【DevOps事例】
ツール連携とAPI開発の勧め

2017/10/27

s.takatsu

はじめに

- 自己紹介

  システム開発側の人間です。

  PHP、Javaとかで、Webシステムのバックエンドをメインでやってきました。とはいえ、がっつり技術の人間ではないです。

  正直ネットワーク全然詳しくないです。

現状

- ネットワーク部隊 唯一の開発要員

  歴史ある社内システムの保守管理を担当

  - 前任のSSE（SuperSE)が残したシステム群を四苦八苦しながら廻してます。

    ⇒　言語はRuby、Perl、Go言語だったり、Dockerで構築されてたり、フレームワークもバラバラ。。

    社内システム・ツールが群雄割拠の状態。

ミッション

「ネットワーク運用の自動化よろしく！ヒントはDevOps！」

エッ(゜Д゜;)！？

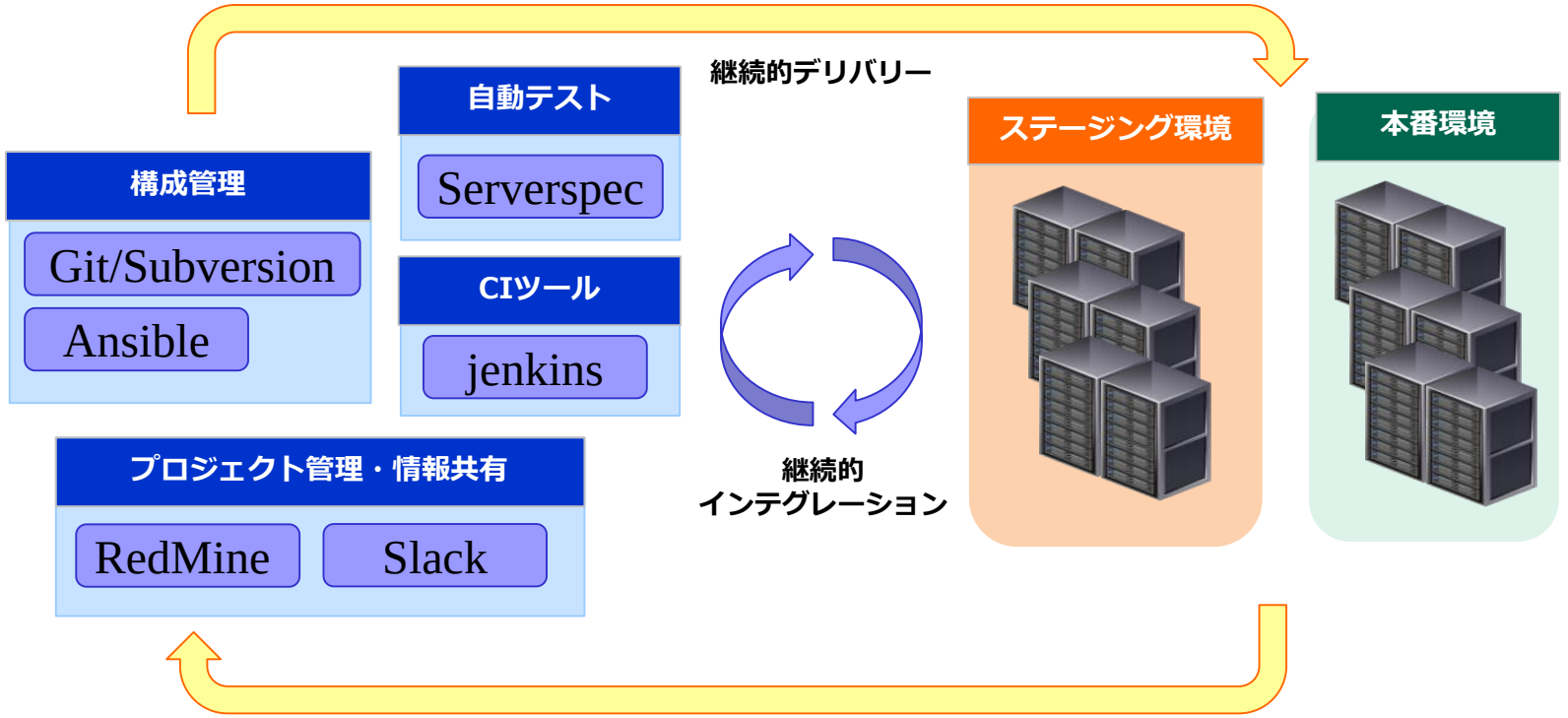
DevOpsとは？

- DevOpsとは、開発(Development)と運用(Operation)を組み合わせたもの
- 開発と運用が密に連携して、運用側が要求する新機能や改修などの開発を行なう手法や概念を指す
- 大規模な開発を長期間かけて行なってリリースするのではなく、小規模な開発とリリースを繰り返す

（日立ソリューションズ IT用語辞典より）

DevOpsとは？

良くあるDevOpsのイメージ図（例）



DevOpsとは？

- DevOpsを支えるツール群
  - バージョン管理システム（Git、subversion、mercurial）
  - バグトラッキングシステム（Redmine、trac、JIRA）
  - 構成管理ツール（Chef、Puppet、Ansible ）
  - 継続的インテグレーション（Jenkins、CirleCI、TravisCI）
  - テスト自動化（ Selenium、Appium）
  - 仮想化（VMware、Docker 、 VirtualBox ）

