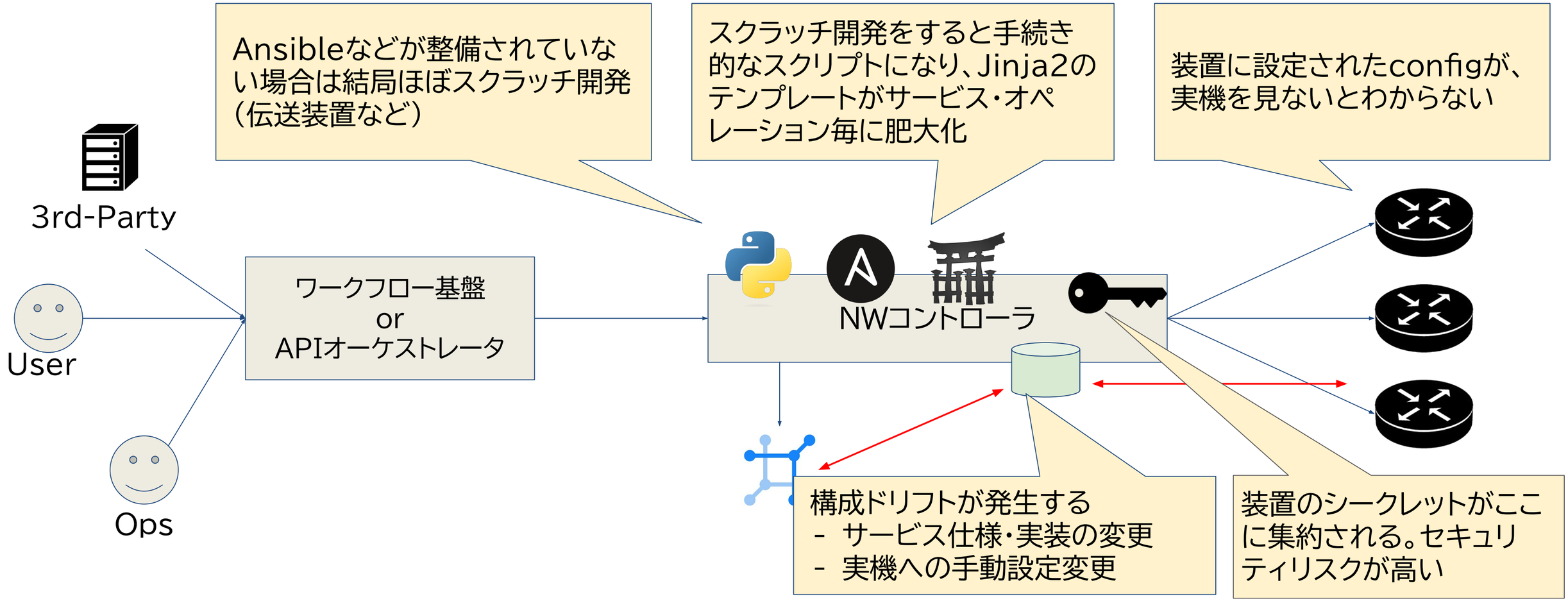
- **複数ノードの状態管理が必要**
  - NWは複数のノードの状態の集合体であり、ノード間で状態や依存関係を考慮する必要がある

  → 異なるNWサービス間での設定競合防止、ノード間をまたぐトランザクション管理
  に対応できなければならない。

© NTT Communications Corporation All Rights Reserved.

既存のNW自動化解法

- Ansible + jinja2を用いた自動化事例が増え、Pythonライブラリなども充実してきた
- 以下構成例



© NTT Communications Corporation All Rights Reserved.

クラウドネイティブ技術で考える

- **Kubernetesによるコンテナ管理**
  - リソースの状態を宣言的に管理し、監視できる。
  - コンテナの増減が容易でスケーラビリティが高い。
  - Secretを活用してセキュアに機密データを管理できる。

- **GitHubによるデータ管理**
  - Web GUIでリソースの状態や変更遷移が確認できる。
  - SSoTにすることで、commit単位での切り戻しが容易になる。

K8s、GitHubを活用して、新NWコントローラーの開発

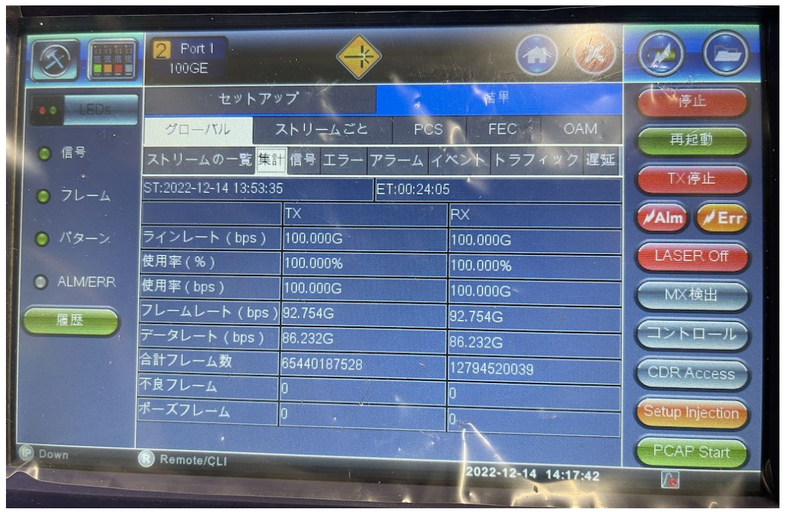
© NTT Communications Corporation All Rights Reserved.

