

**API Meetup Tokyo #2**

# **クラウドサービスの Web APIとそのユースケース**

**Naoya Ito**

KAIZEN platform Inc.

8/29 2014



**KAIZEN PLATFORM**

# お話しする内容

- [過去] 私が見てきた Web API の変遷  
– 帰着のひとつ … \* as a Service
- [現在] それを使うユースケース
- [未来] Webシステムのアーキテクチャトレンドと Programmable Web



# 私が見てきた Web API の変遷



# Why: なんでもそんな話するの？

- 昨今の \* as a Service は10年前に描いていた理想的なユースケースの一つ
  - 「あー、そうそう、これがしたかったんだ」
  - それを支える Web API
- ここに至るまで、どういう変遷があったか
  - my point of view



# 注: "私が見てきた"

- なので、B2C Web業界に話が偏ってます。  
ご了承ください

# はじめての API



日経インターネットテクノロジー  
日経Windowsプロ

この解像度しか残ってないっぽかったw  
2002年の雑誌

## 日経BPパソコンベストムック Webサービス完全ガイド

### ■ 内容紹介

IBM、マイクロソフト、サン・マイクロシステムズ。登場してからまだ日の浅い分野ですと、いよいよ現実味を帯びてきました。ように、インターネット/情報システム分野

本ムックは、SOAPやUDDIといった新技術に必要な実用情報を多角的に取り上げ、国内で利用可能な製品、実際の構築事例からECへの適用方法までを示す解説一冊。この専門用語については「Webサービスを理解

■日経BP社のIT分野の最新情報は[こちら](#)



# 当時の状況

- 上司「伊藤君！XML Webサービスって知ってるか？」
  - 他にも「伊藤君J2EEって」「伊藤君ブログって」などのパターンあり

※おかげでブログ開発担当になれたので、よい上司でした

# XML Webサービス

- 業務ロジックを公開サービスとして、それらのサービスを組み合わせて一つの業務を実現する…
  - 例: 旅行予約 = ホテル + 航空会社 + 個人決済
  - まだ "SOA" という言葉は聞かなかったように記憶している
- SOAP/UDDI/WSDL …





# エンタープライズ

## 日経BPパソコンベストムック Webサービス完全ガイド

### ■ 内容紹介

IBM, マイクロソフト, サン・マイクロシステムズなどが全社をあげて取り組む「Webサービス」。登場してからまだ日の浅い分野ですが、2002年に入って対応製品や構築事例が登場してきたことで、いよいよ現実味を帯びてきました。「Webの誕生よりもインパクトは大きい」とする声もあるように、インターネット/情報システム分野で今後のメイン・テーマとなることは間違いありません。

本ムックは、SOAPやUDDIといった新技術の教科書的な解説ではなく、Webサービスに取り組むために必要な実用情報を多角的に取り上げたところが特徴です。具体的には、各社のねらいと最新の動き、国内で利用可能な製品、実際の構築事例、Javaや.NETを使った実装方法、基本アーキテクチャからECへの適用方法までを示す解説一冊。これらを1冊に盛り込んでいます。調べるのが面倒な最新の専門用語については「Webサービスを理解するためのキーワード50」としてまとめました。

■日経BP社のIT分野の最新情報は[こちら](#)



# 当時困ったこと

- XML Webサービスがバズってるらしいけど、実例が見当たらない
  - 企業内や、業務システムのバックエンドでは動いていたかもしれないが一般消費者からはなかなかそれが見えなかった

当時の自分「SOAPとか  
幻の技術なんでは・・・？」



# 2003年頃

- 上司「伊藤君！ブログって知ってるか？」

naoya 「」



## Blog テクノロジーとWeb サービス

Naoya Ito  
naoya@naoya.dyndns.org  
http://naoya.dyndns.org/~naoya/mt/

## アジェンダ

- Blog とは?
- Blog ツール
- Blog テクノロジ: RSS
  - RSSアプリケーション
  - PerlでRSS
  - RssRolling
- Blog テクノロジ: Weblogs.Com Ping
  - XML-RPC Ping Interface
  - Perl による Ping クライアント
  - Perl による Ping サーバ
  - デモ: Ping Server + IRC
- Amazon Web サービス
  - Perl で AWS
  - Blog と AWS: mt.cgi

2014/08/27

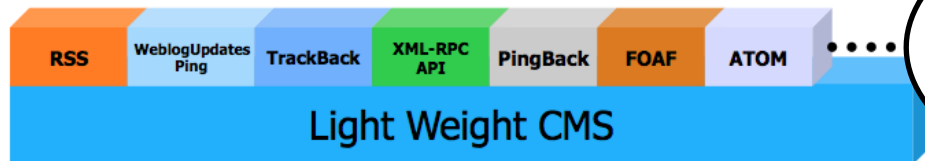
Shibuya.pm Tech Talk #4

2

## Blogツール(2)

### 最近のBlogツールの傾向

サイトのコンテンツやデザインを管理する**軽量なCMS**を、**XML 応用技術**や**ウェブサービス**などでトッピング。



「**BlogはXMLだ!**」...これはちょっと大袈裟

めっちゃ使われてるやん  
(2013/10 の Shibuya.pm での発表資料より)

