ネットワーク運用自動化
のはじめかた

2018/6/14 16:35-17:00 JST
Interop Tokyo 2018 SEIKOブース

ITOCHU Techno-Solutions America, Inc.
Taiji Tsuchiya / 土屋 太二

自己紹介

- 土屋 太二/Taiji Tsuchiya
- Career
  - Solution Engineer @ CTC America (2017.8-Present)
    - 米国進出する日系企業向けのITソリューション提供
    - ハイパースケール企業のネットワーク/インフラ技術の調査
    - アプリケーションソフトウェアのプロトタイプ開発
  - Network Engineer @ BIGLOBE (2011.4-2017.8)
    - DC/バックボーン/Peeringの運用/設計/開発
    - ネットワーク運用自動化システム、SDNシステムの開発
- Community Activities
  - 過去JANOG プログラム委員, 実行委員長など
  - NetOpsCoding 運営委員

インフラ運用の自動化への要望
(ネットワークに限らず)

- 機器作業における人為ミスを無くしたい。
- 運用に関連する作業の効率性を高めたい。
  - 例: 機器一台あたりの作業/運用にかかる時間を短縮したい。
  - 例: 技術者一人あたりの運用対象機器を増やしたい。
- 障害発生時に、迅速に原因を発見および復旧したい。
- 障害発生前に、予兆を発見し、危険因子を取り除きたい。
- サービス開発チームからの要求に応じて、
  迅速にインフラの準備・更新・廃止を実施したい。
- インフラの正常な状態を維持したい、
  もしくは異常発生時に迅速に正常状態を再現したい。

ネットワーク運用自動化の一例

- ネットワーク装置の設定の自動化
  - インタフェース, ACL, BGP, Route Policy, MPLS LSP
- 作業手順書における作成工程の自動化・省力化
- 監視ツール運用の自動化・省力化
- リソース情報管理のDB化・脱Excel化
- ネットワークトポロジー管理の見える化・省力化
- ネットワーク障害の自動検知/自動復旧

ネットワーク運用自動化は何から始めるべきか？

- まずプログラミング言語を覚える？
- まずAnsibleを覚える？
- まず自動化関連の本を読んでみる？
- まず他社の成功事例を調べてみる？

=> 自社の運用の課題や理想の姿を検討するところから始めてみましょう。

ネットワーク運用の改善のはじめかた

現場の課題・問題を見つける

(例)
- この作業、時間かかりすぎている。
- 今の運用の方法では、効率が悪い。
- 新サービス導入には、今の運用では対応できない。
- 品質を上げるために、今の監視では不足している。

(CTO 目線で)
改善の仮説を立てる

- どういったシステム・方針であれば解決可能か？
- 何を作る or 変える必要があるか？
- システムを作らずして、解決できる方法は無いか？
- 顧客や関係者が求めている本質は何か？
- やり方を変えるには、誰に相談する必要があるか？

仮説を検証する

- 関係者と議論し、フィードバックをもらう。
  - 上司, 同僚, 他部署メンバー, システム利用者…
- 検証に時間をかけすぎないように注意。

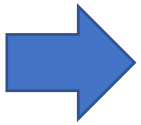
今できることをはじめる

- 今ある予算/メンバー/リソースでできることを始める。
- 開発初心者の場合は、リスクの小さい課題を選ぶ。
- とにかく手を動かす。ある程度の失敗は許容する。
- 小さな成功を繰り返し、徐々に大きな課題にも挑戦していく。

