JANOG55発表

# Kubernetesを活用した、Intent-Drivenアプローチに基づくネットワーク設計・作業自動化の取り組み

KDDI株式会社

渡辺 裕太　福井 良平　佐藤 亮平

2025年1月23日

Tomorrow, Together　KDDI　おもしろいほうの未来へ。　au

# 自己紹介



**渡辺　裕太**
Yuta Watanabe

**福井　良平**
Ryohei Fukui

**佐藤　亮平**
Ryohei Sato

| | 渡辺　裕太 | 福井　良平 | 佐藤　亮平 |
|---|---|---|---|
| 所属 | 次世代ネットワーク整備室 | 次世代ネットワーク整備室 | IPネットワーク部<br>兼 次世代ネットワーク整備室 |
| 経歴 | ◘ サービス保守（2018～2019）<br>IoT系サービス保守、監視実装<br>◘ クラウド開発（2019～2024）<br>クラウド（社内/AWS/OCI）上の<br>インフラ設計開発・SRE業務<br>◘ NW企画・戦略（2024～）<br>NW自動化検討 | ◘ NW運用（2015～2019）<br>FTTH網の運用・保守<br>◘ NW設計・開発（2019～2021）<br>AS2516網の開発<br>◘ NW企画・戦略（2021～）<br>将来NW検討 | ◘ NW運用（2013～2018）<br>AS2516網の運用・保守<br>◘ NW設計・開発（2018～）<br>自動化、将来NW検討、<br>ピアリング、RPKI、CONECT |
| JANOG参加歴 | 54@奈良<br>今回、初登壇 | 53@福岡、54@奈良<br>登壇2回目（54で登壇） | 42@三重～45@札幌、<br>53@福岡、54@奈良<br>登壇2回目（54で登壇） |

# アジェンダ

1. 背景・課題
2. 自律化の将来像
3. 自動化アプローチ概要
4. 技術実証（PoC）
5. 商用導入へ向けた課題・今後の展望
6. まとめ・議論したいポイント

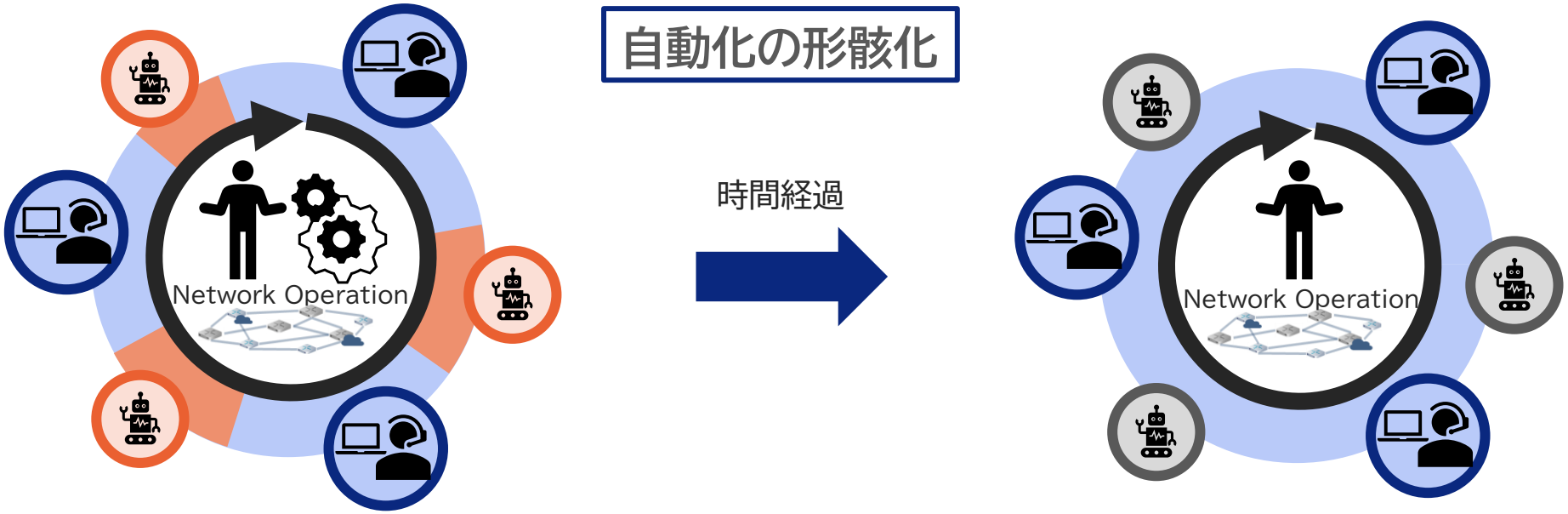


# 1. 背景・課題

# 課題

## 自動化から自律化へ

- 現状のNWはいまだ人手によるオペレーションが大部分を占めている
- 人手を楽にするだけの自動化では高度化していくNW運用に耐えられない
- 人手のプロセスから脱却した世界、手元の**自動化**ではなく統一した**自律化**が必要



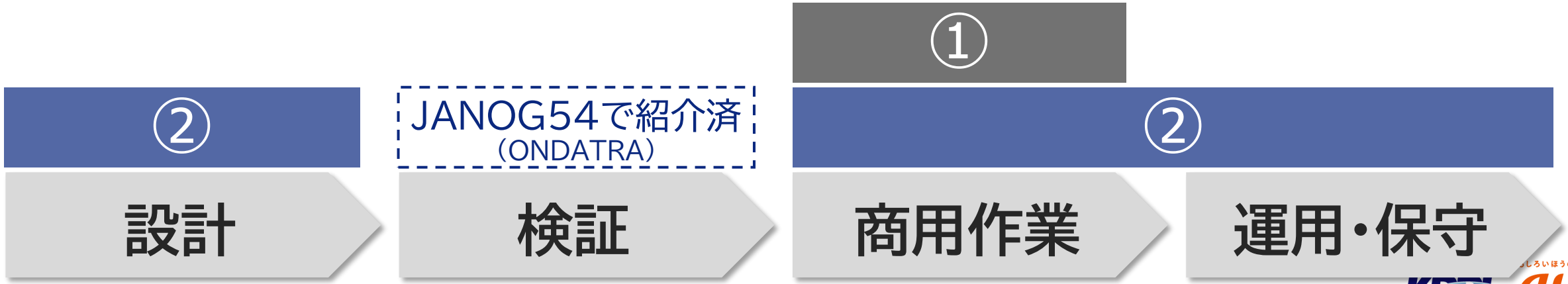

人手の作業を部分的に自動化
一時的な作業効率化を得られる

大きな更改等でメンテナンスが間に合わず形骸化
人手の運用に逆戻り（次の自動化が始まる）

# 【参考】前時間帯の弊社発表内容との違い

## 本取り組みは、中長期的にNW自律化を目指したもの

| | 取り組み①<br>ネットワーク作業自動化の道：<br>信頼性と効率性の両立 | 本発表<br>取り組み②<br>Kubernetesを活用した、<br>Intent-Drivenアプローチに基づく<br>ネットワーク設計・作業自動化の取り組み |
|---|---|---|
| 時間軸 | 短中期 | 中長期 |
| 対象業務<br>（下記フロー図参照） | 商用作業（準備含む） | 設計、商用作業（準備含む）、運用・保守<br>（検証自動化と併せ、NW自律化を図る） |
| 対象設備<br>（新規/レガシー） | 新規/レガシー問わない | 新規設備から対応予定<br>（将来的にはレガシー含む） |



