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 Programable Web

私が見てきた Web API の変遷

Why: なんでそんな話するの?

- 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

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・マイクロス」。登場してからまだ日の浅い分野ですとで，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

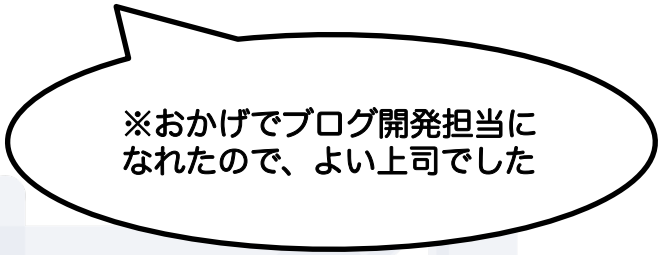
本ムックは，SOAPやUDDIといった新技めに必要な実用情報を多角的に取り上げたき，国内で利用可能な製品，実際の構築事らECへの適用方法までを示す解説ーー。こ専門用語については「Webサービスを理解

■日経BP社のIT分野の最新情報はこちら

この解像度しか残ってないっぽかったw
2002年の雑誌

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



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

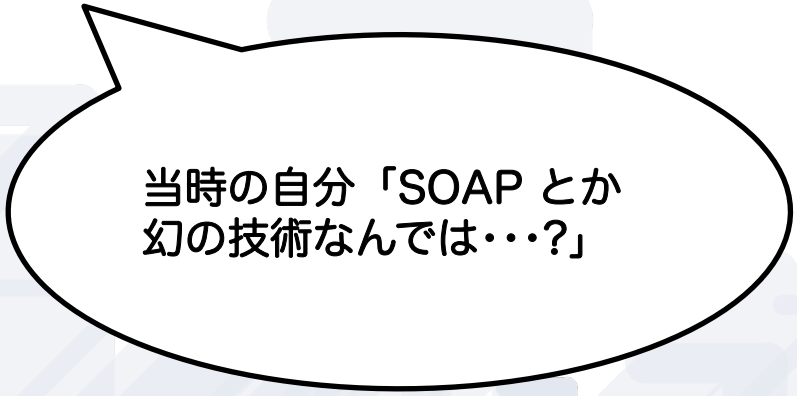
IBM，マイクロソフト，サン・マイクロシステムズなどが全社をあげて取り組む「Webサービス」。登場してからまだ日の浅い分野ですが，2002年に入って対応製品や構築事例が登場してきたことで，いよいよ現実味を帯びてきました。「Webの誕生よりもインパクトは大きい」とする声もあるように，インターネット/情報システム分野で今後のメイン・テーマとなることは間違いありません。

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ーー。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

