mixi
GROUP

走りながら作る
ネットワークテンプレートシステム

株式会社ミクシィ
小島 慎太郎

本日 話すこと

mixi GROUP

現在、ネットワーク運用の自動化に取り組んでおり、
テンプレートから 完全なネットワーク設定を生成しています。

- なぜ そのアプローチか
- テンプレートシステムの開発方法
- 効果・課題の共有 → 議論

完成してから商用投入だと
めちゃくちゃ遅い

"走りながら"
= 未完の状態でデプロイ・商用で動かしながら開発するスタイル

テンプレート？

mixi GROUP

たとえば、この記述は 👉

- BGP ピア
- interface (物理・loopback)
  - packet filter
  - sflow
- DDoS 対策用ルーティングいろいろ

```
bgp:
 peers:
   - type: ddos_transit_xxx
     neighbor_as: xxx
     neighbor_address: x.x.x.x
```

の設定を生成する。「パラメーターを入力すれば関連の設定が生成される」しくみがルーター・スイッチの全機能分あって、パラメーターを組み合わせることで完全な running-config を生成する。

自己紹介

小島 慎太郎

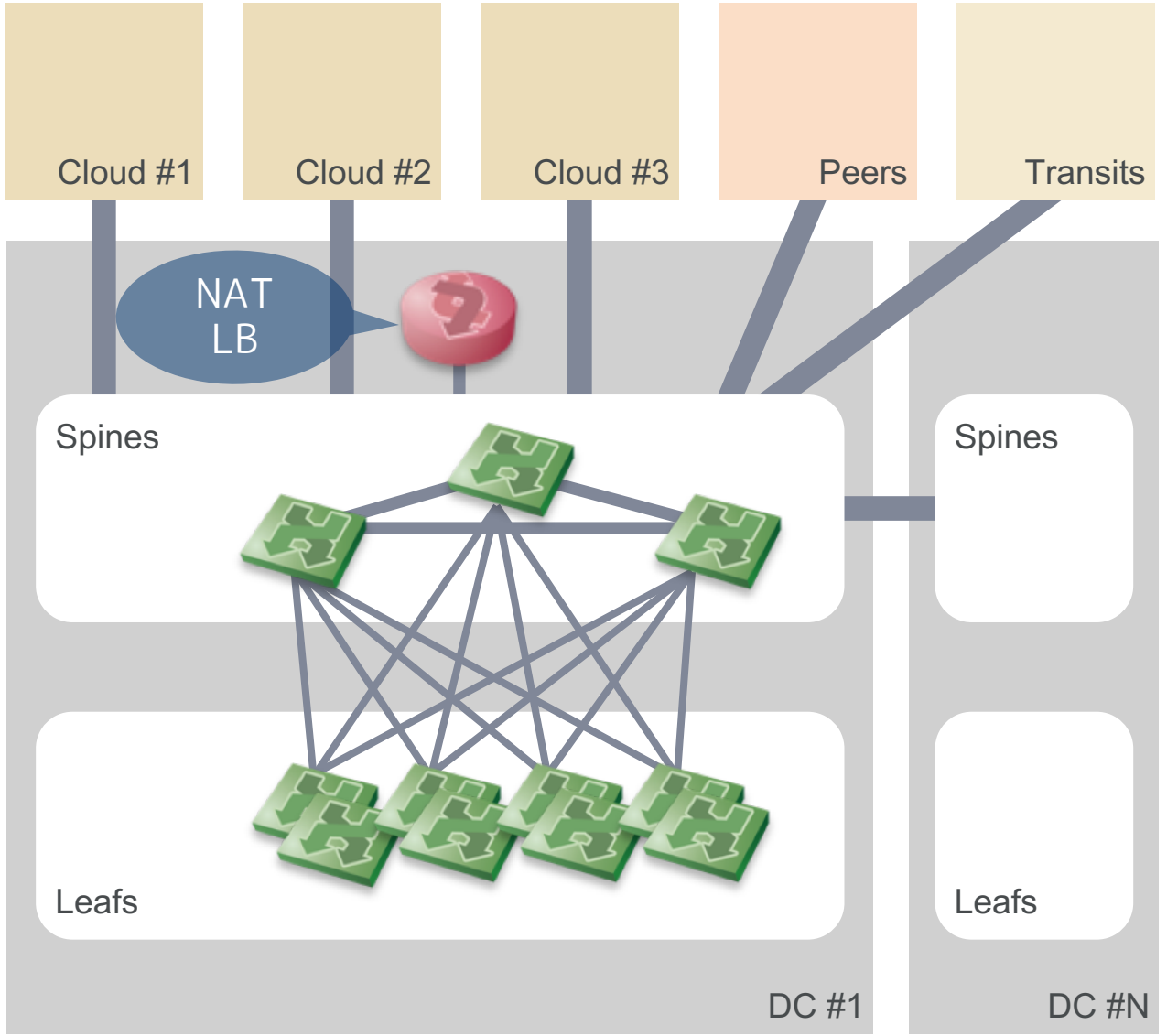
- **codeout** (GitHub / Twitter)
- https://about.me/codeout
- **株式会社ミクシィ 開発本部 インフラ室**
- **やっていること: ネットワーク設計・運用**
- **業務委託メンバー**
  - 本籍：株式会社コーダンス

背景:
ミクシィのネットワーク運用

ミクシィのネットワーク

mixi GROUP

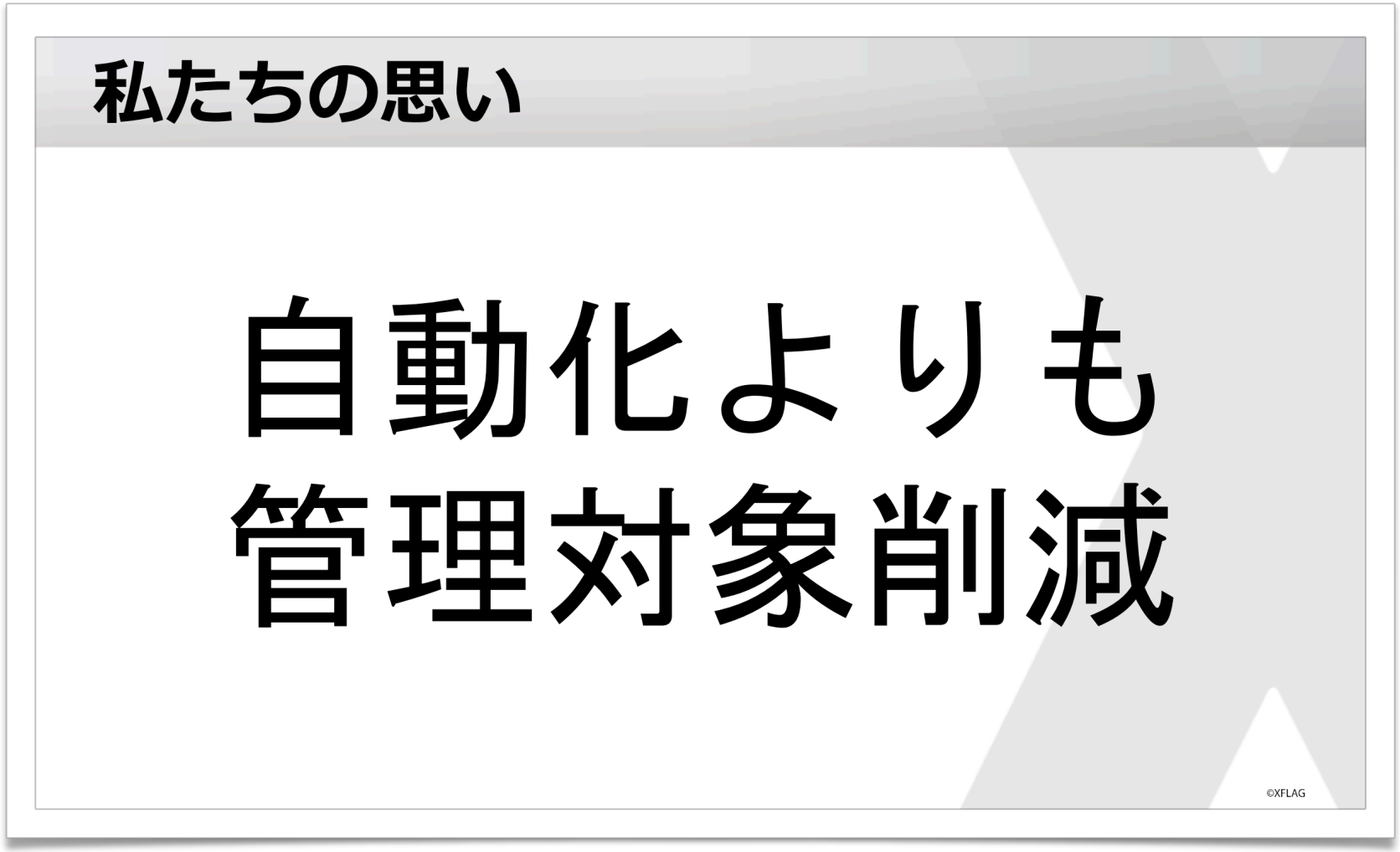
- **Multi-Tenant**
  - MPLS vpn-ipv4 + 6PE
- **Multi-Cloud**
- **Multi-Vendor**
- **BGP CLOS**
  - JANOG40
    "OSPFでToRまでL3にしてみた"
  - JANOG44.5
    "続 ！Goodbye IPv4 On L3 ToR" → **BGP化** 🎉
- **NAT**
- **LB**
- **ベンダー製 L2 / L3 機器中心**
- **L1 機器**
  - JANOG45
    "ホワイトボックス伝送の動向と商用利用について"