DevOpsとは？

- 業務ノウハウの蓄積と、環境（開発手法、ツール）が整った。開発・運用チームが協力し、高速開発・リリースが可能に。

上記を実現するための手段が「DevOps」

巷に溢れる情報は自社サービスの提供を前提としたものが殆ど。

ネットワーク運用の自動化に落とし込むには工夫が必要。

DevOpsとは？

前段で紹介したツール群を導入することで、ネットワークの運用でもメリットがあることは間違いない（と思う）。が、自動化までの道のりは険しい。

「ただでさえ忙しいし、社内システムが乱立してるのに、さらに新しいこと覚えるの？」といった心の声。なるべく楽して効果を得たい。

DevOpsとは？

ツールと組織文化、両面からの**カイゼン**

まずは既存システムから**カイゼン**

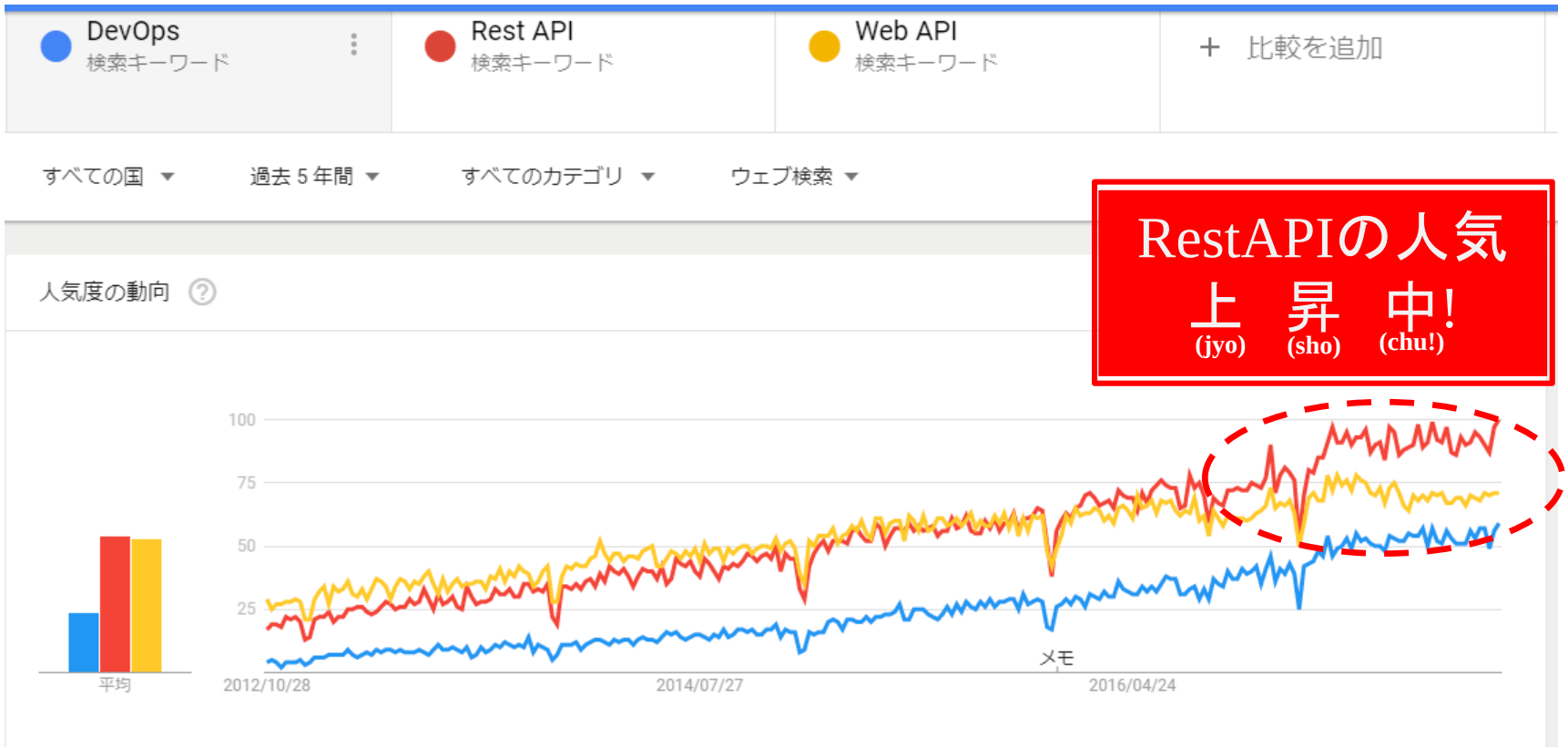
ちょっとした工夫で日々の作業が楽になる、ミスが少なくなるという体験に基づいた意識付け

APIのススメ

（Rest）APIをおススメ！

- 最近のシステム、ツールであれば予めRest（っぽい）APIが用意されていることが多いが、現場では意外と認知されてない。
- 当然、活用されていなければ、アイデアも出てこない。。

おさらい Rest APIとは？



おさらい Rest APIとは？

REST(REpresentational **S**tate **T**ransfer)

HTTPの技術を活用した設計方法

- アドレス指定可能なURIで公開されていること
- HTTPメソッドの利用が統一されていること
- ステートレスであること
  （セッションなどの状態管理は行わない）
- 処理結果がHTTPステータスコードで通知されること