2014/08/27

Shibuya.pm Tech Talk #4

6

# Blogの背後にあった Web API

- 「軽量なWebサービス」
  - RSS、Atom、weblogUpdates.ping
- ひとときわ目を引いたもの: XML-RPC
  - SOAP の初期のドラフトからインスパイア
  - SOAP のエンベロープやヘッダなどを取っ払った仕様

標準化 vs デファクト  
"実践的な" Webサービス



# 結果、"Web業界"で起こったこと

- Web 2.0 (2004、2005 あたり)
  - API を公開するとかっこいい? (中二病)
- さて、業務ロジックを公開しようぜといっても
  - Web業界「業務ロジック」って何だ?
    - 多くは背後にそれほど{大きい, 重要}な業務を持っていなかった
    - ∴ ロジックではなくデータを公開するという方向に流れた
      - Web業界にはメディア企業が多かったし
      - Blog/ニュースの RSS フィード、Amazon Product Advertisement API 等わかりやすい先行事例

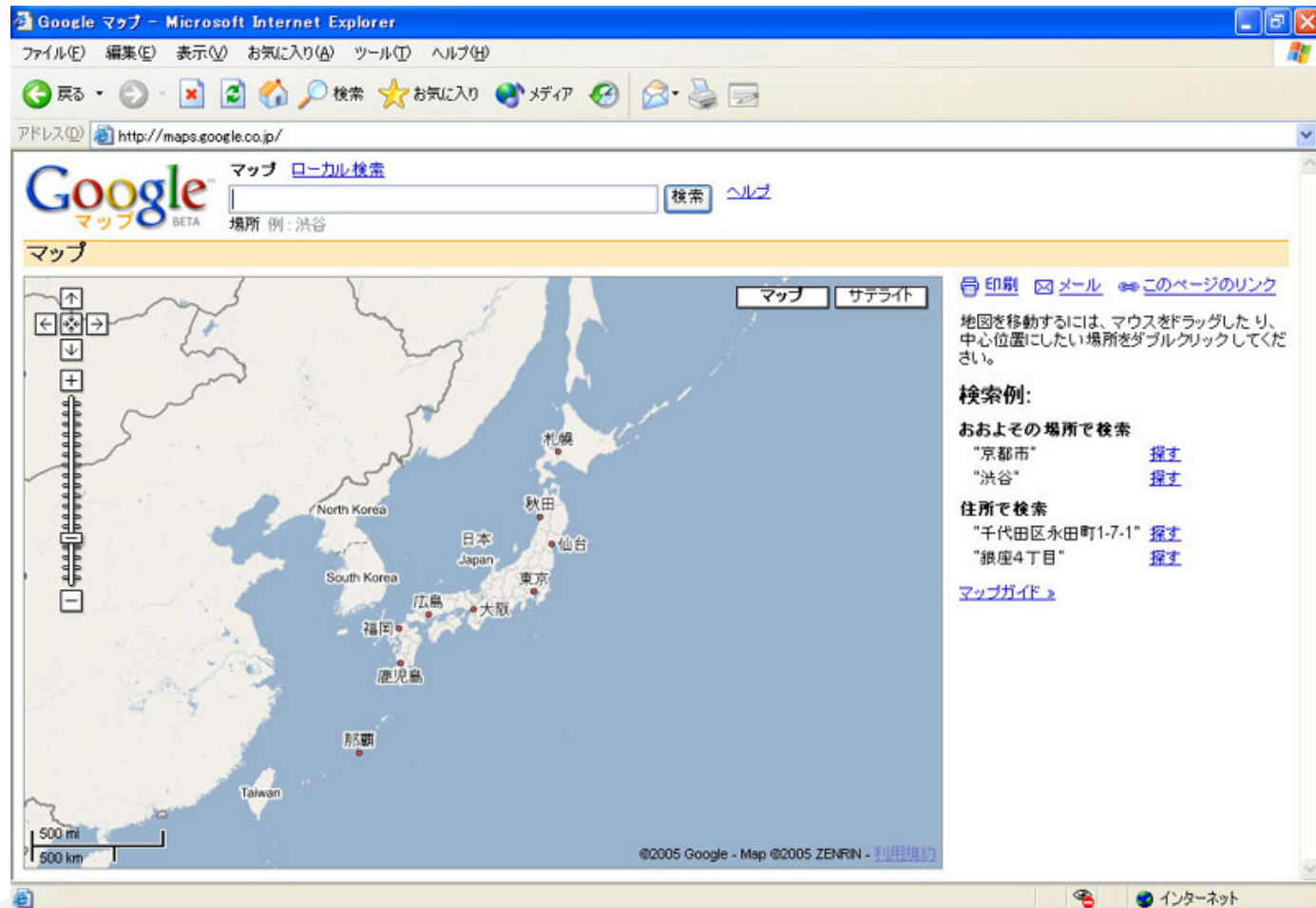


# REST

- 前回の yohei さんの話
- 当時
  - 実装厨 「SOAP とか XML-RPC とか言うけど、結局 XML を HTTP GET でよくね?」
  - アーキ厨 「そもそもそれは Representational state transfer と言ってだな」

少なくとも自分の周囲では  
REST は SOAP などとの  
対比の文脈で語られ始めた

# そうこうしてるうちに Ajax



2005年にBB Watchに寄稿した拙記事より  
<http://bb.watch.impress.co.jp/cda/alphageek/10989.html>



Share Email Embed Like Save

- Ajaxの発見→Google Maps, Livedoor Reader→Single Page Application
- コードがクライアントサイドに移った
  - サーバはデータを提供する
  - ポータブルなクライアントがそれを処理する
- ブラウザが表示装置ではなくなった

22

22 / 39

Share Email Embed Like Save