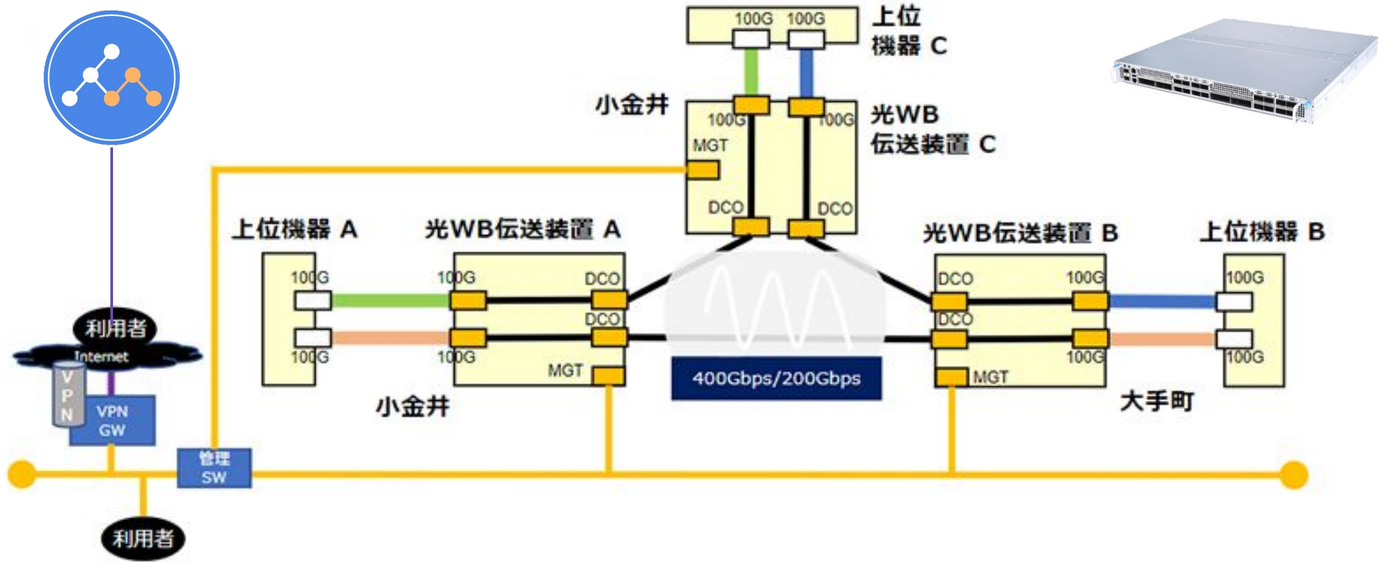
今までの取り組み

© NTT Communications Corporation All Rights Reserved.

