2023/01/26 JANOG51

# もし本番ネットワークを<br>まるごと仮想環境に"コピー"できたら<br>うれしいですか?

萩原 学
(TIS株式会社 / 沖縄オープンラボラトリ)

田島 照久
(NTTコミュニケーションズ株式会社 / 沖縄オープンラボラトリ)

滝口 敏行, 前野 洋史, 川口 永一郎
(ビッグローブ株式会社 / 沖縄オープンラボラトリ)

© 2023 Okinawa Open Laboratory

# 登壇者紹介

- 萩原 学　TIS株式会社
- 田島 照久　NTTコミュニケーションズ株式会社
- 滝口 敏行, 前野 洋史, 川口 永一郎　ビッグローブ株式会社


@沖縄オープンラボラトリ
Model Driven Network DevOps プロジェクト
https://www.okinawaopenlabs.org/mdnd

© 2023 Okinawa Open Laboratory

# このセッションの流れ

- このセッションでやりたいこと　　萩原
- 背景 & ねらい　　田島
- デモ & ユースケース　　滝口
- まとめ・議論したいこと　　萩原

- QA/議論

© 2023 Okinawa Open Laboratory

# このセッションで
# やりたいこと

© 2023 Okinawa Open Laboratory

# このセッションでやりたいこと

- ネットワーク「全体がどう動くのか」を検証したい…が、「全体の構造」を再現するだけのリソースはまずない。

- いくつかの技術・考え方を組み合わせると、ある程度「本番と同等規模・同様の動き」が再現できるようになるのでは?
  - 「やってみないとわからない」を実際にやってみる
  - いままであきらめていたことを再現して検証の品質や効率を上げる

- いま、どんなことが、どのくらいできるのか?

© 2023 Okinawa Open Laboratory

# ポイント

- ターゲット: 既存のネットワークとその運用 (≠Green Field)
- 100%理想的な検証環境はない
  - トレードオフがある
  - やりたいことに対して、何をどこまで再現可能なのか ➡ 見る・見ないのバランスが重要
- このあと紹介する話では、何を見て何を見ないのか?

| 見る | 見ない |
| --- | --- |
| 機能検証 | 性能・キャパシティ検証 |
| 規模の再現<br>(NWの構造とノード数・NW全体の動作) | 個々のノードの再現<br>(特定のハード・ソフト固有の動作) |

- 新しくやれるようになることはどんなことか?
- その考え方・何がどこまでできるのか?

いろんなものを
割り切って捨ててます。
既存の検証が不要になるとか、
夢の何かではありません…。
現実的にどんなことができるよ
うになるか、が論点

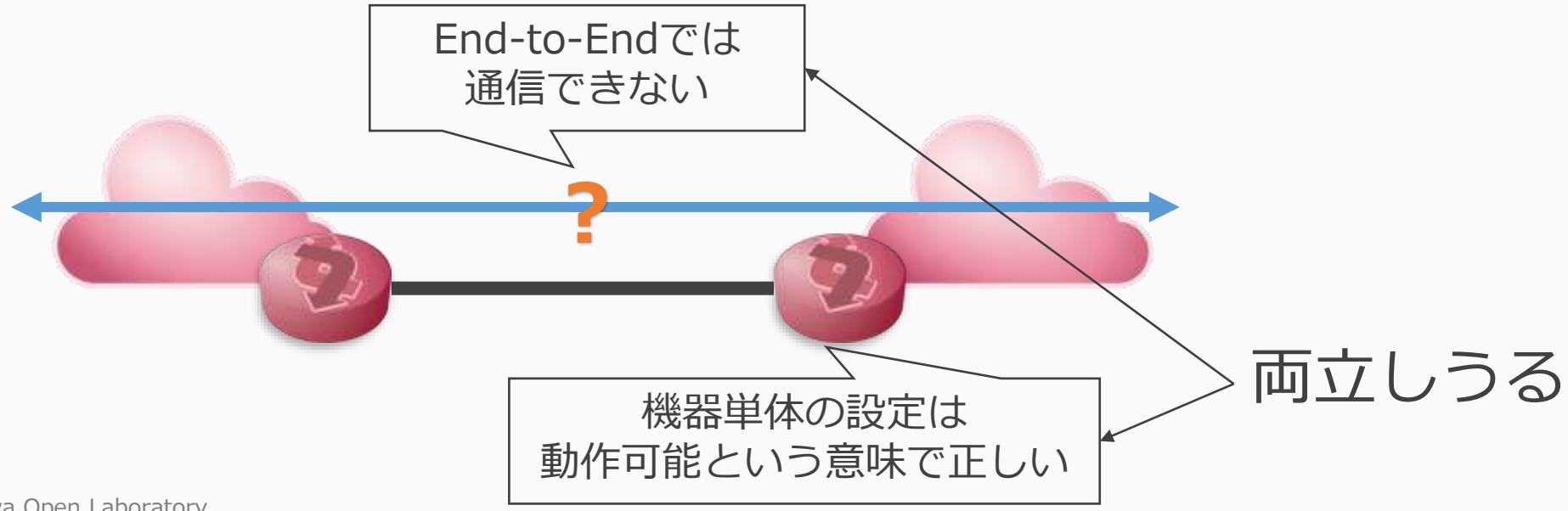
© 2023 Okinawa Open Laboratory

# 背景 & ねらい

© 2023 Okinawa Open Laboratory

# NWの検証環境に求めるもの

- ノード単体だけではなく系としての全体を見たい
- 準備時間を短くしてトライアルアンドエラーを早くしたい
- 各ノードの本番環境での動きを正確に再現したい




© 2023 Okinawa Open Laboratory

# NW運用は全体を見ることが大事

検証が必要＝E2Eでネットワークの要件を満たせているか確認したい

全体を模擬
- 要件をそのままテスト
- Pros:　確実に要件を満たす
- Cons:　リソース準備が極めて困難

⟷

ノードの単体試験の積み上げ
- 各ノードの動作を個別テスト
- Pros:　少ないリソースで実施可能
- Cons:　要件からテストパターン導出が困難

よくある例：折衷案のNWの一部を切り出す検証

➡ 範囲の縮小により発生する見落とし・思い違い・予期せぬ動作
- 「やってみないとわからない」本番一発勝負の発生
- 「本番作業で初めて発覚した」事象の発生

どのようにして全体が見える検証環境を作るか

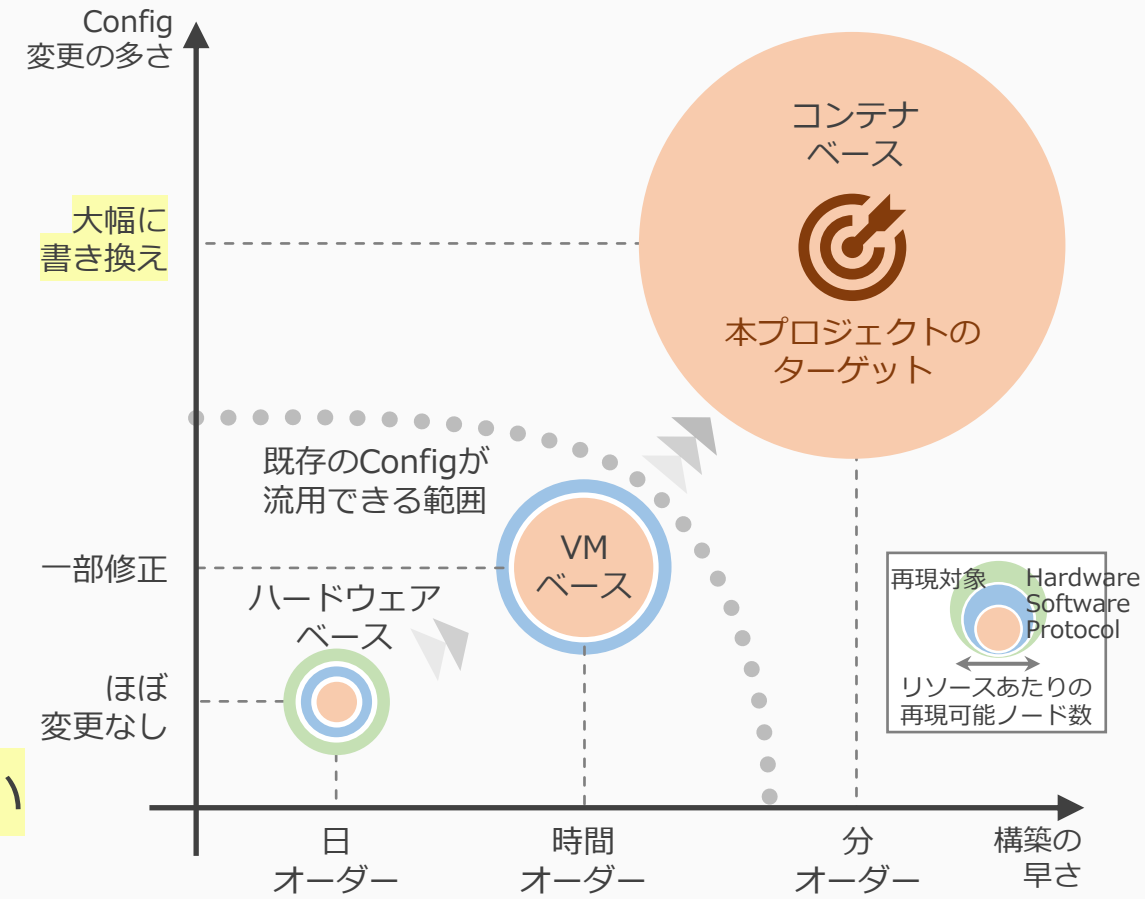
© 2023 Okinawa Open Laboratory

# 検証環境の規模と精度のトレードオフ

要望を全て満たす検証環境は非現実的

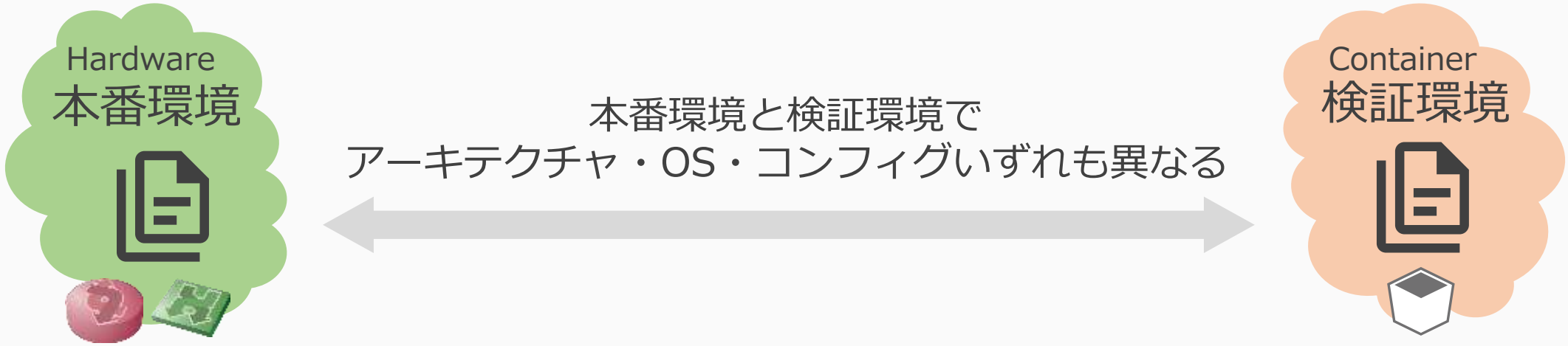
用途に応じた落としどころを探す
- 全体を再現するだけの拡張性が欲しい
- L3以上の構成が同等であれば良い
- ノードの性能や再現度は求めない