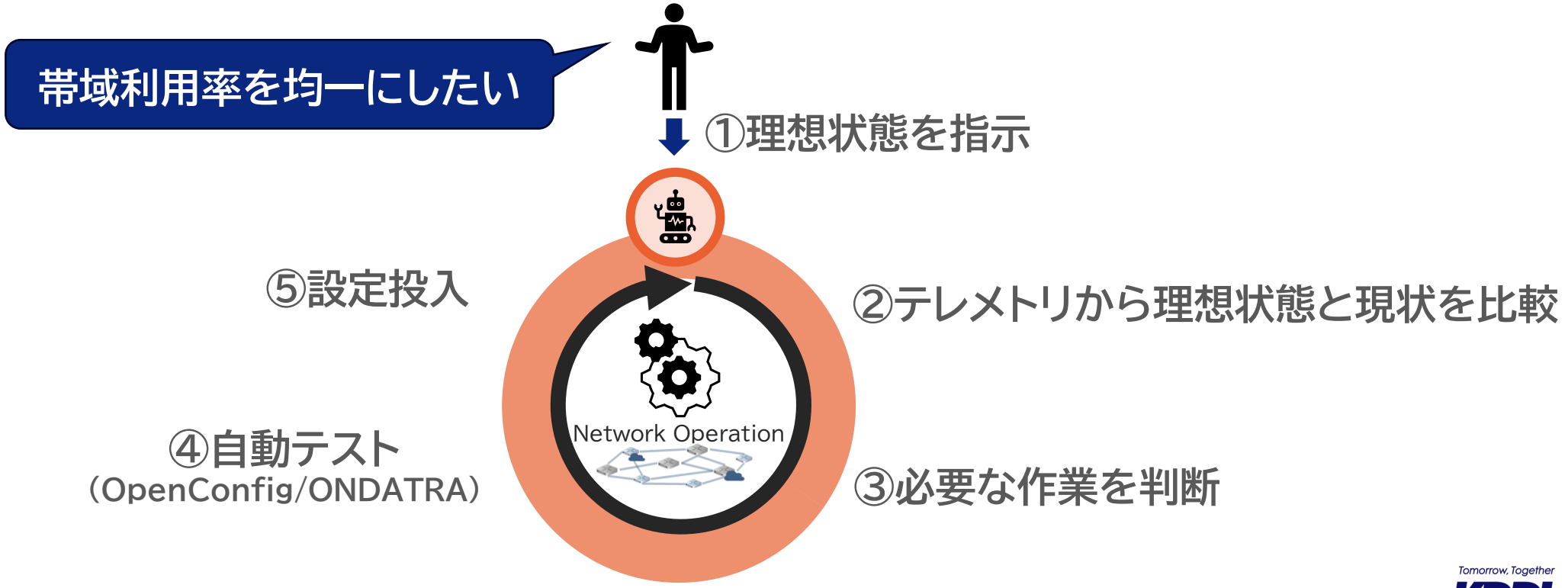


# 2. 自律化の将来像

# 目指している自律化の全体像

## 人手のプロセス実行からの脱却

- 人はネットワークの理想状態を宣言、テレメトリから自律的に判断
- NWに是正が必要な場合は新規コンフィグを自動生成、検証NW上で自動テストし商用適用までを自律的に行う

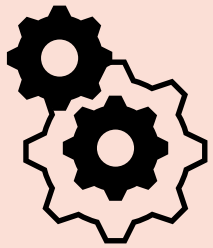


# 自動化レベルのステップ

## 自動化レベルを引き上げ、段階的に完全自動化(＝自律化)を目指す

STEP1 → STEP2 → STEP3 → STEP4

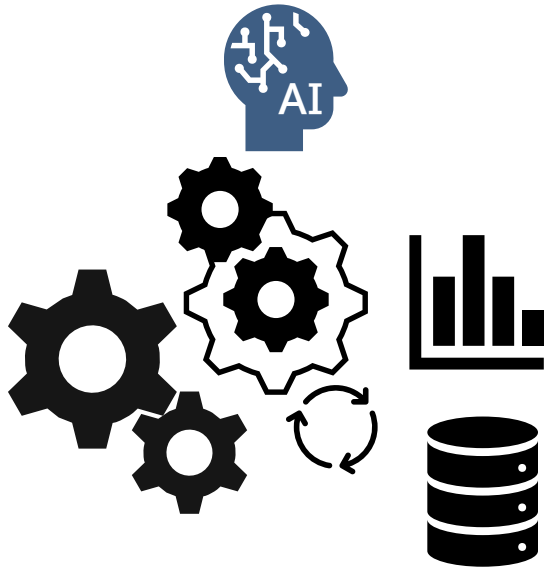
**今回のPOC対象**



ある程度具体的な指示から
自動コンフィグ生成＋投入

テレメトリからの自律的判断
構成管理DBとの連携
自動テスト連携

より抽象的な指示への対応
適用作業範囲の拡大




AI活用による高度化

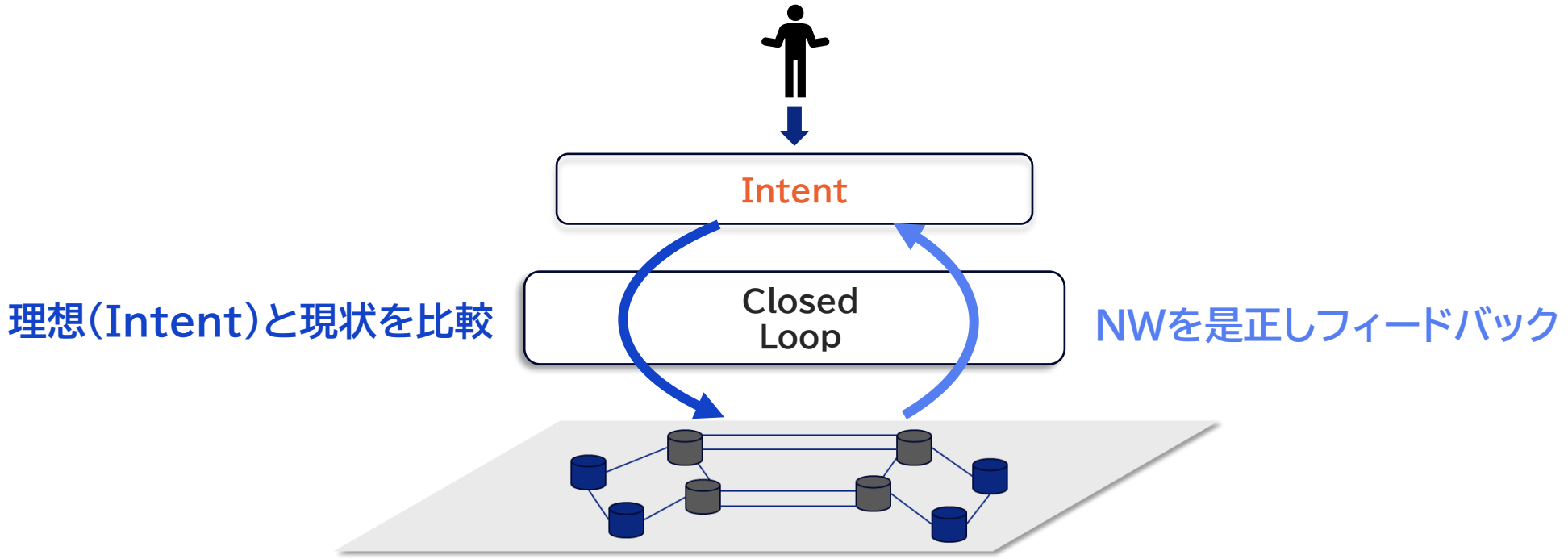


# 3. 自動化アプローチ概要

# Intent-Drivenの概要

## Intent-Drivenとは

- こうあるべきという意図（＝Intent）をベースとした考え方
- さまざまな団体において議論されており、Autonomous Network（自律運用ネットワーク）の実現アプローチとして注目されている