参考: 土屋がはじめにやったこと

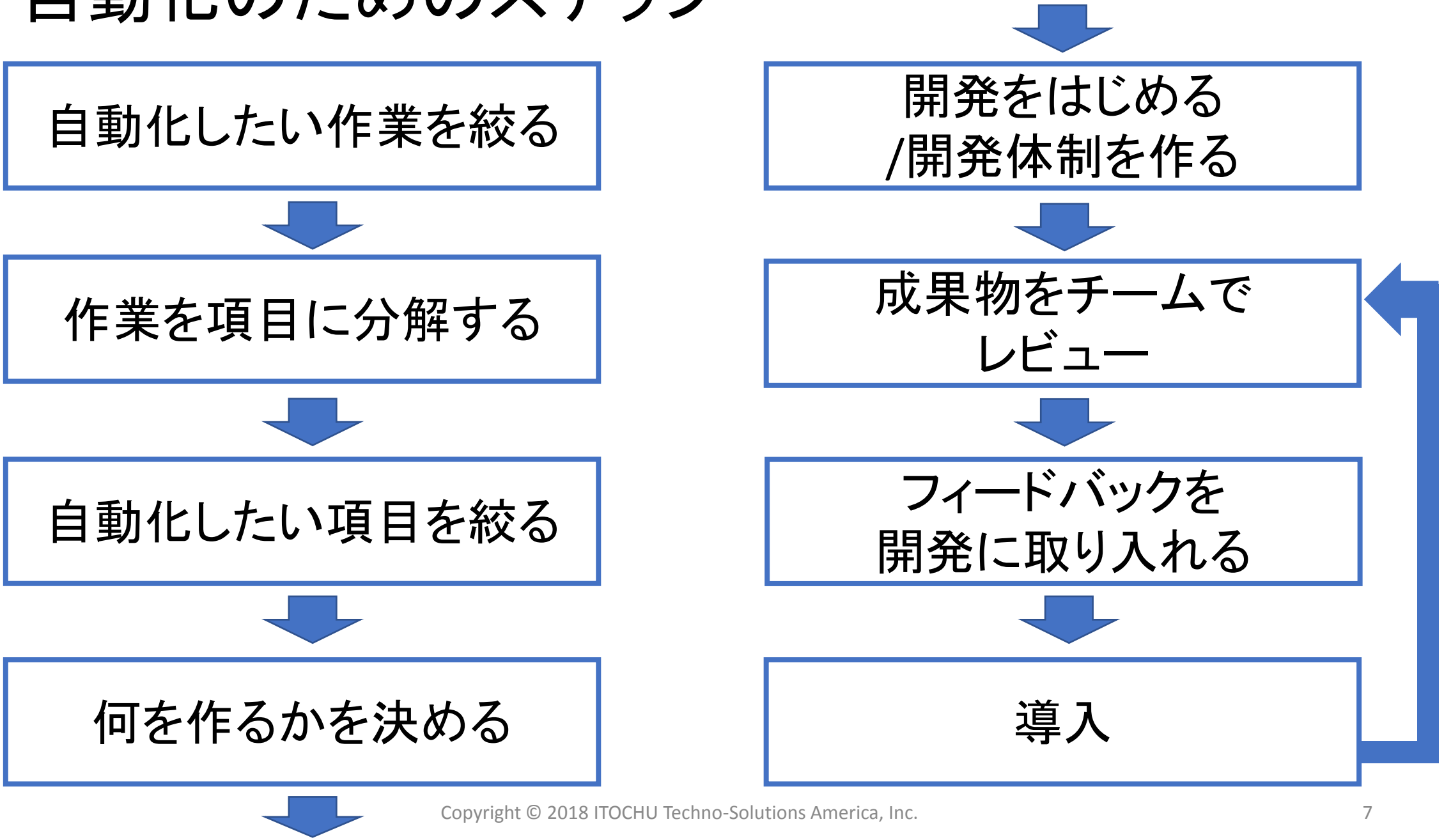
- ネットワークチームを対象にPython勉強会開催(週1回を1年ほど)
  - 社内別チームのPython経験者に協力してもらい、チームメンバーに教えてもらう
  - 数ヶ月後、全メンバーがPythonプログラムを読み書きできるように。
- とにかく動くツールを作ってみて、改善効果を出す
  - 例: Juniperコンフィグの変換ツールの開発(30行程度のプログラム)

```
system {
    host-name R1;
    time-zone Asia/Tokyo;
    root-authentication {
        encrypted-password "$1$9kcwd00g$YDqr8sBMaAh8SOCjQ2f4b0";
    }
}
interfaces {
    fe-0/0/0 {
        unit 0 {
            family inet {
                address 192.168.1.1/24;
            }
        }
    }
}
```



```
edit system
    set  host-name R1
    set  time-zone Asia/Tokyo
    edit root-authentication
        set encrypted-password "$1$9kcwd00g$YDqr8sBMaAh8SOCjQ2f4b0"
    up
up
edit interfaces
    edit fe-0/0/0
        edit unit 0
            edit family inet
                set  address 192.168.1.1/24
            up
        up
    up
up
```