データとコードの分離が進んだことで、クライアントのバリエーションが増えた。  
または  
クライアントのバリエーションが増えたことで、データとコードの分離が進んだ。

↓  
全てがAPIに

23 / 39

Web API のこれまでとこれから (API meetup #1)  
<http://www.slideshare.net/yohei/webapi-36871915>

# Web API 化が進んだ、が

- 「(XML) Webサービス」 → 「Web API」
  - いつの間にかラベルも変わってた
  - 2006 ~ 2008 くらい?
- 「マッシュアップ」
  - でも、多くの例は "○○と△△を一緒に表示してみました"
    - 全体としてデータの公開に目がいったから?
    - 「業務を構築する」というよりは「データを組み合わせた」だけ



# サービスの組み合わせで 業務を構築する？

- 「XML Web サービス」の文脈で語られていた「Web サービスとWebサービスを組み合わせで業務を紡ぐ」という考え方は間違っていない
- Web 2.0 の Web API はなかなかそこまで到達できてなかった
  - Twitter や Facebook など Web 2.0 サービスが担ったのはコンテンツサービスであり「業務の部品」とは違ったし

# ここで突然の

SOA で作った社内サービス、公開したよ



**amazon**  
**web services**<sup>TM</sup>

ref: Stevey's Google Platforms Rant  
<https://plus.google.com/+RipRowan/posts/eVeouesvaVX>



KAIZEN PLATFORM

# サービスとサービスを 組み合わせて

- ストレージの Amazon S3 と〇〇を組み合わせて
  - それらがプログラマブル (API で) に操作できる

「Webアプリケーション」の API ではないが、  
紛れもなく Web API によるサービスであった



# AWS と Web API

- AWS の各コンポーネントは「後付けでAPI化されたサービス」ではなく「API でサービスする」ことを前提にされたもの
  - Programmable 且つ「需要」が先にあった
  - そうそう "API" ってこれのこと

