JPNAPのソフトウェア化事情

インターネットマルチフィード株式会社 技術部
川上 雄也

川上 雄也 (@yuyarin)

2007 学部３年：情報系の学部に進学
全然プログラミングができなくて講義が苦痛だった...

2008 学部４年：インターネット系の研究室に配属
株式会社ドワンゴ 研究開発本部でアルバイト開始
C++/PHPでニコニコ動画のバックエンドシステムの開発

2009 情報系の大学院に進学

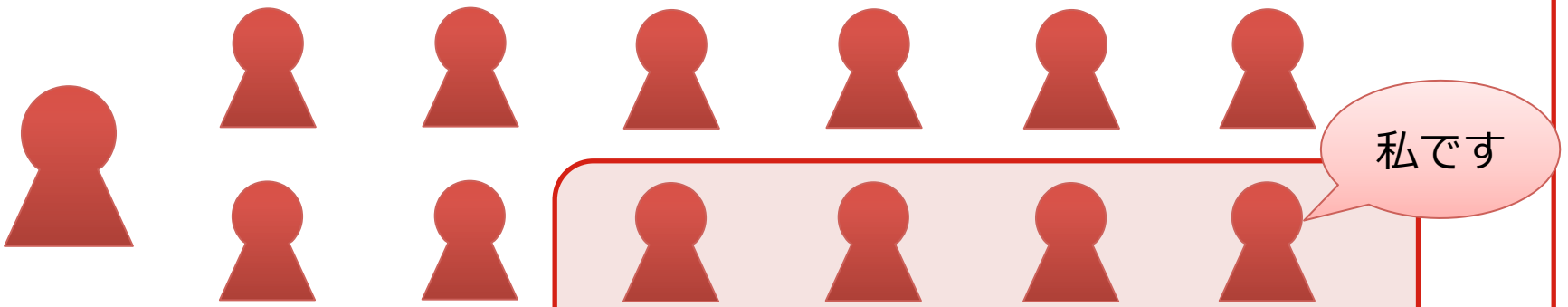
2011 NTTコミュニケーションズ株式会社 入社
インターネットマルチフィード株式会社 技術部 配属
JPNAPのネットワークエンジニア

2014 イマココ

JPNAPを支えるチーム

JPNAPチーム（13名）

- JPNAPの構築・運用、その他技術的事項全般



サーバ・システムチーム（６名）

- 社内システムの運用・管理 ※
- 監視網・社内のサーバの管理
- 顧客向けポータルの開発

v6・インフラチーム（７名）

- IPv6サービスの構築・運用
- 社内ネットワークの管理
- マネジメントネットワークの管理

※かつてはJPNAPの運用向けシステムの開発も行っていた

JPNAPの運用とシステム化の現状

- リソース管理
  - IPアドレス
  - 各種機器・機材のアサイン
- ネットワーク機器の設定
  - IX用L2スイッチ
  - BGPルータ、ルートサーバ
  - 伝送装置
  - 光スイッチ
- サーバの設定
  - MRTG（トラフィック、機器情報）
  - nagios (ping監視)
  - メーリングリスト
  - その他監視運用系のツール類