自動化のためのステップ



自動化したい作業を絞る

- 自動化したい作業をリストアップ
    - IXピア開通作業 / プライベートピア開通作業 /トランジット開通作業 /トランジット Route Policy変更作業 / ACL追加作業 / VLAN追加作業 / 初期構築
- 開発初期は、「リスクの小さい作業」「開発の敷居が低い作業」を自動化対象に選ぶと、成功実績を作りやすい。
- 自動化に向けて作業内容/コンフィグのフォーマット化標準化も同時並行で行う。

作業を項目に分解する/自動化する項目を絞る

作業項目の一例

作業計画の立案 / パートナーとの調整

↓

リソースの空き状況の調査

↓

作業で利用するリソース情報を確定

↓

作業用のコンフィグおよび手順書を作成

↓

第三者からのレビュー・承認

↓

作業実施

- 装置にログイン
- 装置の状態/ネットワーク正常性を確認
- コンフィグを投入 / Commit実施
- 装置の状態/ネットワーク正常性を確認
- コンフィグを管理システムに登録

作業完了報告

どの工程に時間がかかっているか？

削減効果が大きい工程はどこか？

楽に自動化できそうな項目はどこか？
- 人の判断を挟まずに実施できるか
- 失敗したとしても影響が小さいか
- 技術的に難易度が低いか
- 短期間で実現できるか

不必要な工程、省略可能な工程は含まれていないか？
- コードを書きはじめる前に、この機会に無くせる工程は無いか？
- 必要に応じてキーパーソンと調整し工程の見直しを検討。

何を作るかを決める

- 何を作り、何を作らないか、を決める。
  - GUI？CLI？
  - データベースは必要？
  - 対象機種? OS？ Version？
  - どの工程は機械に任せ、どの工程は人が判断/実施すべきか?
  - 既存のオープンソースやライブラリ, ツールを流用できないか？
自身がコードを書く量を最低限にできないか？

- 「誰が使うか」を明らかにする。
  - 自分自身 or 同僚 or 運用担当者 or 運用委託会社
  - 他者から常時フィードバックをもらえる環境/良好な関係性を構築する
  - 利用者の技術レベルを把握しておく。ベテラン技術者 or 非技術者?

- 「１つのことを確実に完了する小さいツール」を目指す。
「あれもこれも、なんでもできる最強ツール」は目指さない。

開発をはじめる/開発体制を作る

- 誰がどのように開発するか
  - 成果物のイメージと期限を予め決めておく
  - 小さく成果を出し続けられるように計画する。高望みしない/させない。
  - 属人化/ブラックボックス化を防ぐため、一人開発は避ける。
- いつでも開発業務を引き継げるように、
  開発とドキュメント作成を連動して実施する。
- Git や GitHub/GitLab のようなソースコード共有の仕組みを
  取り入れる

成果物をレビューする/
フィードバックを開発に取り入れる

- 定期的にデモ会を実施
  - ２週間に一度程度が目安。
  - デモ会を通して運用担当者(ツール利用者)から
フィードバックをもらいやすい雰囲気を作る
  - 定期的に成果物や進捗、詰まっているポイントを共有。
    - 詰まっていることは、チーム全体の課題であるケースが多い。
    - 一人ではどん詰まりしやすく、問題を抱え込みやすい。
    - チーム全員で問題を解決するための手段を模索する

導入 → フィードバック

- 導入時に気をつけること
  - 開発したツールの普及活動を行う。
    レクチャーや技術支援などを怠らない。
  - 利用者のフィードバックを漏らさないようにする。
  - ドキュメント化も同時に進める。