おさらい Rest APIとは？

- **アドレス指定可能なURIで公開されていること**

有名どころはドキュメントが公開されているので参考に。

GitHub Developer　https://developer.github.com/v3/

取得（GET）例：https://api.github.com/repositories

登録（POST）例：https://api.github.com/user/following/:username

Twitter　https://developer.twitter.com/en/docs

取得（GET）例：https://api.twitter.com/1.1/statuses/retweets_of_me

登録（POST）例：https://api.twitter.com/1.1/statuses/update

おさらい Rest APIとは？

- HTTPメソッドの利用が統一されていること

処理	HTTPメソッド	CRUD
登録	POST	CREATE
取得	GET	READ
更新	PUT	UPDATE
削除	DELETE	DELETE

例えば下記URIにGETでアクセスすれば、メールの取得。
POSTでアクセスすればメールの送信。

https://example.com/user/emails

おさらい Rest APIとは？

・ステートレスであること

同じ入力に対する出力は常に同じになる。

同じURIであれば、出力結果は同一ということ。

※認証情報（セッション）はどうなんだという話題は、
良く理解できていないのでスルー

おさらい Rest APIとは？

- 処理結果がHTTPステータスコードで通知されること

取得（GET）

成功の場合は、200 OK、失敗の場合は内容に応じたステータスコード（404 Not Found、500 INTERNAL SERVER ERROR等)を返す。

登録（POST）

成功の場合は、201 CREATED、失敗の場合は内容に応じたステータスコードを返す。(GETと同様）

まとめ Rest APIとは？

あくまで設計思想。

利用者からすれば、RestAPIを実装しているシステム（ツール）が持つリソースをお手軽に利用（登録、取得、更新、削除）できる。

（個人の感想ですが）おまけ機能なんてとんでもない、むしろ本体。活用しなきゃもったいない！

事例紹介

弊社で稼働している社内システム・ツールをAPIを利用して連携した例を紹介します。

※APIの利用前提というか、結果としてAPIを使わざるを得ませんでした。使ってるうちにRestAPIやらマイクロサービスアーキテクチャに行きついた、というのが本音です。

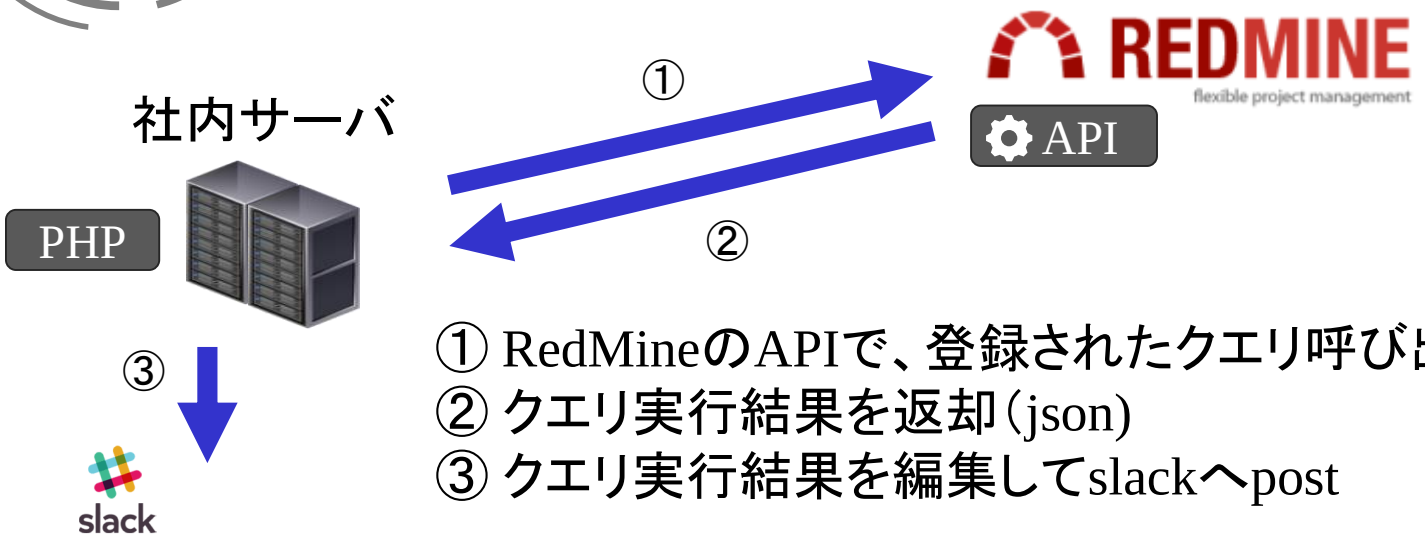
事例1：RedMineとSlackの連携

弊社ではチケット管理にRedmineを使用しています。運用チームでは毎日定時（リマインドはSlack）にスタンドアップミーティングを実施し、チケット進捗状況などを確認しています。