コンテナを使えば良さそう
- 必要リソースが少ない
- L3以上のルーティング機能を備えている
- ただし本番環境のConfigはそのまま使えない




© 2023 Okinawa Open Laboratory

# Configの可搬性を確保




## NWの構造を抽象化

- 環境に依存する文法ではなく
  トポロジとしてとらえる
- 各レイヤのトポロジを
  グラフとして表現する

## 各環境へ翻訳

- 単純にデータ変換ではなく
  「同等なもの」にする
- 各環境の文法をテンプレ
  エンジンで生成する

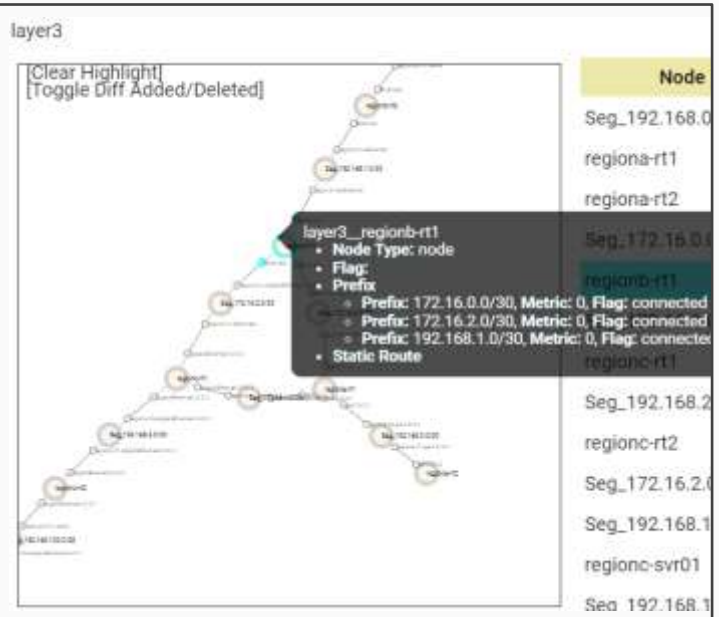
© 2023 Okinawa Open Laboratory

# 抽象化したモデルデータ


- 各レイヤをグラフとして表現したデータ（RFC 8345）
  - L1/L2/L3/OSPFそれぞれのレイヤごとにノードとエッジを持つ
  - レイヤ間の関係性をノードの参照・被参照で表現する

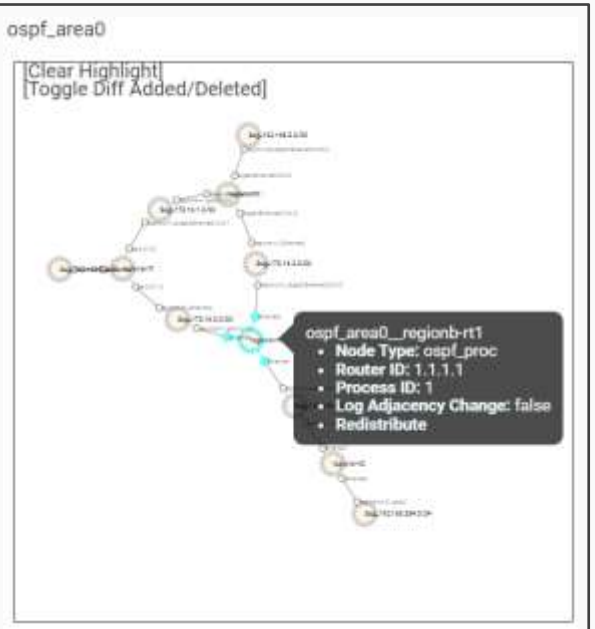
## Layer3

- L3(IPv4)のトポロジ
- インターフェースやIP設定から構成される
- L1/L2の冗長化は1つのPoint-to-MultiPoint Node (Segment)として集約される




## OSPF (Area 0)

- OSPF neighbor のトポロジ
- L3情報とOSPF設定を基に neighbor を判定




© 2023 Okinawa Open Laboratory

RFC 8345, A YANG Data Model for Network Topologies, https://datatracker.ietf.org/doc/rfc8345/

# 実現に必要なツールの登場

NW構成情報の処理・ポータブルなNWを作るツールを組み合わせる

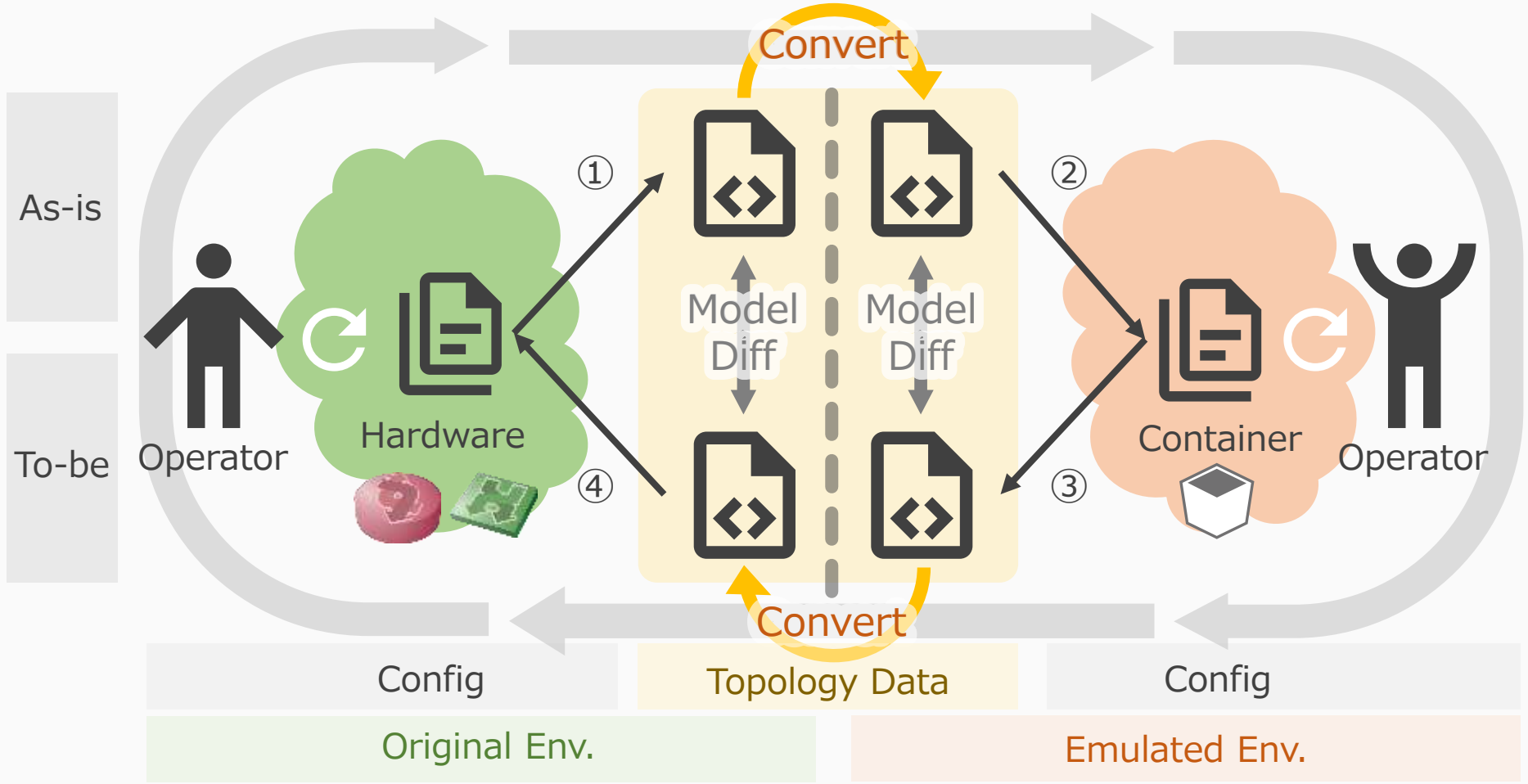
- ネットワークのトポロジを把握するためのツール
  - 物理トポロジ管理: Netbox
  - NW機器コンフィグパーサ/シミュレータ: Batfish
- ネットワークノード操作
  - 自動化: Ansible
- 軽量なネットワークノード
  - CNF (Cloud-native Network Function), コンテナルーティングエンジン
    - OSS: VyOS, FRR, …
    - 製品: Juniper cRPD, Arista cEOS, Nokia SR Linux, Cisco XRd …
  - ソフトウェアL2スイッチ
    - OSS: Open vSwitch
- NW検証用コンテナオーケストレータ
  - Containerlab