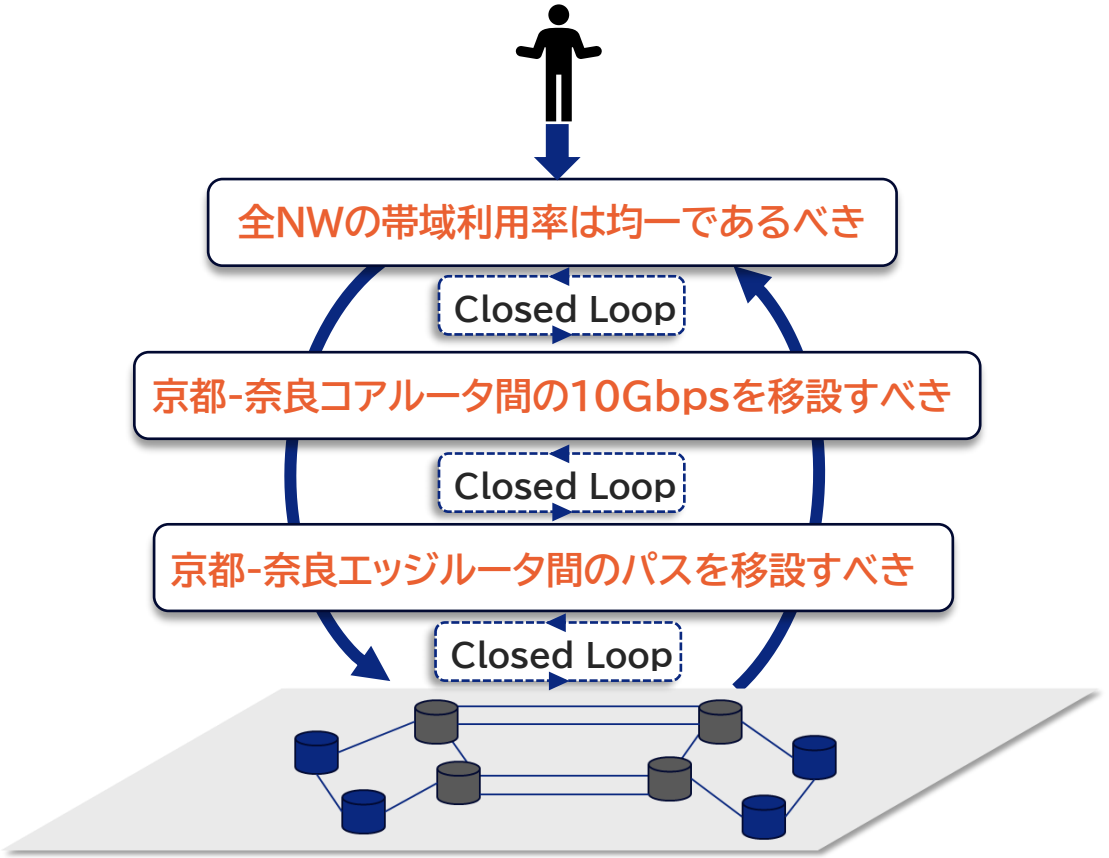



下記資料を参考に作成
https://www.tmforum.org/resources/introductory-guide/autonomous-networks-framework-v1-1-0-ig1218f/

# Intent-Drivenをネットワーク作業に適用した例

## （適用例）トラフィックチューニングの場合

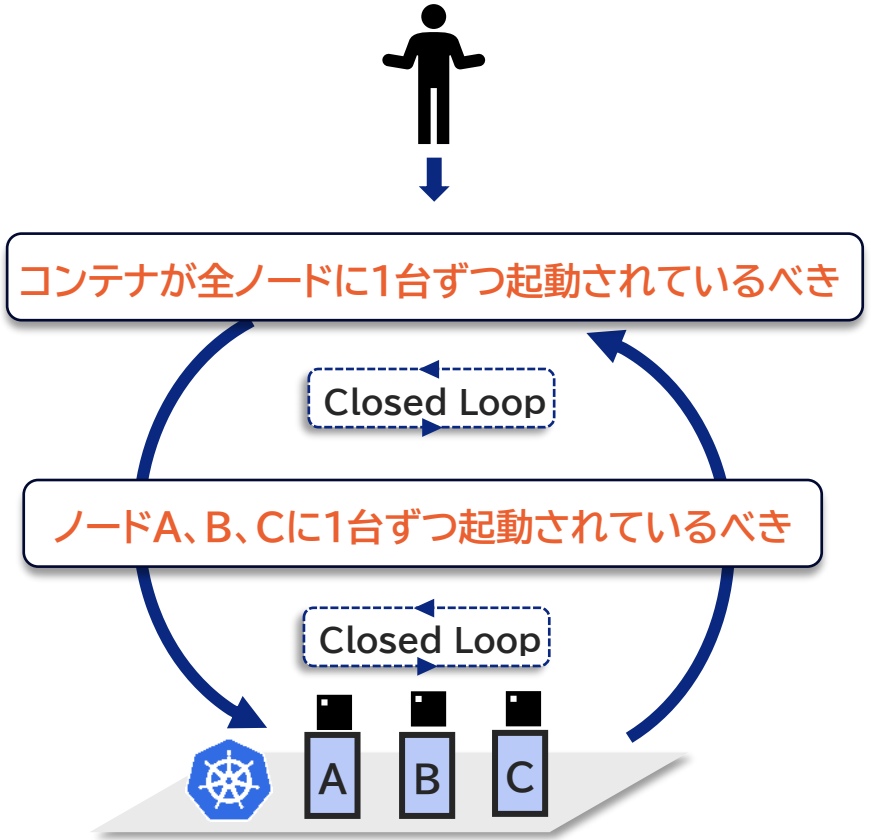
- Intentは抽象度に応じて段階的に定義される
- Closed Loopの中で理想と現状が比較され、Intentが具体化される



# Kubernetesとは

- 今回コンテナ管理ソフトウェアとして知られるKubernetesを活用
- KubernetesはIntent的な考え方に基づいた宣言的APIアーキテクチャ*をもつ

＊あるべき状態を定義し、それを維持するよう働く仕組み




Kubernetes(DaemonSet)

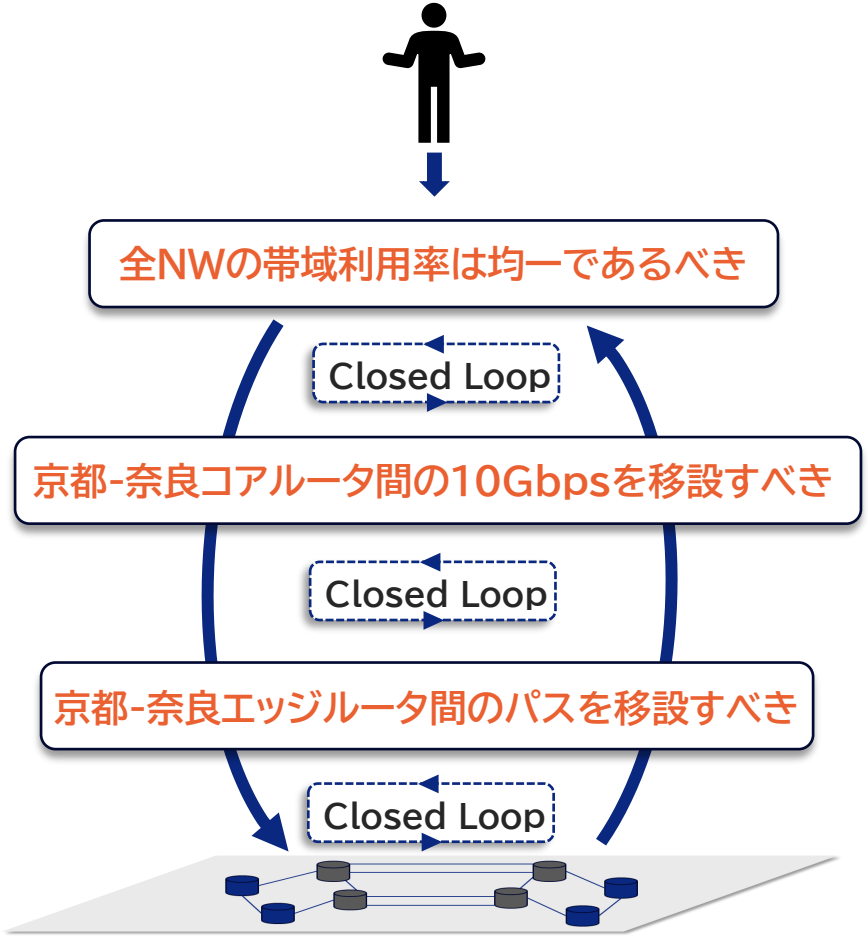
| | |
|---|---|
| | マニフェスト(Resource定義ファイル) |
| ds | Resource(DaemonSet) |
| c-m | Controller |
| pod | Resource(Pod) |
| sched | Controller |

コンテナ情報を宣言的に定義(Resource定義) ≒ Intent

# Custom Resource/Custom Controllerについて

## KubernetesのAPI拡張機能により、独自のResourceを定義可能

- ネットワークのあるべき状態(Intent)をCustom Resourceとして実装
- Intentの具体化や適用を行なう機能をCustom Controllerとして実装




Kubernetes(Intent)

| | |
|---|---|
| | マニフェスト(Resource定義ファイル) |
| crd | Custom Resource(High-Level Intent) |
| | Custom Controller |
| crd | Custom Resource(Mid-Level Intent) |
| | Custom Controller |
| crd | Custom Resource(Low-Level Intent) |
| | Custom Controller |