昨年度: フィールド環境での検証（伝送装置）

- NICT様環境でのフィールド検証
- Kuestaを用いてライン側を導通し、クライアント間を疎通させることに成功*1





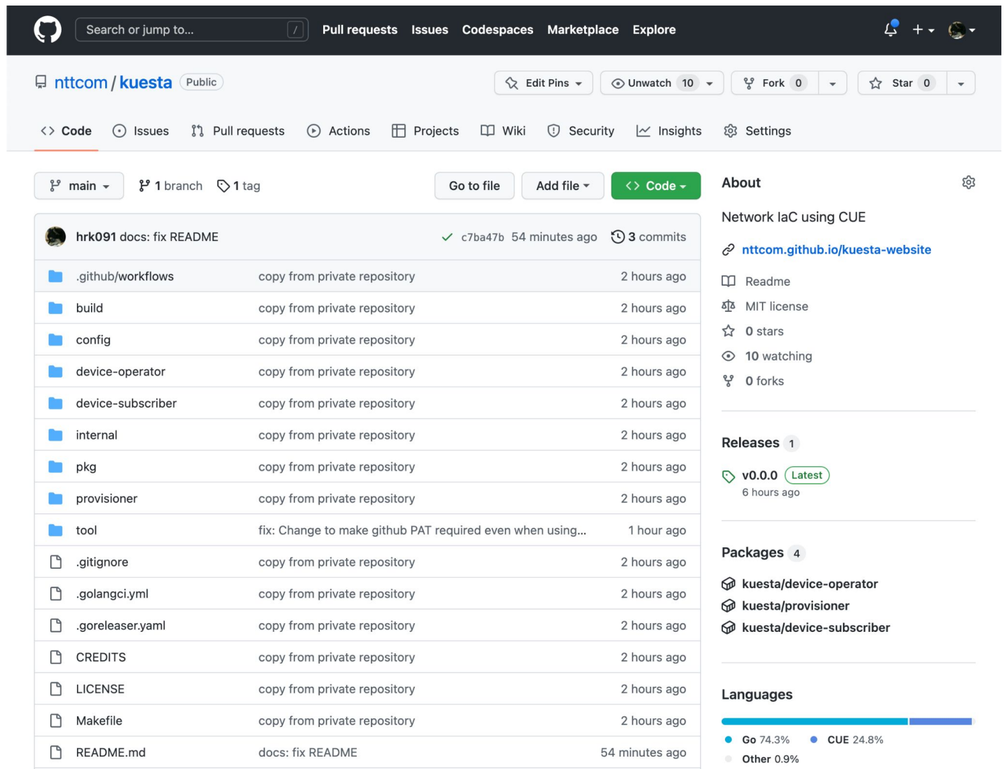
Galileo Flex T

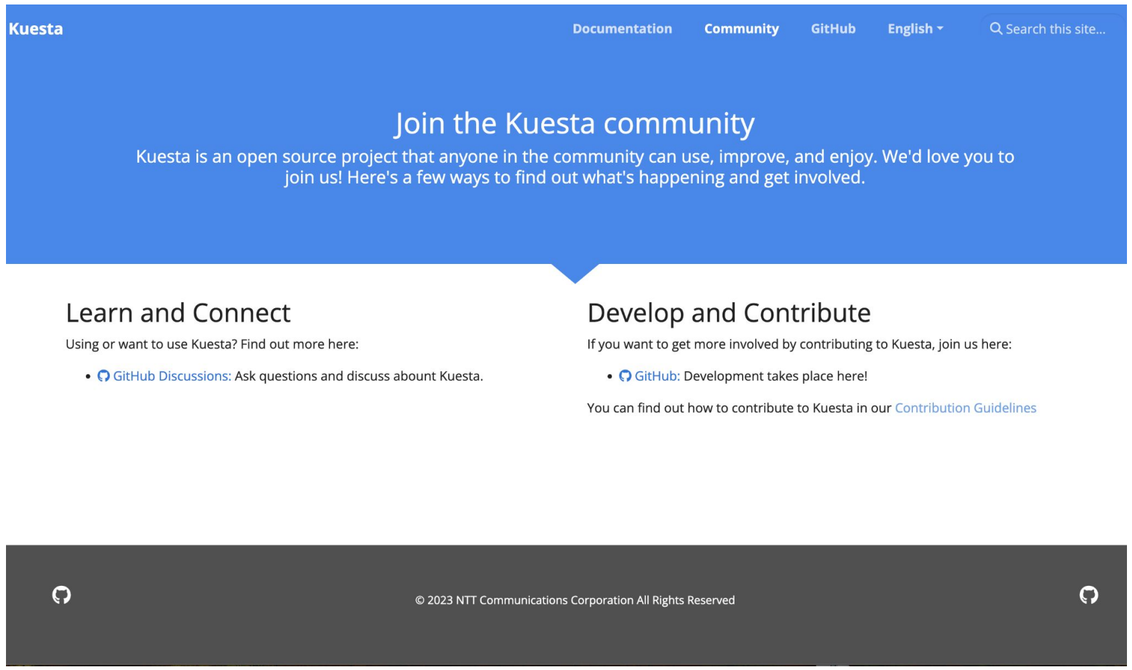
*1: 本研究は、国立研究開発法人 情報通信研究機構（NICT）が運用する NICT 総合テストベッド「B5G高信頼仮想化環境」を用いて行われました。

© NTT Communications Corporation All Rights Reserved.

昨年度: OSS化

- GitHub: https://github.com/nttcom/kuesta
- コミュニティサイト: https://nttcom.github.io/kuesta-website/community





© NTT Communications Corporation All Rights Reserved.

実用化に向けた4つの課題と解決策

⛰️ 課題① 外部システムとの連携プロトコルが限られている

→ 解決策①: NorthboundのWebAPI化

⛰️ 課題② 制御できる装置の種類が少ない

→ 解決策② 汎用的・拡張性の高いdriverの開発

⛰️ 課題③ 装置設定導出ロジックの開発効率が悪い

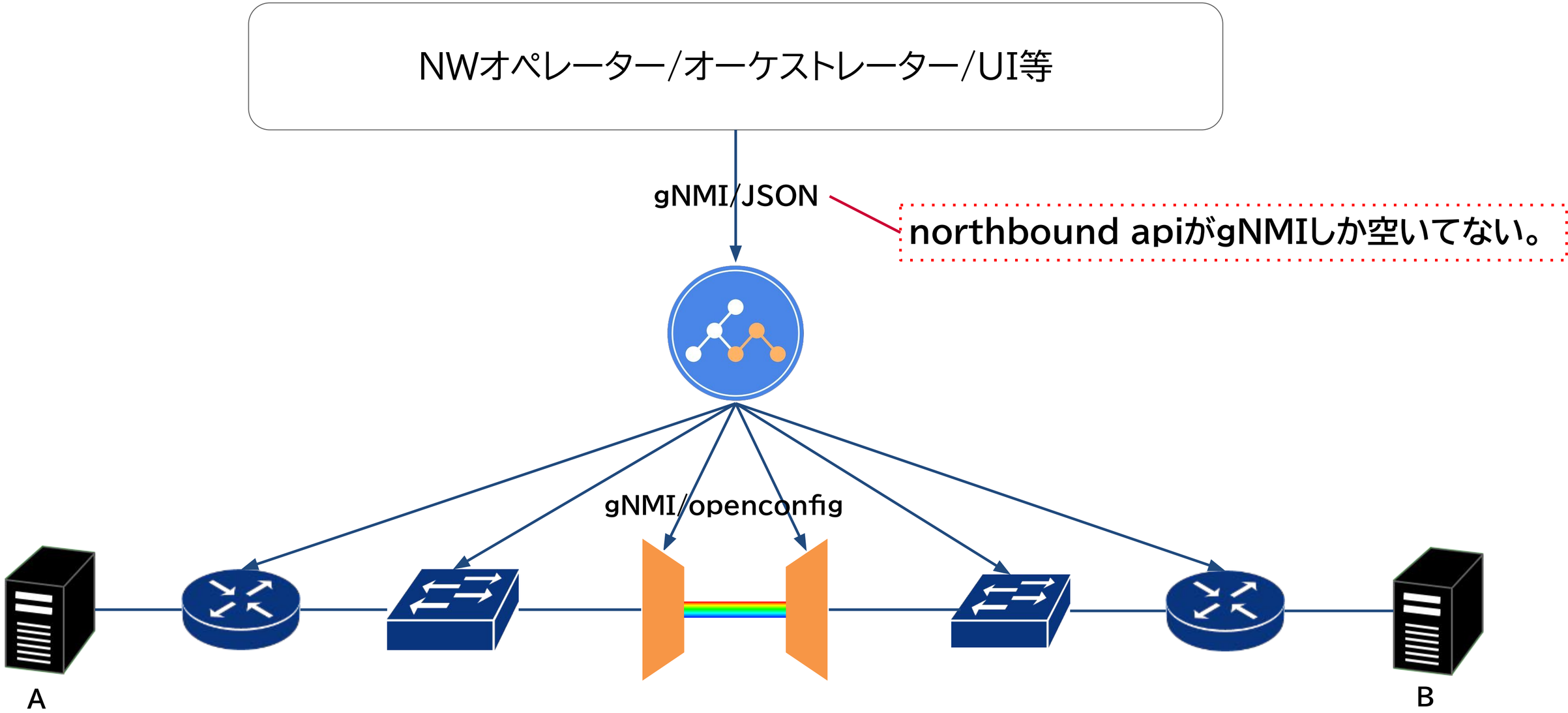
→ 解決策③ Xpathによる装置コンフィグ導出

⛰️ 課題④ 削除オーダーが実施できない

→ 解決策④ 既存データとのCRUDを意識した差分計算（検討中）

© NTT Communications Corporation All Rights Reserved.