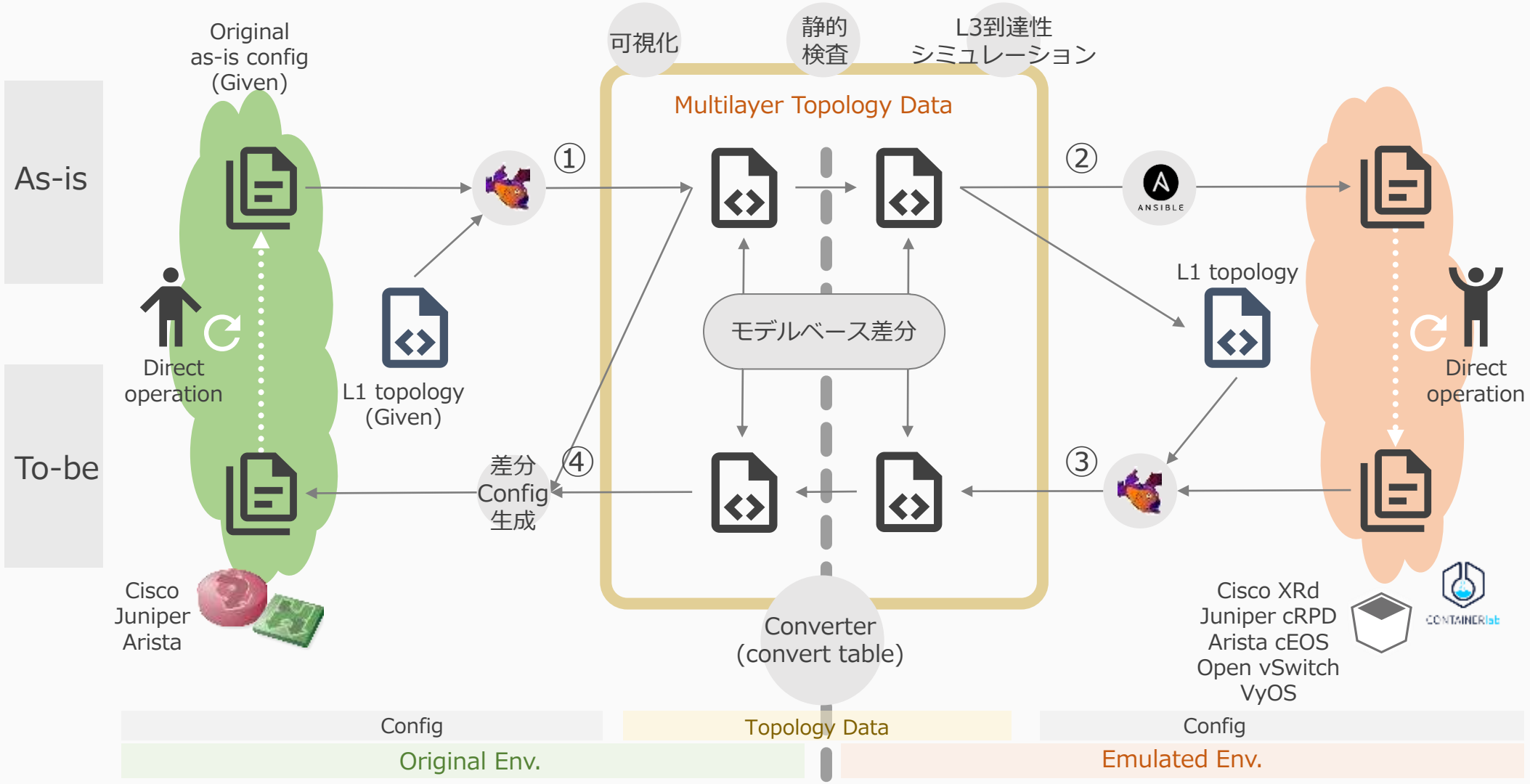

© 2023 Okinawa Open Laboratory

# 本番環境と検証環境のあいだで構成をやりとりするシステム




© 2023 Okinawa Open Laboratory

# システムコンポーネント詳細

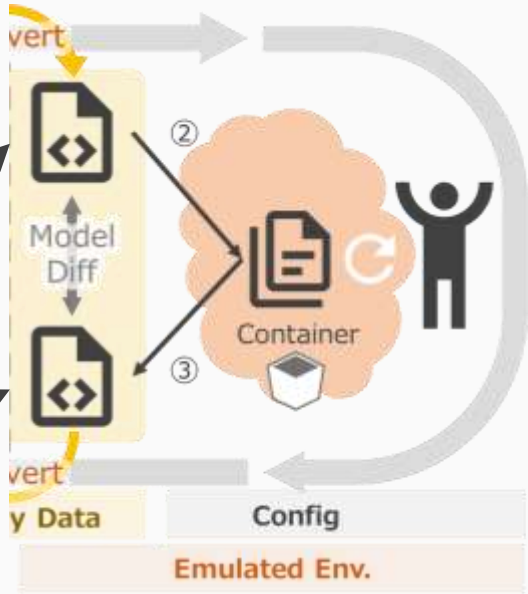



© 2023 Okinawa Open Laboratory

# 正しく「コピー」されているのか

コンテナへの「コピー」に起こりがちな問題
- モデルデータからの翻訳実装のバグ
- コンテナルータ自身の問題や仕様差分
- コンテナオーケストレータの設定ミス

実際に動作させた元/先環境の状態を比較
- 例) Routing Table, OSPF Neighbor
- 構成を再現した結果、状態が同等（差分なし）であればヨシ
- 設定ではなく状態を比較することで正しいかどうかを判断
- 対象NWの構成変更やツールの修正等のタイミングで実施




© 2023 Okinawa Open Laboratory

# デモ & ユースケース

© 2023 Okinawa Open Laboratory

# デモ & ユースケース

- デモ
  - 架空の小規模NWを使った仕組みの解説
  - 複数リージョンが接続されたNWでのセグメント移転シナリオ
- ユースケース
  - NTTコミュニケーションズ社の検証網をベースにした再現実験
  - 実際に運用されているサイズのNWで利用可能かどうか
- それらをやってみてわかったこと

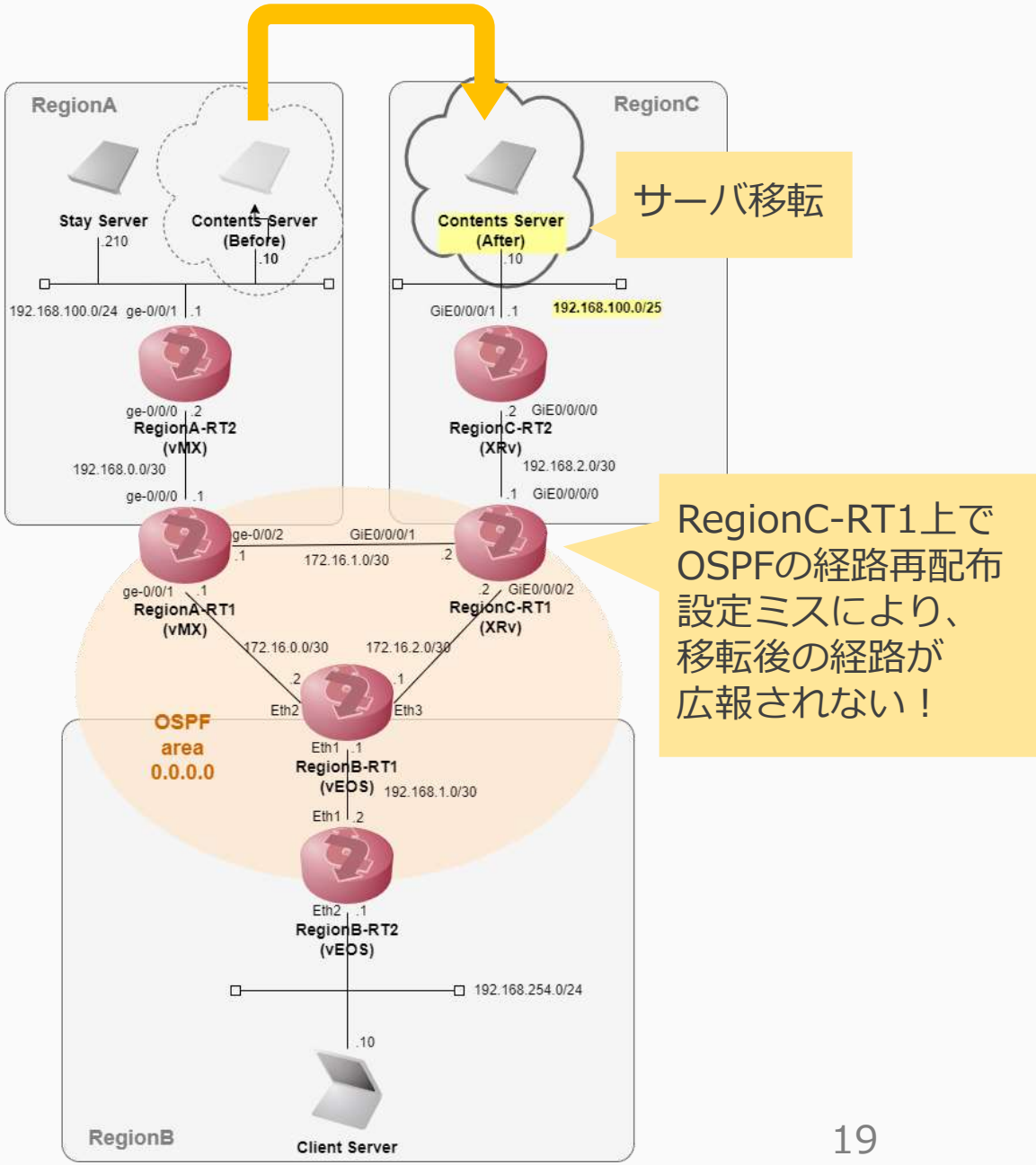
© 2023 Okinawa Open Laboratory

# デモ: セグメント移転

デモシナリオ

- 一部のセグメント移転(移動)
- 移転に伴うネットワークの経路制御変更
  - OSPFの経路再配布での設定ミス
  - 移転後サーバの通信トラブル
- 従来だと…
  - 異なるトポロジ・縮小したNWでの検証
  - 検証に含まれなかった箇所・パターンは不確実なままのこってしまう

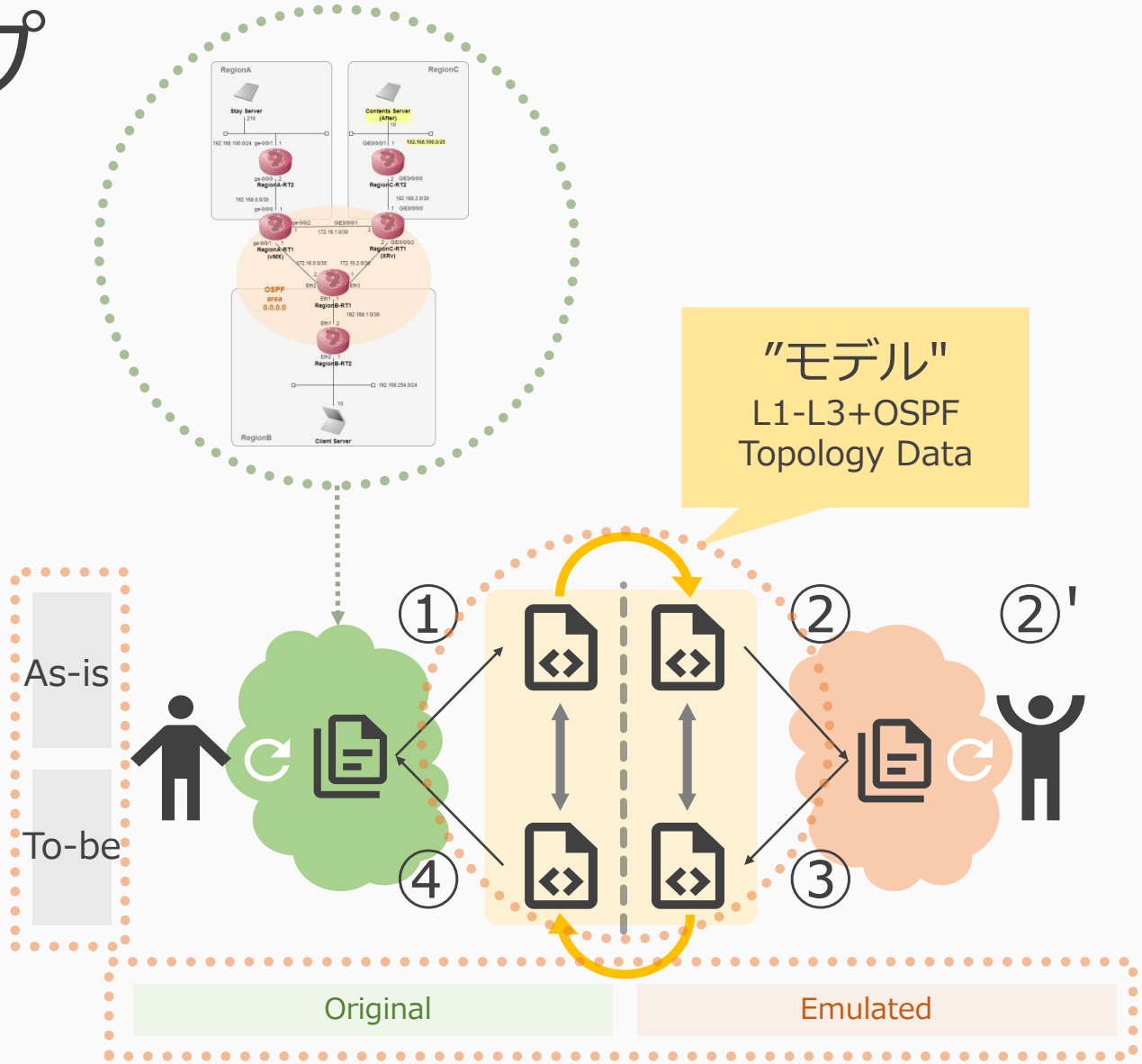
"システム全体の動き"の問題を発見できるか?
どれくらいのコスト(リソース)で可能か?