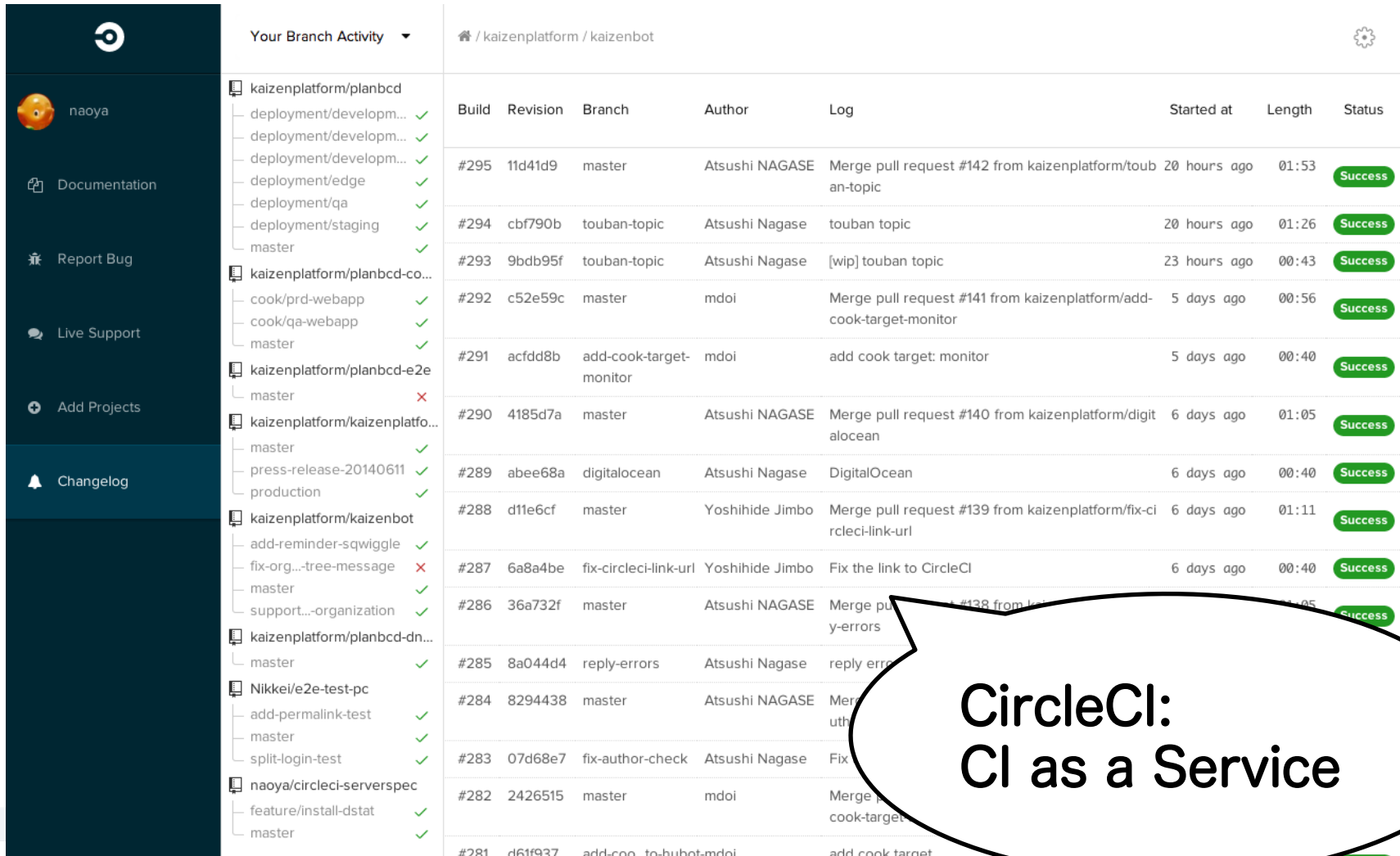
Webサイトに後から部分的にAPIを生やす、のではなく文字通り "Application Programming Interface"

# そしてクラウドの時代へ・・・



※途中、JSON とかモバイルとかソーシャルゲーム (OpenSocial) とかいろいろあるけど割愛

# \* as a Service



The screenshot displays the CircleCI interface with a sidebar on the left containing navigation options: naoya, Documentation, Report Bug, Live Support, Add Projects, and Changelog. The main area shows 'Your Branch Activity' for the repository 'kaizenplatform / kaizenbot'. A table lists recent builds with columns for Build ID, Revision, Branch, Author, Log, Started at, Length, and Status. Most builds are marked as 'Success'.

Build	Revision	Branch	Author	Log	Started at	Length	Status
#295	11d41d9	master	Atsushi NAGASE	Merge pull request #142 from kaizenplatform/touban-topic	20 hours ago	01:53	Success
#294	cbf790b	touban-topic	Atsushi Nagase	touban topic	20 hours ago	01:26	Success
#293	9bdb95f	touban-topic	Atsushi Nagase	[wip] touban topic	23 hours ago	00:43	Success
#292	c52e59c	master	mdoi	Merge pull request #141 from kaizenplatform/add-cook-target-monitor	5 days ago	00:56	Success
#291	acfdd8b	add-cook-target-monitor	mdoi	add cook target: monitor	5 days ago	00:40	Success
#290	4185d7a	master	Atsushi NAGASE	Merge pull request #140 from kaizenplatform/digitalocean	6 days ago	01:05	Success
#289	abee68a	digitalocean	Atsushi Nagase	DigitalOcean	6 days ago	00:40	Success
#288	d11e6cf	master	Yoshihide Jimbo	Merge pull request #139 from kaizenplatform/fix-circleci-link-url	6 days ago	01:11	Success
#287	6a8a4be	fix-circleci-link-url	Yoshihide Jimbo	Fix the link to CircleCI	6 days ago	00:40	Success
#286	36a732f	master	Atsushi NAGASE	Merge pull request #138 from kaizenplatform/fix-reply-errors	6 days ago	01:05	Success
#285	8a044d4	reply-errors	Atsushi Nagase	reply errors	6 days ago	00:40	Success
#284	8294438	master	Atsushi NAGASE	Merge pull request #137 from kaizenplatform/uth	6 days ago	01:05	Success
#283	07d68e7	fix-author-check	Atsushi Nagase	Fix author check	6 days ago	00:40	Success
#282	2426515	master	mdoi	Merge pull request #136 from kaizenplatform/add-cook-target-monitor	6 days ago	01:05	Success
#281	d61f937	add-cook-target-monitor	mdoi	add cook target	6 days ago	00:40	Success

CircleCI:  
CI as a Service



# The complete mobile app platform

Focus on creating unique & engaging apps on any platform. We take care of everything else your app needs, from the core of your app to analytics and push notifications.