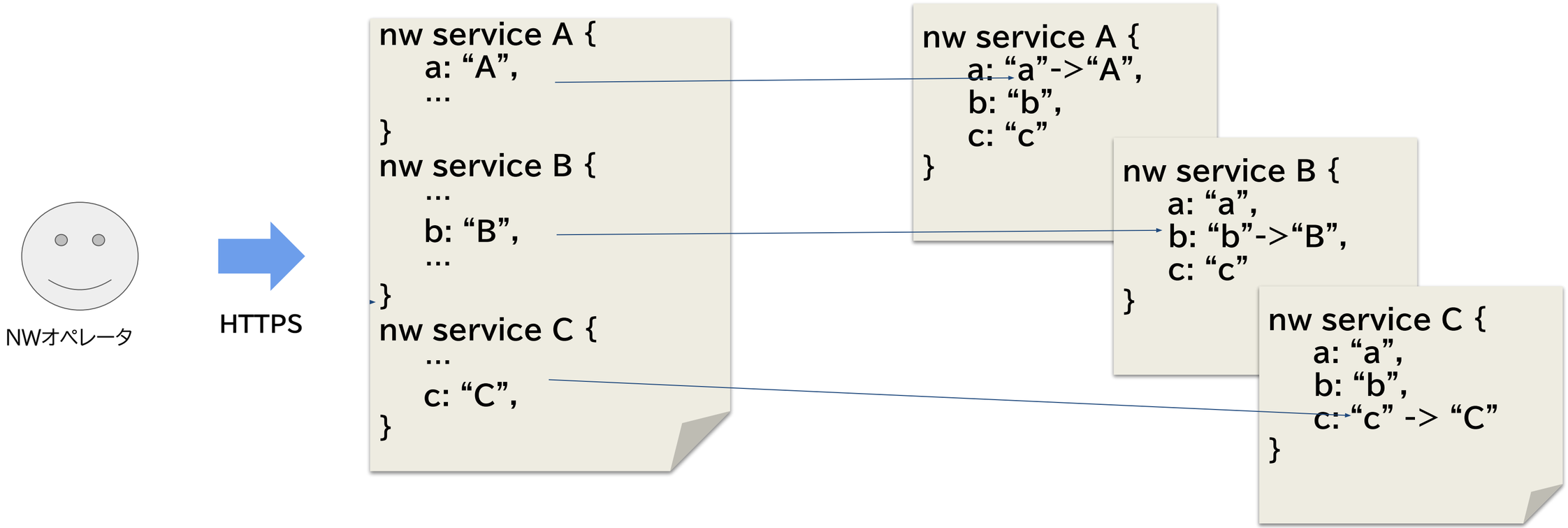
課題① 外部システムとの連携プロトコルが限られている



© NTT Communications Corporation All Rights Reserved.

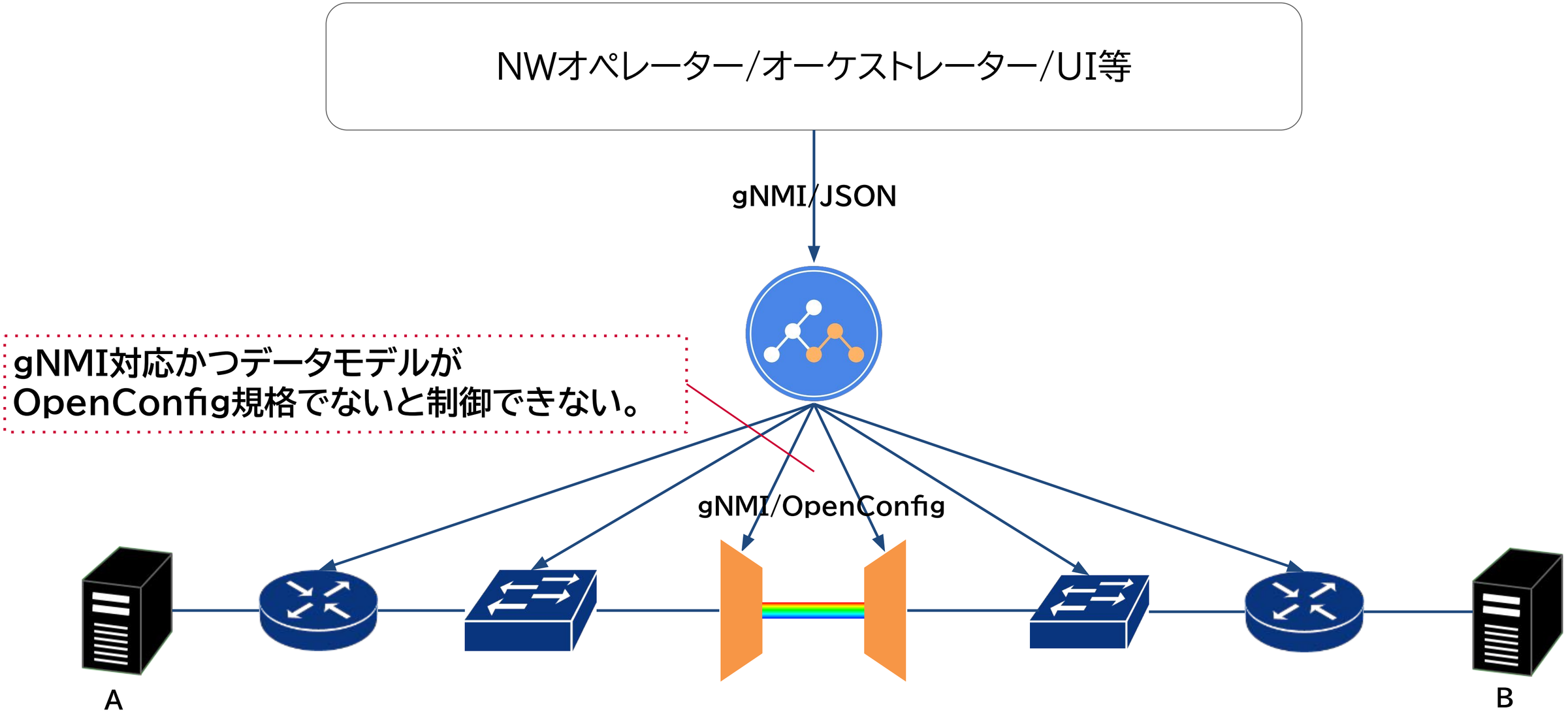
解決策① NorthboundのWebAPI化

- Web IFを追加し抽象モデルをJSONでHTTP送信できるように実装を変更



© NTT Communications Corporation All Rights Reserved.