- 作って終わり、導入すれば終わり、ではない。
  導入開始した後から本当の運用改善が始まる。
    - 不具合対応
    - 想定してなかったユースケース
    - 新たに出る利用者からの要求

サンプルツール

- 本発表では、BGP Peering作業(Private Peer)を例に、自動化ツールの作り方を紹介します。
- 制作したツールはGithubにて公開しています。
  https://github.com/taijiji/peerup
- 制作期間: 2日

プライベートピア作業の自動化ツールの流れ

リソース情報パラメータを入力

　ルータ名, インタフェース名, IPアドレス, Peer先AS番号, Peer先IPアドレス…

ルータ ログイン

ルータ状態確認: インタフェース状態

コンフィグ投入: インタフェース設定

ルータ状態確認: インタフェース状態

コンフィグ投入: BGPネイバー設定

ルータ状態確認: BGPネイバー状態, BGP経路送受信状態

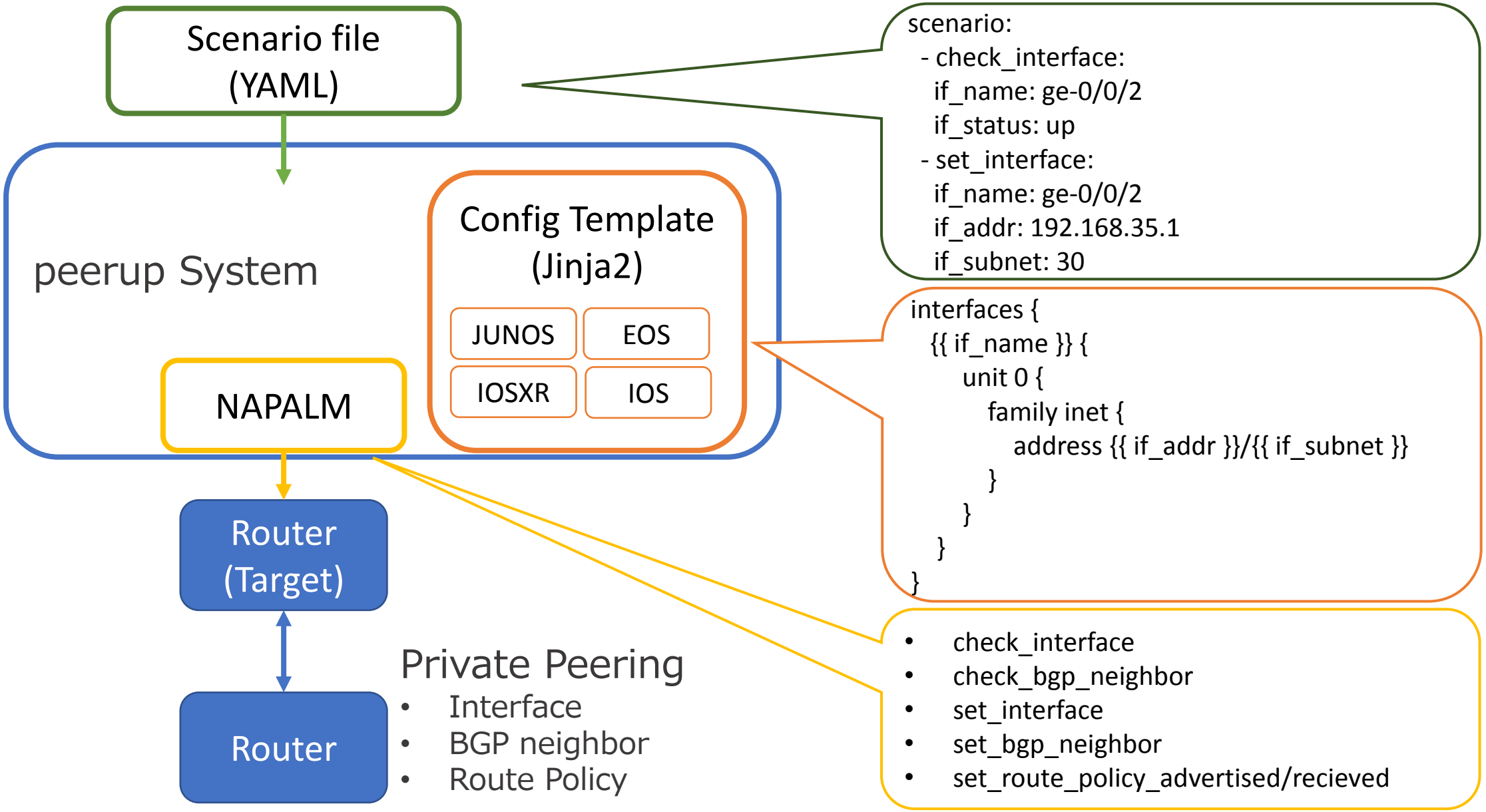
コンフィグ投入: Route Policy export設定変更

ルータ状態確認: BGP経路 受信状態

コンフィグ投入: Route Policy import設定変更

ルータ状態確認: BGP経路 送信状態

サンプルツール: システム概要



サンプルツール: Scenario file

```yaml
purpus:  |
  本作業の目的は、ABC社（AS65002）との BGPプライベートピアするものである。
  The target of operation is BGP private peering with ABC company(AS65002).
operator: Taiji Tsuchiya
operation_date: 20180410
hosts:
  hostname: vsrx1
  os: junos
  username: user1
  password: password1
  mgmt_ipaddress: 192.168.33.2
scenario:
  - check_hostname:
      hostname: vsrx1
  - check_model:
      model: FIREFLY-PERIMETER
  - check_os_version:
      os_version: 12.1X47-D15.4
  - check_interface:
      if_name: ge-0/0/2