© 2023 Okinawa Open Laboratory

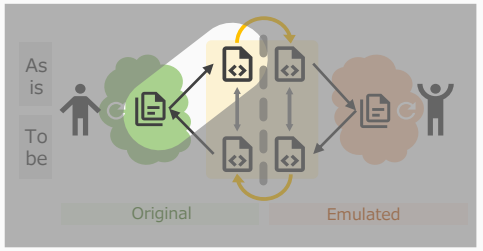
# デプロイのステップ

① As-Is (現状) モデル作成
② As-Is 仮想環境作成
　②' 仮想環境上での検証
③ To-Be (理想) モデル作成
④ To-Be 実環境への適用




© 2023 Okinawa Open Laboratory

# ①As-Is (現状) モデル作成




**Input:**
NW機器設定ファイル
現状使っているもの
(Original/as-is)

**Process:**
Batfish+自製ツール
L1-L3+OSPF設定情報の抽出と
各レイヤのトポロジ・
ノード属性データの組み立て

Hardware

ANSIBLE

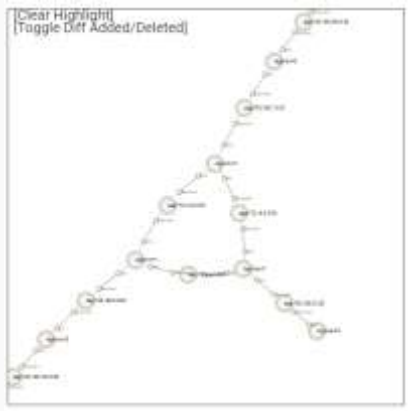
Descr
変換

デモでは
Interface
description
から自動生成

**Input:**
物理トポロジ情報(json)

可視化

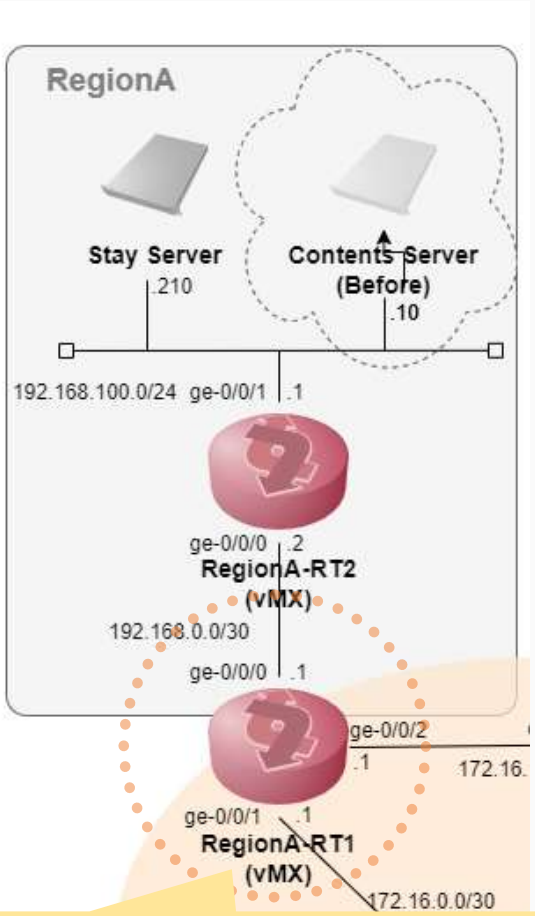
**Output:**
マルチレイヤのトポロジデータ(RFC8345)
各レイヤ内およびレイヤ間の
ノード関係性+ノード属性情報
(Original/as-is)



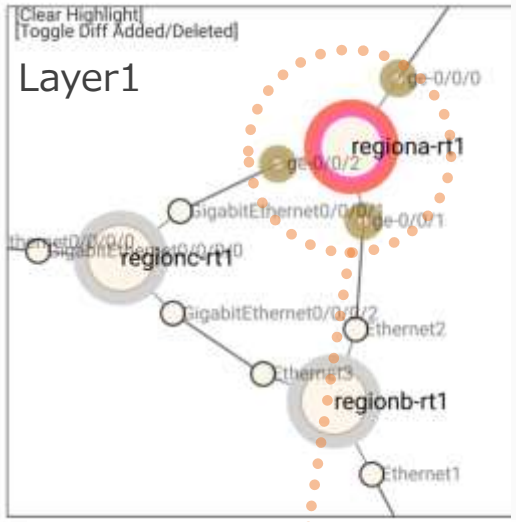

Original Env.

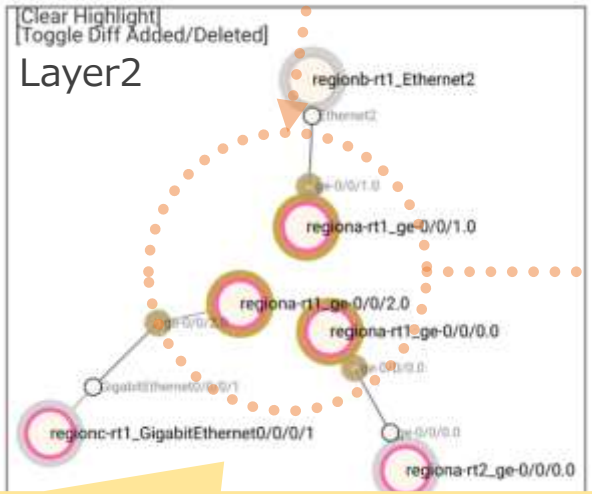
© 2023 Okinawa Open Laboratory

# ①モデルデータの具体例

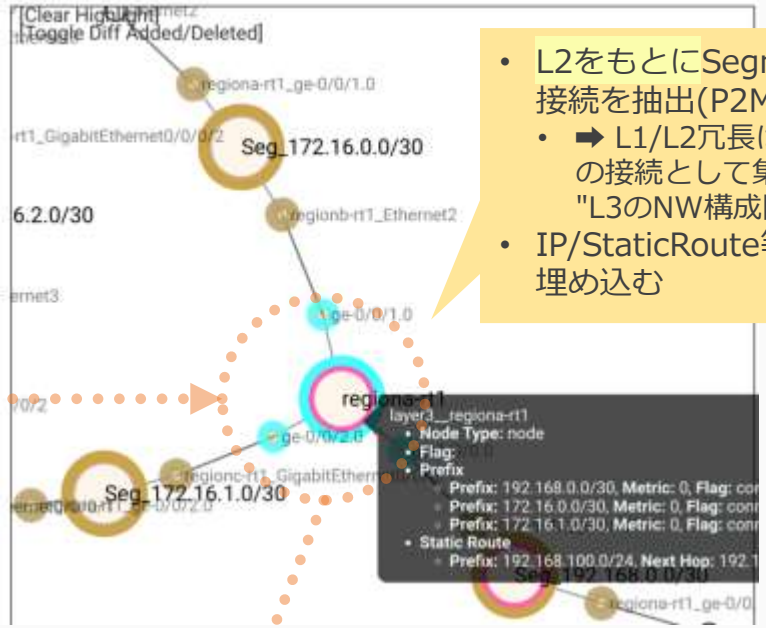



RegionA-RT1
- Static: 192.168.100.0/24 NextHop 192.168.0.2
- Redistribute static & connected

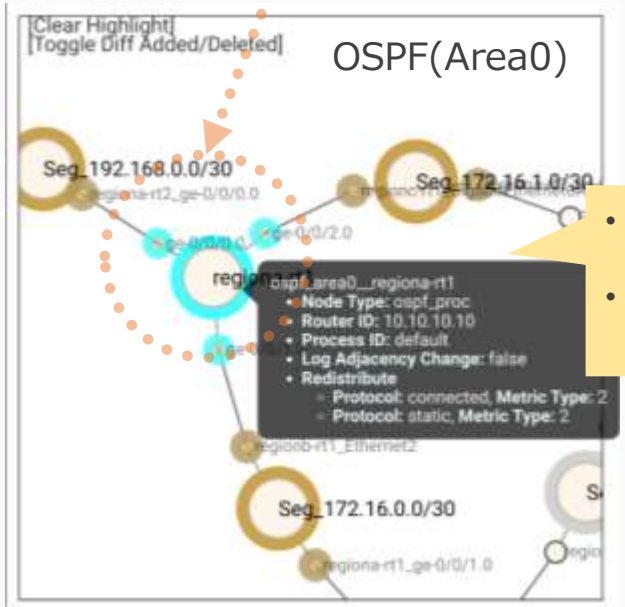



L1をもとにL2接続(VLAN, Bridge)接続の抽出
(デモNWでは /30 直接接続)




- L2をもとにSegment~Node間接続を抽出(P2MPも表現)
  - ➡ L1/L2冗長は "Segment" との接続として集約…いわゆる "L3のNW構成図" と同等
- IP/StaticRoute等の属性情報を埋め込む




- L3をもとにOSPF proc間接続を抽出
- OSPF Redistribute等の属性情報を埋め込む

# ①実行後の状態

## Configのパースとトポロジデータが生成される

```
$ sudo bash demo_step1.sh
Creating playground_netomox-exp_run ... done
# Make directories
mkdir -p /mddo/netoviz_model
mkdir -p /mddo/models
# Clean models dir
rm -rf /mddo/models/*
# Pass snapshot pattern generation
rm -f /mddo/configs/mddo-ospf/original_asis/snapshot_patterns.json
rm -f /mddo/configs/mddo-ospf/emulated_asis/snapshot_patterns.json
rm -f /mddo/configs/mddo-ospf/emulated_tobe/snapshot_patterns.json
# Generate model data
- POST: http://batfish-wrapper:5000/api/networks/mddo-ospf/queries, data={}
# Generate netoviz index file
# Generate topology files
find /mddo/netoviz_model -type d -name '*_linkdown_*' | xargs rm -rf
find /mddo/netoviz_model -type d -name '*_drawoff' | xargs rm -rf
mkdir -p /mddo/netoviz_model/mddo-ospf/emulated_asis
bundle exec ruby model_defs/mddo_trial.rb -i /mddo/models/mddo-ospf/emulated_asis > /mddo/netoviz_model/mddo-ospf/emulated_asis/topology.json
mkdir -p /mddo/netoviz_model/mddo-ospf/emulated_tobe
bundle exec ruby model_defs/mddo_trial.rb -i /mddo/models/mddo-ospf/emulated_tobe > /mddo/netoviz_model/mddo-ospf/emulated_tobe/topology.json
mkdir -p /mddo/netoviz_model/mddo-ospf/original_asis
bundle exec ruby model_defs/mddo_trial.rb -i /mddo/models/mddo-ospf/original_asis > /mddo/netoviz_model/mddo-ospf/original_asis/topology.json
# Copy netoviz layout files
# Generate diff data
find /mddo/netoviz_model -name '*.diff' | xargs rm -f
```