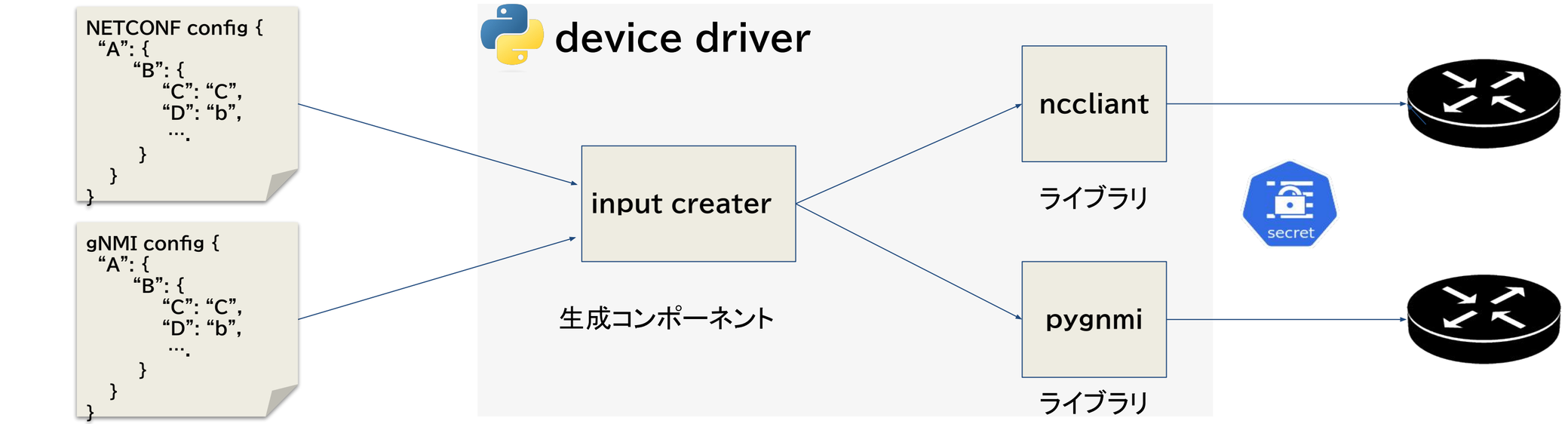
課題② 制御できる装置の種類が少ない



© NTT Communications Corporation All Rights Reserved.

解決策② 汎用的・拡張性の高いdriverの開発

- 装置から取得したJSONをモデルとして管理するように変更、driver内部でライブラリへの入力を生成するコンポーネントとライブラリ利用部分に分けて実装
- **NETCONFに対応、追加開発を実施すればrestconfへの対応も容易。**

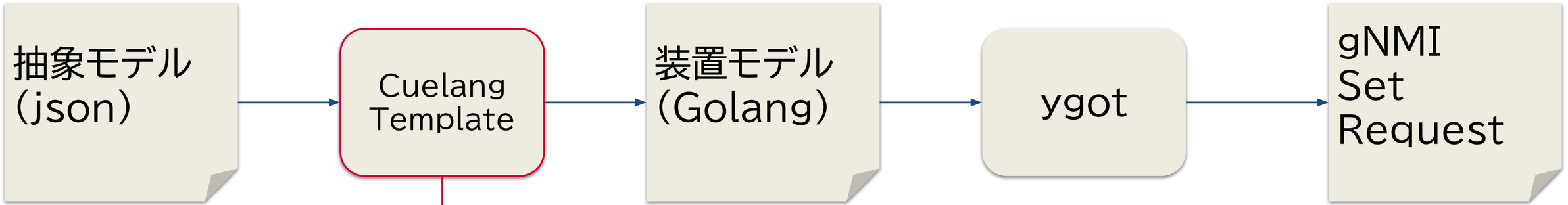


装置コンフィグをJSONとして管理

各種ライブラリからコンフィグ投入

© NTT Communications Corporation All Rights Reserved.