Power your app using Parse

Try it for free

**Parse.com:  
mobile Backend as  
a Service**

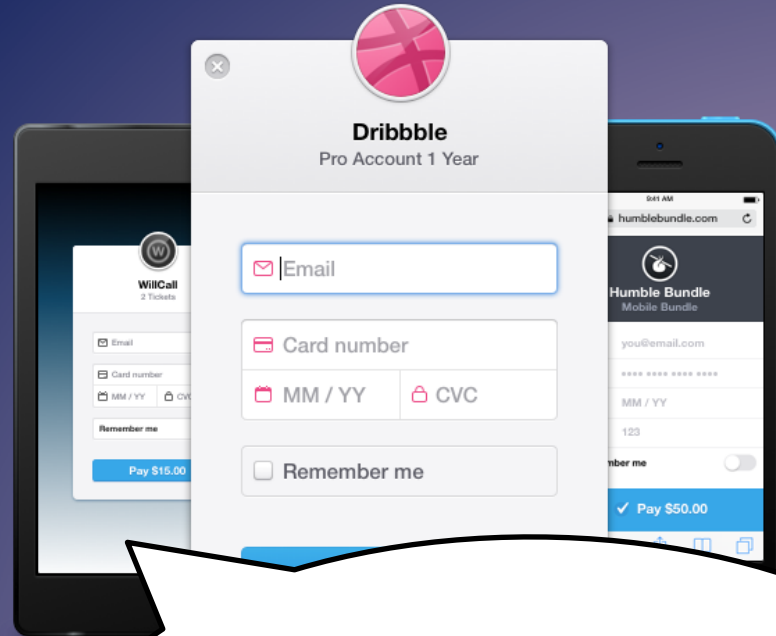


## Checkout

A beautiful, optimized, cross-device payment form, with support for single click payments.

[Explore Checkout](#)

[Documentation](#)



Stripe:  
Payment Platform  
as a Service

★ Stripe + Alipay. Integrate Checkout and instantly accept payments from

COMPOSE QUERY

Query History

Job History

planbcd ▾

- ▶ gi
- ▶ logapp
- ▶ webapp

---

- ▶ publicdata:samples

## Table Details: nginx\_access

Schema

Details

Query Table

### Schema

time	INTEGER	NULLABLE	Describe this field...
host	STRING	NULLABLE	Describe this field...
method	STRING	NULLABLE	Describe this field...
uri	STRING	NULLABLE	Describe this field...
status	INTEGER	NULLABLE	Describe this field...
size	INTEGER	NULLABLE	Describe this field...
ua	STRING	NULLABLE	Describe this field...
referer	STRING	NULLABLE	Describe this field...
reqtime	STRING	NULLABLE	Describe this field...
vhost	STRING	NULLABLE	Describe this field...
environment	STRING	NULLABLE	Describe this field...

Google BigQuery:  
BigData as a Service

# \* as a Service と Web API

- 昨今の \* as a Service の多くは Web (HTTP) を介してサービスを提供する
- プログラムのための部品であるという性格上 API を有する



# \* as a Service を駆使して 業務を構築する (イマココ)

- 例: Parse.com でモバイルバックエンドを賄い BigQuery でログ解析して Stripe で課金する
- クラウドサービスの台頭という形で、かつて「XML Web サービス」が目指していた世界が現実になった
  - SOAP/UDDI/WSDL的な手段ではなかったが、それを繋いだのが "Web API"
  - (※REST、とも言ってない。念のため。)



