# クラウドネイティブな新しいOSSのネットワークコントローラ『K-SOT』の3年間の進化と実用化に向けた取り組み

2024年10月10日

NTTコミュニケーションズ株式会社

# 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: K-SOT(ケーソット)開発
- メールアドレス: tatsuki.sakai@ntt.com

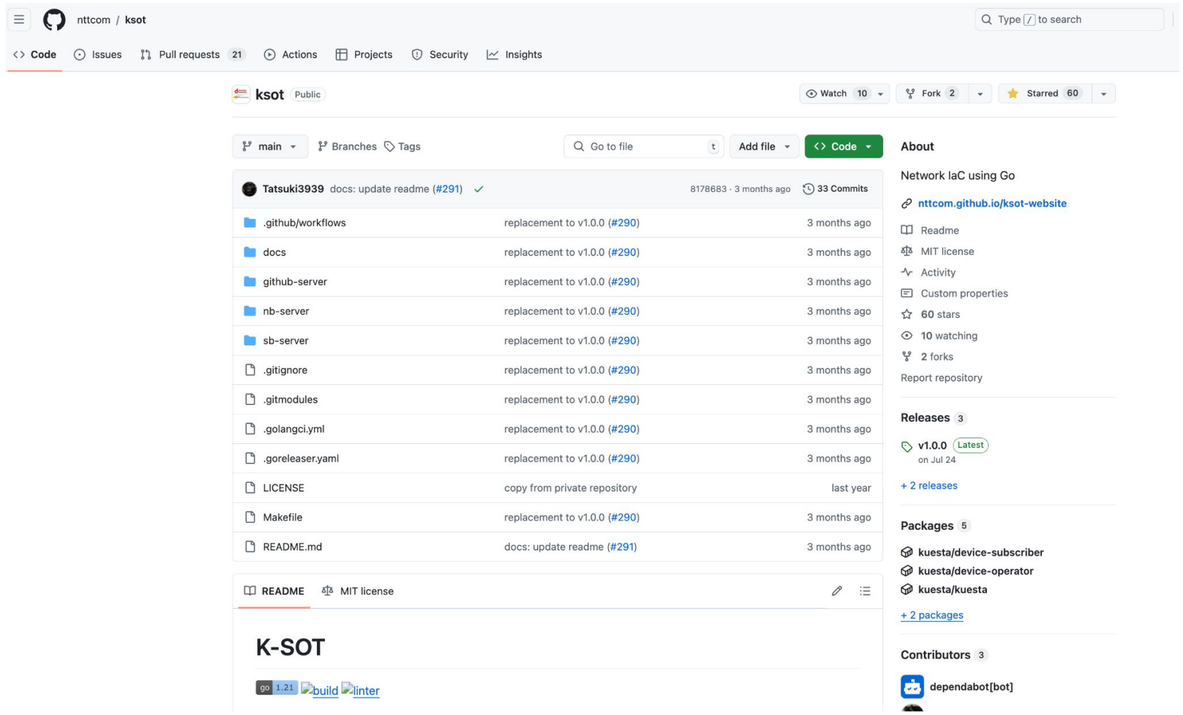
# 自己紹介

- 為末 航平（ためすえ こうへい）
- 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
- 略歴
  - 2019～2023
    - ソフトバンク株式会社入社
      - 伝送NWの保守・運用
  - 2023～2024
    - NTTコミュニケーションズ株式会社へ転職
      - K-SOT（ケーソット）開発、伝送装置検証
        - ソフトウェア開発は勉強中！
- メールアドレス: k.tamesue@ntt.com