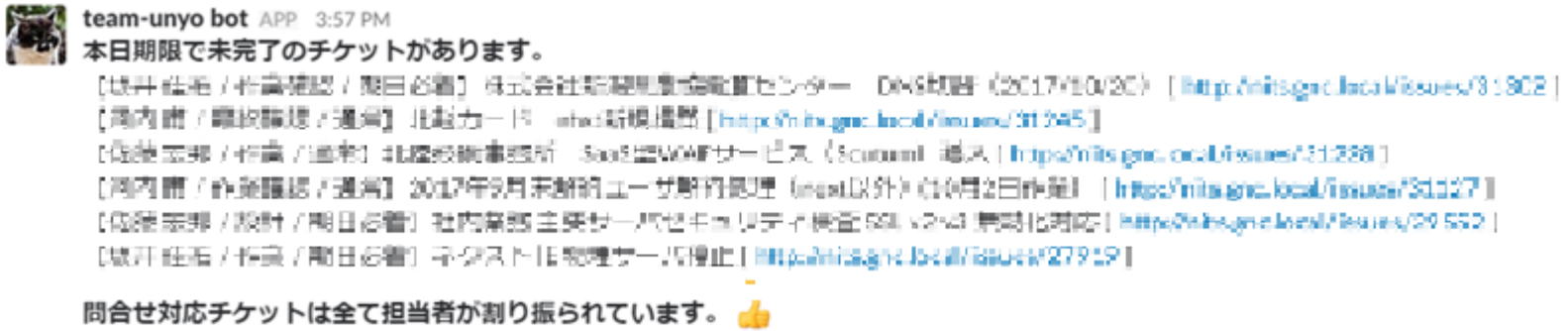
※最新のチケット状況の把握が難しい。

　自己申告なので、報告漏れがあったり。

事例1：RedMineとSlackの連携



① RedMineのAPIで、登録されたクエリ呼び出し
② クエリ実行結果を返却（json)
③ クエリ実行結果を編集してslackへpost



事例1：参考

- RedMineからの情報取得

```
$url = 'http://REDMINE_SERVER/projects/PROJECT_KEY/issues.json
?key=API_ACCESS_KEY&query_id=QUERY_ID’;
$ret = exec_curl($url);
```

- Slackへのポスト

```
$webhook_url =SLACK_URL;
$options = array(
    'http' => array(
      'method' => 'POST',
      'header' => 'Content-Type: application/json',
      'content' => json_encode($message)));
  $response = file_get_contents($webhook_url, false, stream_context_create($options));
```

事例1：余談

REST APIのアプリケーション例としてスマホ向けアプリが挙げられていました。

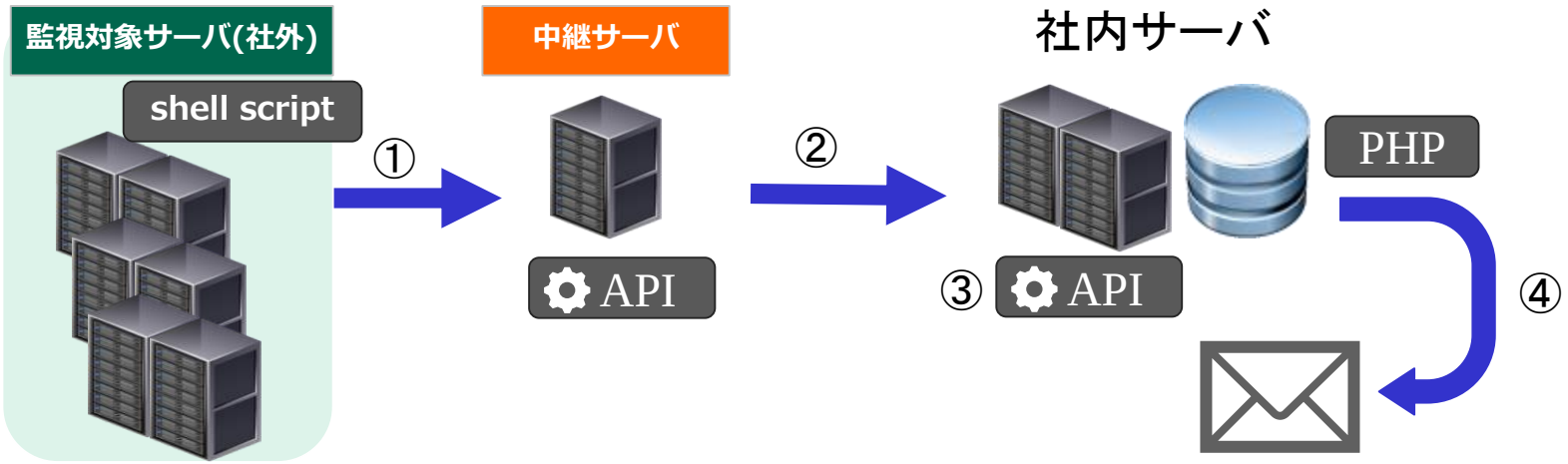
iPad / iPhone / Android対応のRedmineクライアントアプリ「RedminePM」
http://blog.redmine.jp/articles/redminepm/
(Redmine.JP Blog)