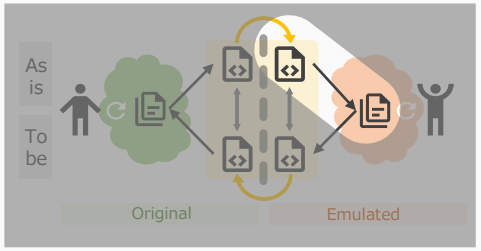
(処理時間 約10秒)

Batfish
Config parse

トポロジデータ
の組立

© 2023 Okinawa Open Laboratory

# ②As-Is 仮想環境作成




**Input:**
トポロジデータ
(Original/as-is)

**Process:**
"名前空間"の変換

**Output/Input**
トポロジデータ
(Emulated/as-is)

Converter

**Process:**
コンテナ用
コンフィグ生成

**Output:**
コンテナ(CNF)用コンフィグ
(Emulated/as-is)

**Process:**
データ抽出

Node-RED

**Output:**
トポロジ情報(json)

**Process:**
clabコンフィグ生成
データ生成

**Output:**
Containerlab用
コンフィグ

Outputを使って
Emulated env を
構築・起動する
(Containerlab)

Container

L3/OSPFレイヤ
のトポロジを
ベースに処理

Original Env.

Emulated Env.

実環境(Original)をコンテナベース環境(Emulated)で再現するため、NWの構造(アーキテクチャ)を変える:
- 使用しているツール・トポロジが変わる(L3以上のみ再現する) ➡ インタフェース名などの識別子を変換する ➡ "名前空間" の変換
- 異なる "名前空間" のトポロジデータは直接比較できない

# ②実行後の状態

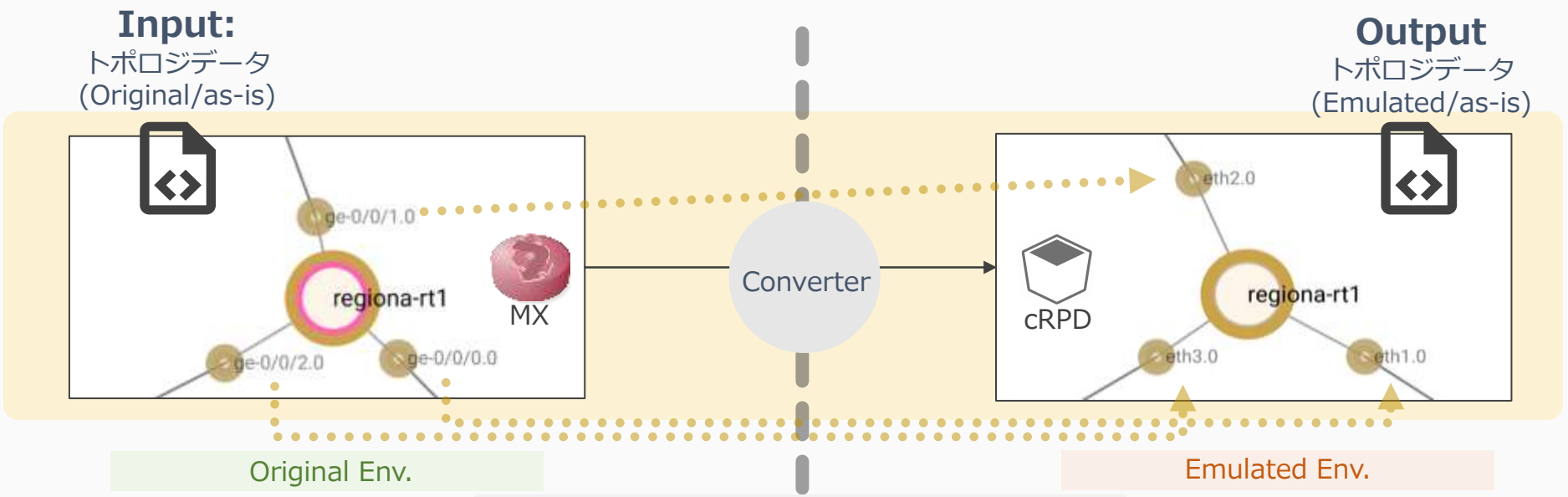
## 各種のコンテナが必要ノード数(18台)だけ起動する

(処理時間 約90秒)

```
INFO[0000] Parsing & checking topology file: clab-topo.yml
+----+-------------------------------------+--------------+---------------------------------+-------------+---------+----------------+----------------------+
| #  |                Name                 | Container ID |              Image              |    Kind     |  State  |  IPv4 Address  |     IPv6 Address     |
+----+-------------------------------------+--------------+---------------------------------+-------------+---------+----------------+----------------------+
|  1 | clab-demo202301-Seg-172.16.0.0-30   | 3d2c74566097 | ghcr.io/ool-mddo/clab-ovs:latest | linux       | running | 172.20.20.45/24 | 2001:172:20:20::2d/64 |
|  2 | clab-demo202301-Seg-172.16.1.0-30   | 83c9bb042a99 | ghcr.io/ool-mddo/clab-ovs:latest | linux       | running | 172.20.20.56/24 | 2001:172:20:20::38/64 |
|  3 | clab-demo202301-Seg-172.16.2.0-30   | 841967394a6a | ghcr.io/ool-mddo/clab-ovs:latest | linux       | running | 172.20.20.42/24 | 2001:172:20:20::2a/64 |
|  4 | clab-demo202301-Seg-192.168.0.0-30  | 0550e0aa5224 | ghcr.io/ool-mddo/clab-ovs:latest | linux       | running | 172.20.20.48/24 | 2001:172:20:20::30/64 |
|  5 | clab-demo202301-Seg-192.168.1.0-30  | d8b4e58be508 | ghcr.io/ool-mddo/clab-ovs:latest | linux       | running | 172.20.20.46/24 | 2001:172:20:20::2e/64 |
|  6 | clab-demo202301-Seg-192.168.100.0-24 | 00ba1c3bab94 | ghcr.io/ool-mddo/clab-ovs:latest | linux       | running | 172.20.20.47/24 | 2001:172:20:20::2f/64 |
|  7 | clab-demo202301-Seg-192.168.100.0-25 | 3e832c7a04c8 | ghcr.io/ool-mddo/clab-ovs:latest | linux       | running | 172.20.20.53/24 | 2001:172:20:20::35/64 |
|  8 | clab-demo202301-Seg-192.168.2.0-30  | 460d23ce3cdf | ghcr.io/ool-mddo/clab-ovs:latest | linux       | running | 172.20.20.43/24 | 2001:172:20:20::2b/64 |
|  9 | clab-demo202301-Seg-192.168.254.0-24 | 7d26b2910e68 | ghcr.io/ool-mddo/clab-ovs:latest | linux       | running | 172.20.20.44/24 | 2001:172:20:20::2c/64 |
| 10 | clab-demo202301-regiona-rt1         | fe958ce578d8 | crpd:22.1R1.10                  | juniper_crpd | running | 172.20.20.49/24 | 2001:172:20:20::31/64 |
| 11 | clab-demo202301-regiona-rt2         | 459e07769ba2 | crpd:22.1R1.10                  | juniper_crpd | running | 172.20.20.41/24 | 2001:172:20:20::29/64 |
| 12 | clab-demo202301-regiona-svr02       | 54398f3e3abe | crpd:22.1R1.10                  | juniper_crpd | running | 172.20.20.40/24 | 2001:172:20:20::28/64 |
| 13 | clab-demo202301-regionb-rt1         | f82e1729677f | crpd:22.1R1.10                  | juniper_crpd | running | 172.20.20.51/24 | 2001:172:20:20::33/64 |
| 14 | clab-demo202301-regionb-rt2         | b35a8ed5d6d3 | crpd:22.1R1.10                  | juniper_crpd | running | 172.20.20.55/24 | 2001:172:20:20::37/64 |
| 15 | clab-demo202301-regionb-svr01       | 191343b8a557 | crpd:22.1R1.10                  | juniper_crpd | running | 172.20.20.54/24 | 2001:172:20:20::36/64 |
| 16 | clab-demo202301-regionc-rt1         | b09031fe80ae | crpd:22.1R1.10                  | juniper_crpd | running | 172.20.20.52/24 | 2001:172:20:20::34/64 |
| 17 | clab-demo202301-regionc-rt2         | 65e9a0fde471 | crpd:22.1R1.10                  | juniper_crpd | running | 172.20.20.57/24 | 2001:172:20:20::39/64 |
| 18 | clab-demo202301-regionc-svr01       | dd4be2e17612 | crpd:22.1R1.10                  | juniper_crpd | running | 172.20.20.50/24 | 2001:172:20:20::32/64 |
+----+-------------------------------------+--------------+---------------------------------+-------------+---------+----------------+----------------------+
```

© 2023 Okinawa Open Laboratory

# ②名前空間の変換




```
node:
  original: regiona-rt1
  clab: regiona-rt1
iflist:
  - original: ge-0/0/0.0
    clab: eth1.0
    ifDescr: to_Seg-192.168.0.0-30_Ethernet1
  - original: ge-0/0/1.0
    clab: eth2.0
    ifDescr: to_Seg-172.16.0.0-30_Ethernet1
  - original: ge-0/0/2.0
    clab: eth3.0
    ifDescr: to_Seg-172.16.1.0-30_Ethernet1
```

L3ノードはcRPDへ、
L3セグメントはブリッジインスタンス(OVSコンテナ)へ変換

© 2023 Okinawa Open Laboratory

# ②名前空間の変換

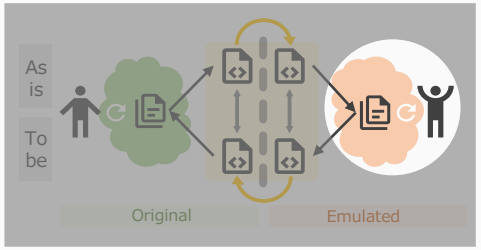
- L1/L2をもとにL3だけ再現
  - L1/L2は翻訳: L1/L2冗長は単一の"Segment"に集約してL3を再現
- 識別子(インタフェース名)の変換

構成情報の抽出・変換・翻訳が複数はいっているため
ミスやバグ等が入った時に見つけにくい

元のNWをちゃんと「再現」できてるかを
確認する方法あります。
(後述: 実環境ユースケースで解説)