# K-SOT(ケーソット)概要/今までの取り組み

# K-SOT（ケーソット）とは

- K8s、GitHub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




現在のOSSレポジトリ

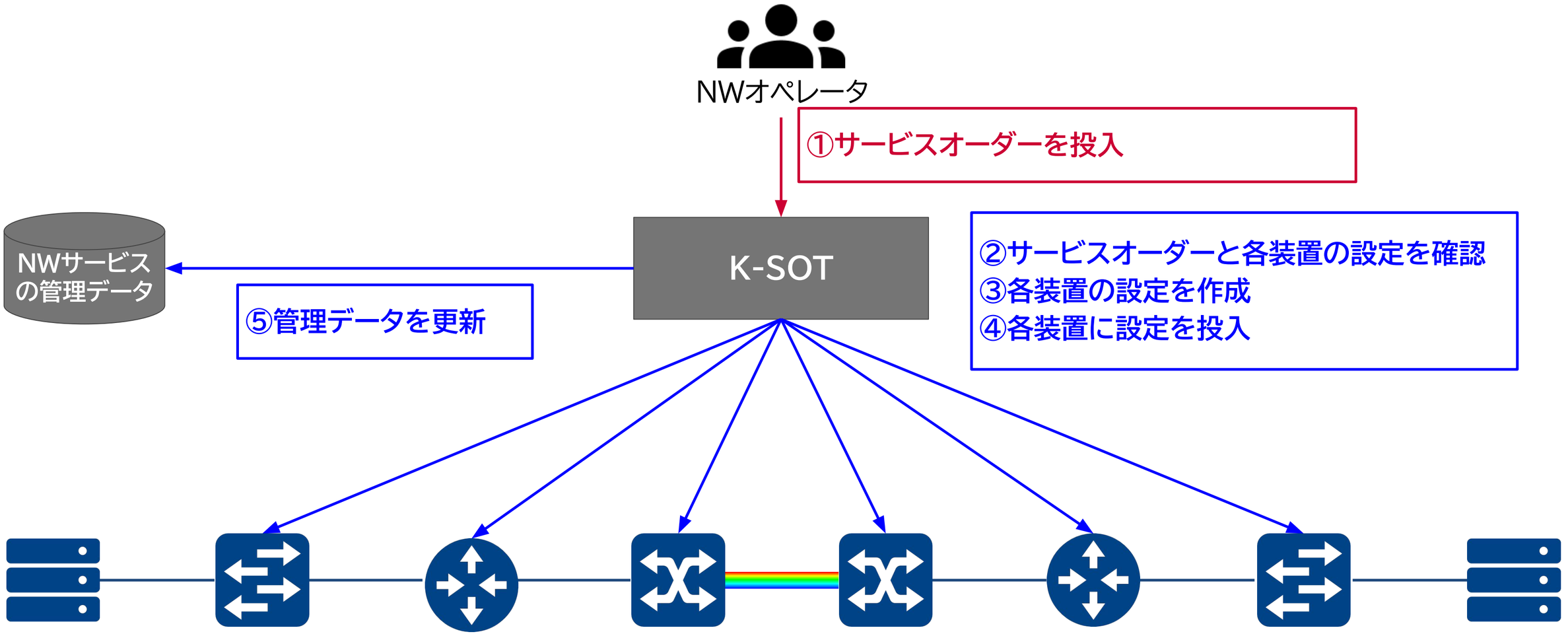
※名前をK-SOTに変更してます！
（2024.04～商標出願中）

[オフィシャルサイト](https://nttcom.github.io/ksot-website/ja/)

https://nttcom.github.io/ksot-website/ja/

# K-SOTを使った構成

- K8s、GitHub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
- NWオペレーターが各装置の接続情報や設定手順を把握していなくても、オペレーションが可能