ミクシィのネットワーク運用

mixi GROUP

JANOG44



JANOG44

私たちの思い

DONE !!!

自動化よりも

管理対象削減

JANOG44

私たちの思い

自動化まわり

管理対象削減

次は自動化!!!

私たちの思い

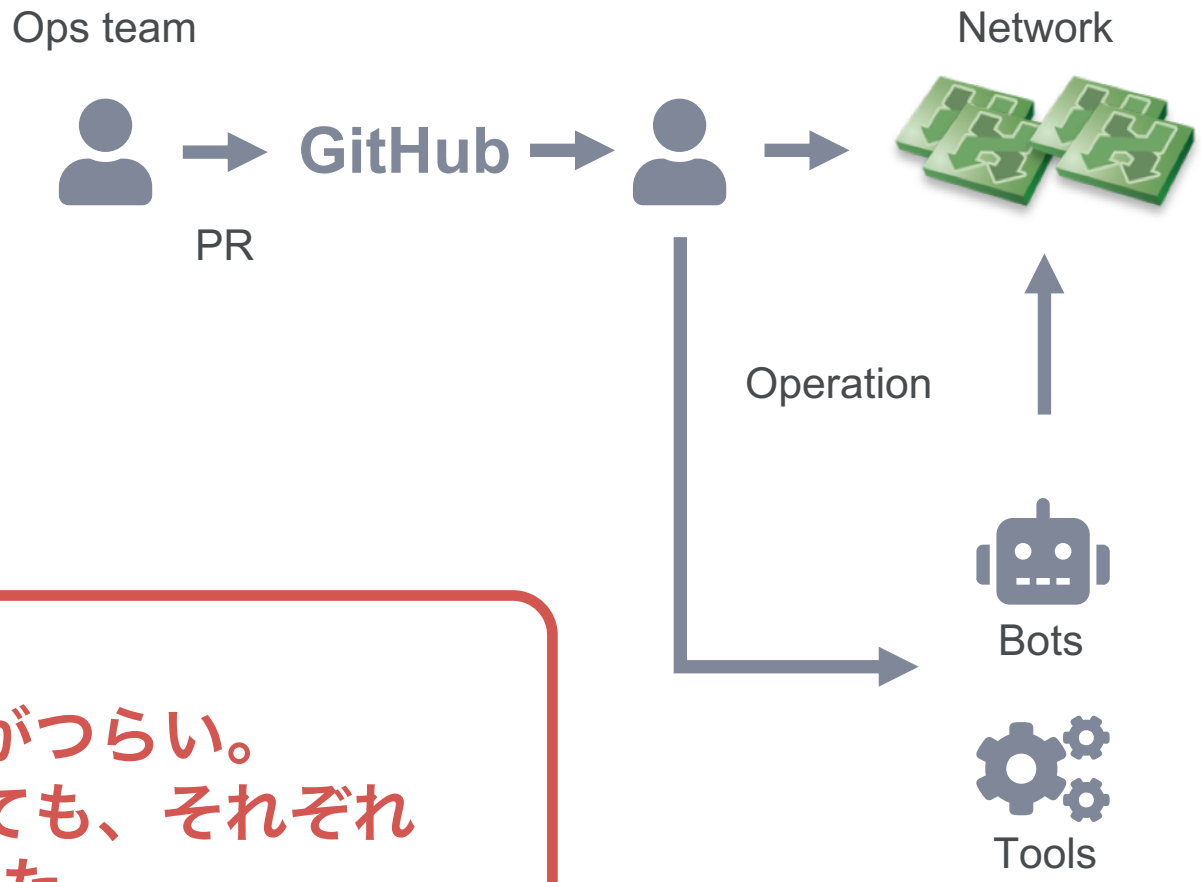
- **ビジネスを止めない。進化を速く**
  - ビジネスはどんどん作られる。ネットワークをボトルネックにしない
  - 「システムが未対応なので無理です」はダメ 🙅‍♂️
- **ネットワークを止めない**
  - 自動化も安全にやる
  - JANOG46
    "突撃！隣の品質設計 ～評価・採用基準どうしてる？～"

ミクシィのネットワーク運用

mixi GROUP

手動オペレーション中心

- GitOps / ChatOps
- ツールの手は借りるが、自動設定に至っていない



課題

Pull Request 作成・レビューがつらい。
ルーティングポリシーを理解していても、それぞれ数時間かかったりしていた

設定が標準化されてなかった 😭

mixi GROUP

peer group
面倒くさい

表記ゆれ

- FooBar-transit or FB-transit
- in/out or import/export
- deny/permit or reject/accept
- "-" or "_"
- 大文字・小文字
- 無限にあった

ルーティングポリシーゆれ

- prefix list
- as-path list
- max prefix limit
- 名前は同じだが、スイッチごとに
  中身が違う
- 数こそ少ないが、
  修正に気を使うやつ

自動化検討フェーズ

構想

mixi GROUP

- **ビジネスを止めない**
  - 「自動化の前に 標準化しましょう」→ **正しい。でも遅い。標準化しながら自動化したい**
  - **標準化されていない設定も許容したい**
  - **自動の仕組みが定着するまで、手動オペレーションを許容したい**
- **特定ハードウェア・ソフトウェアにロックインされない**
  - 設備はマルチベンダー。オーケストレーターも変えていきたい
- **Closed loop automation を目指したい**
  - ネットワークイベント発生 → 自動対処 → ネットワークイベント発生、のループ
  - 例: 通信品質が閾値を下回った & 迂回路があれば、迂回する

Closed Loop Automation?