事例2：ウィルスパターンファイル監視

- 知らぬ間にロックファイルが作成され、パターンファイルの更新が行われていなかった！！
- ウィルスチェック自体は正常に行われていたので、気付くのが遅れてしまった。。

⇒　属人的な監視は厳しい。。

事例2：ウィルスパターンファイル監視



① 定期的にパターンファイル更新状況を取得、中継サーバAPIを叩く
② 中継サーバAPIでチェック後に、社内サーバAPIへリレー
③ 社内サーバのAPIで受取り、更新状況をDBに格納
④ 異常が発生していたら即通知メール、正常であれば即通知はしないが、
定期的に正常稼働通知メールを送信

事例2：ウィルスパターンファイル監視

- 監視対象サーバ側から定期的に情報を送ってくる仕組み

  ⇒ 片道一方通行なので、シンプル。

  情報が取れない≒何かしらの異常

- 正常稼働の場合、複数サーバの状況を一つのメールにまとめて通知

  ⇒ 確認が楽。

  届かなければ何か起きたと気づく

事例３：監視ツールの動作確認、再起動

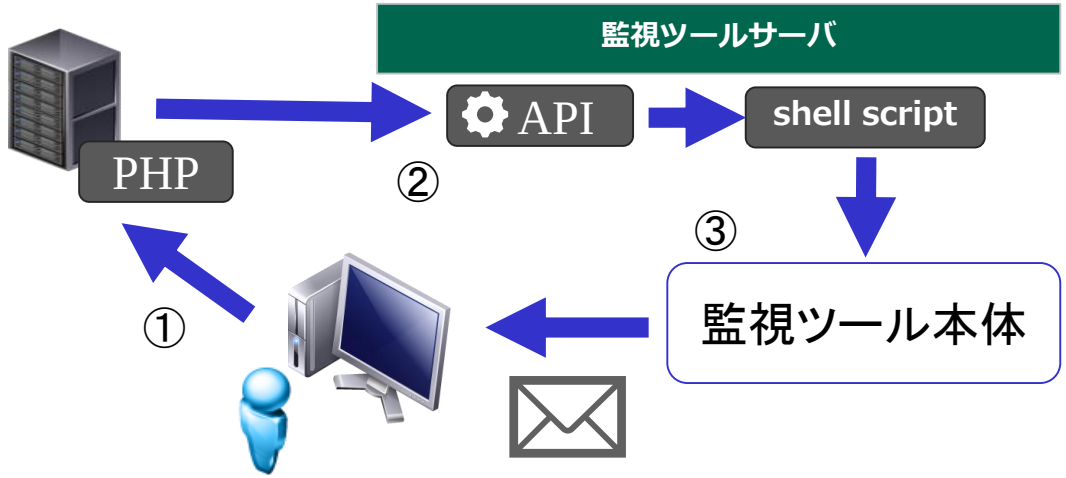
- 管理画面から設定を変更するごとに、動作確認メールを送信する運用ルール。
- トラブル発生の場合、再起動の必要あり。サーバにログイン出来ないユーザが操作していた場合、運用担当者へ依頼して再起動。