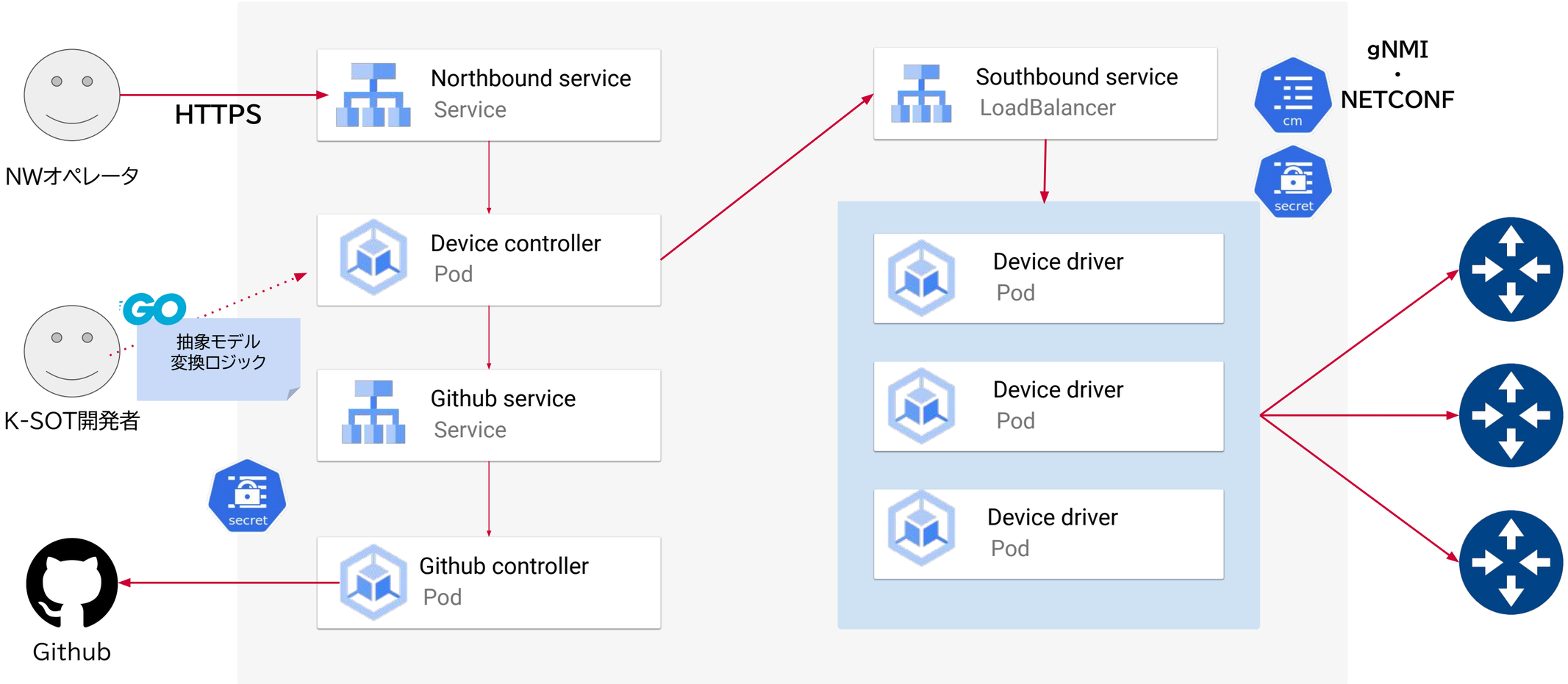


# 3年間の取り組み

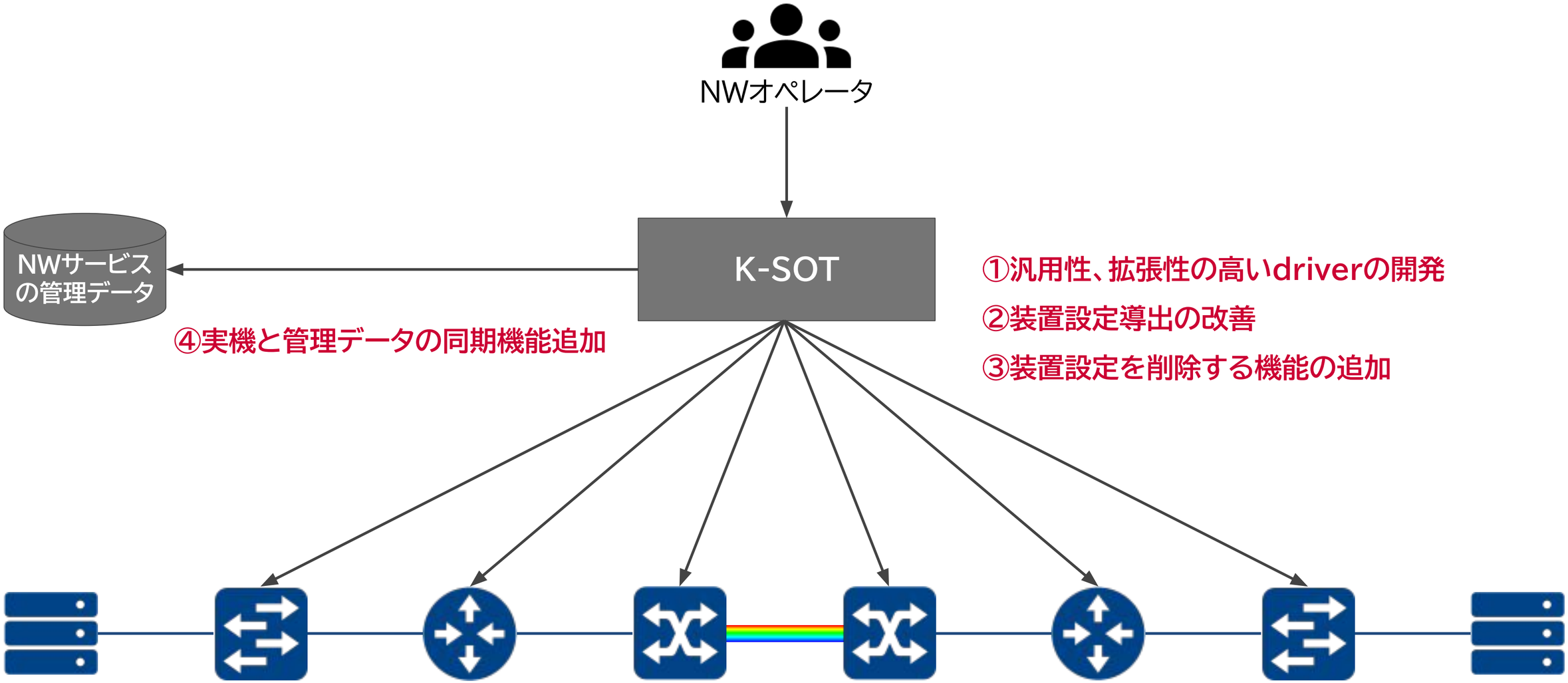
| | | |
|---|---|---|
| ONIC2022 | K-SOT初版完成 | ✓ クラウドネイティブ技術（K8s, GitHub）をNWコントローラーで採用 |
| | | ✓ gNMIによるマルチベンダ・デバイスの制御 |
| ONIC2023 | OSS化/<br>伝送装置の動作実績 | ✓ WEB API対応 |
| | | ✓ 伝送装置のフィールド検証 |
| | | ✓ 新バージョン（v1.0.0）向けアーキテクチャ考案 |
| ONIC2024 | V1.0.0リリース/<br>転送装置の動作実績 | ✓ 汎用性、拡張性の高いdriverの開発 |
| | | ✓ 装置設定導出の改善 |
| | | ✓ 装置設定を削除する機能の追加 |
| | | ✓ 実機と管理データの同期機能追加 |
| | | ✓ 転送装置のラボ検証 |