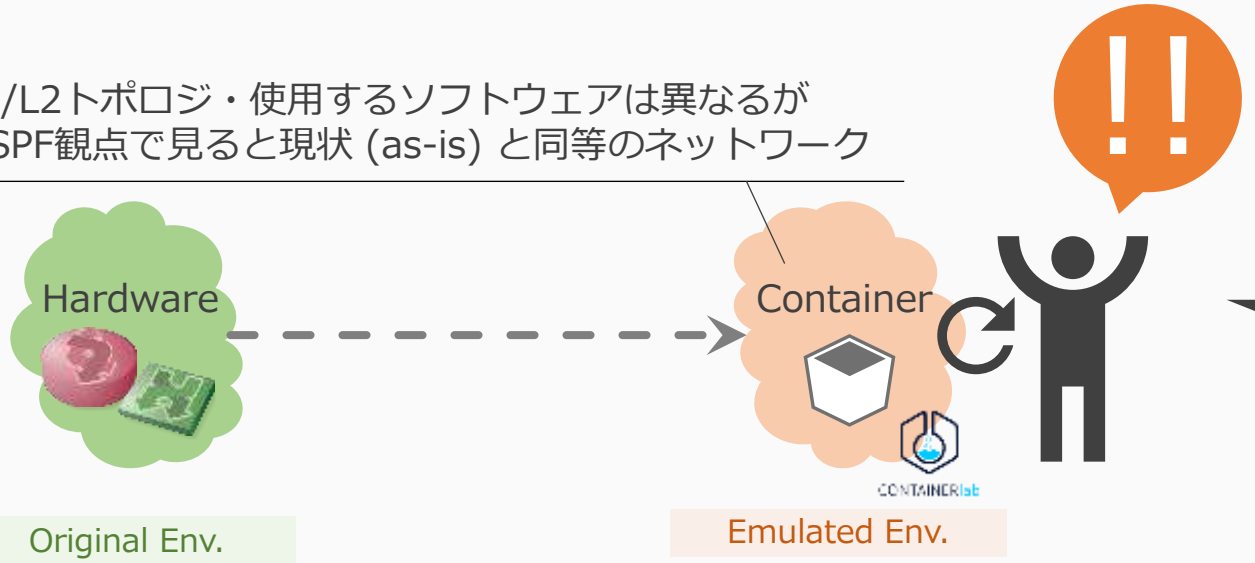
© 2023 Okinawa Open Laboratory

# ②' 仮想環境上での検証




"システム全体の動き"の問題を発見できるか?

L1/L2トポロジ・使用するソフトウェアは異なるが
L3-OSPF観点で見ると現状 (as-is) と同等のネットワーク




As-is

"システム全体の"動作確認・問題探索
- OSPFの状態確認
- 最終的なルーティングテーブルの確認
- L3 reachabilityチェック(ping/traceroute)

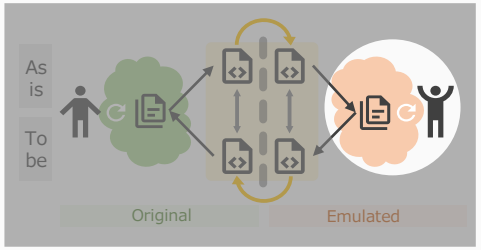
To-be

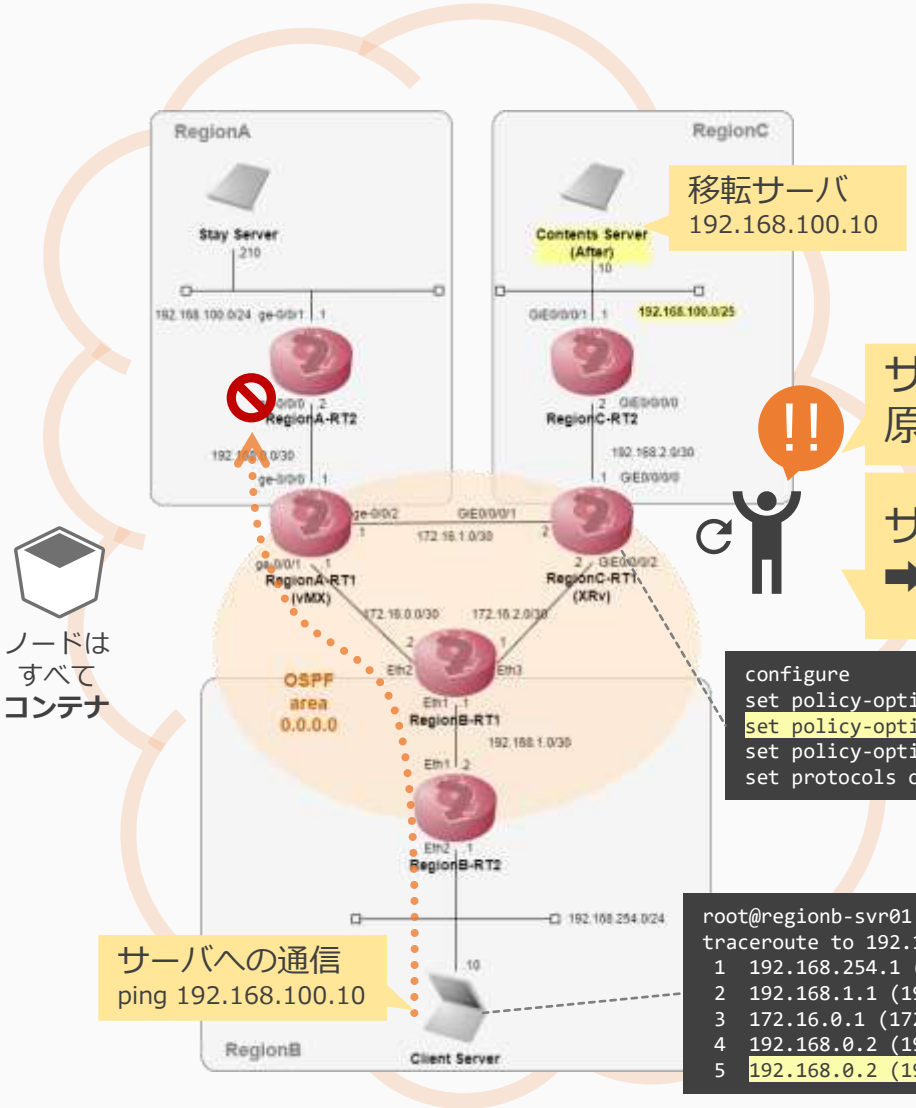
問題の修正
"システム全体の"動作の再確認

デモNW程度の規模だと1-2分程度で検証可能になる

© 2023 Okinawa Open Laboratory

# ②' 仮想環境上での検証




サーバへの通信トラブル発見
原因調査

サーバ移転に伴う経路制御の変更
➡ OSPF再配布設定の修正

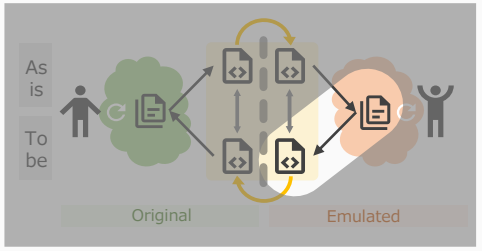
```
configure
set policy-options policy-statement ospf-redistribute from protocol direct
set policy-options policy-statement ospf-redistribute from protocol static
set policy-options policy-statement ospf-redistribute then accept
set protocols ospf export ospf-redistribute
```

```
root@regionb-svr01:/# traceroute 192.168.100.10
traceroute to 192.168.100.10 (192.168.100.10), 30 hops max, 60 byte packets
 1  192.168.254.1 (192.168.254.1)  4.540 ms  3.914 ms  4.254 ms
 2  192.168.1.1 (192.168.1.1)  5.653 ms  4.949 ms  5.266 ms
 3  172.16.0.1 (172.16.0.1)  6.726 ms  6.818 ms  6.645 ms
 4  192.168.0.2 (192.168.0.2)  8.979 ms  9.168 ms  8.845 ms
 5  192.168.0.2 (192.168.0.2)  3069.192 ms !H  3069.168 ms !H  3069.147 ms !H
```

動的経路制御や
経路再配布などの動作は、
構成・状況によって変化するため
正確に予測したり
事前に問題に気付くには
知識や経験が必要

"システム全体"を再現し、
動作を検証することで
問題を発見・対処し
不確実さを減らす

# ③To-Be (理想) モデル作成




トポロジデータ
(Emulated/as-is)

モデルベース差分

**Output**
トポロジデータ
(Emulated/to-be)

**Process:**
Batfish+自製ツール
①同様

**Output/Input:**
コンテナ(CNF)用コンフィグ
(Emulated/to-be)

**Input:**
トポロジ情報(json)

修正後
(to-be)

Container

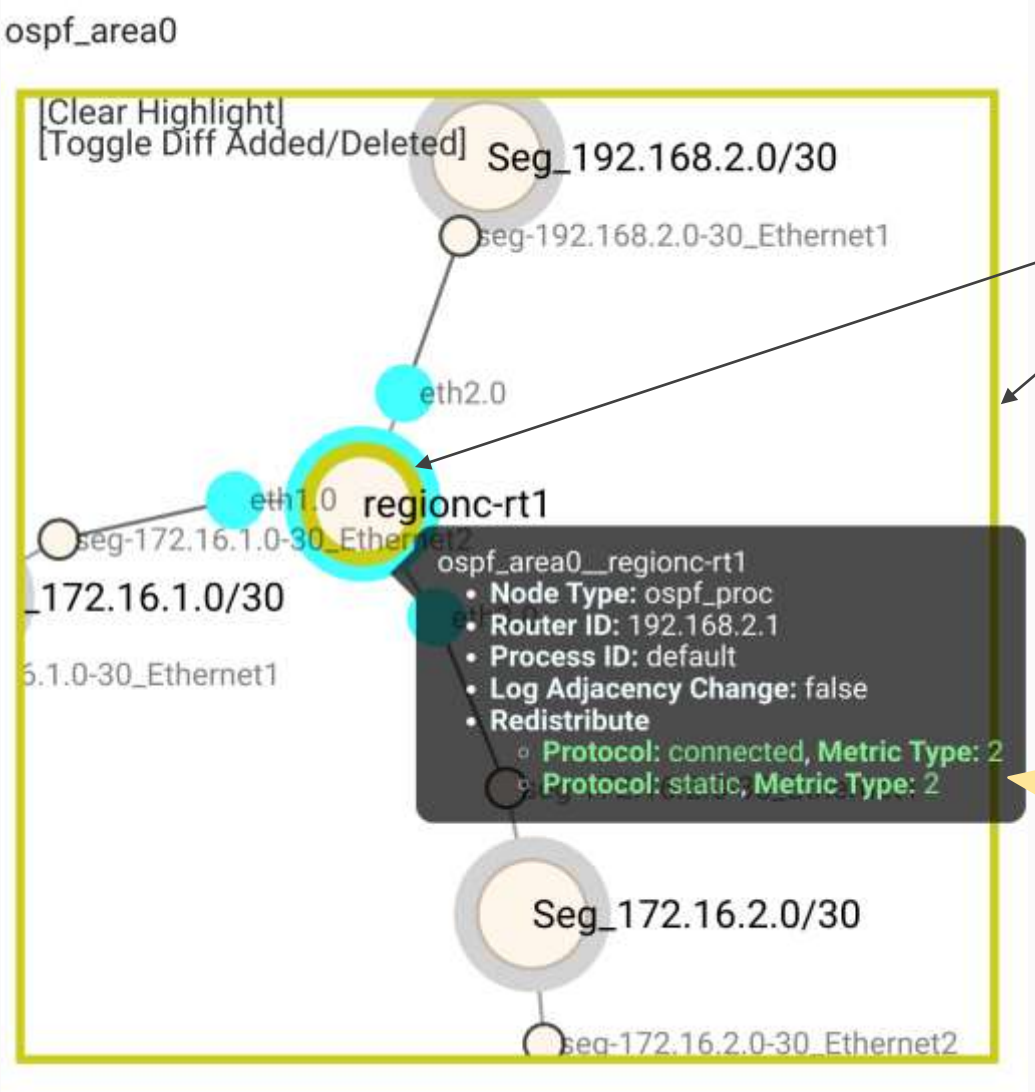
Config
保存

処理対象にするコンフィグが
異なるだけで①と同様

Emulated Env.