      if_status: up
  - set_interface:
      if_name: ge-0/0/2
      if_addr: 192.168.35.1
      if_subnet: 30
      if_description: AS65002_peer
  - check_interface:
      if_name: ge-0/0/2
      if_status: up
```

```yaml
  - set_bgp_neighbor:
      if_name: ge-0/0/2
      neighbor_asnum: 65002
      neighbor_addr: 192.168.35.2
      neighbor_description: AS65002_peer
  - check_bgp_neighbor:
      neighbor_addr: 192.168.35.2
      neighbor_status: up
  - check_bgp_route_advertised:
      neighbor_addr: 192.168.35.2
      advertised_route_num: 0
  - check_bgp_route_received:
      neighbor_addr: 192.168.35.2
      received_route_num: 1
  - set_bgp_route_advertised:
      policy_name: as65002-out
      advertised_route_addr: 172.16.1.0
      advertised_route_subnet: 24
      if_name: ge-0/0/2
      neighbor_addr: 192.168.35.2
  - check_bgp_route_advertised:
      neighbor_addr: 192.168.35.2
      advertised_route_num: 1
  - set_bgp_route_received:
      policy_name: all-accept
      if_name: ge-0/0/2
      neighbor_addr: 192.168.35.2
  - check_bgp_route_received:
      neighbor_addr: 192.168.35.2
      received_route_num: 1
```