- 本質でないところは他人にまかせる、が当然に
- そのプラットフォームは必ずAPIを持つ

- APIをRESTfulにするかどうかは要件しだい

- 要件に合ったアーキテクチャを

- 例：WebRTCはRESTじゃない

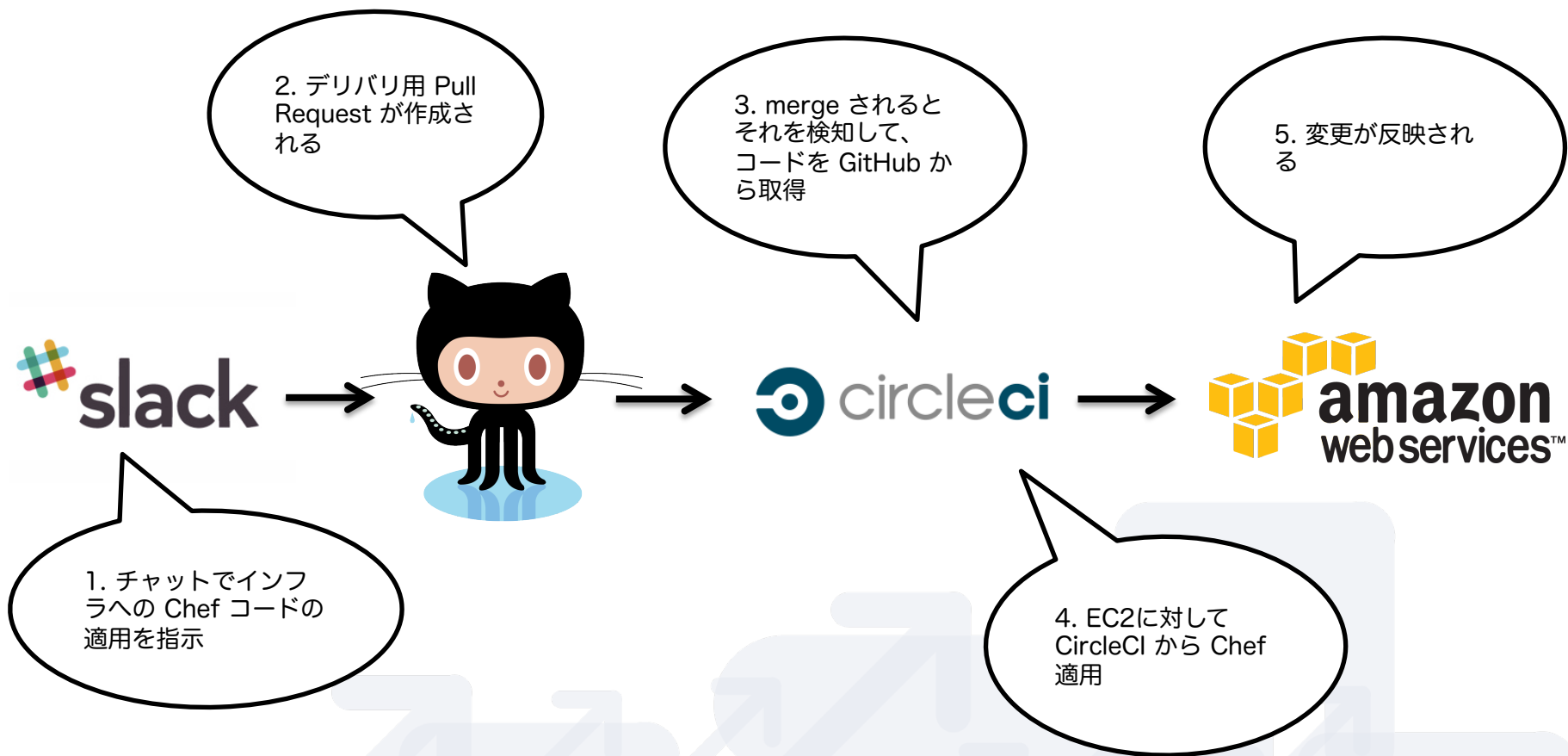
- APIの利用と開発は基本スキルに

まさにこの話

# どんな風に使っているか (ユースケース)



# インフラ構成の継続的デリバリー





# 1. チャットで指示



**mdoi** 12:22 PM

hubot cook run qa-proxy



**hubot** 12:22 PM ★

Creating Pull Request: master -> cook/qa-proxy

cook/qa-proxy

continue manual merge <https://github.com/kaizenplatform/planbcd-cookbooks/pull/659> to cook

check progress <https://circleci.com/gh/kaizenplatform/planbcd-cookbooks/tree/cook%2Fqa-proxy>

just close pull request to cancel



**mdoi** 12:31 PM

hubot cook run qa-editor



**hubot** 12:31 PM

Creating Pull Request: master -> cook/qa-editor

continue manual merge <https://github.com/kaizenplatform/planbcd-cookbooks/pull/661> to cook

check progress <https://circleci.com/gh/kaizenplatform/planbcd-cookbooks/tree/cook%2Fqa-editor>