# 4. 技術実証（PoC）

# PoCにおける対象作業

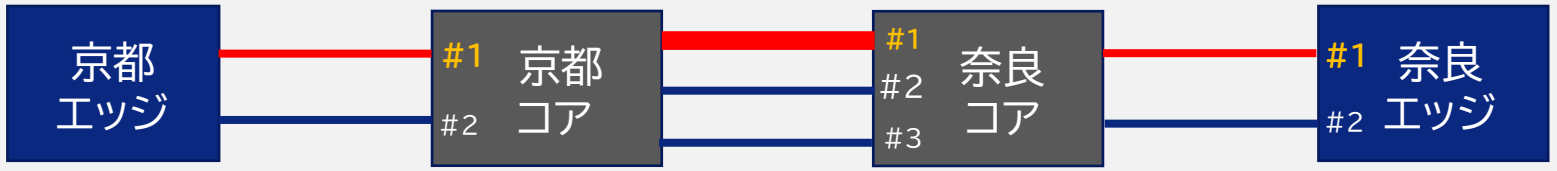
## IPバックボーン網のトラフィックチューニング作業

### 作業概要

- RSVP-TEによるトラフィックエンジニアリング
- 経由するノードのIFを明示的に指定し、**トラフィック量に応じて経由リンクを変更**することで、各ノード間のトラフィック量のバランスを保つ

作業稼働が多い ＆
トラフィック量に応じて
動的に設定変更が必要

事前状態




大阪 ➡ 京都（#1）➡ 奈良（#1）➡ 東京（#1）

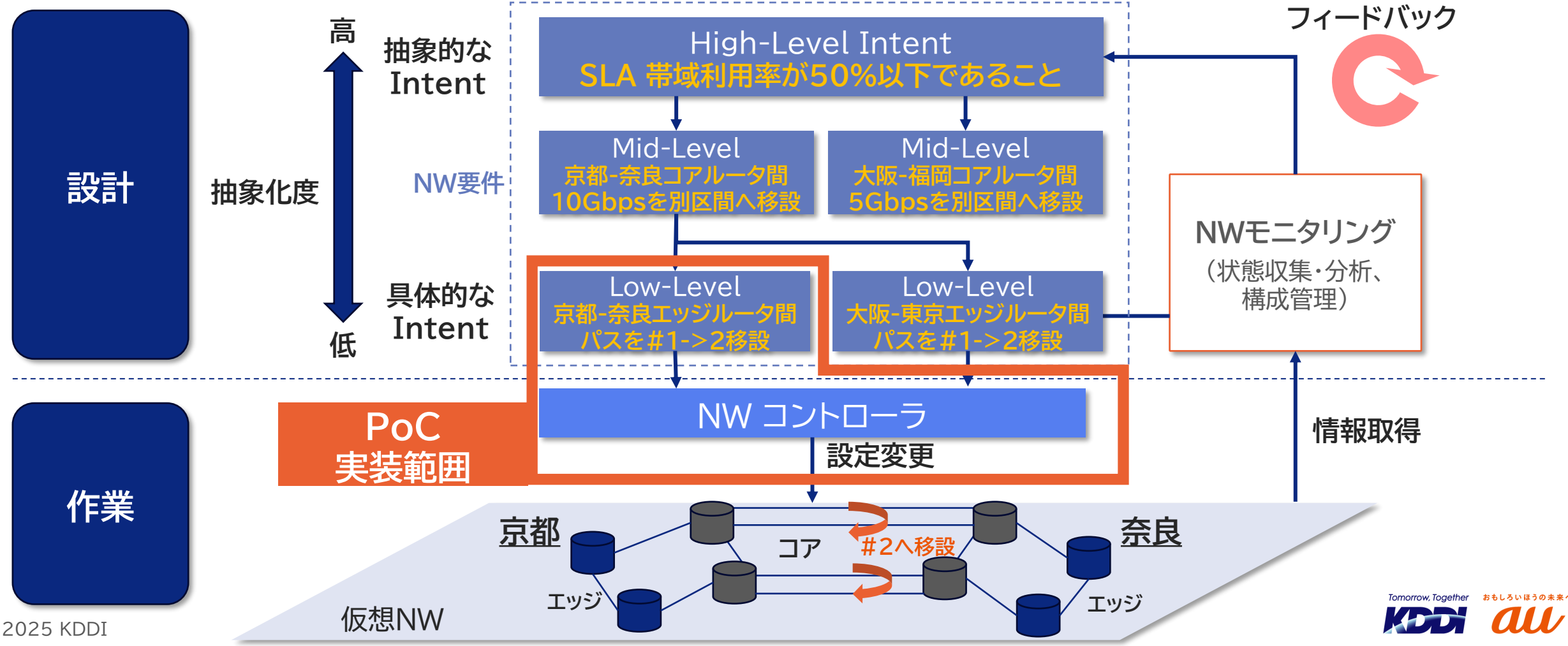
トラフィックチューニング後


大阪 ➡ 京都（#1）➡ 奈良（#2）➡ 東京（#1）

# PoCの実装範囲

## Low-Level Intentの定義内容をNWコントローラ経由でNW機器に反映

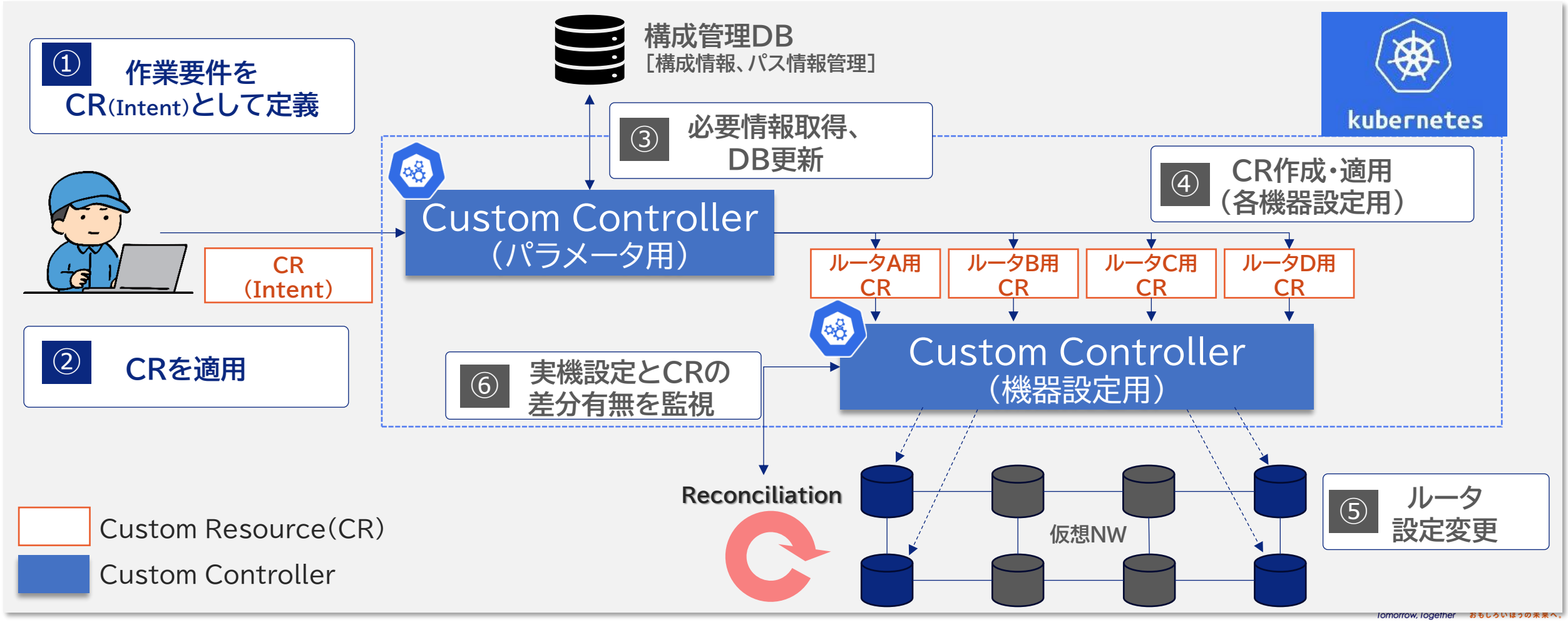
■Intentの具体化(High⇒Low)やモニタリングは、PoCスコープ外(今後の課題)



# アーキテクチャ概要

## CR/Custom Controllerにより、Intent定義内容をNWに反映

■ 人によるオペレーションは、設計フェーズのIntent(CR)の作成・適用のみ(①②)