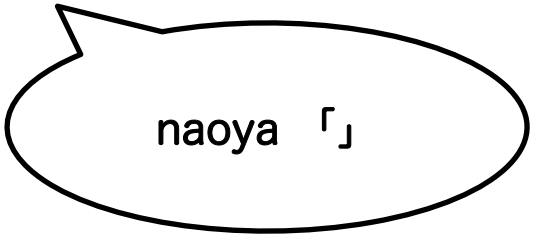
当時困ったこと

- 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



2003年頃

- 上司「伊藤君！ブログって知ってるか？」



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/

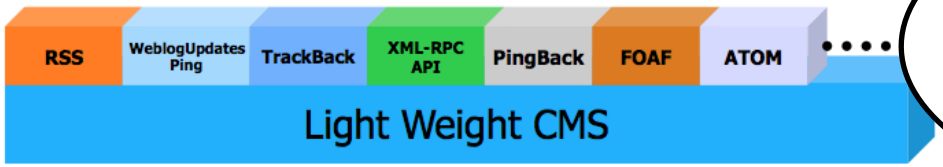
アジェンダ

- 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

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。

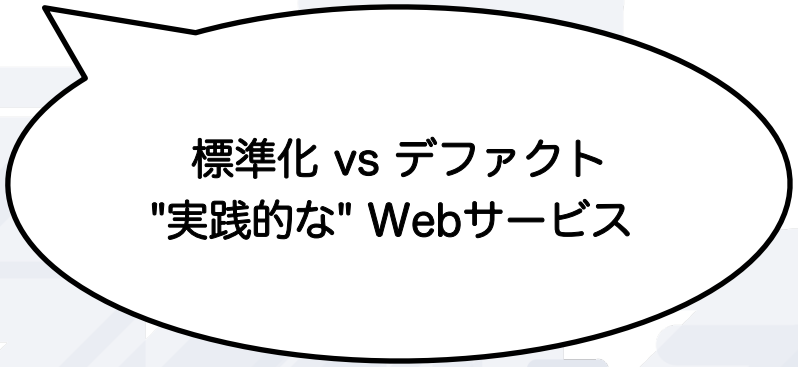


「**BlogはXMLだ!**」…これはちょっと大袈裟

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