# 新バージョン(V1.0.0)リリースまでの取り組み

# 新バージョン(V1.0.0) K-SOTアーキテクチャ 概要図

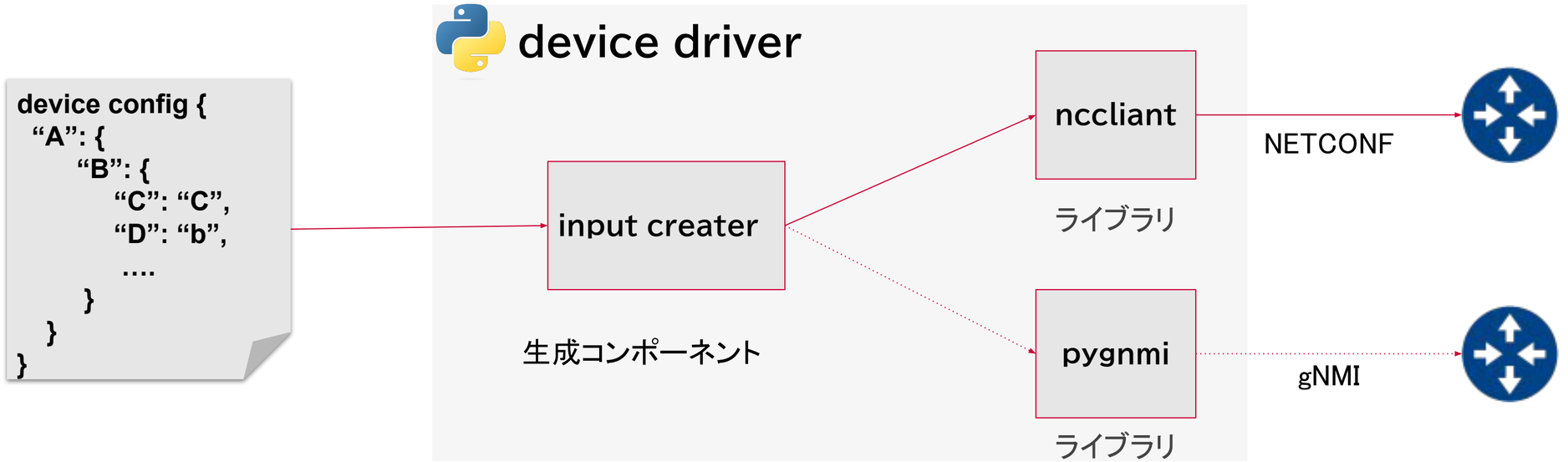


# 新バージョン(V1.0.0)の追加機能



# ①汎用性、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（gNMI, restconfへの対応も容易）

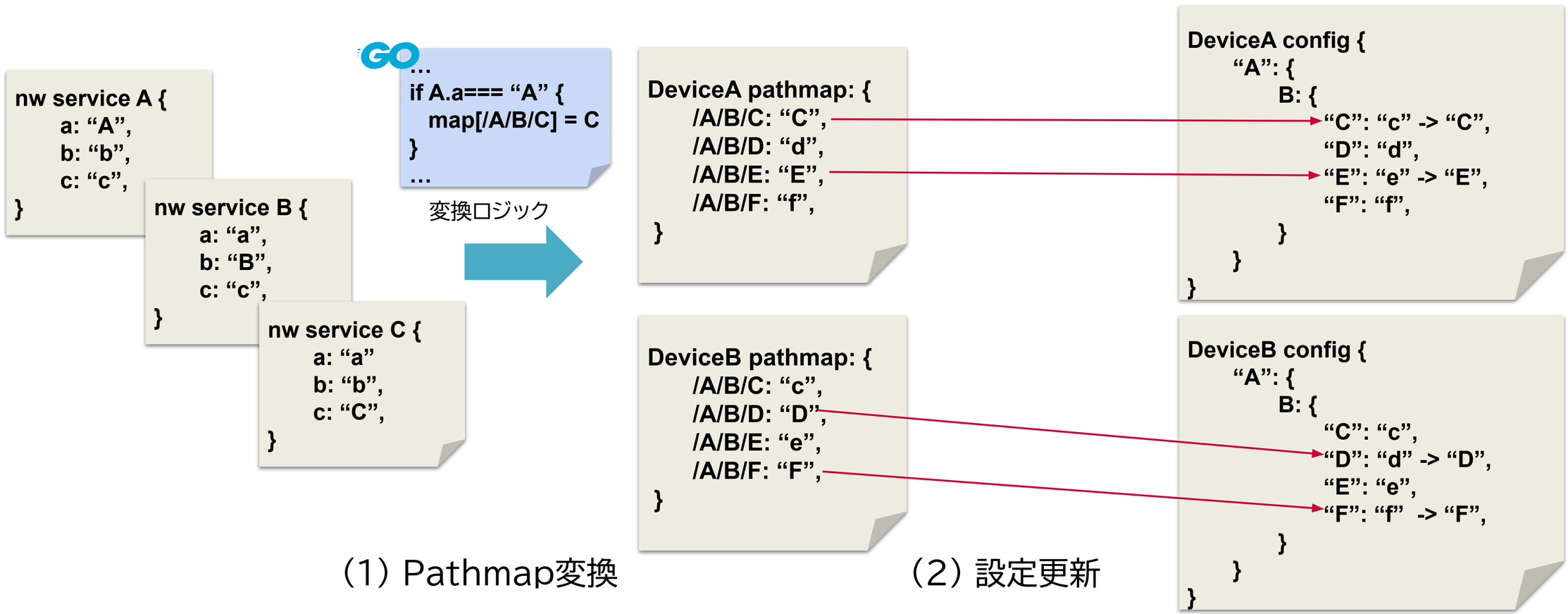