# CR(Intent)

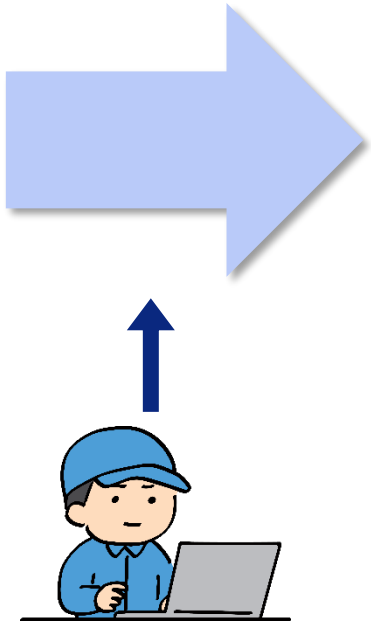
## Intentの内容をCRとして作成（YAML形式）

- Intent：京都～奈良エッジのパスについて、コア間を2番線に移す

**CRD**
**（CRの定義ファイル）**

```
kind: CustomResourceDefinition
spec:
  names:
    kind: HighLevelTrafficTuning
    schema:
    properties:
          spec:
            properties:
              Node:
                type: string
              EndpointNode:
                type: string
              TunnelID:
                type: integer
              ExplicitPath:
                properties:
                  hop1:
                    properties:
                      Host:
                        type: string
                      PhysicalLink:
                        type: integer
                  hop2:
                  …（以下省略）
```

※一部抜粋



Intentを
CRとして定義

**CR**
**（Intent）**

```
kind: HighLevelTrafficTuning
metadata:
  name: intent-test
spec:
  Node: kyotoEDGE001
  EndpointNode: naraEDGE001
  TunnelID: 20000
  ExplicitPath:
    hop1:
      Host: kyotoCORE001
      PhysicalLink: 1
  hop2:
      Host: naraCORE001
      PhysicalLink: 2    # 変更前は 1
  hop3:
      Host: naraEDGE001
      PhysicalLink: 1
  …（以下省略）
```

※一部抜粋

# Custom Controller（パラメータ用）及び、機器設定用CR

## 前工程で作成されたCRと構成管理情報を基に、Controller（パラメータ用）が機器設定情報をCRとして生成

CR
（Intent）

CRD
（機器設定用CRの
定義ファイル）

CRが生成
（機器設定に必要な情報）

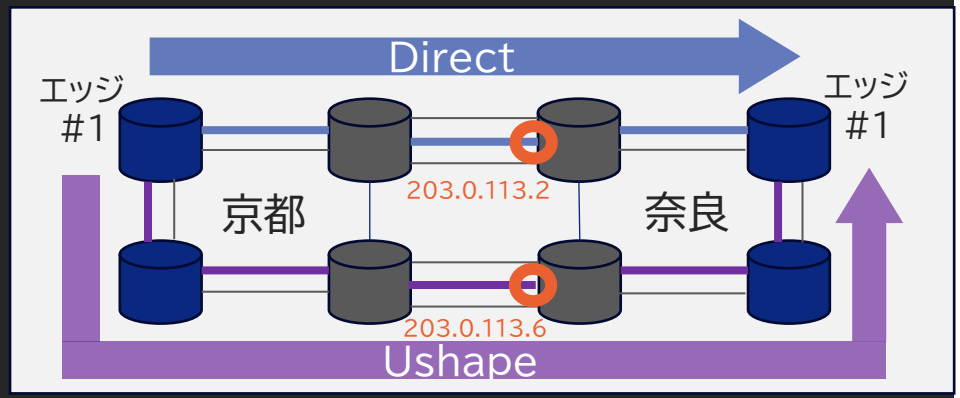
Custom
Controller
（パラメータ用）

情報取得/
更新

構成管理DB

CR
（機器設定用）

```
kind: TrafficTuning
spec:
  nodeIP: 192.168.0.1
  EndnodeIP1: 192.168.0.7
  EndnodeIP2: 192.168.0.8
  Tunnel1:
    TunnelID: 20000
    ExplicitPathDirect:
      option: 10
      hop1: 10.0.0.2
      hop2: 203.0.113.2   #2番線のIPアドレス
      hop3: 10.0.5.2
      hop4: 192.168.0.7
      hop6: null
      hop7: null
      hop8: null
      hop9: null
      hop10: null
    ExplicitPathUshape:
      option: 20
      hop1: 10.0.7.2
      hop2: 10.0.9.2
      hop3: 203.0.113.6 #2番線のIPアドレス
      hop4: 10.0.14.2
      hop5: 10.0.16.1
      hop6: 192.168.0.7
        …（以下省略）
```




※一部抜粋

# Custom Controller（機器設定用）

## 前工程で作成されたCRを基に、Controller（機器設定用）が ルータへの設定投入を実施（今回の例は、迂回⇒パス変更⇒迂回戻し）

京都エッジ#1用 CR

京都エッジ#2用 CR

奈良エッジ#1用 CR

奈良エッジ#2用 CR

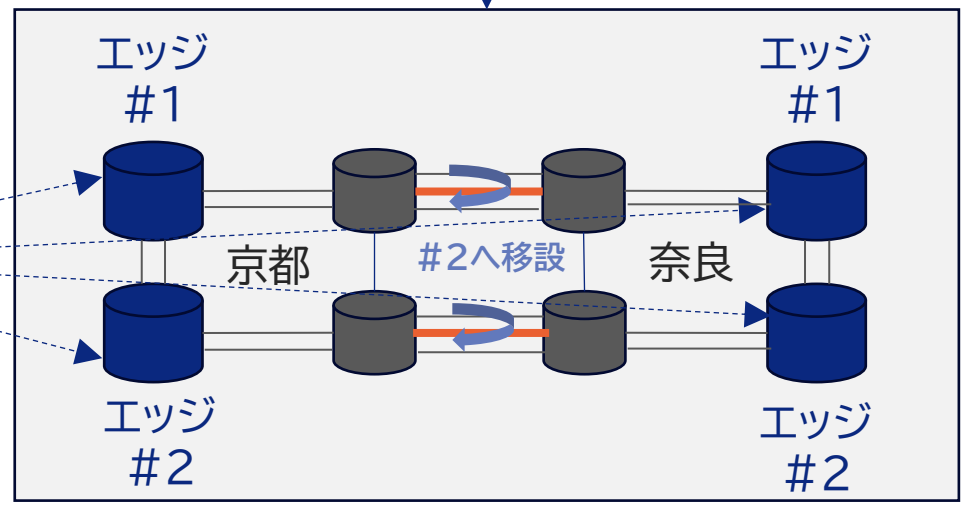
パラメータ情報

Custom Controller（機器設定用）

Reconciliation

CRの内容と差異を監視

設定変更/状態確認




### gNMIを利用

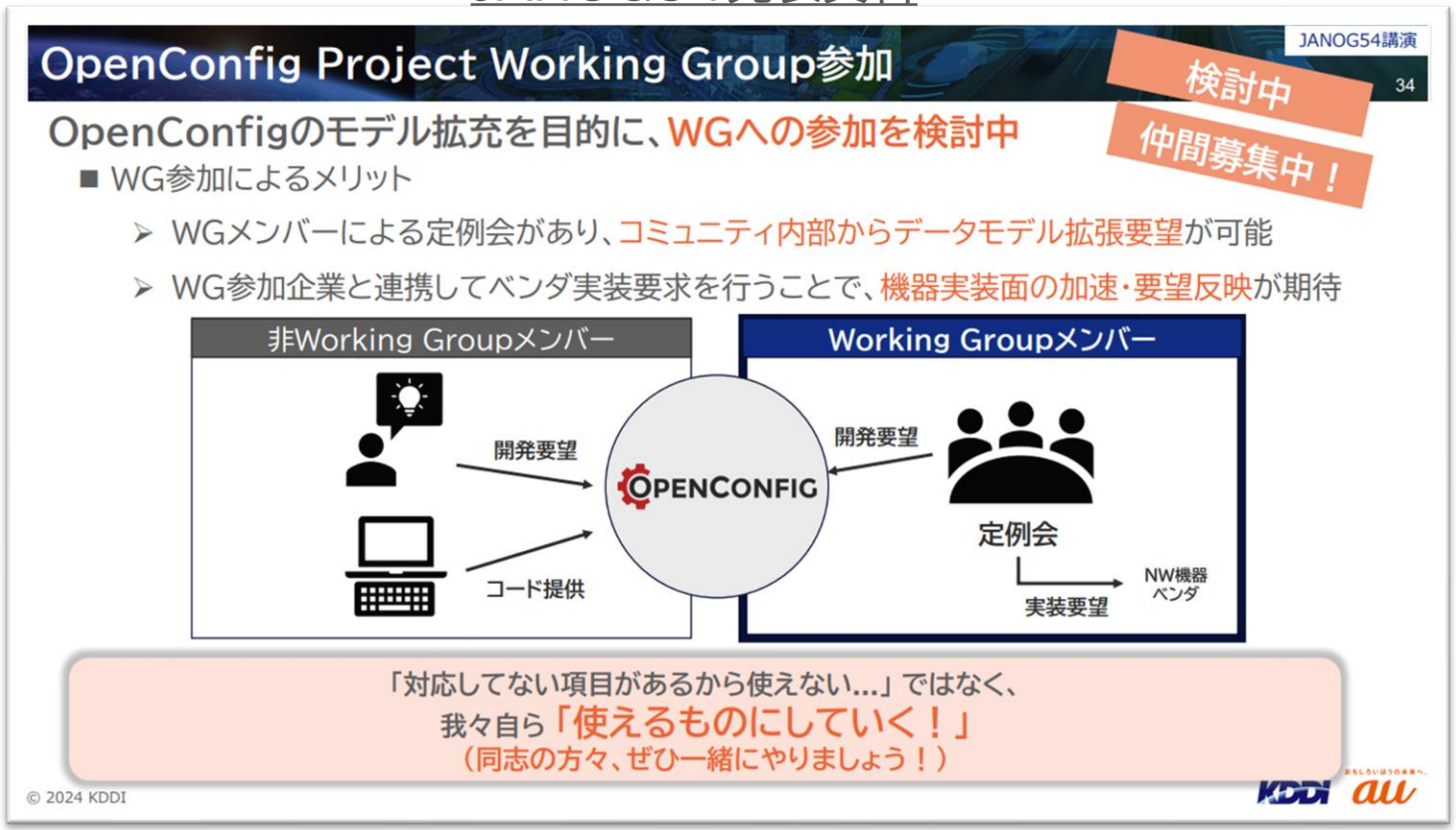
現在： gNMI x（OpenConfig（優先）＋ CLI）
将来： gNMI x（OpenConfig（優先）＋ Vender Native）