課題③ 課題: 装置設定導出ロジックの開発効率が悪い



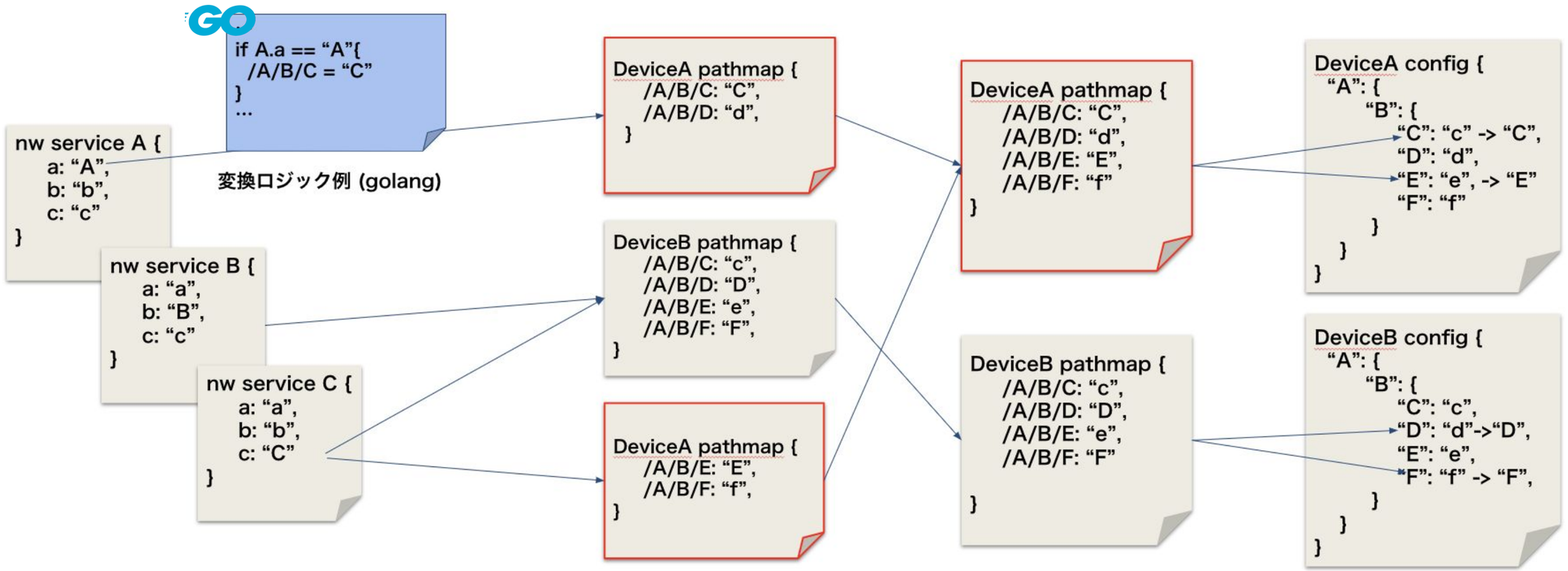
CuelangのTemplateを利用して実装する必要があるが、複雑なロジックを実装すると冗長な実装になりやすい。

Cuelang: https://cuelang.org/ (JSONのスーパーセットでスキーマ定義やバリデーションができる)

© NTT Communications Corporation All Rights Reserved.

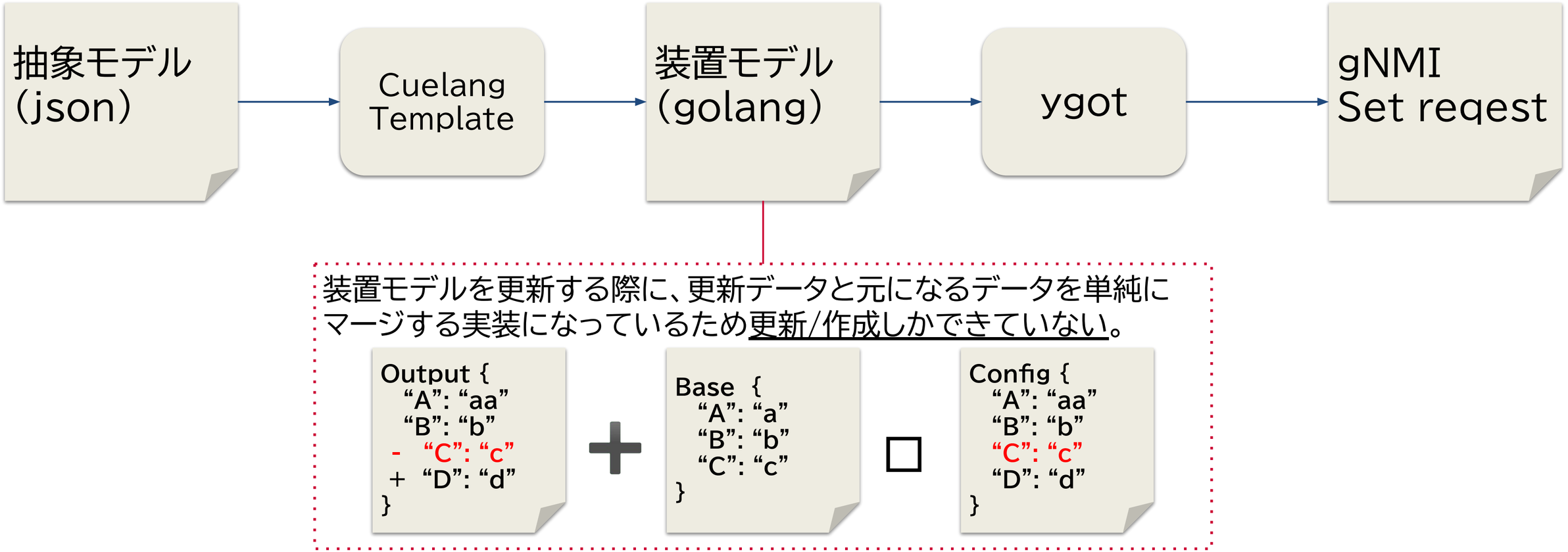
解決策③ Xpathによる装置コンフィグ導出

- 装置コンフィグ導出の際、JSONからgolangによってpathmap（Xpathと値のセット）に変換し、装置コンフィグのJSONを更新することで導出する実装に変更。



© NTT Communications Corporation All Rights Reserved.