© 2023 Okinawa Open Laboratory

# ③変更内容の確認

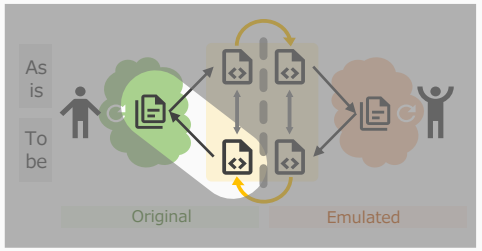



変更されたオブジェクト
(黄色にマーキング)

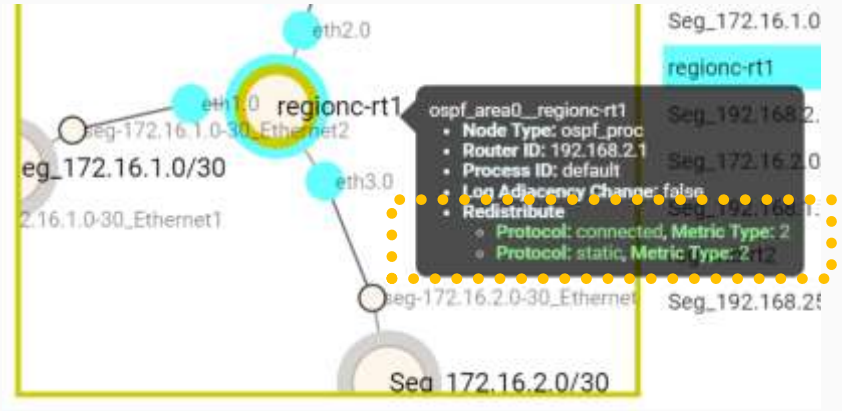
変更されたアトリビュート
(緑➡追加)

© 2023 Okinawa Open Laboratory

# ④To-Be 実環境への適用




OSPF(Area0)




**Output:**
トポロジデータ
(差分情報)

可視化

**Process:**
差分情報の埋め込み

**Input:**
トポロジデータ
(as-is)

**Process:**
Model diff

トポロジデータ
(to-be)

Original Env.

Converter

**Input:**
トポロジデータ
(to-be)

Emulated Env.

**Process:**
差分コンフィグ生成

**Output:**
差分コンフィグ

```
$ bash demo_step4.sh
[
regionc-rt1,
router ospf 65000
 redistribute connected metric 1 metric-type 2
]
[
regionc-rt1,
router ospf 65000
 redistribute static metric 1 metric-type 2
]
```

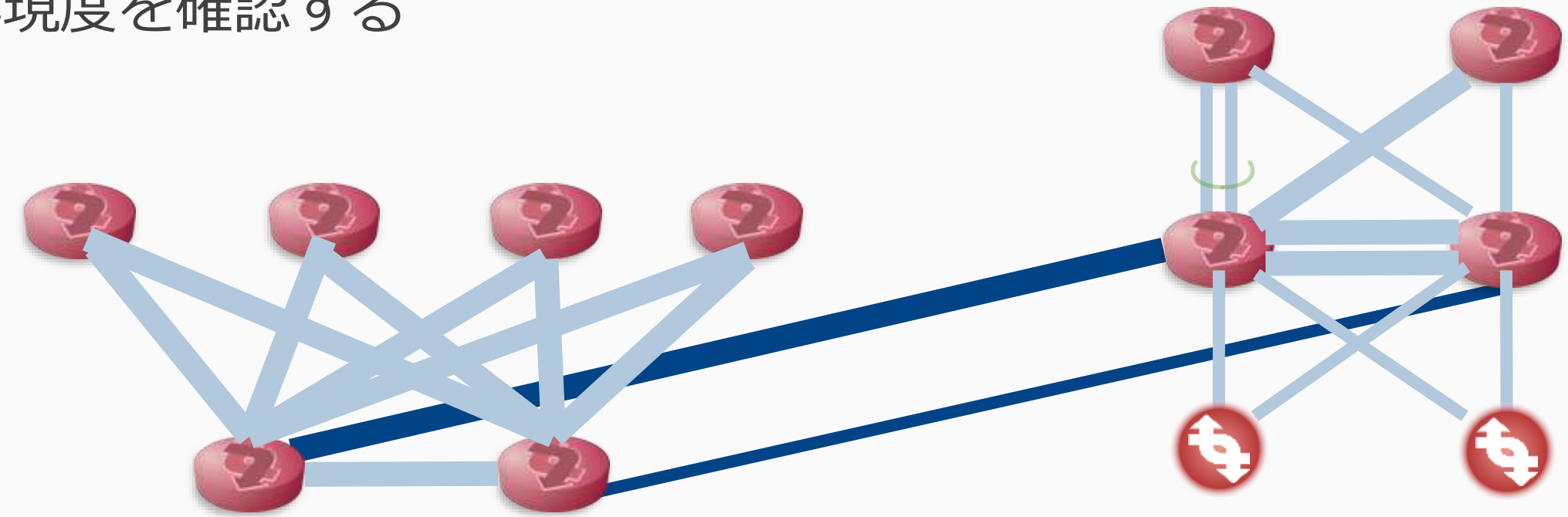

実環境 (Original env) への
フィードバック
➡ 既存の運用方針に従う

同じ"名前空間"であれば
新旧(as-is/to-be)の比較(diff)ができる

© 2023 Okinawa Open Laboratory

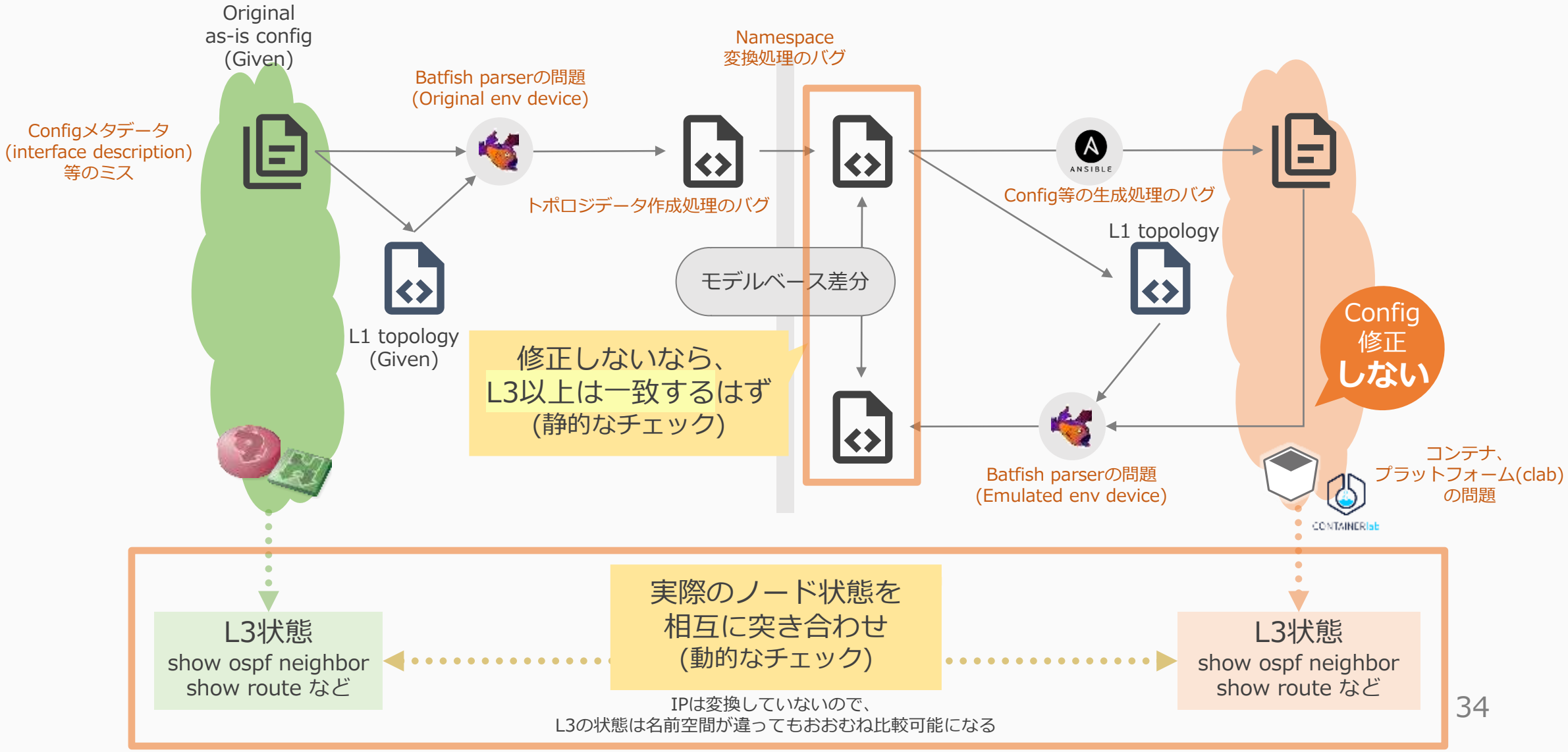
# 実環境ユースケース: NTTCom検証網を再現

- 複数メーカのルータが混在する、
  運用中の構成をContainerlab上に再現する
- 再現度を確認する



© 2023 Okinawa Open Laboratory

# 元のNWの「再現」確認


Original
as-is config
(Given)
Configメタデータ
(interface description)
等のミス
Batfish parserの問題
(Original env device)
Namespace
変換処理のバグ
トポロジデータ作成処理のバグ
L1 topology
(Given)
モデルベース差分
修正しないなら、
L3以上は一致するはず
(静的なチェック)
ANSIBLE
Config等の生成処理のバグ
L1 topology
Config
修正
しない
Batfish parserの問題
(Emulated env device)
コンテナ、
プラットフォーム(clab)
の問題
CONTAINERlab
L3状態
show ospf neighbor
show route など
実際のノード状態を
相互に突き合わせ
(動的なチェック)
L3状態
show ospf neighbor
show route など
IPは変換していないので、
L3の状態は名前空間が違ってもおおむね比較可能になる