# 【参考】OpenConfig WGへの加入

## 25年1月より、OpenConfig WGへの加入

- OpenConfigモデル拡張を目指し、活動中
- 設定投入/情報取得部分は、**優先してOpenConfigを活用**

JANOG54発表資料




出典:https://www.janog.gr.jp/meeting/janog54/wp-content/uploads/2024/05/janog54-openconfig-sato-fukui-2.pdf

# Intent-Drivenによる自動化後のオペレーション

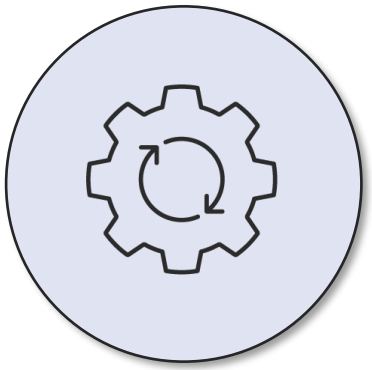
## デモ動画

当日の発表で紹介

# PoC結果

## 期待通り、設計/作業の自動化が可能


■ ただし、一部は課題が残る



### 1. 設計/作業の自動化

期待通りの**自動化が可能**

Kubernetesのカスタム機能の活用により、**Intent-Driven手法の自動化実現性**を確認

### 2.スキルセット明確化

**開発範囲/必要スキル**を明確化

主な開発範囲
- **CRD**作成
- **Custom Controller**作成

必要スキル
- 要件定義 :**NWスキル**
- コーディング/実装:**SWスキル**

### 3.課題抽出

**今後対応すべき課題**を抽出

- 作業順序性を考慮した複数ルータの**同期作業への対応**
- **K8s関連ツール**や**開発言語**の選定（今回は、Kubebuilder及びGo言語を利用）