mixi GROUP

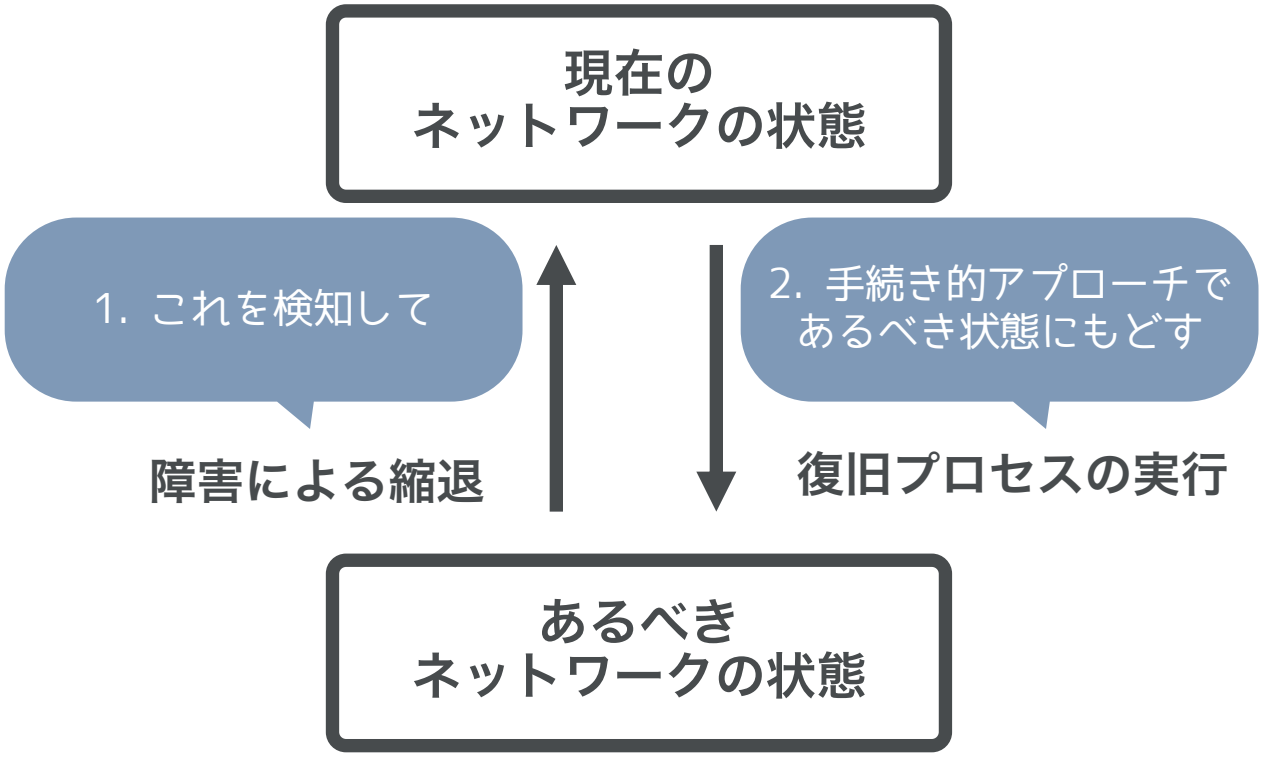
Closed loop automation をやろうとしたら、「宣言的ネットワーキング」に行き着いた

- 「ネットワーク全体のあるべき形」を宣言的に記述し、そこに収束させるネットワーク制御の形
- Kubernetes がお手本
- ぜんぜん道半ばだが、一歩目を踏み出せたと思う
- 今日はここを掘り下げる時間がないので、概要だけ