# 実環境を使うことで発覚した問題

- 変なデータや中間処理の問題があっても何らかの結果は出る
  - 元データのチェック重要 (interface description: L1トポロジ)
  - 元環境(Original)のNW設計の理解が重要
    - 大きな環境で試すほど「こうなるはずだ」を提示するのが難しくなる
    - 機械的なチェックも必要 👉「再現」チェック…静的・動的なチェック
- IF 名前正規化できない問題 (後述)
  - 多数のルール: 同一ベンダ製品でも違う
  - Batfish出力も微妙に正規化のルールにばらつきがある (10G/100Gで違う)
- OSPF Proc ID消える問題
  - cRPD: process-id持たない ➡ IOS config を cRPD config に変換したときにprocess-id情報が消えてしまう (cRPDからIOSに変換しなおそうとしたときに情報がない)
  - 👉名前空間の変換テーブルに含める：IF名以外にも変換すべき識別子がある

© 2023 Okinawa Open Laboratory

# どれくらいのサイズまで作れそうか?

| | | 実験用NWデモ | NTT Com 検証構成 |
|---|---|---|---|
| 規模(サイズ) | | 6ノード<br>9セグメント | 12ノード<br>12セグメント |
| 機器 | | CPU ：Xeon Silver 4216(16C/32T)<br>Mem：64GB | |
| 使用リソース | CPU(peak) | 40% | 42% |
| | Mem(増分) | +2GB | +2GB |
| 処理時間 | 合計 | 153sec | 187sec |
| | Step① | 14sec | 14sec |
| | Step② | 93sec | 125sec |
| | Step③ | 41sec | 43sec |
| | Step④ | 5sec | 5sec |

規模に対して:
- 消費リソースはそれほど増えなかった
- デプロイ処理時間②は増加傾向

今回使用しているサーバでも、
30ノード程度のサイズなら環境起動できるはず。

© 2023 Okinawa Open Laboratory

# つくりこみ: 名前空間 & Config parse

- 検証作業時(②')に、対応する元(Original)インタフェース名がわからない
  - 👉Interface descriptionに元のインタフェース名を埋めて対応 (正直イマイチ)
- CNFだとネーミングルールが大きく違う
  - Loopback系：`lo0` or `lo.0` or `lo0.0` ? (cRPDが`lo.0`とかなり特殊)
  - トラフィックIF系：`eth`*X* (NW機器とは異なるルールが出てくる)
  - 👉Batfishでオレオレパッチを作って対応
- Batfish (Config parser) 問題
  - IOS系インタフェース名の正規化ルールの違い (`TenGigE`⇔`HundredGigabitEthernet`)
  - 一部コンフィグの "誤読" (誤parse)
    - 👉Batfishオレオレパッチ
  - OVS非対応
    - 👉OVSにする前にcEOSを使っていたので、cEOSのコンフィグをBatfish向けに生成して、OVSの代わりのコンフィグとしてコンフィグパースさせている

© 2023 Okinawa Open Laboratory

# つくりこみ: Containerlab & CNF

- Containerlab Linux Bridge問題
  - ホストOSのLinux Bridgeを使うため、ホスト側のDocker上に存在しているContainerlab以外の仮想ブリッジの経路とEmulated環境上で経路が重複する可能性があり、使えなかった。
  - 👉OVSコンテナを使うことで対応
- 管理アクセスIF問題
  - コンフィグに見えてこないのにルーティングテーブルに見えてくる
  - OSPFで余分に管理IFの経路を広報
  - 管理IFでOSPFの経路交換してしまう
  - 👉コンフィグで打ち消せる設定なら、ゼロタッチコンフィグ生成時に打ち消しコンフィグ入れ込む

© 2023 Okinawa Open Laboratory

# まとめ

© 2023 Okinawa Open Laboratory

# まとめ

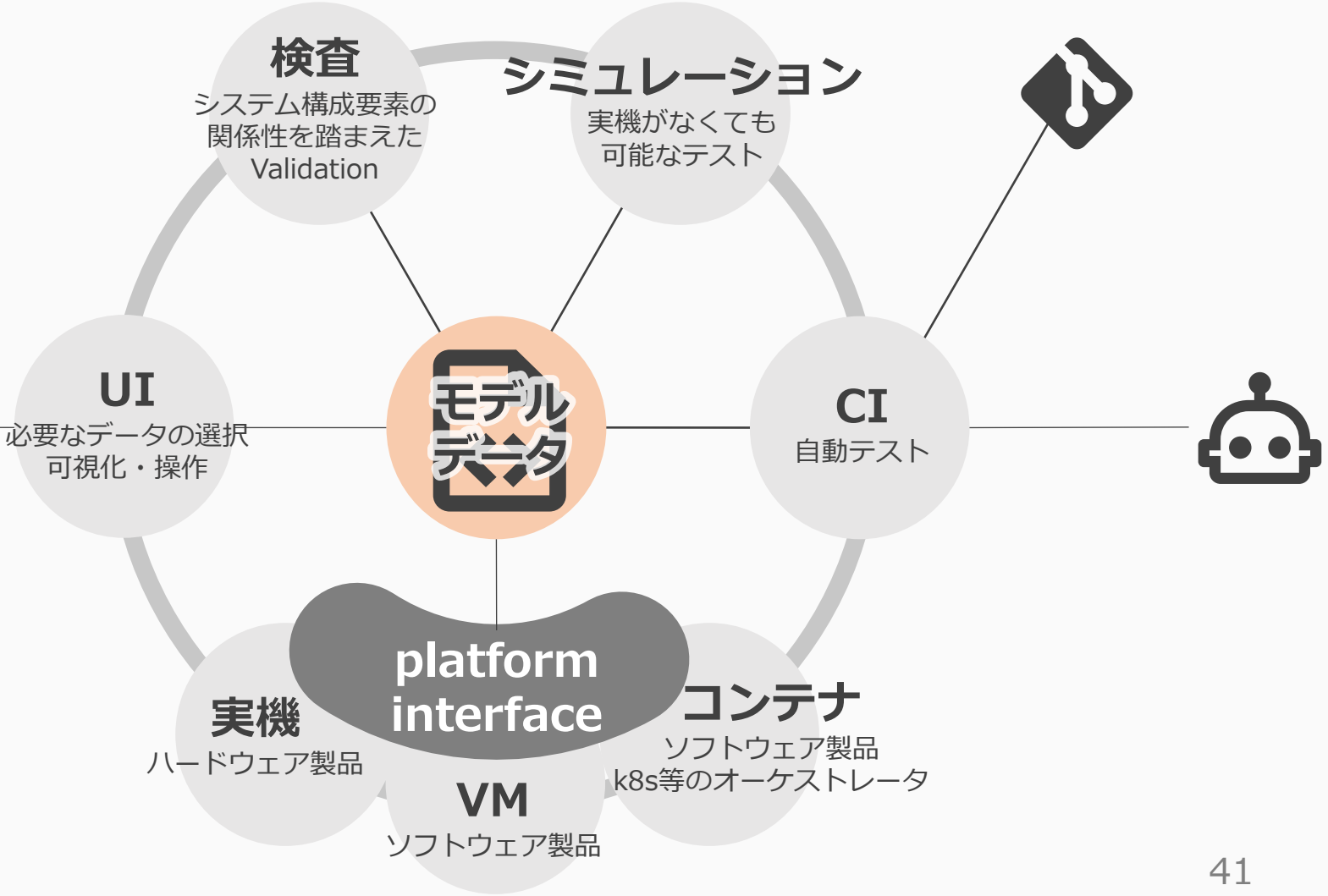
- NW「全体の動作」を再現して検証する
  - コンテナを使ってリソース制約を変える
  - 既存環境の構成情報を抽出して複数レイヤの構成情報(モデル)に再構築する
  - モデルベースに「同等のNW」を再現する (単純な"丸写し"ではない)
- できること・できないこと
  - ☼ネットワークの構成をモデル化してポータブルにする
  - ☼多数のノードを使った「全体の構造」の再現ができそう
    ➡「その構成(トポロジ)」でなければ検証にならないケースのフォロー
  - 🐡識別子(名前)の読み替え問題
  - 🌥使うもの(コンテナ)によって再現可能な範囲は決まってくる
- 「やってみないとわからない」を「やってみてわかった」に変える

© 2023 Okinawa Open Laboratory

# システムの”モデルデータ”を核とした運用プロセス

デプロイ前のテストと予測可能性
静的な検査・動的なテスト
練習・トレーニング・育成

宣言的なシステム運用
CI/自動テストなど
ソフトウェア開発やクラウド開発の
ベストプラクティスのとりこみ




© 2023 Okinawa Open Laboratory

# 議論したいこと

- こういうことができるとしたら実際に使ってみたいですか?
  - コンフィグベースのレビューってしんどくない?
  - 運用のサイクルが、どう変わる(変えたい)と思いますか?
- 実際に現状の運用業務にどんなインパクトがありそう?
  - 想定されるメリット/デメリット
  - 導入にあたっての課題(何がハードルになる?)
- こういうことがやれるといいなあ…
  - 夢とか希望とか
  - こういうこともやれるのでは?

© 2023 Okinawa Open Laboratory

# 補足資料

© 2023 Okinawa Open Laboratory

# 関連資料

- Project Information
  - Model Driven Network DevOps | 沖縄オープンラボラトリ https://www.okinawaopenlabs.org/mdnd
  - NWのモデルベース検査と障害シミュレーションのデモンストレーション - YouTube https://youtu.be/wu9IWRbiKKU
  - ool-mddo – Github https://github.com/ool-mddo

- NW運用におけるモデル定義とReconciliation Loopへの挑戦 - Speaker Deck https://speakerdeck.com/tjmtrhs/nwyun-yong-niokerumoderuding-yi-toreconciliation-loophefalsetiao-zhan
  - NTT Tech Conference 2022 https://ntt-developers.github.io/ntt-tech-conference/2022/
- 沖縄オープンラボラトリ Model Driven Network DevOps (MDDO) Project の紹介 https://enog.jp/wordpress/wp-content/uploads/2022/06/20220610_enog74_takiguchi.pdf
  - ENOG74 Meeting https://enog.jp/archives/2572
- 機器設定ファイルからのトポロジモデル抽出による机上検査を含めたネットワーク設計支援システム https://ken.ieice.org/ken/paper/20220708FCkR/
  - 電子情報通信学会 ICM研究会 (2022/7月) https://ken.ieice.org/ken/program/index.php?tgs_regid=2999890161ea46d8a46d7d0ab86457b95ea553f8b858d0678bf9a3535b3e8b1d&tgid=IEICE-ICM
- モデルを基に本番環境を再現して事前に検証可能にする運用サイクル https://speakerdeck.com/corestate55/ood2022
  - Okinawa Open Days 2022 https://www.okinawaopendays.com/

© 2023 Okinawa Open Laboratory