サンプルツール: 実行結果の一部抜粋

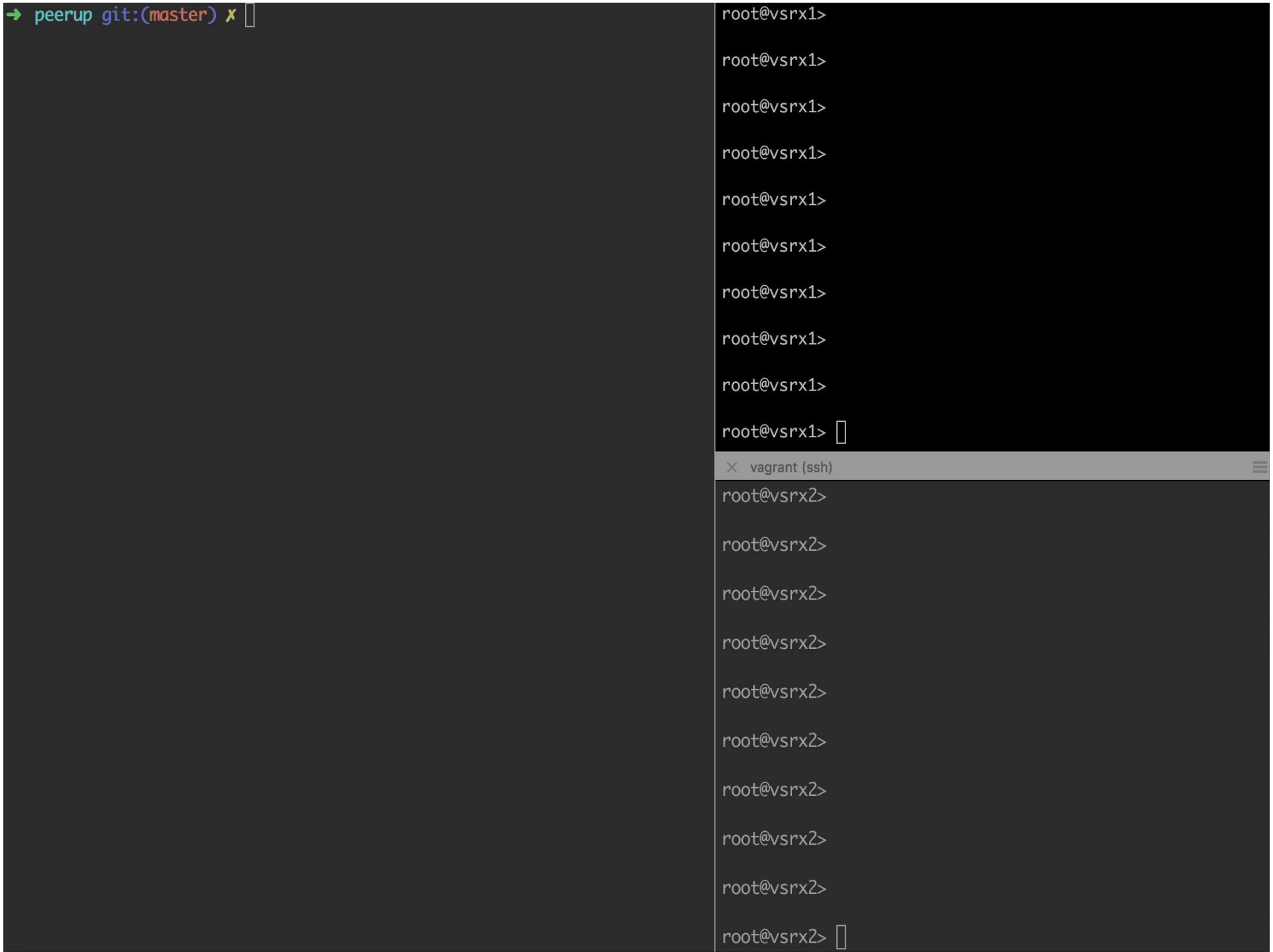
```
➜  peerup git:(master) python3 peerup.py
===== Operation Information =====
purpus:
本作業の目的は、ABC社 (AS65002) との BGPプライベートピアするものである。
The target of operation is BGP private peering with ABC company(AS65002).

Operator        : Taiji Tsuchiya
Operation Date  : 20180410
===== Host Information =====
Host Name       : vsrx1
os              : junos
User Name       : user1
Password        : password1
mgmt_ipaddress  : 192.168.33.2
===== Login Router =====
login router : OK
===== Run Scenario =====
------------------------------
Check Hostname : OK
expected : vsrx1
actual   : vsrx1
------------------------------
Check Model : OK
expected : FIREFLY-PERIMETER
actual   : FIREFLY-PERIMETER
------------------------------
Check OS Version : OK
expected : 12.1X47-D15.4
actual   : 12.1X47-D15.4
------------------------------
Check Interface : OK
expected : up
actual   : up
------------------------------
```

```
------------------------------
Set Interface :
--- Generate Config ---
interfaces {
   ge-0/0/2 {
        unit 0 {
            family inet {
                address 192.168.35.1/30
            }
        }
    }
}
--- Load Config ---
Load: OK
--- Compare Diff ---
[edit interfaces ge-0/0/2 unit 0]
+     family inet {
+         address 192.168.35.1/30;
+     }
--- Commit ---
Do you commit? y/n
y
Commit config: OK
Wait 5 sec
------------------------------
Check Interface : OK
expected : up
actual   : up
------------------------------
```

```
--- Commit ---
Do you commit? y/n
y
--- Commit config ---
OK
Wait 5 sec
------------------------------
Check BGP Neighbor : NG
expected : up
actual   : down
------------------------------
Check BGP Roue Adcertised : OK
expected : 0
actual   : 0
------------------------------
```