Blogの背後にあった Web API

- 「軽量なWebサービス」
  - RSS、Atom、weblogUpdates.ping
- ひ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



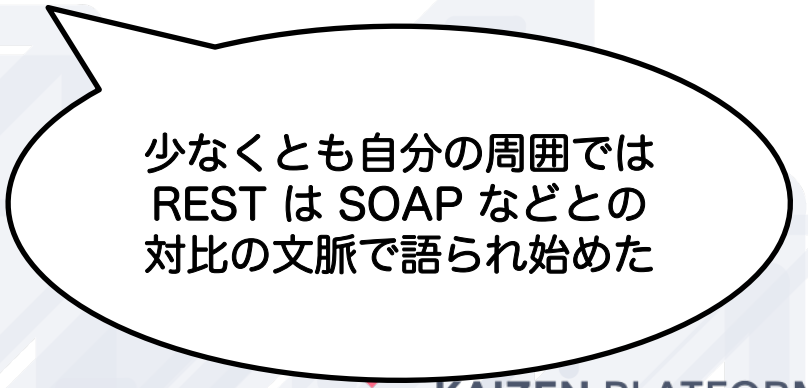
結果、"Web業界"で起こったこと

- Web 2.0 (2004、2005 あたり)
  - API を公開するとかっこいい? (中二病)

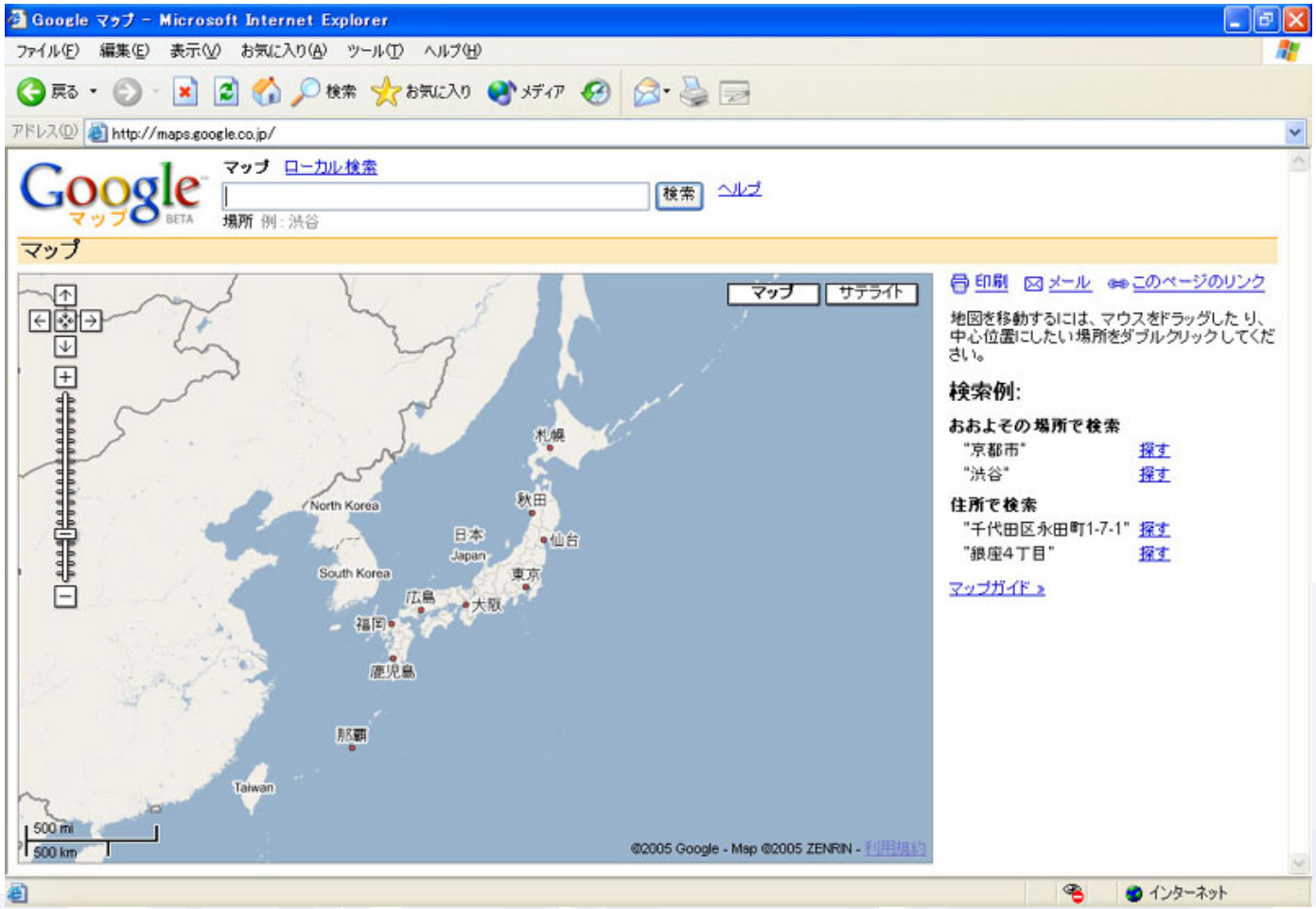
- さて、業務ロジックを公開しようぜといっても
  - Web業界「業務ロジック」って何だ?
    - 多くは背後にそれほど{大きい, 重要}な業務を持っていなかった
    - ∴ ロジックではなくデータを公開するという方向に流れた
      - Web業界にはメディア企業が多かったし
      - Blog/ニュースの RSS フィード、Amazon Product Advertisement API 等わかりやすい先行事例