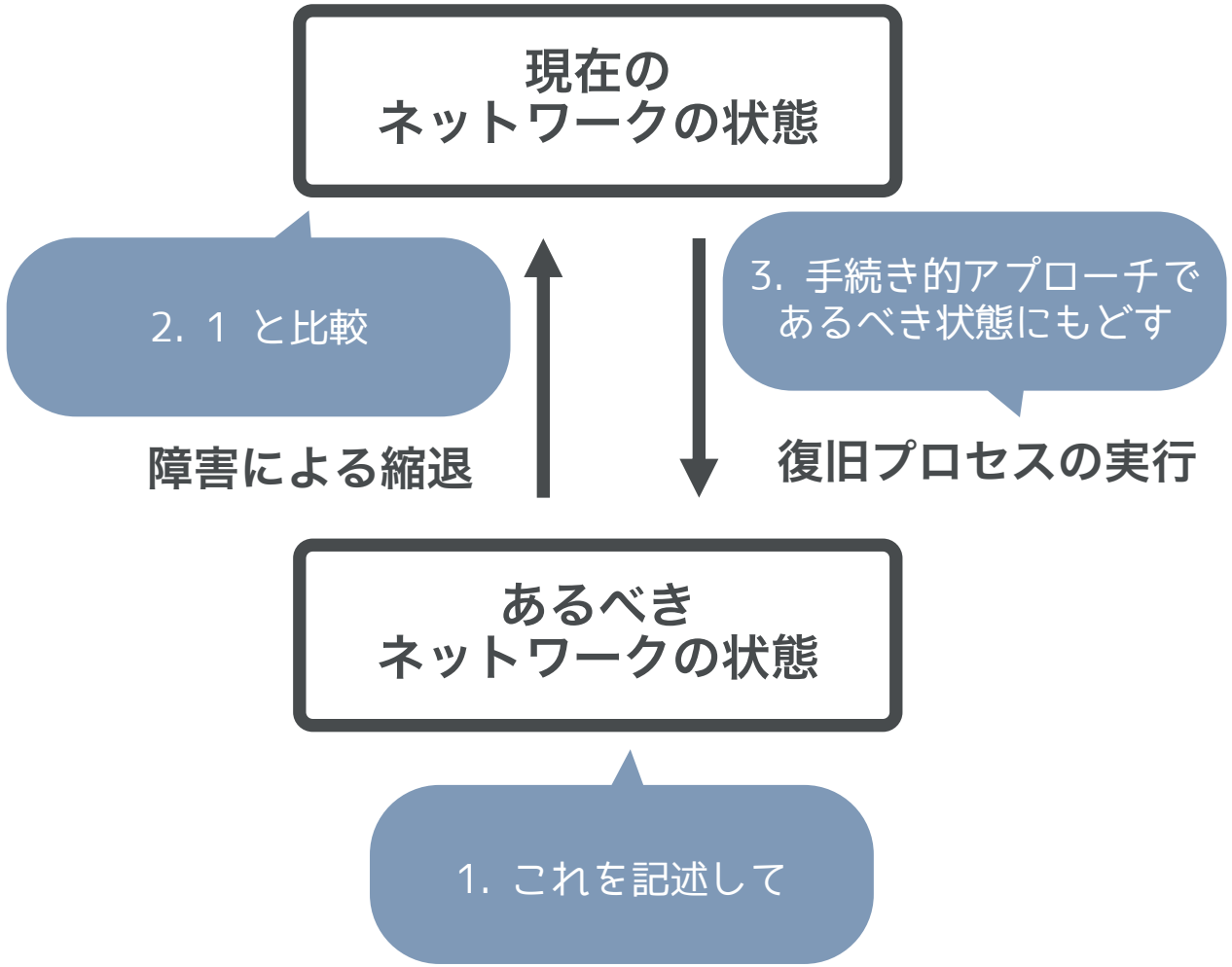
宣言的ネットワーキングと従来手法の対比

mixi GROUP

従来のネットワーク運用

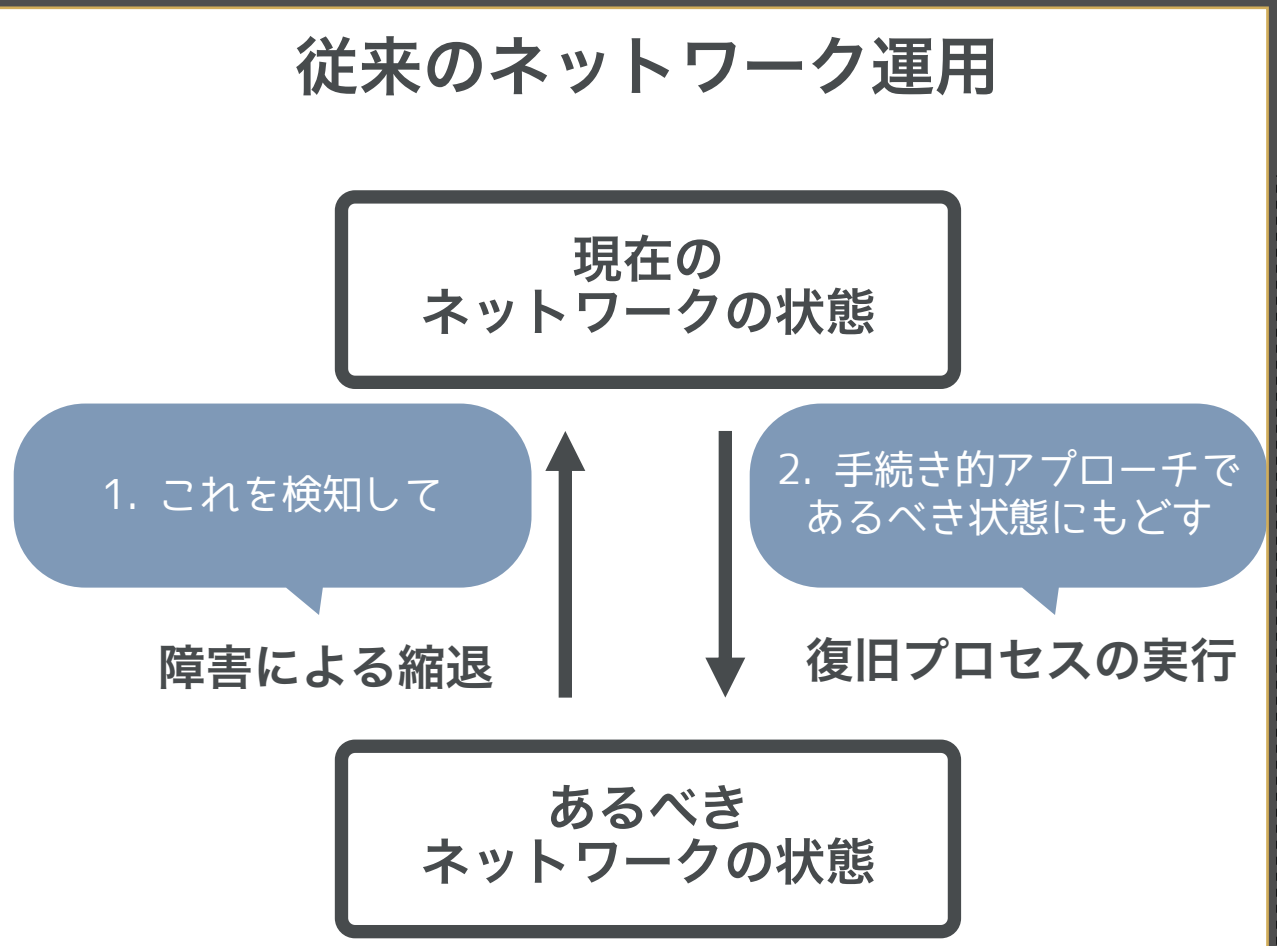


宣言的ネットワーキング



宣言的ネットワーキングと従来手法の対比

mixi GROUP



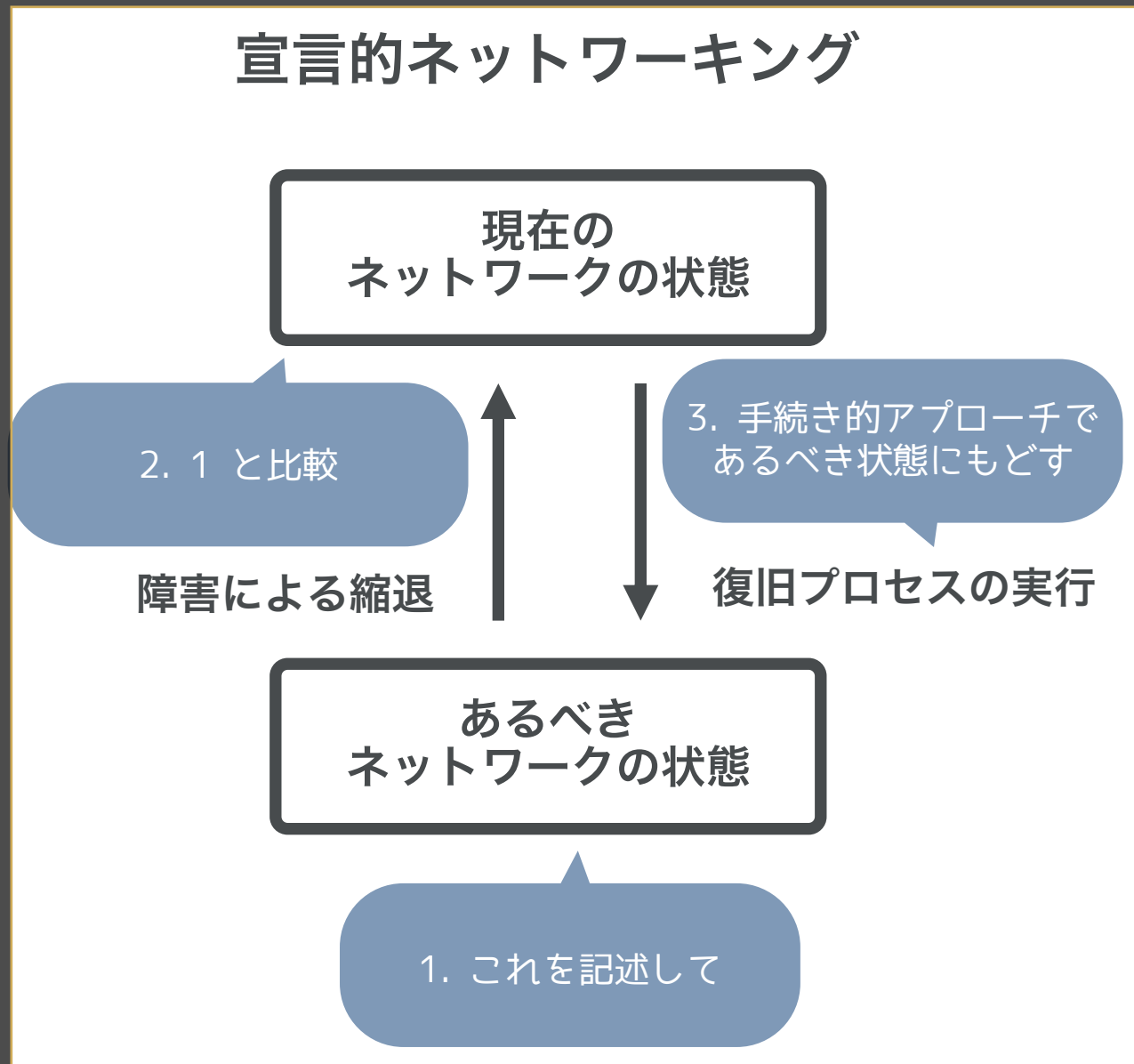
私たちの場合、やりたいのは
「業務ごとの手続きや
障害ごとの手続きを
自動化する」
ではなかった。

パターンの網羅は時間が
かかり、手動オペを行うので
ゴミが生まれる。ゴミは掃除
するべき。

宣言的ネットワーキングと従来手法の対比

mixi GROUP

むしろ
「あるべき形をどう
記述するか」
「どう比較するか」
「収束させるための
手続きを どう
汎用化するか」
という問題だった



「うまく宣言的にやっている」と言われる Kubernetes と比較してみる

- コンテナは自立分散的に動かないので、コンテナ管理・調停役である コントローラーが必要
- **ネットワークがプロトコルレベルで自立分散的に動く**ので、 ネットワーク装置に「あるべき形」を伝えれば、その後の管理・調停は プロトコルが担ってくれる
- ただし、プロトコルが想定しないレベルの制御は効かない