JANOG55発表

# 5. 商用導入へ向けた課題・今後の展望

# 商用導入へ向けた課題・今後の展望

## 今後の展望としては、PoC範囲の商用利用 及び NW自律化に向けた自動化範囲拡張

### ■ 本PoC範囲の商用利用

**1 NWコントローラ**

Kubernetesを利用するインフラ環境の検討

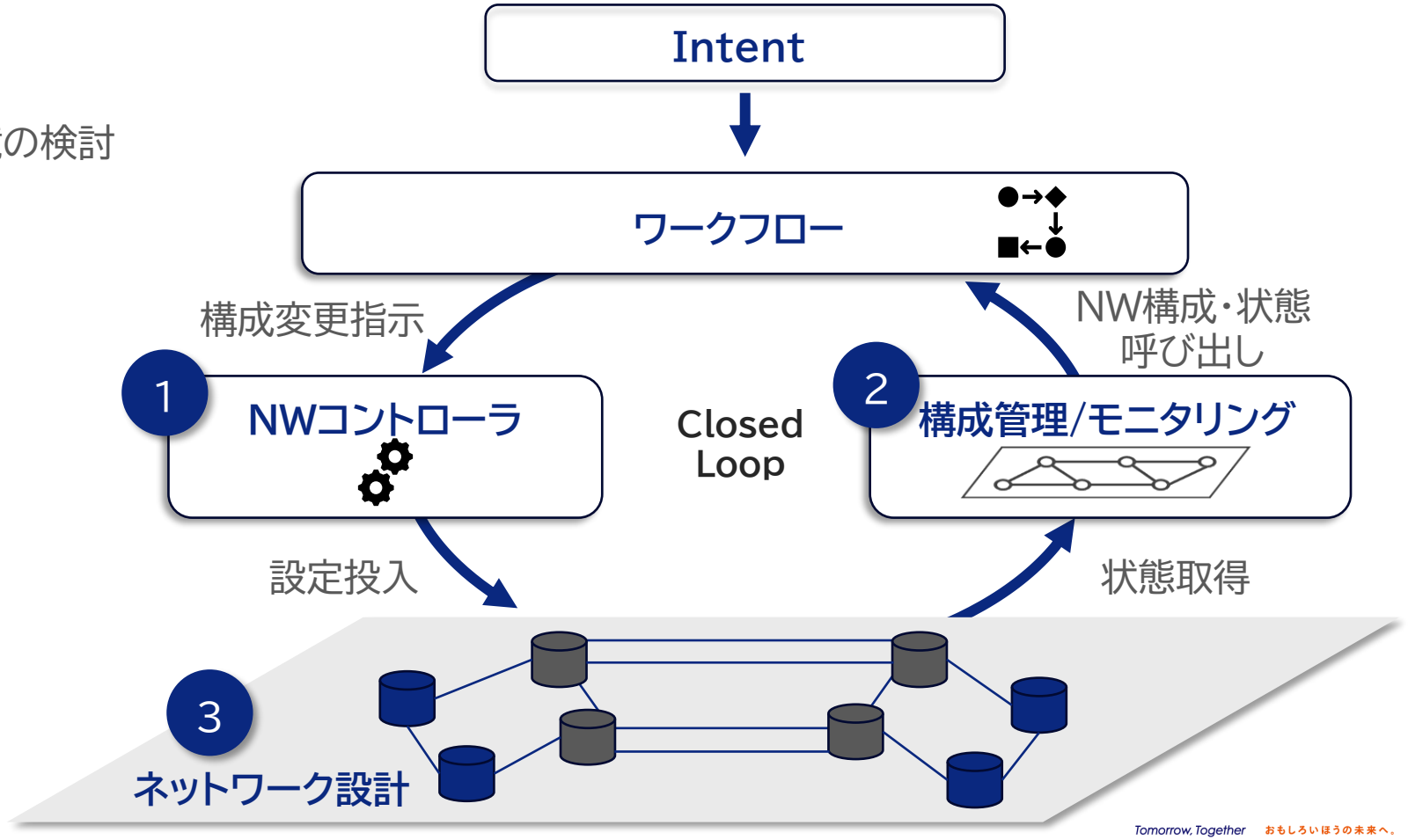
### ■ 自動化範囲の拡張

**2 構成管理・モニタリング**

NWの構成及び状態を定義する
NWモデリングの検討

**3 ネットワーク設計**

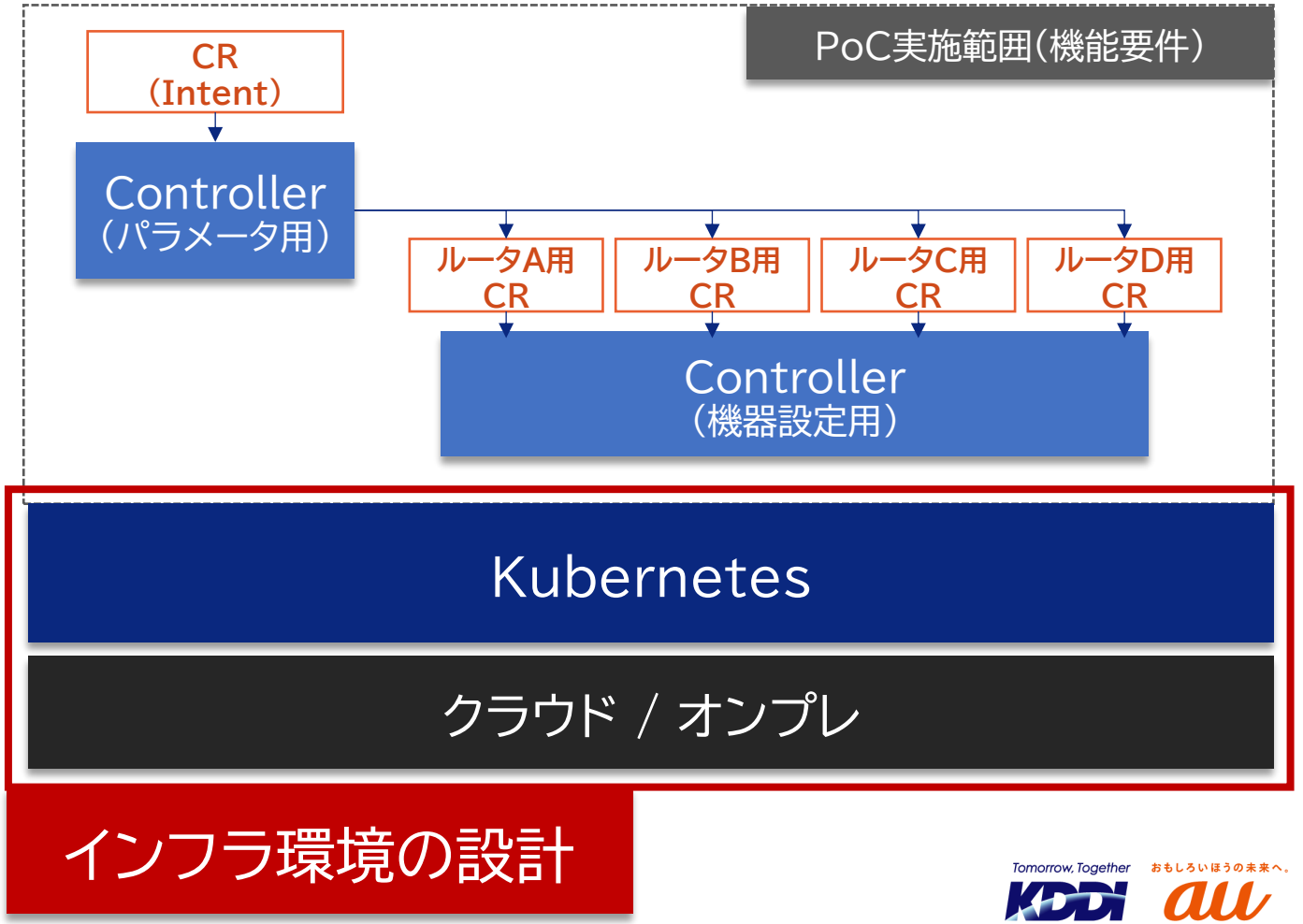
NW構成パターンの削減など、
NWのシンプル化が必要



# 本PoC範囲の商用利用課題① NWコントローラ

商用ネットワークで利用するには、Kubernetesを利用するインフラ環境の検討が必要

- インフラ環境の検討課題
  - 可用性
    - 冗長構成やバックアップ機能
  - 性能
    - スケーラビリティ（CR/Controllerの拡張性）
  - 運用面
    - Kubernetesの管理