依頼する方、される方ともにムダな時間が割かれていた。

そこで、ブラウザ上ボタン一つで確認メール送信、再起動を実行する画面を作成しました。

事例３：監視ツールの動作確認、再起動

事例3：監視ツールの動作確認、再起動



① ブラウザ上でボタンポチッ（メール送信 or 再起動）
② 監視サーバ上に仕込んでおいたスクリプトをAPI経由でキック
③ 監視ツール本体からメール送信、または再起動
④ 実行ユーザはメール確認、または画面上で再起動実行結果を確認

事例4：音声通報システム連携

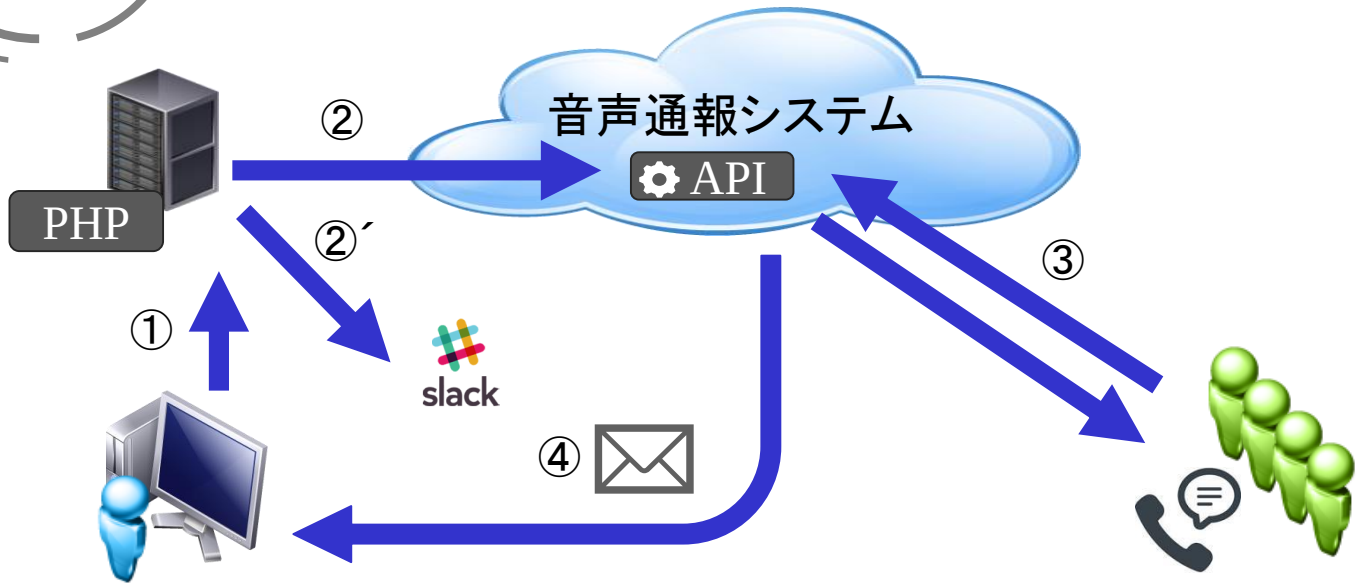
障害発生時の連絡を円滑にするために、音声通報システムを導入しました。

が、UIが運用に合わなかったり、シフトの管理が面倒でした。

APIが実装されていたため、通報する担当者向けのUIを内製し、操作をシンプルに。

事例4：音声通報システム連携

事例4:音声通報システム連携



① ブラウザ上でシフトを確認し、同画面の通報ボタン押下
② ブラウザ上の順番で、音声通報実行API呼び出し
②´ 通報中であることSlackへ周知(Slack Bot)
③ 音声通報システムからの呼び出し、応答結果取得
④ 応答内容を通報者へメールで通知
※シフト表はAPI経由で自動更新

事例まとめ

- 既存システム・ツールの復権
  ⇒ レガシーな資産を活用出来た！連携が出来ると分かれば、現場からアイデアが出てくる（といいなぁ）
- 製造コストの削減
  ⇒ APIが実装されていれば、基本的なデータの取得、更新は手間いらず。ビジネスロジックに集中出来る。お試しで製造も気軽に。
- 煩雑な画面操作からの解放
  ⇒ やりたいことは大体決まってる。手順をAPIでまとめて、ボタン一つで実行。誰がやっても結果は同じ。皆が幸せに。