装置コンフィグをJSONとして管理

各種ライブラリからコンフィグ投入

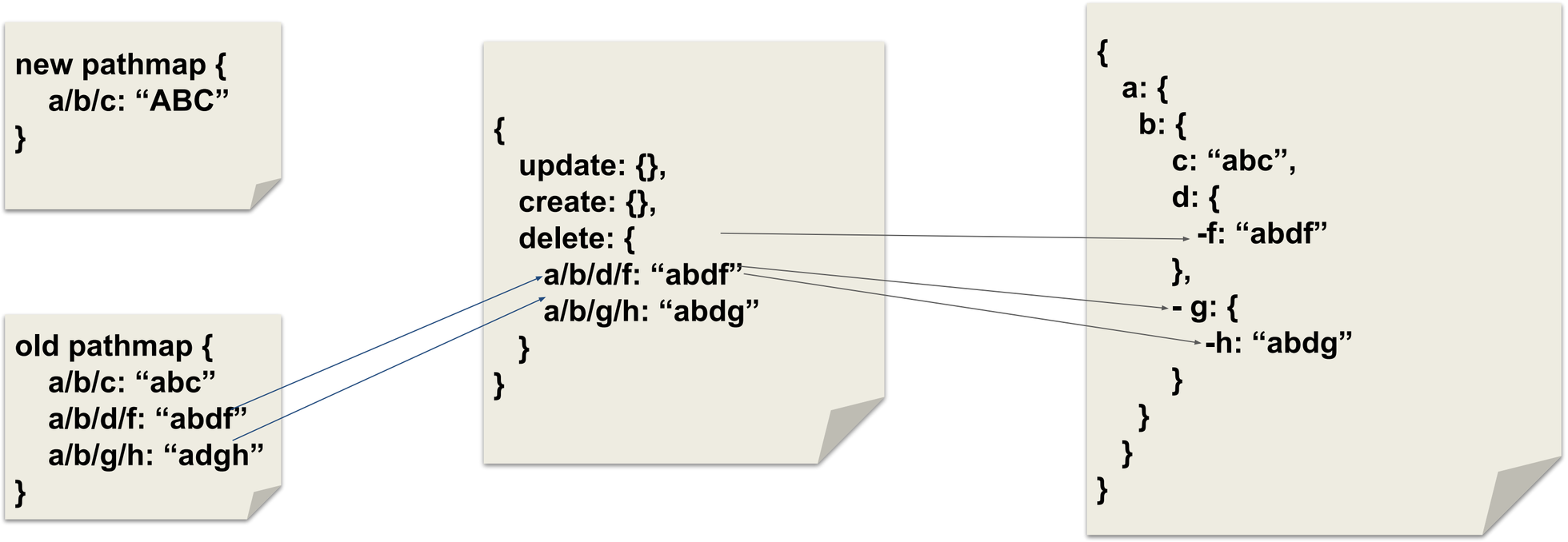
# ②装置設定導出の改善

- 装置コンフィグ導出の際、JSONからGoによってPathmap（YANGの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
  - ✓ 変換ロジックをCue（データスキーマ言語）で実装していた旧方式と比較し簡潔な実装ができるようになった