REST

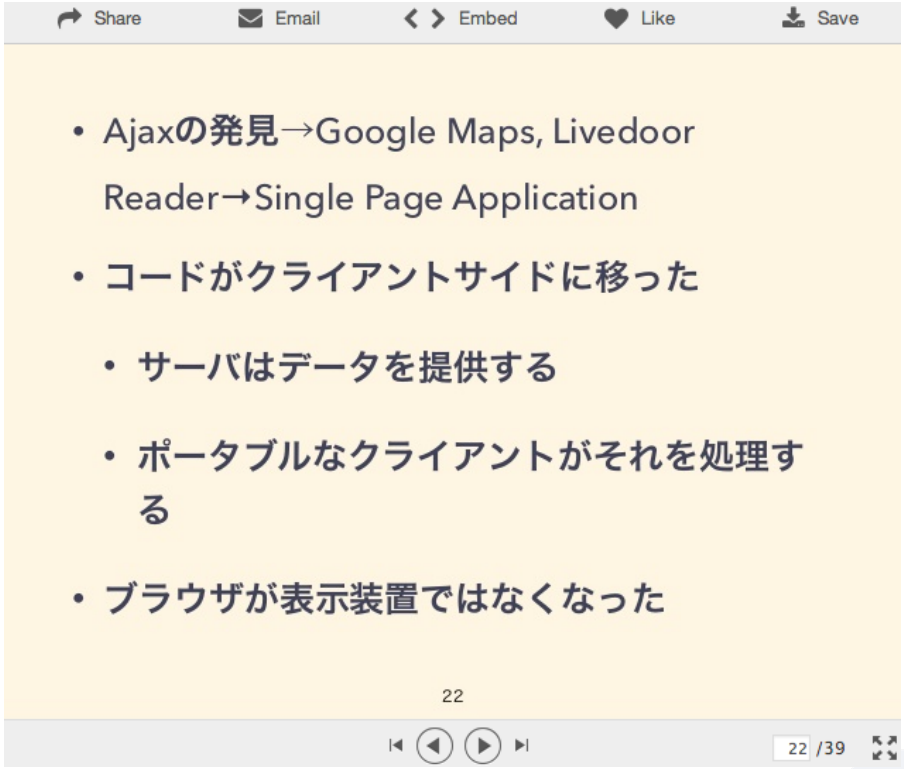
- 前回の yohei さんの話
- 当時
  - 実装厨 「SOAP とか XML-RPC とか言うけど、結局 XML を HTTP GET でよくね?」
  - アーキ厨 「そもそもそれは Representational state transfer と言ってだな」