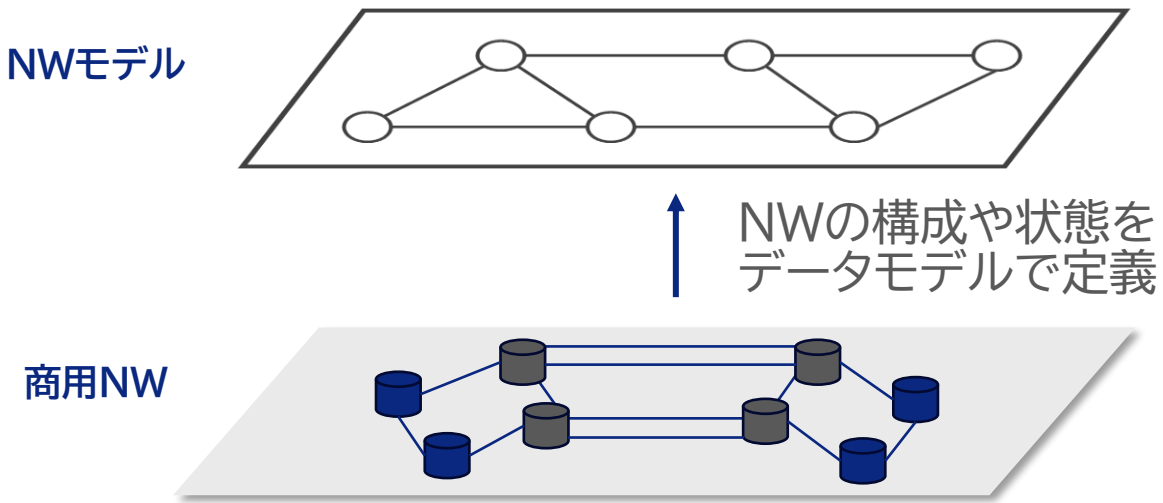


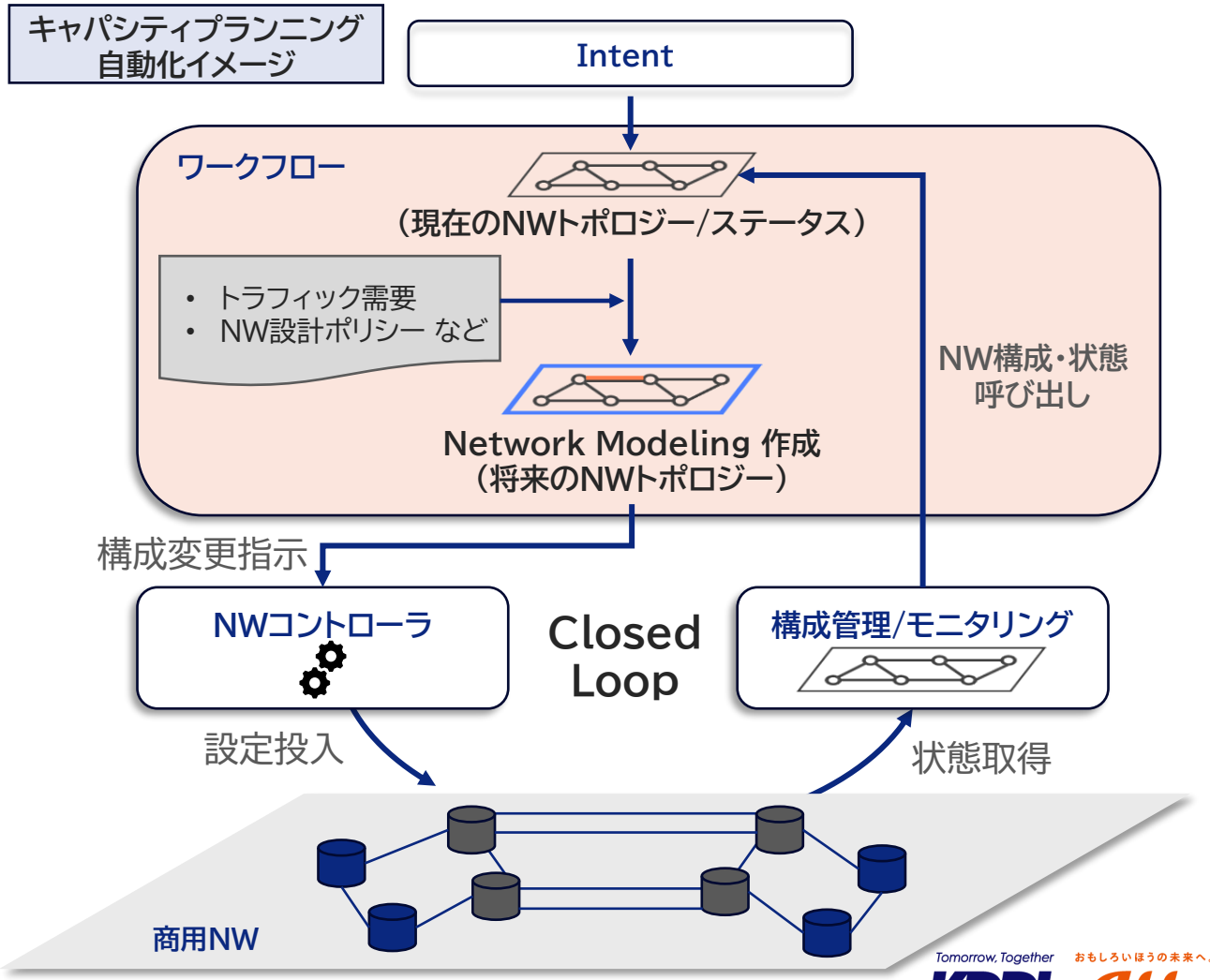
# 自動化範囲の拡張課題② 構成管理・モニタリング

## NW構成及び状態をSWの世界でデータとして扱うには、ネットワークのモデリングが必要

### ● ネットワークモデリングとは

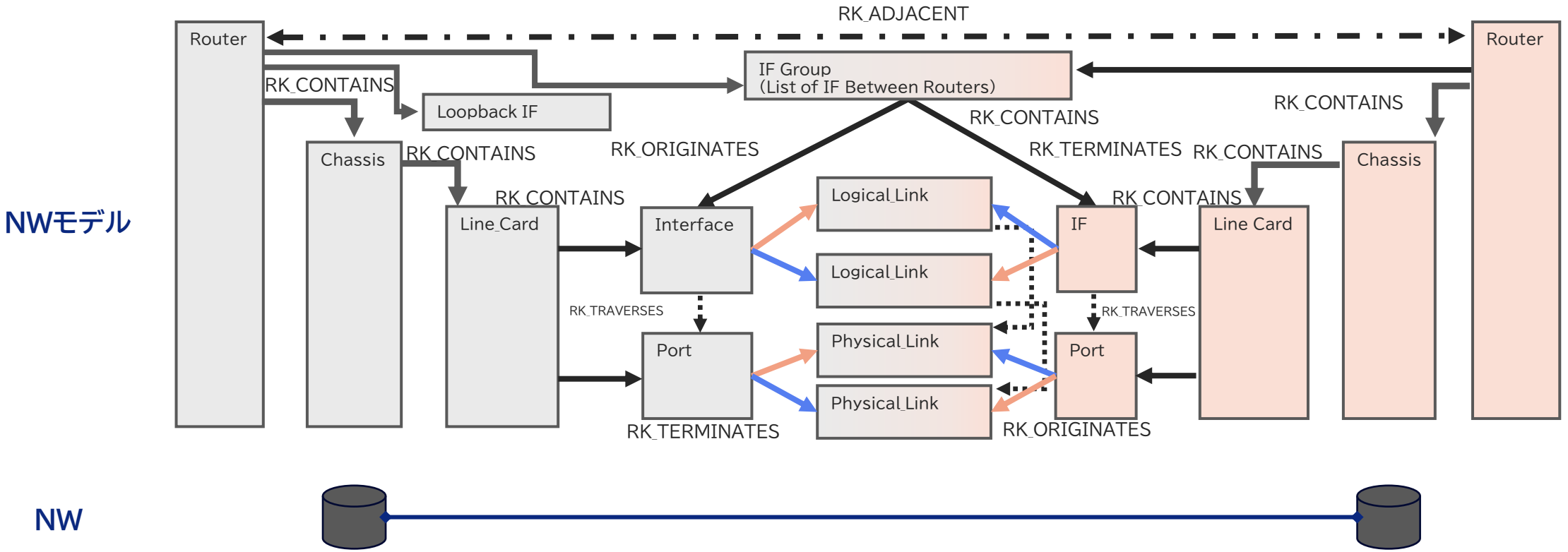
- NWの構成や状態をデータモデルで定義
- NWの未来の状態も表現可能なため、Intentとしても利用



# 検討課題② ネットワークモデリング課題

- ネットワークモデリング課題
  - どこまでの情報をモデリングしておくべきか
  - データモデルの定義（TM Forum, IETF, Google MALTなど）として、参考にすべきものはあるか
  - モデリング定義後のデータベース実装はどうするか

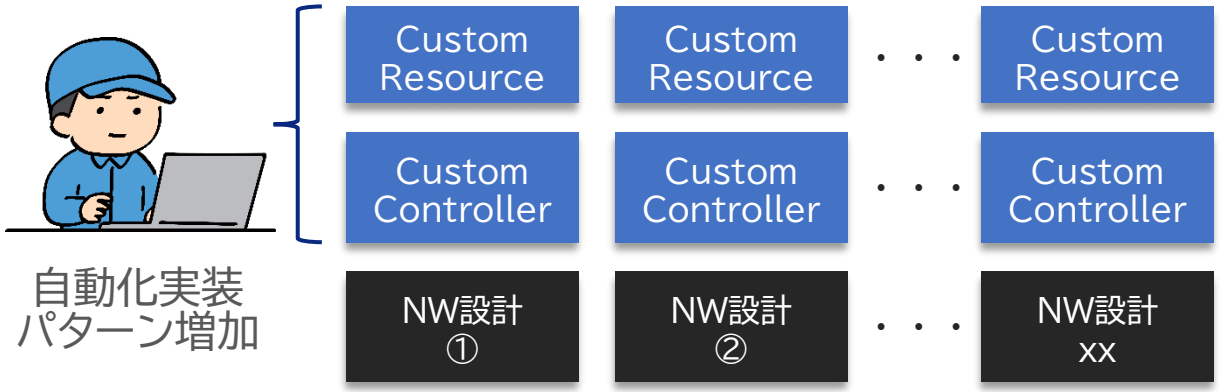


# 自動化範囲の拡張課題③ ネットワーク設計

自動化を実現する上では、**NW設計も自動化に適した形**に変更が必要

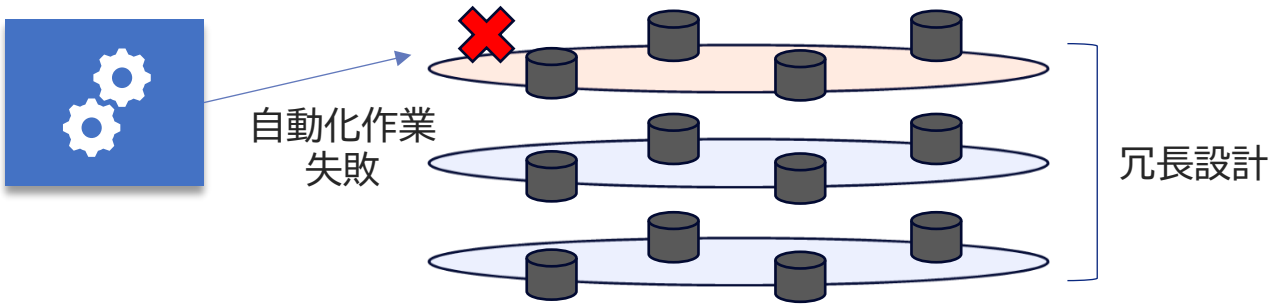
- 自動化に適したネットワークとは

1. 自動化開発の負荷が少ない
   - 自動化実装する際の開発負荷を低減する




NW構成・設計パターンが多い場合

2. 自動化が導入し易い
   - 自動化作業で問題が発生しても
     サービス影響が出ない/即時対応が不要




自動化作業トラブルを想定したNW設計


# 6. まとめ・議論したいポイント

# まとめ

| | |
|---|---|
| **モチベーション** | 人手によるオペレーションから脱却した**自律型ネットワーク**の実現 |
| **アプローチ** | **Intent-Driven**による自動化 |
| **PoCを通して分かったこと** | コンテナオーケストレーションツールである**Kubernetes**のカスタム機能の活用により、**Intent-Driven手法の自動化実現性を確認**<br>【Kubernetesのカスタム機能】<br>• Custom Resource ：Intentの定義に利用<br>• Custom Controller：Intentを実現するためのロジック実装に利用 |
| **今後** | 1）本PoC範囲（NWコントローラ）の**商用利用**<br>2）Intent-Drivenによる**自動化範囲の拡張（設計・運用工程）**<br>• NWモデリング（構成管理/モニタリング）<br>• 自動化に適したNW設計 |

# 議論したいポイント

## Kubernetes活用について

- Kubernetes導入時の課題について
- 開発言語の選定について

## NWモデリングについて

- どの範囲の情報をモデリングしておくべきか。
- データモデルの定義において、参考にすべきものは何か。
  （例：TM Forum、IETF、Google MALTなど）

## NW設計について

- 自動化に適したネットワークとはどのようなものか。
- 自動化に適したネットワークをどのように実現しているのか。

Tomorrow, Together

KDDI

# 「つなぐチカラ」を進化させ、
# 誰もが思いを実現できる社会をつくる。

## KDDI VISION 2030