Excelから
内製のWebUI/DBへ移行

典型的な手順書はツールで
生成できるが修正が必要。
手作業でコンフィグ

コンフィグを手書きから
DBに基づく自動生成へ移行

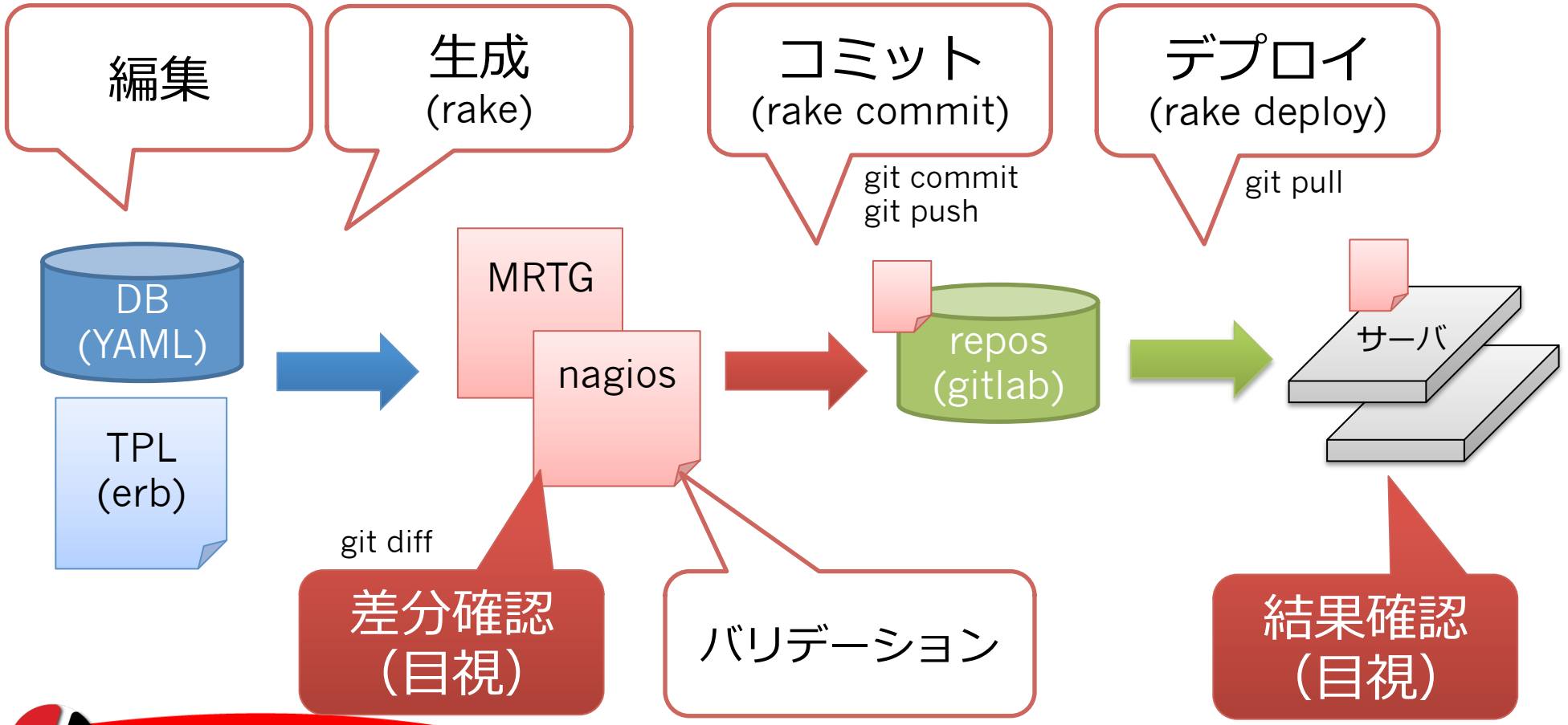
サーバの設定を自動化する前は

RCS管理されたコンフィグファイルを１つずつ漏れがないように手書きで編集していく…

- 不幸な人為ミスが多い
  - 本来人間が気を遣うことに注力すべきでないフォーマットのミスが発生
  - 編集すべきファイルを編集し忘れる
- 作業量が多く稼働がかかる
  - 典型的な新規開通に４～５時間、慣れても２～３時間
- ポリシーやフォーマットがバラバラ
  - 名称や監視項目・監視基準がサービス・拠点で差分がある
- とにかくストレス！！！