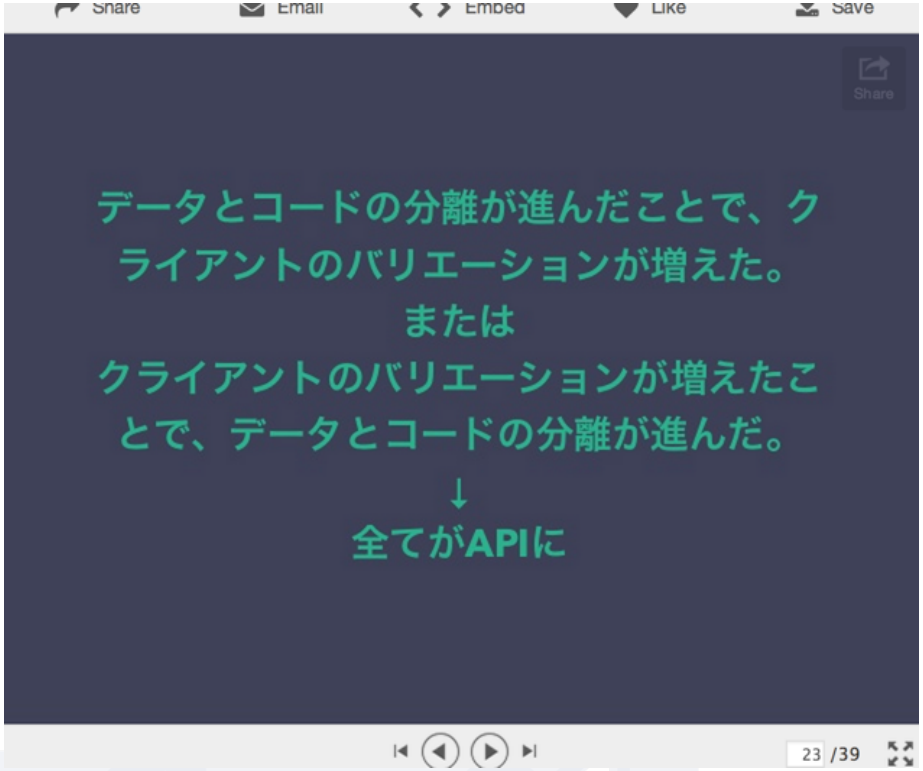


そうこうしてるうちに Ajax



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





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てたから?
    - 「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て業務を紡ぐ」という考え方は間違ってない
- 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