just close pull request to cancel

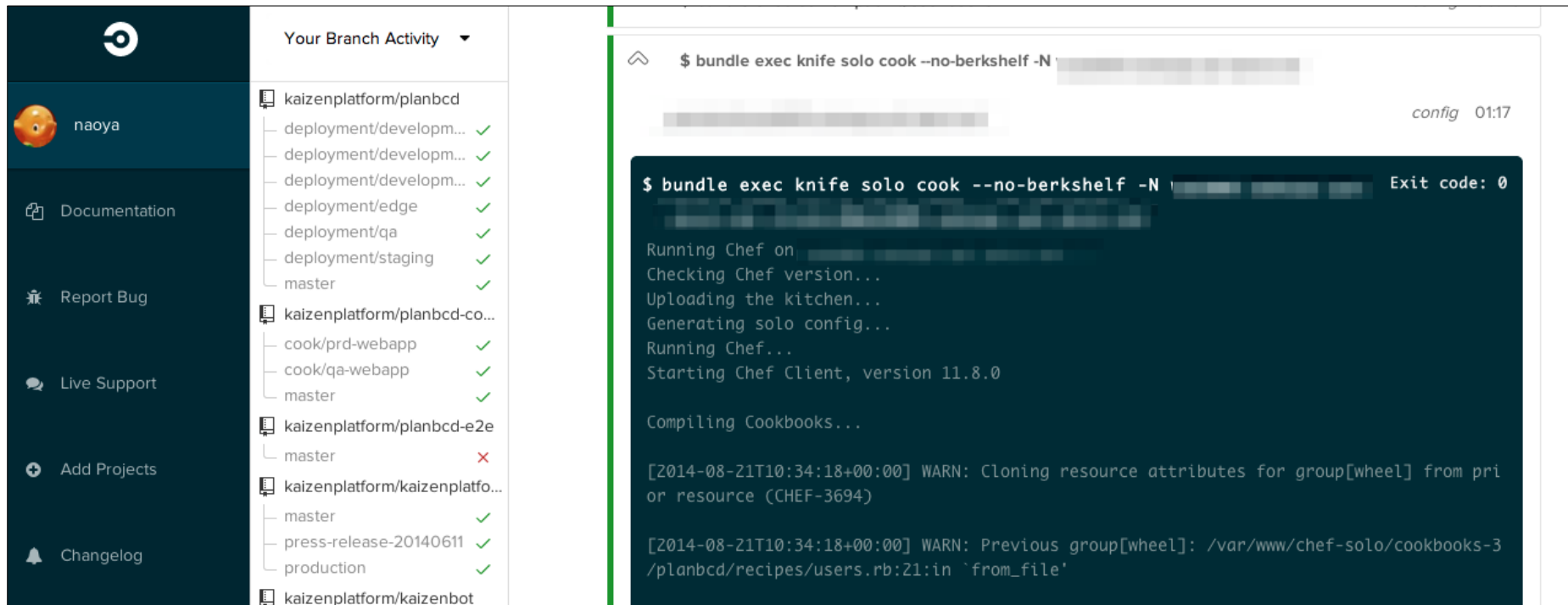


# 2. デリバリー用 Pull Requestが作られる

The screenshot shows a GitHub Pull Request interface. At the top, it says 'This repository Search' and 'Explore Gist Blog Help'. The repository name is 'kaizenplatform / planbcd-cookbooks' with a 'PRIVATE' label and 'Unwatch 11' button. The PR title is '2014/08/21 19:11 cook qa-webapp by naoya #699'. It is marked as 'Merged' and states 'naoya merged 23 commits into cook/qa-webapp from master 6 days ago'. Below this, there are statistics: 'Conversation 0', 'Commits 23', and 'Files changed 54'. A comment from 'kaizenbot' is visible, stating: 'Created by naoya on infra-team Room (via hubot)' and 'Build status will be shown: https://circleci.com/gh/kaizenplatform/planbcd-cookbooks/tree/cook%2Fqa-webapp'. Below the comment, a list of commits is shown, including 'wip', 'Upgrade fluent-plugin-bigquery to v0.2.0 for enabling fetch\_schema', 'Revert "Upgrade fluent-plugin-bigquery to v0.2.0 for enabling fetch\_s...', 'Upgrade fluent-plugin-bigquery to v0.2.0 for enabling fetch\_schema', 'Upgrade fluent-plugin-bigquery to v0.2.1', 'fix logapp env', and 'add cook target'. On the right side, a list of commit hashes is visible, some with red 'X' marks and some with green checkmarks.

デリバリー用ブランチにメインブランチを merge する

# 3. CircleCI が検知して実行



The screenshot displays the CircleCI web interface. On the left is a navigation sidebar with the CircleCI logo and user profile 'naoya'. The main area is titled 'Your Branch Activity' and lists various branches for different projects, such as 'kaizenplatform/planbcd' and 'kaizenplatform/kaizenbot', with green checkmarks indicating successful builds. On the right, a job execution log is shown for the command '\$ bundle exec knife solo cook --no-berkshelf -N'. The log output includes 'Running Chef on...', 'Checking Chef version...', 'Uploading the kitchen...', 'Generating solo config...', 'Running Chef...', and 'Starting Chef Client, version 11.8.0'. It also shows two warning messages from Chef regarding resource attributes and group definitions. The job status is 'Exit code: 0'.

```
$ bundle exec knife solo cook --no-berkshelf -N [redacted] config 01:17

$ bundle exec knife solo cook --no-berkshelf -N [redacted] Exit code: 0

Running Chef on [redacted]
Checking Chef version...
Uploading the kitchen...
Generating solo config...
Running Chef...
Starting Chef Client, version 11.8.0

Compiling Cookbooks...

[2014-08-21T10:34:18+00:00] WARN: Cloning resource attributes for group[wheel] from pri
or resource (CHEF-3694)

[2014-08-21T10:34:18+00:00] WARN: Previous group[wheel]: /var/www/chef-solo/cookbooks-3
/planbcd/recipes/users.rb:21:in `from_file'
```