サーバの設定をDBから自動生成

- 各作業のトリガーは人間がコマンドを打つこと
- 差分や実行結果は必ず人間の目で確認している



サーバの設定を自動化して…

凡ミス・変更漏れ

稀によくあった → いまのところ無し

作業時間

３時間前後 → 30分未満！

品質

バラバラ → 画一化・統一

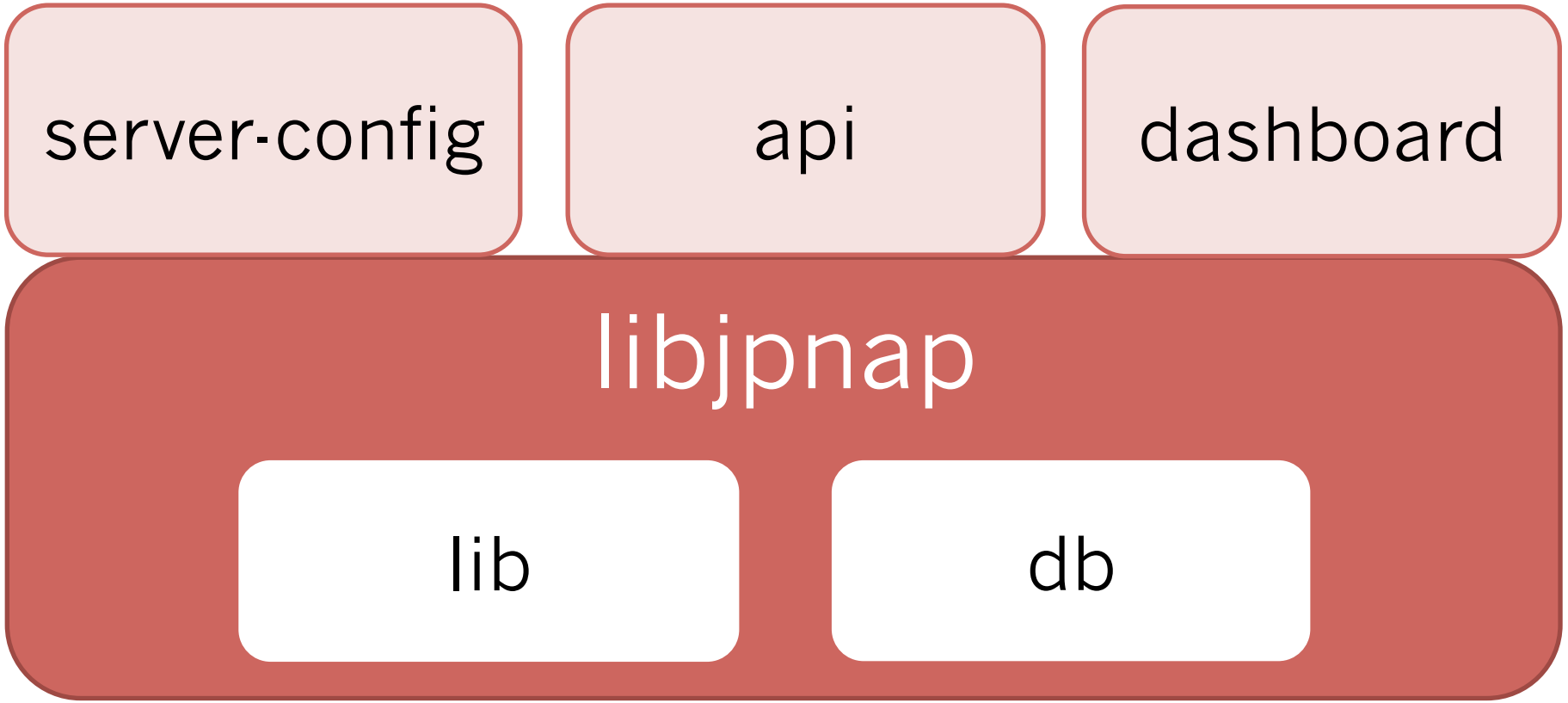