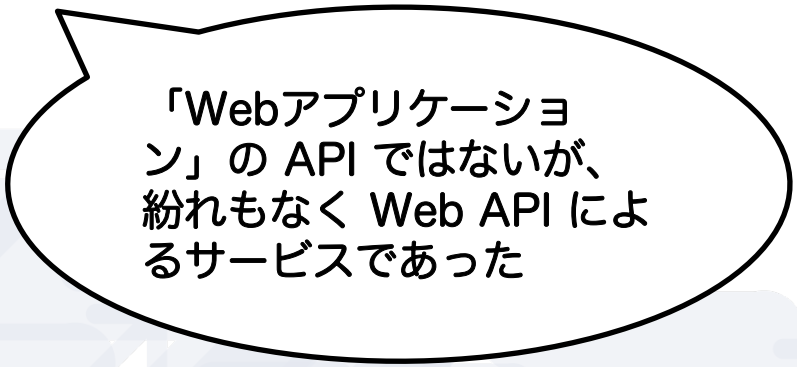
SOA で作った社内サービス、公開したよ

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

KAIZEN PLATFORM

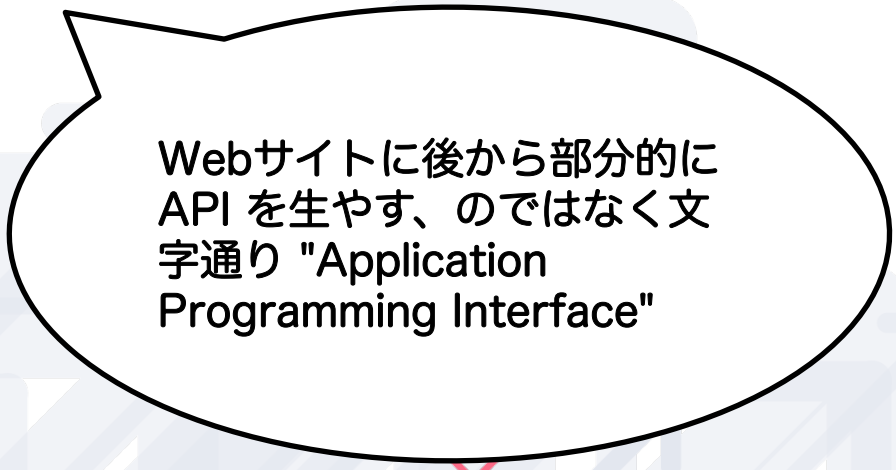
サービスとサービスを組み合わせて

- ストレージの Amazon S3 と◯◯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



AWS と Web API

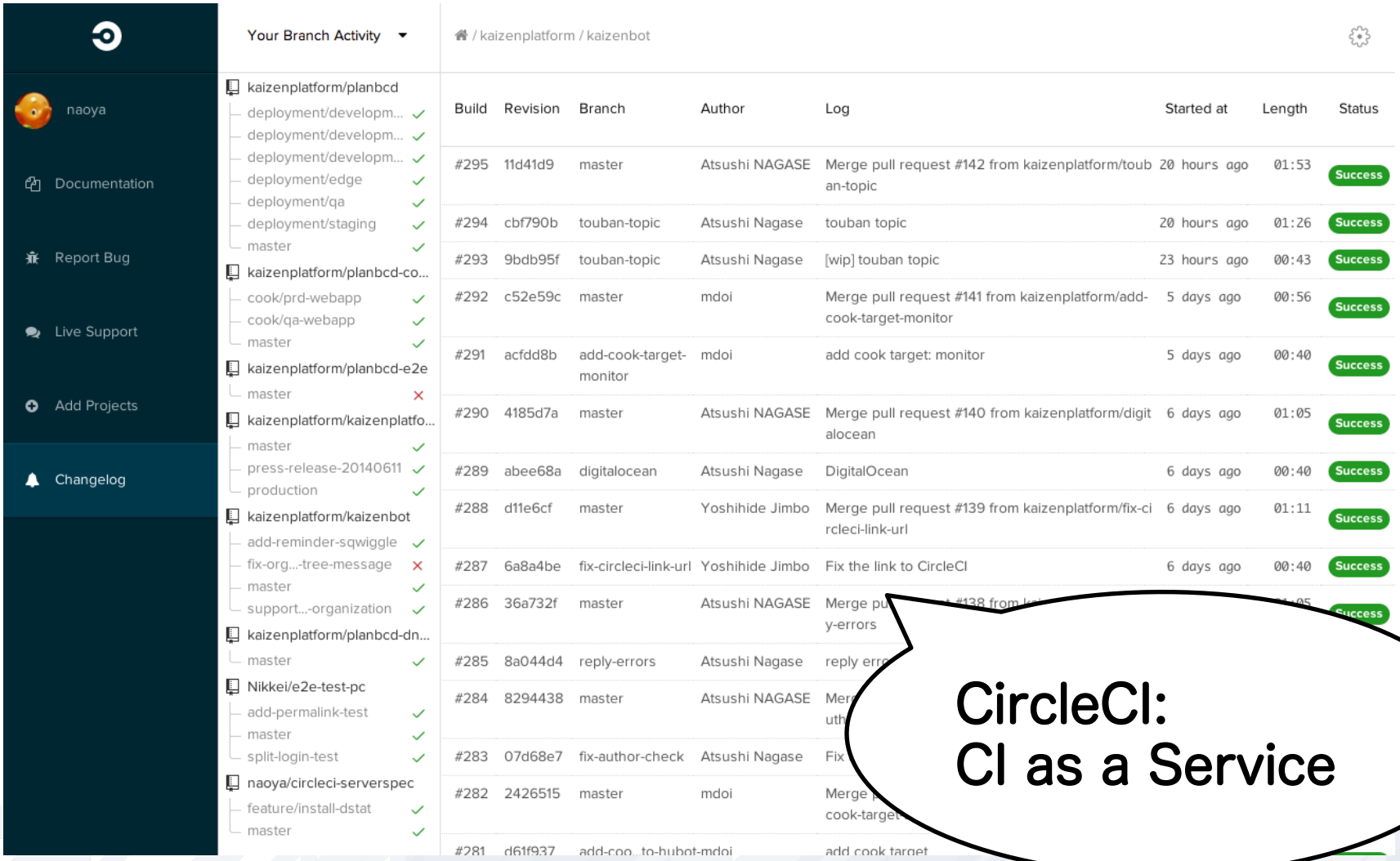
- AWS の各コンポーネントは「後付けでAPI化されたサービス」ではなく「API でサービスする」ことを前提にされたもの
  - Programmable 且つ「需要」が先にあった
  - そうそう "API" ってこれのこと