緑文字: 異常なし
赤文字: 異常あり
黄色文字: ユーザ判断


➜ peerup git:(master) ✗
root@vsrx1>
root@vsrx1>
root@vsrx1>
root@vsrx1>
root@vsrx1>
root@vsrx1>
root@vsrx1>
root@vsrx1>
root@vsrx1>
root@vsrx1>
root@vsrx1>
vagrant (ssh)
root@vsrx2>
root@vsrx2>
root@vsrx2>
root@vsrx2>
root@vsrx2>
root@vsrx2>
root@vsrx2>
root@vsrx2>
root@vsrx2>
root@vsrx2>
root@vsrx2>

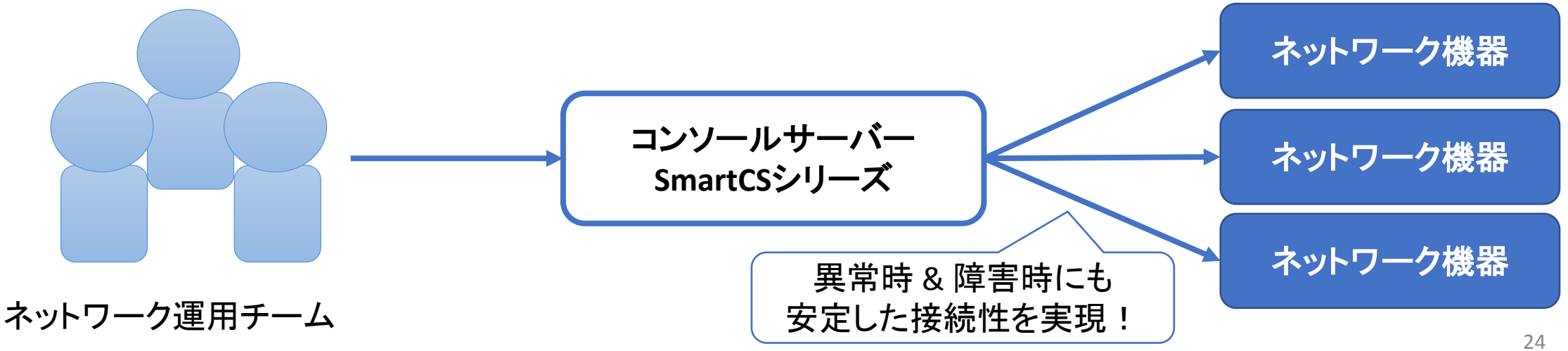
まとめ

- ネットワーク運用自動化のはじめかた
  についてお話しました。
- BGPプライベートピアを想定したサンプルツールを
  作ってみました。
- ネットワークの運用自動化や運用改善の実現には、
  システムを作ること以上に「運用のあるべき姿」や「組織としての取り組み方」が問われます。
- ぜひみなさんの職場で一度議論してみてください。

セイコーソリューションズ x CTC
= 高速 & 安定のIT運用を提供！

コンソールサーバー SmartCS (NS-2250シリーズ)

SmartCSシリーズを中心に国内IT企業にて多数の導入実績あり！



The corporate logo mark embodies our burning vision “to target more than swift perception of global changes and proper response to market shifts -- aspiring to be a part of inducing those transitions.”

Beneath the logo, this desire is expressed in the phrase “Challenging Tomorrow's Changes.”