ストレス

非常に高い → ほぼゼロ！

libjpnap

ライブラリを使って色々なツールを作る

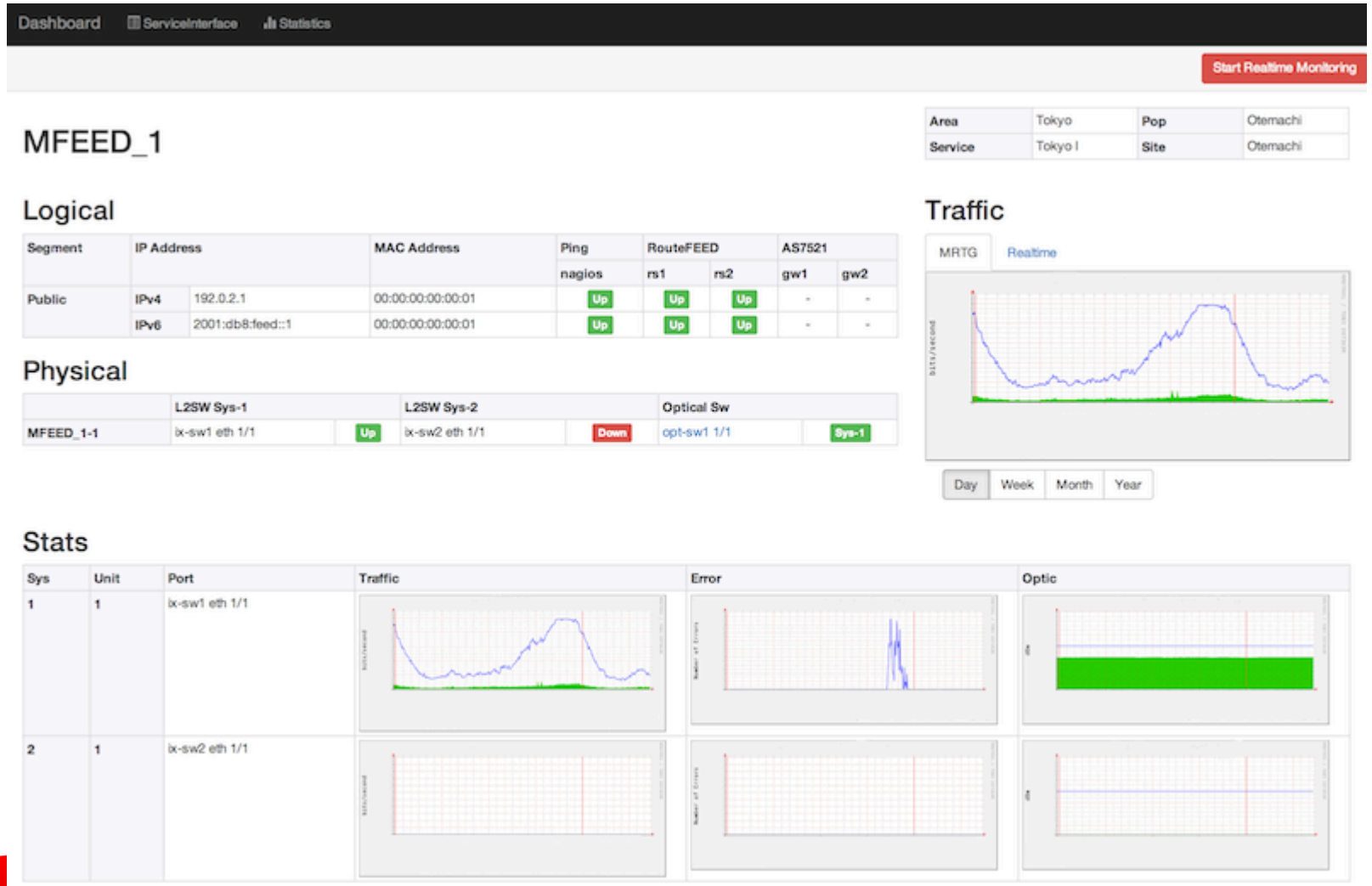
- 再利用可能なライブラリを作ることで、新しいツールの開発コストを下げる
- ライブラリを作ることで、運用を標準化することができる



※たくさんノウハウが溜まったのですが、今回は紹介する時間がありません...