宣言の抽象度・粒度の問題

簡単なレベルでよければ、
あるべき形の宣言 = ネットワーク装置の設定 なのでは？

- ネットワーク装置を設定すると その形に収束しようとする。ただし宣言として使うために条件がある
  - candidate config を、atomic に commit できること
  - 存在しない設定は (no / delete しなくても) 消えること
- 本当に求めているものではない。プロトコルが想定する「あるべき形」は本来のビジネス要件に合わない
  - ❌ Keepalive PDU が Nコ連続で落ちない
  - ⭕ Packet loss < N%
  - ⭕ キャパシティの N% までトラフィックを乗せる

宣言の抽象度・粒度の問題

mixi GROUP

- あるべき形をどう記述するか
- どう比較するか
- 収束させるための手続きを どう汎用化するか

という問題は、あるべき形の宣言粒度を
プロトコルに合わせることで、「ネットワーク
機器の設定をどうモデル化するか」まで簡単に
できる。
(私たちは初手として、それでOK)

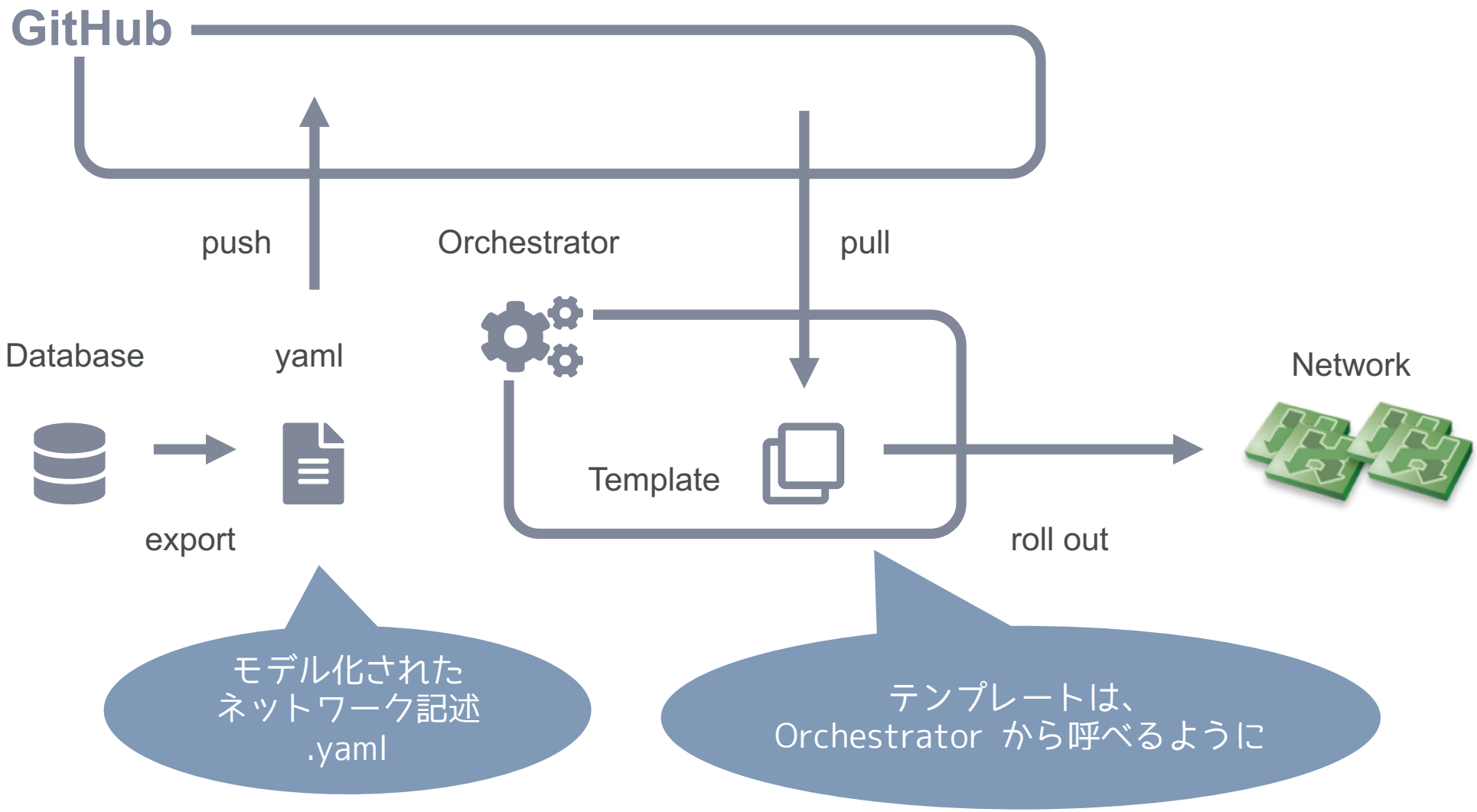
ここまでのまとめ: テンプレート方式を選択した理由

mixi GROUP

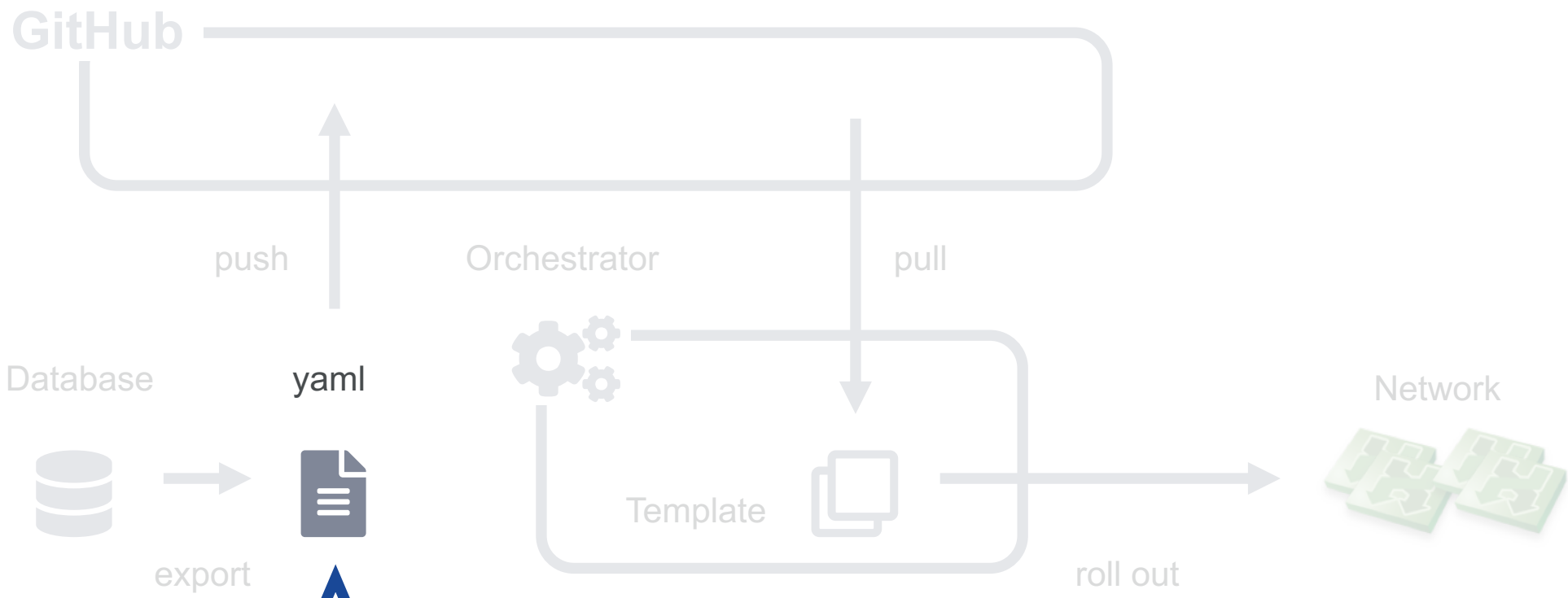
- **業務・障害パターンの網羅に時間がかかると判断したから**
  - Ansible でいう playbook を網羅的に書けない
- **手動オペレーションを許容したいから**
  - 手動オペレーションがミスっていたら検知したい
- **Closed loop automation を目指しているから**
  - 「あるべき形」の宣言が必要になる