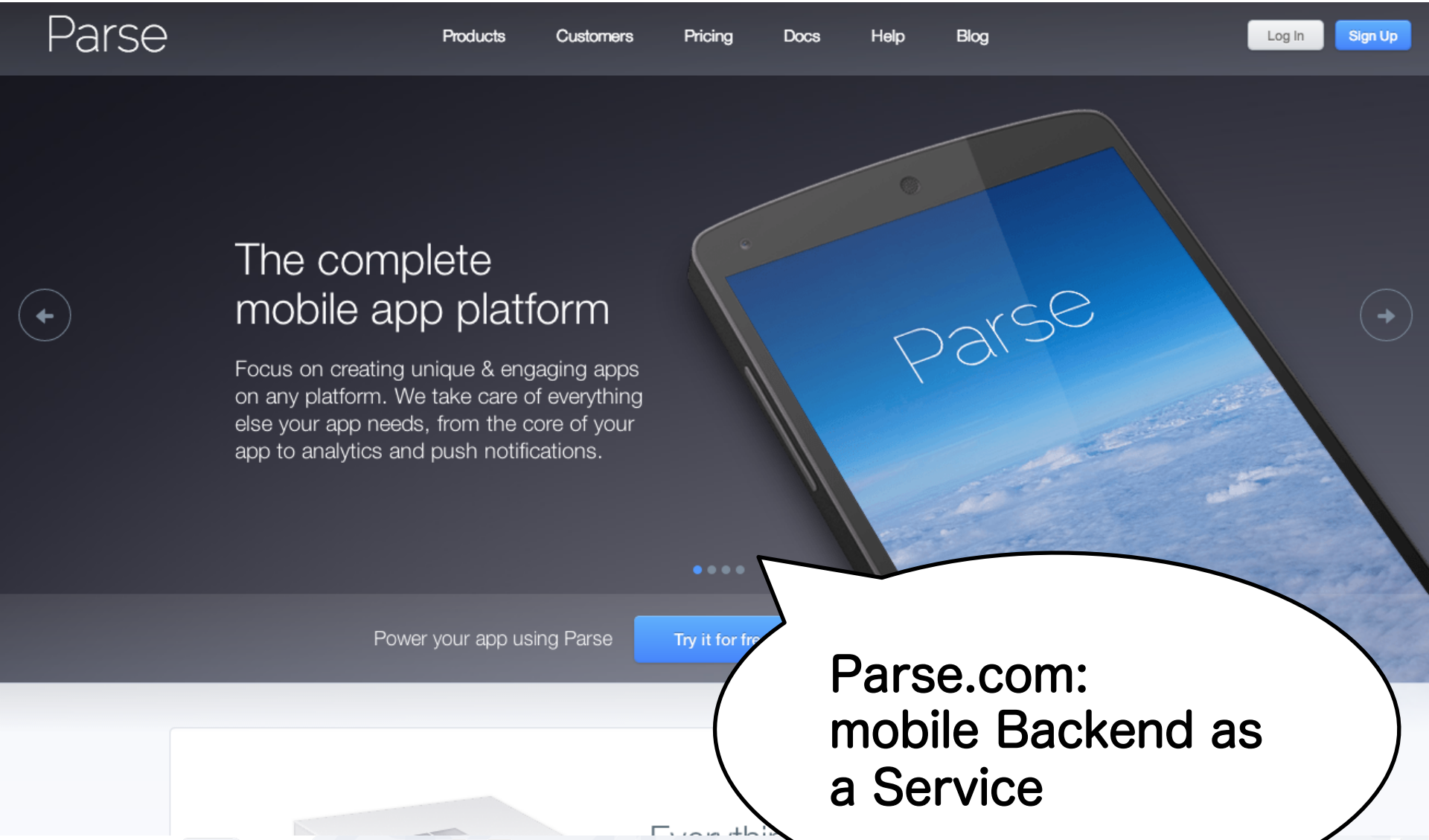
そしてクラウドの時代へ・・・

※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

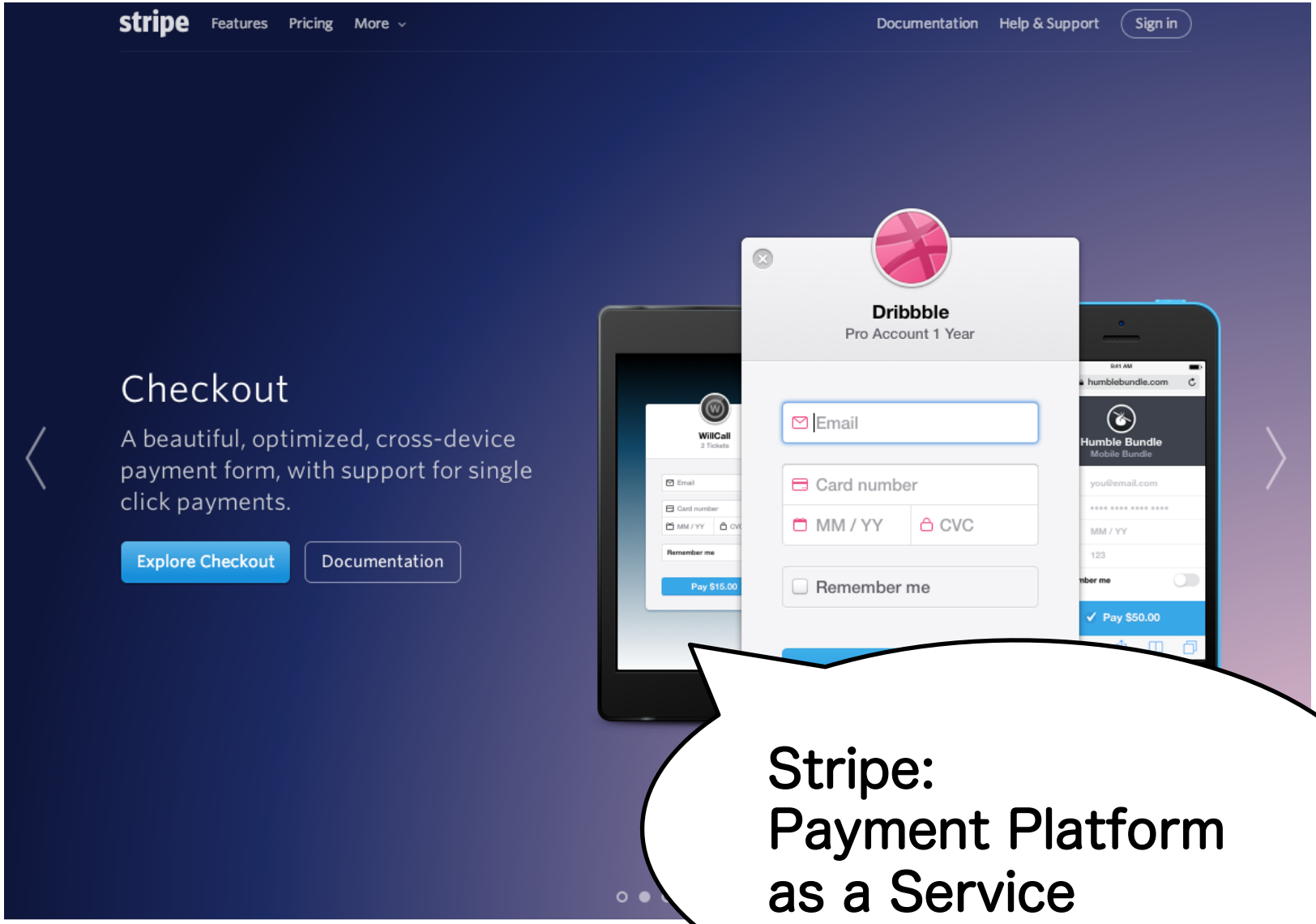


KAIZEN PLATFORM



Parse.com:
mobile Backend as
a Service