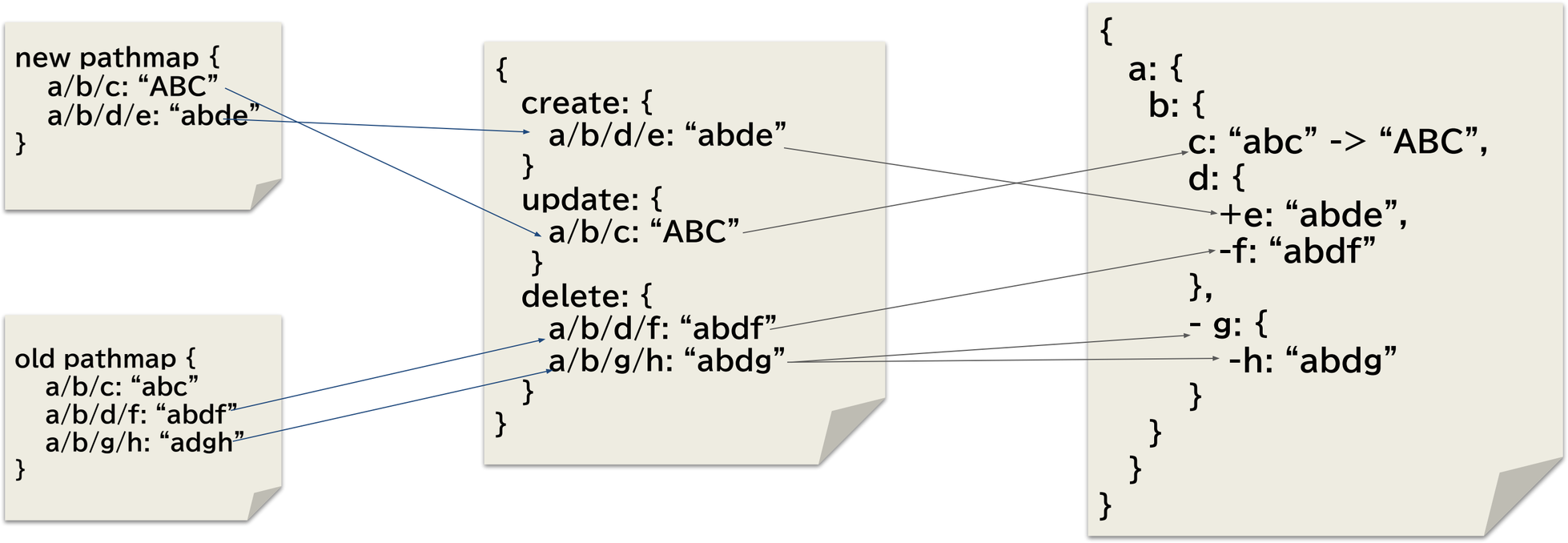
課題④ 削除オーダーが実施できない



© NTT Communications Corporation All Rights Reserved.

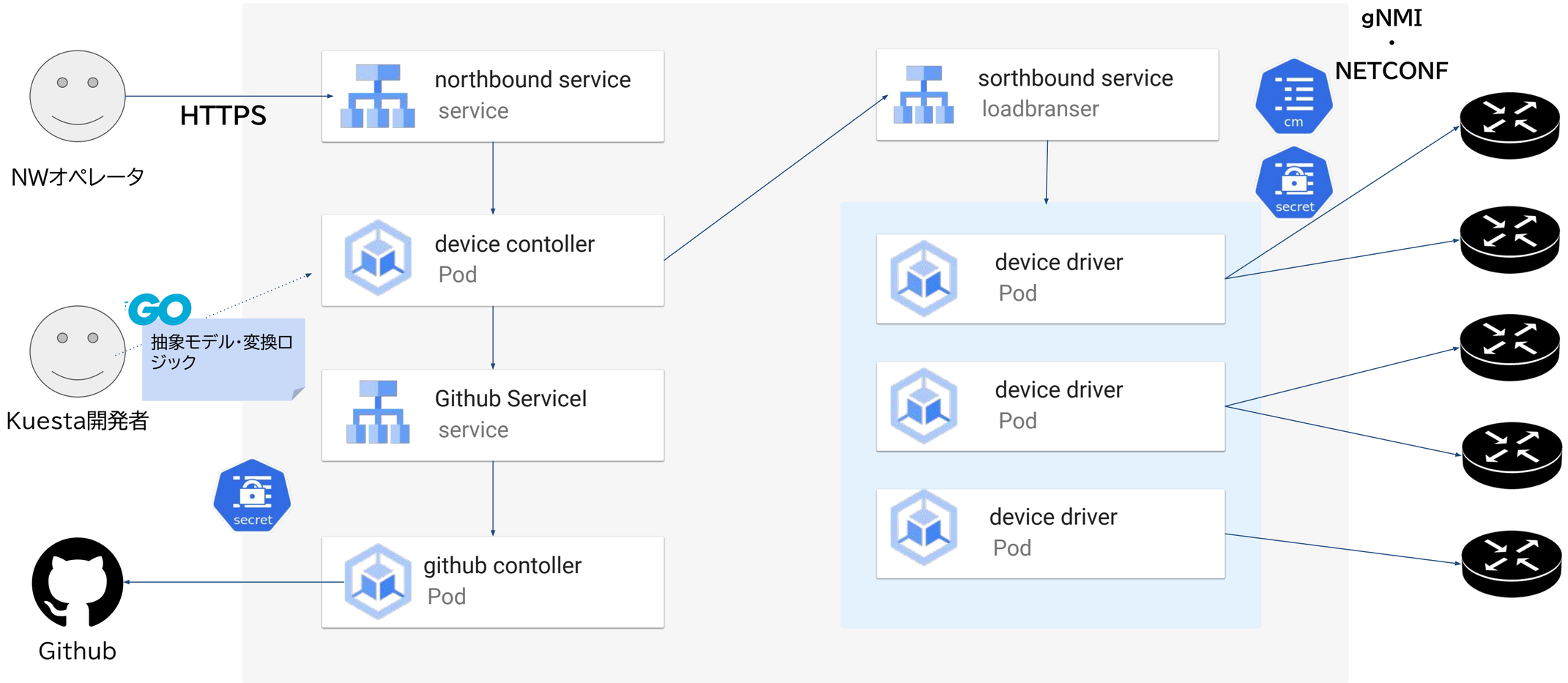
解決策④ 既存データとのCRUDを意識した差分計算

- 装置のPathmapを更新するときに、既にPathmapが存在する場合はデータとの差分を計算
- **create, update, deleteに分けて更新データを装置のJSONを更新**
  - **装置のコンフィグの削除が可能になった**
  - **ただし、まだ問題あるため現在議論中**(親要素にKuestaから入れてない要素があった場合の考慮、単にKuesta管理から外したい場合(pathmapからは消すがdeleteはしたくない)場合のオプション追加)



© NTT Communications Corporation All Rights Reserved.