設計フェーズ:
どういうものを作るか

全体像



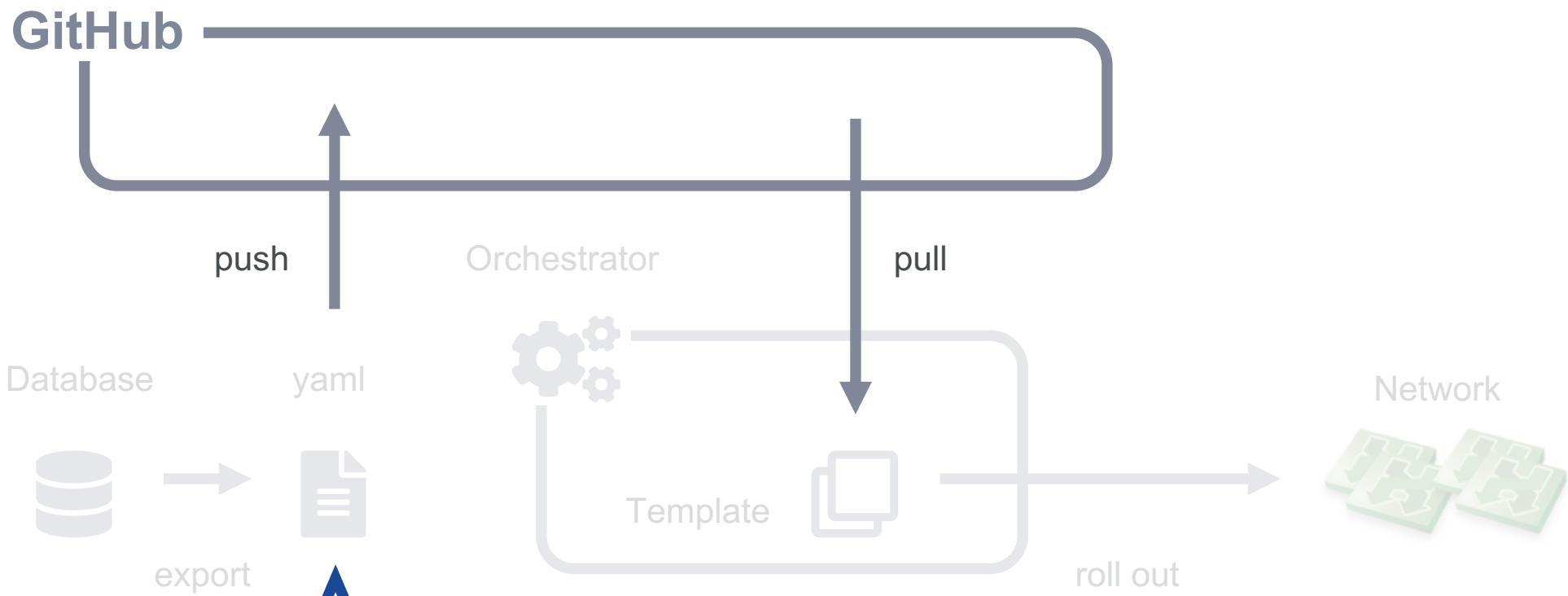
.yaml を DB と テンプレートの interface にする



自分たちがスキーマを決め、定義したモデル。
スキーマに準拠している限り、
DB / テンプレートは好きに開発してOK

マージは git にまかせる

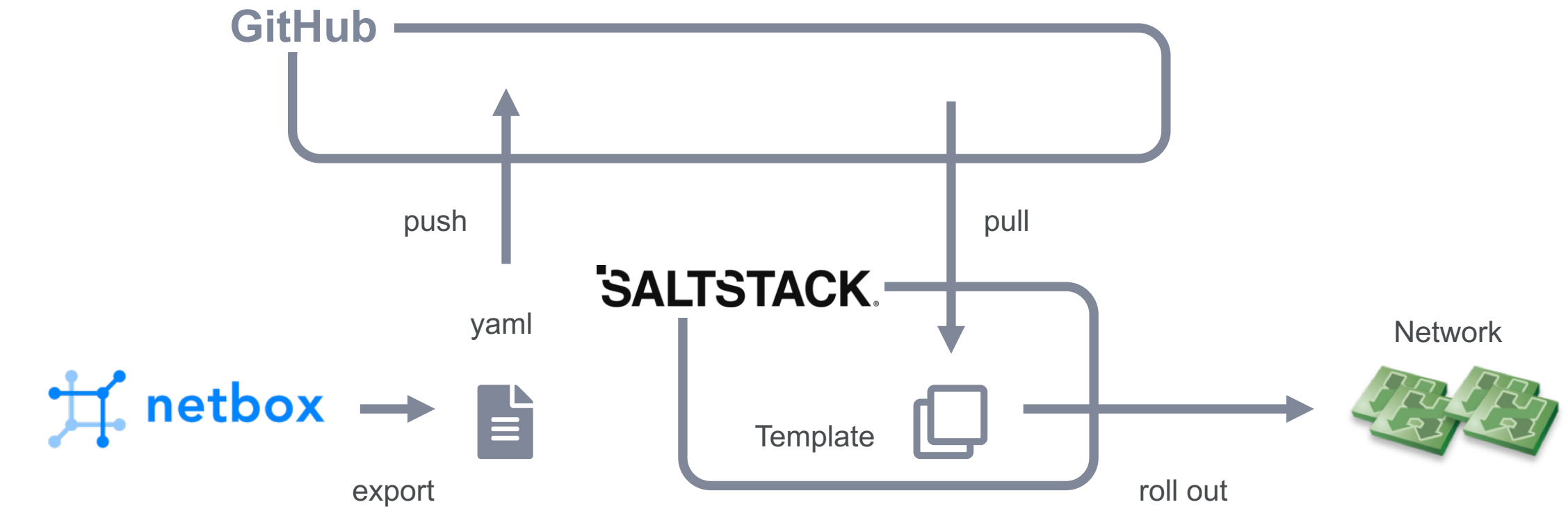
mixi GROUP



変更したいと思ってPR作成したあと、
モデルが変わるかもしれない。
デプロイ前に rebase する。

現在の開発状況

mixi GROUP



WIP。いまは
オペレーターが .yaml を編集

ワークフロー整理中。
既存の GitOps + 手動オペレーション
で設定している

テンプレートが出力する設定

標準化しながら自動化したい・標準外も許容したい

mixi GROUP

ビジネス要件が目まぐるしく変化する

- 標準化の完了を目指していたら終わらない
- 「ポリシーとして標準化して、
  ツール化しないとネットワークに入れられません」はダメ 🙅‍♂️

標準化しながら自動化したい・標準外も許容したい

mixi GROUP

Patch

ポイント！

- モデルには **自由記述欄**（＝標準化がまだ）があり、テンプレート出力（＝標準化済み）とくっつけて出力
- 徐々に標準化を進められ、「標準外の設定」を管理できる
- Patch 0 = 100% 標準 を目指す。定量評価できる

手動オペレーションを許容したい

mixi GROUP

Diff

- 設定順序・削除 などをよしなに考慮したdiff。
  テンプレート出力と running-config を毎日比較
- 構文解析
- **「あるべき形」が設定されているか、厳密にチェック**