# ③装置設定を削除する機能の追加

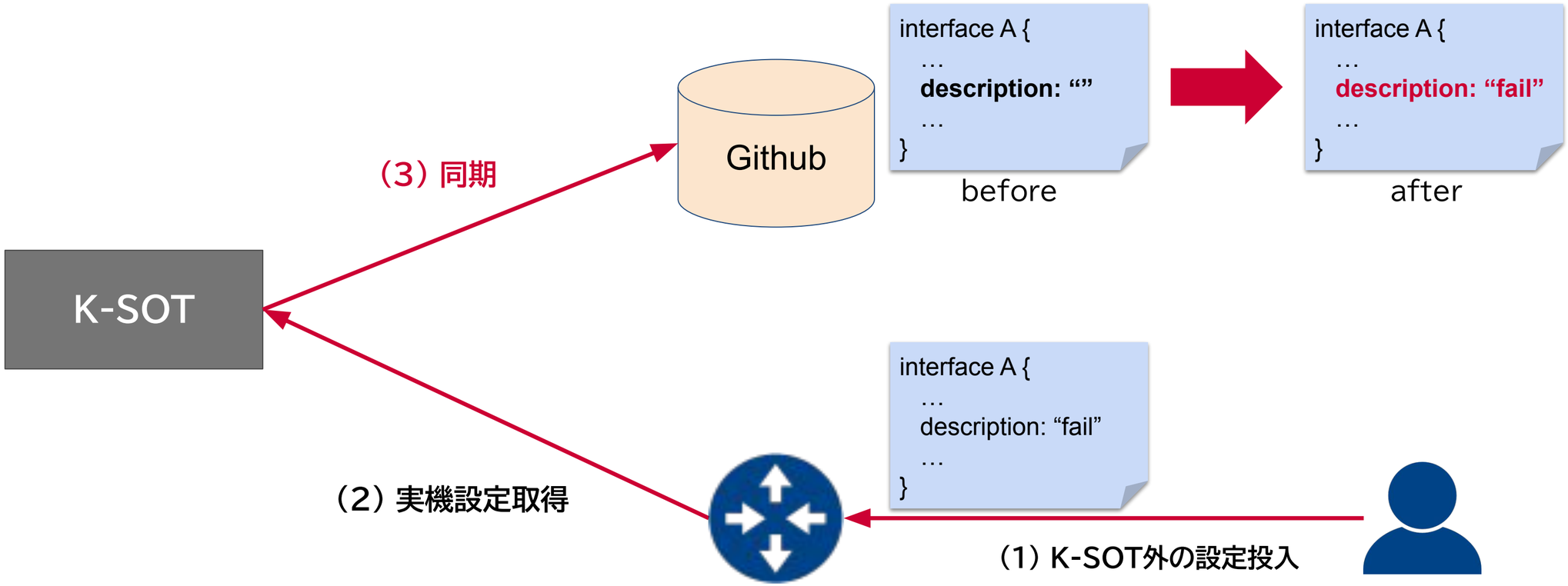
- 装置のPathmapを更新時、データの差分を計算し、create, update, deleteに分けて変更内容を装置のJSONに反映させる機能を実装
  - ✓ 従来では更新/作成のみしかできていなかったが削除にも対応ができるようになった




（1） 差分導出　　（2） 変更反映

# ④実機と管理データの同期機能追加

- 装置のコンフィグをGithubに吸い上げるAPIを追加
  - ✓ コントローラー外から管理装置への設定変更やトランザクションが完了しなかった場合(一部のケース)に管理データを同期させることができるようになった



# 転送装置のラボ検証

# ラボ検証

- 検証概要
  - KSOTからのルータ制御の確認のため、商用でも採用実績のある仮想ルータ2台について、K-SOTから制御（開通、廃止）を検証

- 想定するサービス
  - 2つの拠点で構成されるL3接続サービスで、2種類で構成される①WAN（RT間）、②LAN（宅内）
  - ユーザは「どの拠点とどの拠点を接続したいか」を指定することで、指定する拠点間のルーティングが設定され、宅内アドレス間で疎通