新Kuestaアーキテクチャ 概要図



© NTT Communications Corporation All Rights Reserved.

これからの取り組み・まとめ

© NTT Communications Corporation All Rights Reserved.

実用化に向けて超えたい山はまだまだある

削除オーダーが実施できない（引き続き）

・親要素にKuestaから入れてない要素があった場合の考慮

・単にKuesta管理から外したい場合（pathmapからは消すがdeleteはしたくない）場合のオプション追加

装置とKuesta管理データとの同期機能がない

・現状、実機のコンフィグとKuestaの管理データがずれた場合、同期機能がないため対応できない

導出した装置設定のバリデーションチェック

・装置にリクエストを投げる前に装置設定値のバリデーションチェックができない

アイデア募集中です！！！

設定投入前に装置設定の変更差分を確認する機能がない

・設定を投入する前に装置コンフィグがどのように変更されるかオペレーターが確認できない

© NTT Communications Corporation All Rights Reserved.

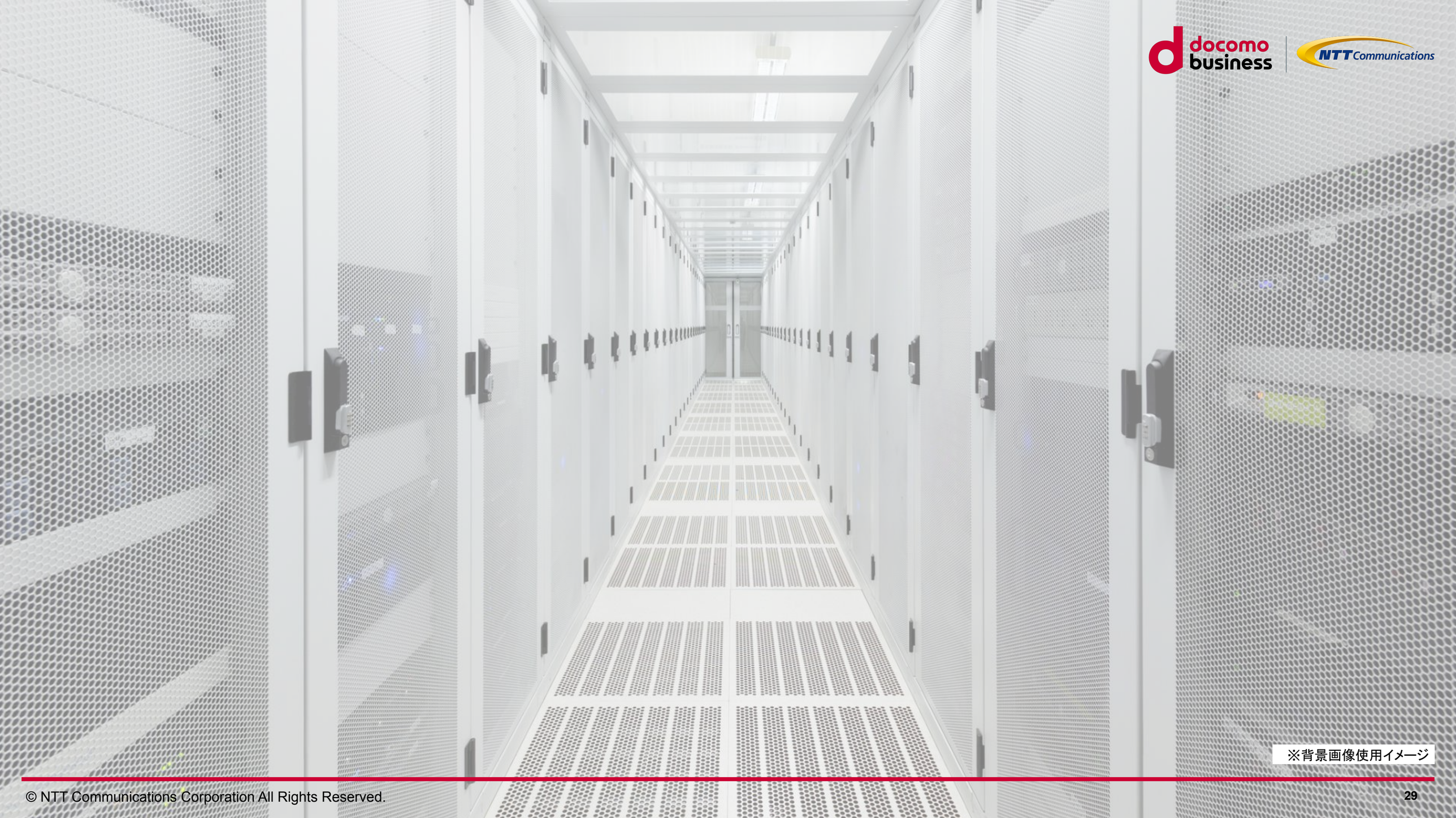
まとめ

- K8s + Githubを使って宣言的NWコントローラーを開発しました
  - ネットワーク機器のコンフィグを抽象化したモデルによって操作
  - K8sリソースによる負荷分散・セキュアな接続情報管理
- フィールドトライアル実施に向けて課題を整理、追加開発を実施中
  - 今日発表内容は次年度Q1、Q2、Q3で順次リリース予定
  - フィールドトライアルで実績を積んで、2025年度に事業導入を目指す
- コミュニティで一緒に開発・検証してくださる方募集中です。**是非Star、コメントよろしくお願い致します！！**🙇
  - OSSレポジトリ: https://github.com/nttcom/kuesta
  - Getting started: https://nttcom.github.io/kuesta-website/docs/getting-started/

© NTT Communications Corporation All Rights Reserved.

ご清聴ありがとうございました！！

© NTT Communications Corporation All Rights Reserved.



※背景画像使用イメージ

© NTT Communications Corporation All Rights Reserved.