運用高度化 — JPNAP Dashboard

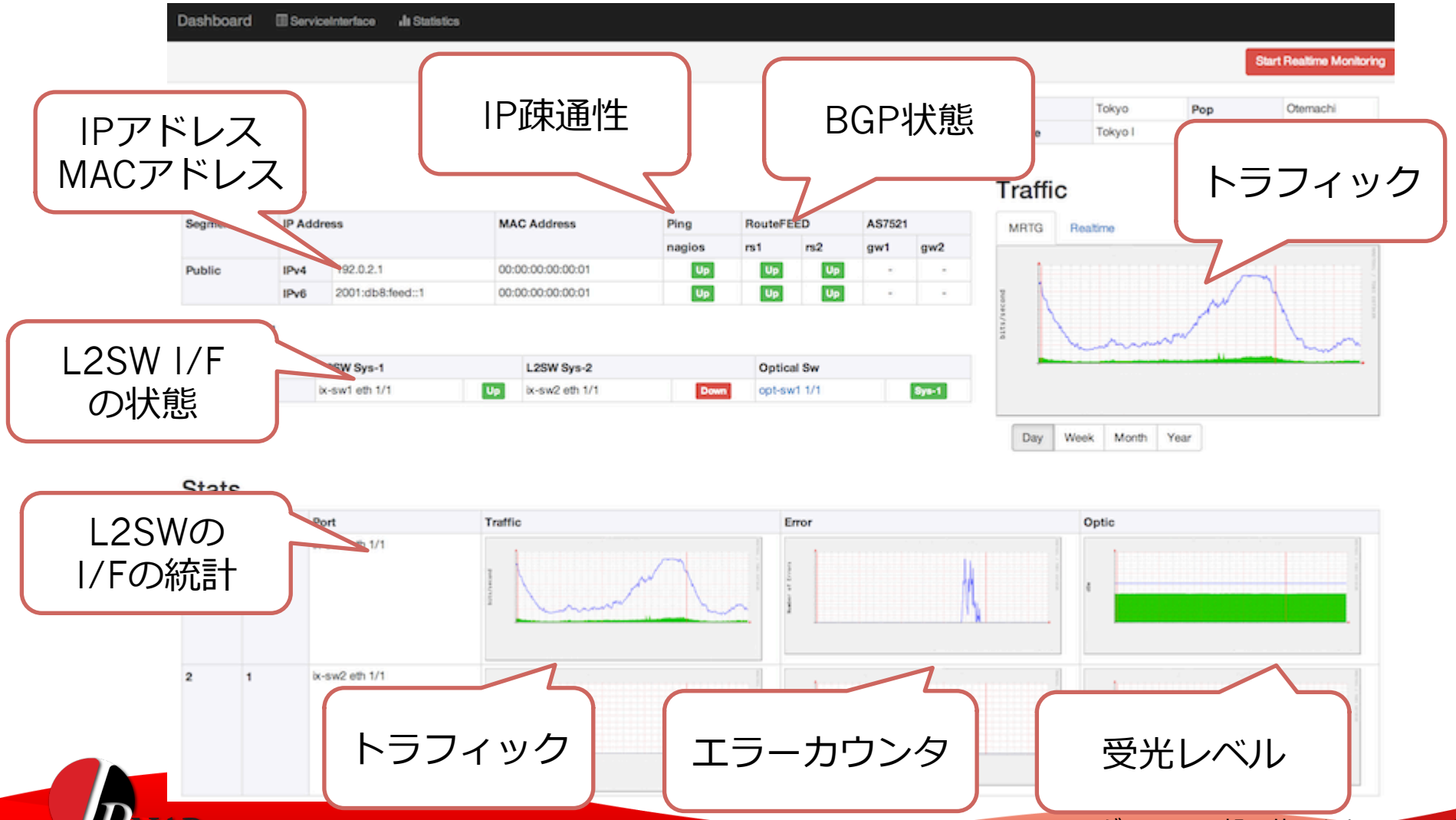
- IXの統計情報や顧客の接続状況を１画面で全て把握できる



※グラフの一部は伏せられています

運用高度化 — JPNAP Dashboard

- IXの統計情報や顧客の接続状況を１画面で全て把握できる



※グラフの一部は伏せられています

運用高度化 — JPNAP Dashboard

- 色々な統計情報を並べて表示することができれば、運用や障害対応に必要な相関性を簡単に見つけることができる

ユーザ収容ポートでエラーが出た時の例

トラフィック　　エラーカウンタ

帯域幅上限

※グラフの一部は伏せられています

運用高度化 — JPNAP Dashboard

- 色々な統計情報を並べて表示することができれば、運用や障害対応に必要な相関性を簡単に見つけることができる

顧客の回線が副系に切り替わった例

トラフィック　エラーカウンタ　受光レベル

主系

副系

※グラフの一部は伏せられています

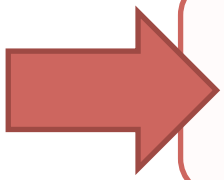
ネットワーク機器の自動制御は？

- JPNAPの運用の中ではまだ人間が対応したい領域
  - 時間に依らずにどうしても行わなくてはいけない、一部の定形作業は自動化している
  - でもある程度の定形作業は自動でやりたい

自動	BGPルータの経路フィルタの設定 特定のアラート対応
手動	L2SWの顧客収容ポートの初期設定 顧客接続時の開通作業 バックボーン増速作業

あえて全自動化しない理由

- 運用者の「目」を信用している
  - 直感的な違和感を感じ取る能力を信じている
  - 自分たちの目でも確認することで品質を担保している
- 例外もあるし人為ミスもあるしバグもある
  - 例外が発生した時や意図しない変更があったときに誤ったコンフィグを勝手にデプロイされると困る
  - 人間とプログラムのお互いが得意分野で支えあうことで品質を高めたい



チームの状況や方針に合わせて
どこまで自動処理するかを見極める

変わった文化

1. コンフィグもツールもgit管理してgitlabにpush
   - すべてを差分管理する
   - マスターを明確にする
2. ツールのドキュメントはWikiじゃなくてREADME.md
   - ツールとドキュメントを同時に管理する
3. コンフィグやツールの変更者と変更理由を差分に残す
   - 共通アカウントで変更作業をしない
4. デプロイ前にバリデーション

まとめ

- 自分たちの運用を省力化・高度化するために、運用者自身がソフトウェアを書いて幸せになろう
- 自分たちの運用スタイルやチーム状況に適した自動化・高度化を目指そう
- ソフトウェアに強いチームを作ろう！