今後の展開1

Ansibleを利用した構成管理にも取り組んでいます。少しづつですが、Ansibleを利用しようと自発的に活動する人も。。

そんな中、「Ansibleでルータの設定入れたいんだけどモジュール何使えばよいの？」といった質問を受けました。

今後の展開1

- 確かに構成管理ツール（Ansible等）でもネットワーク機器に対応するモジュールが増えてきましたが、まだ不足気味。
- 過去 ENOG37 で海老澤さんの発表にもありましたが、ネットワーク機器もRest（風）APIに対応しています。（未だメーカ、機器ごとに癖がある？）

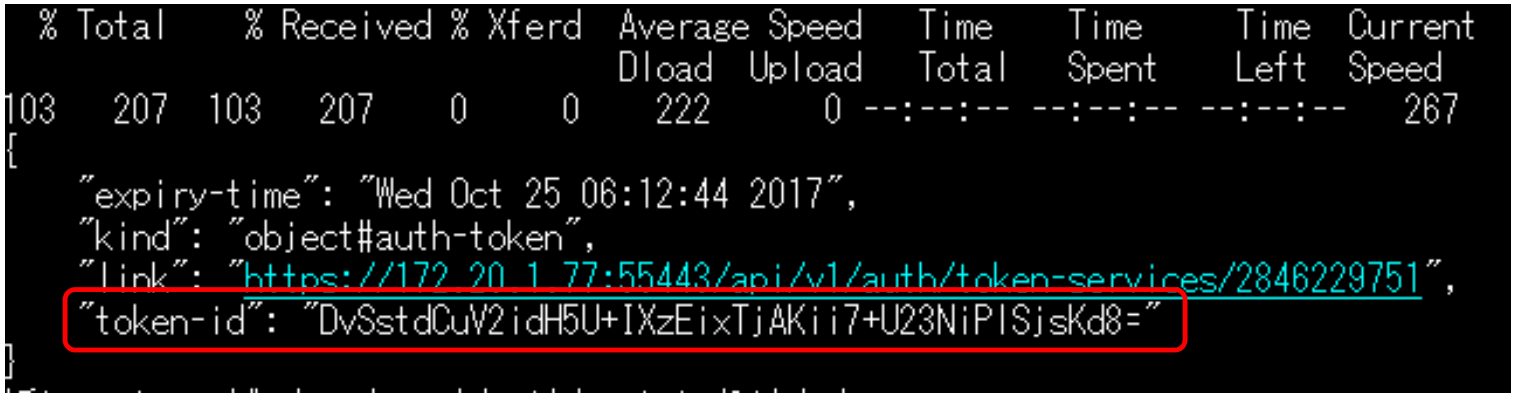
相互に補う形で、ネットワーク機器の自動設定が可能では？

参考：ネットワーク機器でのAPI利用

仮想ルータ（Cisco CSR 1000V ）でお試し

- セッションを開始

```
$ curl -k -X POST https://172.20.1.77:55443/api/v1/auth/token-services -H
"Accept:application/json" -u "user:password" | python -mjson.tool
```



参考：ネットワーク機器でのAPI利用

- トークンを使用してAPIにアクセス

```
$ curl -k -X GET https://172.20.1.77:55443/api/v1/interfaces -H "Accept:application/json" -H 'X-auth-token:DvSstdCuV2idH5U+IXzEixTjAKii7+U23NiPlSjsKd8=' -u "user:password" | python -mjson.tool
```

例：Interfaceリストの取得

今後の展開2

RestAPIを使う側から作る側へ

- 自作したAPIはRestとは言えない。。自社向けのAPIフレームワーク作りたい。
- Dockerと相性良さそう。苦手意識あるので克服しなければ。。
- マイクロサービスアーキテクチャの考え方に行き着く。点と点が繋がってきた感じ。何か作ってみたい。

最後に

ご清聴ありがとうございました。

※懇親会に出席しますので、ぜひお声がけください。