テンプレート出力

```
set interfaces xe-0/0/0 unit 10 family inet filter input foo_filter
set interfaces xe-0/0/0 unit 10 family inet address 10.0.10.1/30
set interfaces xe-0/0/0 unit 20 family inet filter input foo_filter
set interfaces xe-0/0/0 unit 20 family inet address 10.0.20.1/30
```

```
delete interfaces xe-0/0/0 unit 10
set interfaces xe-0/0/0 unit 11 family inet filter input bar_filter
set interfaces xe-0/0/0 unit 11 family inet address 10.0.11.1/30
```

==

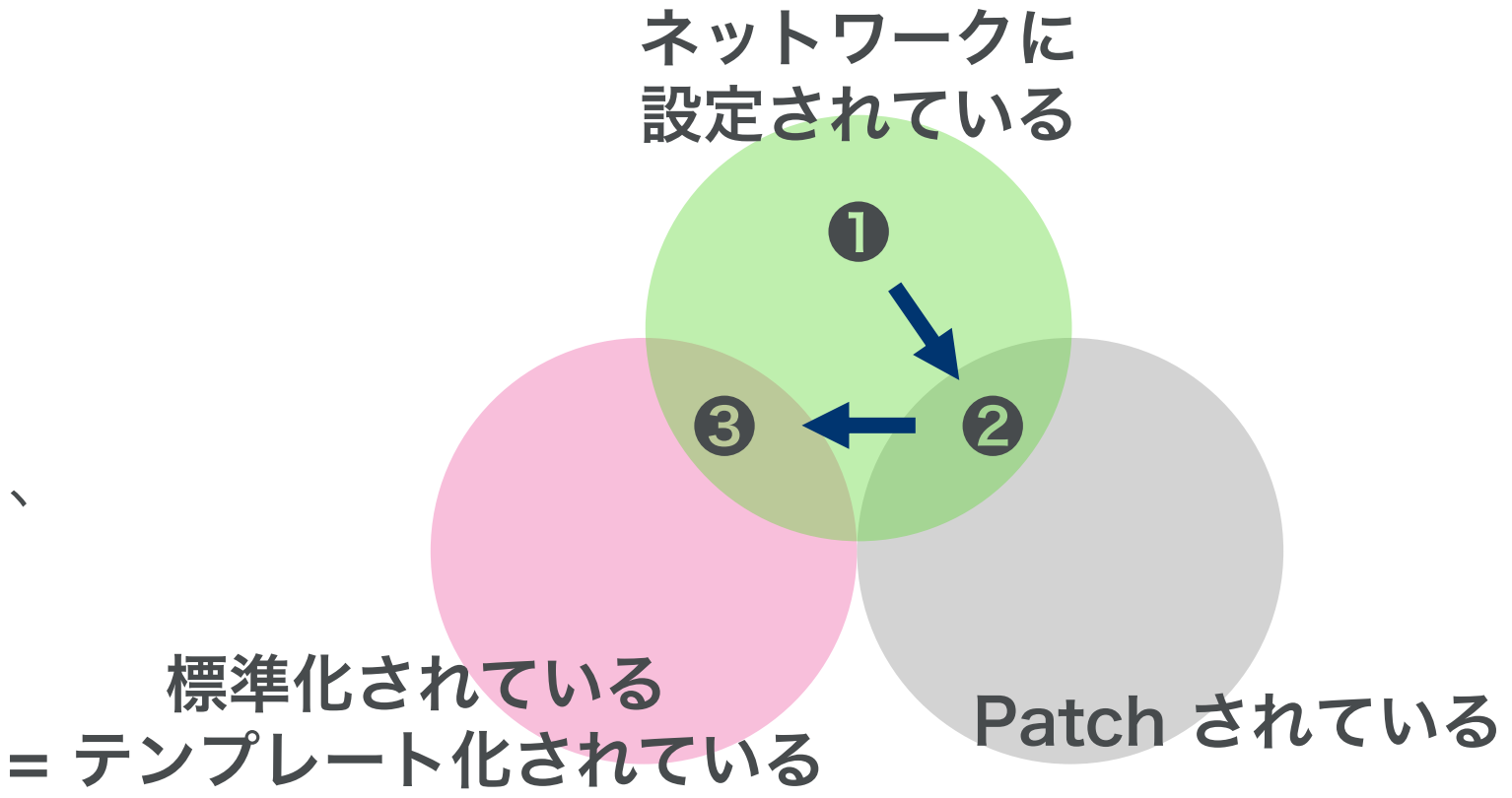
running-config

```
set interfaces xe-0/0/0 unit 11 family inet filter input bar_filter
set interfaces xe-0/0/0 unit 11 family inet address 10.0.11.1/30
set interfaces xe-0/0/0 unit 20 family inet filter input foo_filter
set interfaces xe-0/0/0 unit 20 family inet address 10.0.20.1/30
```

手動で入れた設定のライフサイクル

mixi GROUP

1. ビジネス判断により、
   ネットワークに入れる
2. diff によって検知され、
   Patch する
3. 標準化するべきか議論し、
   テンプレート化する

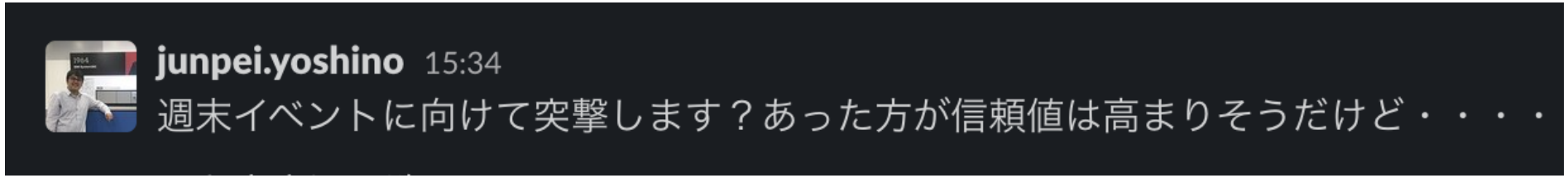


効果:

得られたもの・支払ったもの

まれにあるやつ

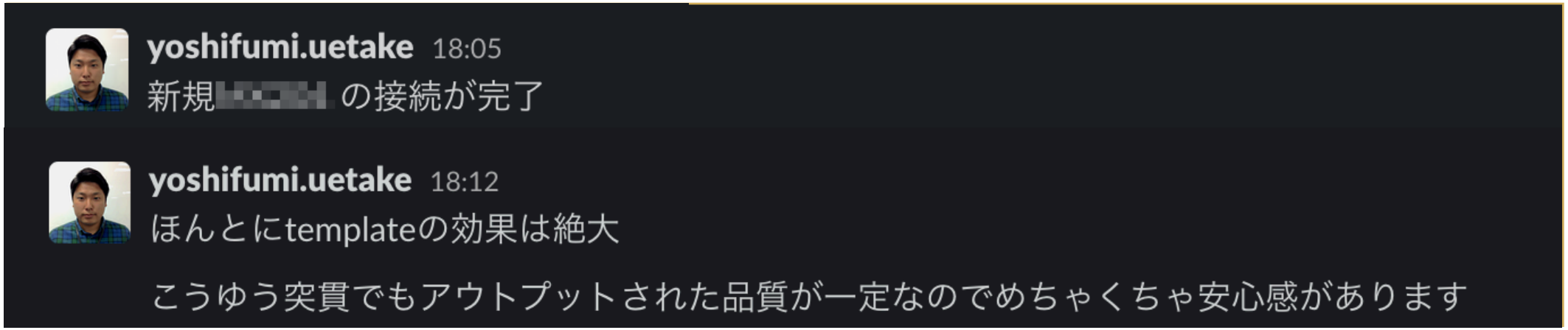
mixi GROUP



突然のコアスイッチ追加…！！
(設定台数としては <20 くらい)