# 間をつなぐ Web API

- Slack API
  - Hubot が Slack API 経由でチャットに常駐
- GitHub API
  - Hubot から GitHub API で Pull Request 作成
- CircleCI API
  - GitHub の WebHook で CircleCI へ通知
- AWS
  - (場合によっては) AWS の API で構成を調整



# ほかにもこんな

- BugSnag … Rails / JS の例外検知
- Stripe … 決済
- PagerDuty … インフラのアラート
- Pingdom … 監視
- Mackerel … リソースモニタリング
- Mandrill / MailChimp … メール
- Google BigQuery … ログ解析、DWH
- BrowserStack … クロスブラウザテスト
- ほかには … DMP (Data Management Platform) とかも



サーバーサイドの  
Rails アプリがグル  
ー (glue) になって、  
業務を紡ぐ



# 発想の転換

- かつて
  - 何か自動化しようかな
  - 手持ちのライブラリ、OSSなどを選択肢に案を練る
  - よし、コード書こう
- いま
  - 何か自動化しようかな
  - クラウドサービス探す
  - あのサービスとこのサービスを組み合わせせて…
  - よし、コードを書こう

「コンテンツのマッシュアップ」のような限られた状況での思考プロセスだったのが、そうでない状況でも



# Webシステムのアーキテクチャトレンドと Programmable Web (ここからは妄想)



# (自分的な) 注目キーワード

- microservices
- Immutable Infrastructure

Ajax や \* as a Service が Web API の発展を拡大させたように、これらのキーワードもそれを後押しする予感



# microservices

<http://martinfowler.com/articles/microservices.html>

MARTIN FOWLER

[Intro](#) [Design](#) [Agile](#) [Refactoring](#) [NoSQL](#) [DSL](#) [Delivery](#) [About Me](#) [ThoughtWorks](#) [RSS](#) [Twitter](#)

## Microservices

*The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.*

25 March 2014



### James Lewis

James Lewis is a Principal Consultant at ThoughtWorks and member of the

### Contents

- Characteristics of a Microservice Architecture
- Componentization via Services
- Organized around Business Capabilities
- Products not Projects
- Smart endpoints and dumb pipes
- Decentralized Governance

# microservices

- モノリシックなWebアプリケーションがでかくなるとメンテナビリティが落ちる
- 適当な役割でくくりだして Web API (など) で繋げましょう
- エンタープライズ派出の "SOA" に対して Practical な "microservices"
  - "XML Webサービス" と "Web API" に似てる



# Immutable Infrastructure

- a.k.a. Disposable Component
  - 一度稼働させたサーバーは二度と変更しない
  - 必要になったら新しいの作って古いの捨てる
- 例: Heroku のコンテナ

# Immutable Infrastructure と Web アプリケーションの交換可能性


- Immutable: その上で動く Web アプリは再現可能でなければならない
  - いつ捨てられても、全く同じ構成でデプロイ可能になるということ
  - キー: Gemfile、Procfile、Shared Nothing
- 再現可能 = 状態を持たない (ステートレス) = 人に渡せる



# Heroku Button

README.md

## Markdown to Inao-Form

 Deploy to Heroku

build passing coverage 96%

短冊



KAIZEN PLATFORM

# Heroku Button の仕組み

- [heroku.com/deploy](https://heroku.com/deploy) にリンクしてる
- [heroku.com/deploy](https://heroku.com/deploy) を HTTP GET すると、リファラから GitHub レポジトリを特定
- `app.json` を読み取り、それに従いセッション主の Heroku アカウントにデプロイ



# Heroku Button をある 視点で見ると

- URI で Webアプリケーション交換
  - HTTP GET という統一インターフェースでアプリケーションというリソースを交換している
  - Immutable Infrastructure → Stateless → URI があれば、交換できる

Programmable Web  
との交差点がなんとなく  
見えてきますよね



# まとめ

- Web API は、データを公開するためのものから、本来の意味の API に
- かつて「XML Web サービス」が目指していた世界が実現された (近づいた)
  - \* as a Service
- Web システムのアーキテクチャの進化が今後さらにそれを後押しする
  - microservices / Immutable Infrastructure ...





# Thanks!



絵 by あわゆき