本検証の構成図

# 検証したNWサービスのYANGモデル

- 2つのサービスモデルを用意
  - locationサービス: 各拠点の装置を管理するモデル
  - static-routiongサービス: 拠点間のルーティングを管理するモデル、接続したい拠点をconnectedに指定する。

```
module locations {
    list locations {
        key "name";
        // 拠点の名称
        leaf name {
            type string;
        }
        // 拠点にある装置群
        container devices {
            list devices {
                // ip, interface情報を記載
                …
            }
        }
    }
}
```

locationサービス

```
module static-routings {
    list static-routings{
        key "name";
        // 拠点の名称
        leaf name {
            type string;
        }
        leaf-list connected{
            // 接続する拠点の名称群
            type string;
        }
    }
}
```

static-routingsサービス

# 検証での装置コンフィグ設定フロー

3. 変更があった装置に対して、
コンフィグ投入

```
{
  “staticRoutings:staticRoutings": [
      {
            "name": "拠点1",
            "connected": [
              "拠点2"
            ]
      }
  ]
}
```

1. static-routingサービス
を拠点1と拠点2が繋がるよ
うに更新

K-SOT

2. K-SOTがlocationサービ
スから各拠点の装置情報を取得
し、装置コンフィグを計算

```
{
  "locations:locations": [
      {
            "name": "拠点1",
            "devices": [
                  {
                        wan": "10.0.0.1",
                        interfaces[...]
                        …
                  }
            ]
      }
  ]
}
```

拠点1

拠点2

# 検証結果・まとめ

- 社内のラボ環境の2拠点で、K-SOTを用いてL3サービスの開通、廃止を一通り実施することができた
  - **転送装置環境での動作実績を得ることができた！！**
- 一方で課題についても再確認
  - 開発ライブラリが整備されておらず、**サービスモデルの開発に時間がかかった。**
    - 今回の検証では複数のサービスモデルを用意する必要があり、かなり手間取った
    - サービスモデル向けの開発ライブラリを整備する必要がある
  - **サービス変更の処理時間が遅い。**
    - 数行の変更で2分半, 装置に変更がない（設定投入が走らない）場合でも同じくらい（2分）時間がかかる
    - サービスの差分検出やGithubにデータを上げる部分で処理が重くなっている予想で、ロガー整備して各処理時間の計測・分析後、高速化の検討が必要

# 今後の取り組み・まとめ

# 今後の取り組み

- **開発効率の改善（検証から得た課題）**
  - サービスモデルの装置コンフィグへの変換部分について、現状Goを用いたスクラッチ開発になっており効率が悪い
    ため、ライブラリを整備し、より効率的かつ一貫した手順でサービス開発をできるようにする予定
  - 処理時間が遅いため、解析してアルゴリズムの見直し・改善を実施する予定
- **システムログ整備（検証から得た課題に関連）**
  - 現在各Podがログを出力している状況で、ログサーバー相当のPodが存在しないため、各処理の分析やデバッグが難しい。→ fluentbitを導入することで1つのPodにログを集約する機能を実装中
  - 出力ログについてもフォーマットが曖昧かつ実装も未完全なので、見直す予定
- **障害対応**
  - 管理している装置の故障やNWエラー時の対応オペレーション整備が未実施
  - 障害バリエーション毎に検討予定
- **導入先検討**
  - まずは社内で既にコントローラーを使っている領域・サービスについて、NETCONF対応装置で構成されている環境をスコープにしていく予定

# まとめ

- K8s + Githubを使って宣言的NWコントローラー『K-SOT』を開発中
  - 今年度はV1.0.0リリースに向けて課題を整理して、追加開発を実施
  - NETCONF対応や設定の削除機能追加等、**段々と実用的なNWコントローラーになりつつある**
- 装置検証も昨年度から段階的に実施
  - 前年度の伝送環境でのフィールド検証に加えて、今年度はラボ環境でのL3サービス検証も実施し**着実に動作実績を積んでいる。**
  - 引き続き検証・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/ksot/
  - Getting started: https://nttcom.github.io/ksot-website/