1日で完了 🎉

- ラッキング・モジュール搭載・version up
- 配線
- 設定



ネットワークは正しく設定されている、という安心感

得られたもの

mixi GROUP

効率

- PR作成 ・レビューが一瞬
- モデル.yaml だけでOK
- ベンダーの違いを気にしなくて良くなった 🎉

ブレないネットワークポリシー

- ネットワークが正しく設定されている、という安心感
- ミス・ゴミが混入しない
- ネットワーク全体で整合性がとれる

得られたもの

まだ使いこなせてないが…

あるべき状態の記述

- いろんなシステムから参照できる

冪等性

- テンプレート出力した結果を commit すれば、必ず「あるべき状態」になる
- 差分がなければ、副作用もない
- Closed loop automation / 宣言的ネットワーキング への布石

支払ったもの

mixi GROUP

開発コスト

- 2019/03 開発スタート ~ 2019/06 運用スタート
- 今日まで ざっくり1000h くらい
- 「開発が遅れているため XXX できません」はなかったはず

まとめ

まとめ

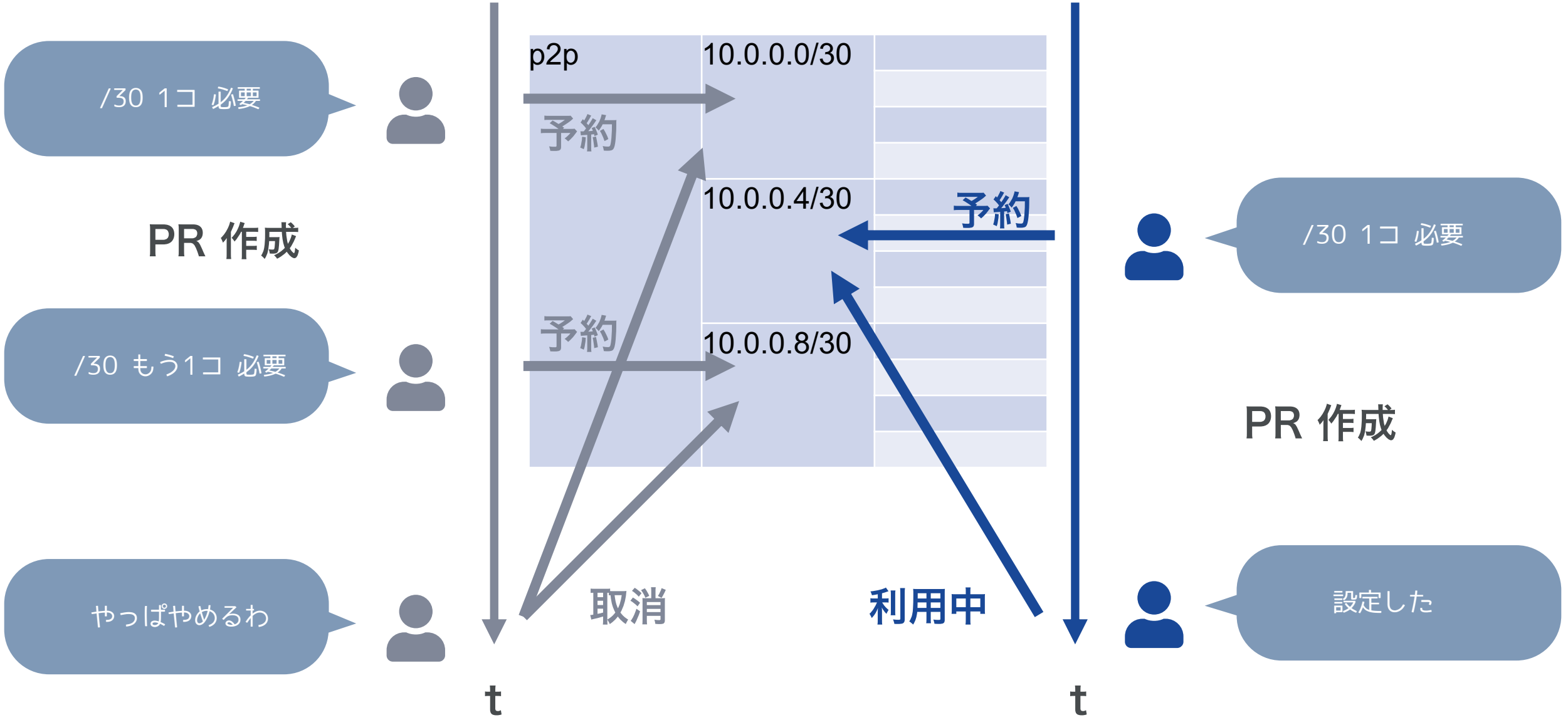
完全なネットワーク設定を出力する、
テンプレートシステムを開発・運用している

- OPEX 削減
- 手動オペレーション・標準外設定を許容しつつも、
ネットワークポリシー強制力を得られた
- **未完でも早くデプロイして、メンバーに見えるところで改善するのが効いた**
    - **ネットワーク運用時、システムとの噛み合わせについて常に考えながら
みんなで改善できる 💪**

課題の共有・議論

課題1: 自動リソースアサイン

mixi GROUP



課題1: 自動リソースアサイン

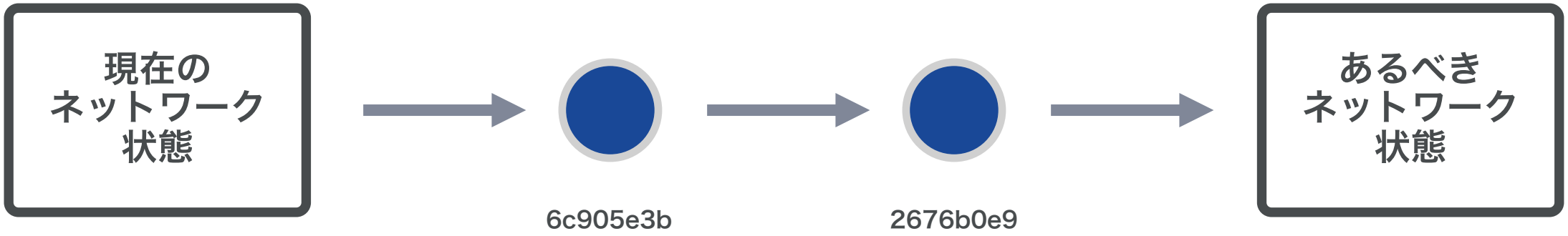
mixi GROUP

- アサインするタイミングは、git commit より前
  - というか任意タイミング
- **PR merge ・ close のタイミングで、アサインしたものを確定・取消**
  - PR に IPアドレスが並んでいたとして、
    どれが固定でどれが自動アサインか判断できるだろうか？
- 人力では可能だった「連番で欲しいのでここ譲ってもらえませんか」とかは諦めないといけない？
- IPAM は PR と連動できるのか？

課題2: 設定デプロイ

mixi GROUP

- あるべき形は分かっている。差分も取れる
- **破壊的な変更は手動でやりたい。実績ある変更は自動投入したい**
- 変更をどうやって分割する？
  - 手動で git commit 分けるアプローチを考え中
  - 業務ごとの自動化では気にしなくてよかった点
- 分割したそれぞれが破壊的かどうか、判定可能か？



課題3: モデルの抽象度

mixi GROUP

- 本来定義したい「あるべき形」
  - たとえばQoS 👉
  - 計測したいメトリックは、
    複数要素の影響を受ける 🤔
- 現状「ネットワーク設定をモデル化した」にすぎない
  - 抽象度を上げられるのか？
  - Intent-based で書ける？
    - 例: クラウドと十分な帯域で接続したい

```
bgp:
 peers:
   - type: ddos_transit_xxx
     neighbor_as: xxx
     neighbor_address: x.x.x.x

     limits:
       packet_loss:
         - target: 10.0.0.1
           threshold: 10**-5
       latency:
         - target: 10.0.0.1
           threshold: 10  # ms
       action: depref

       bandwidth: 10  # Gbps
       action: notify
```

mixi
GROUP