KAIZEN PLATFORM



Stripe: Payment Platform as a Service

KAIZEN PLATFORM

Google BigQuery:
BigData as a Service

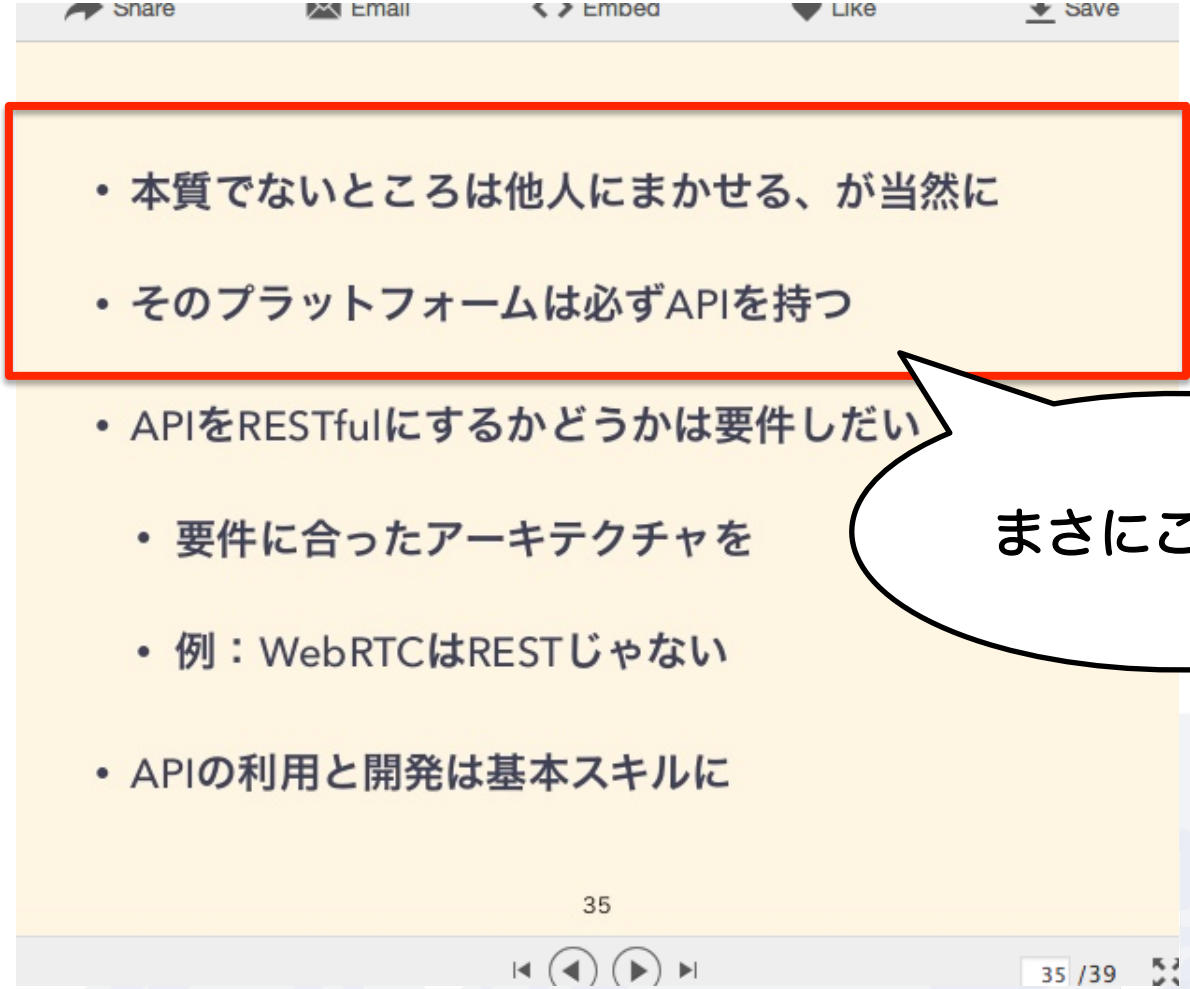
KAIZEN PLATFORM

* 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

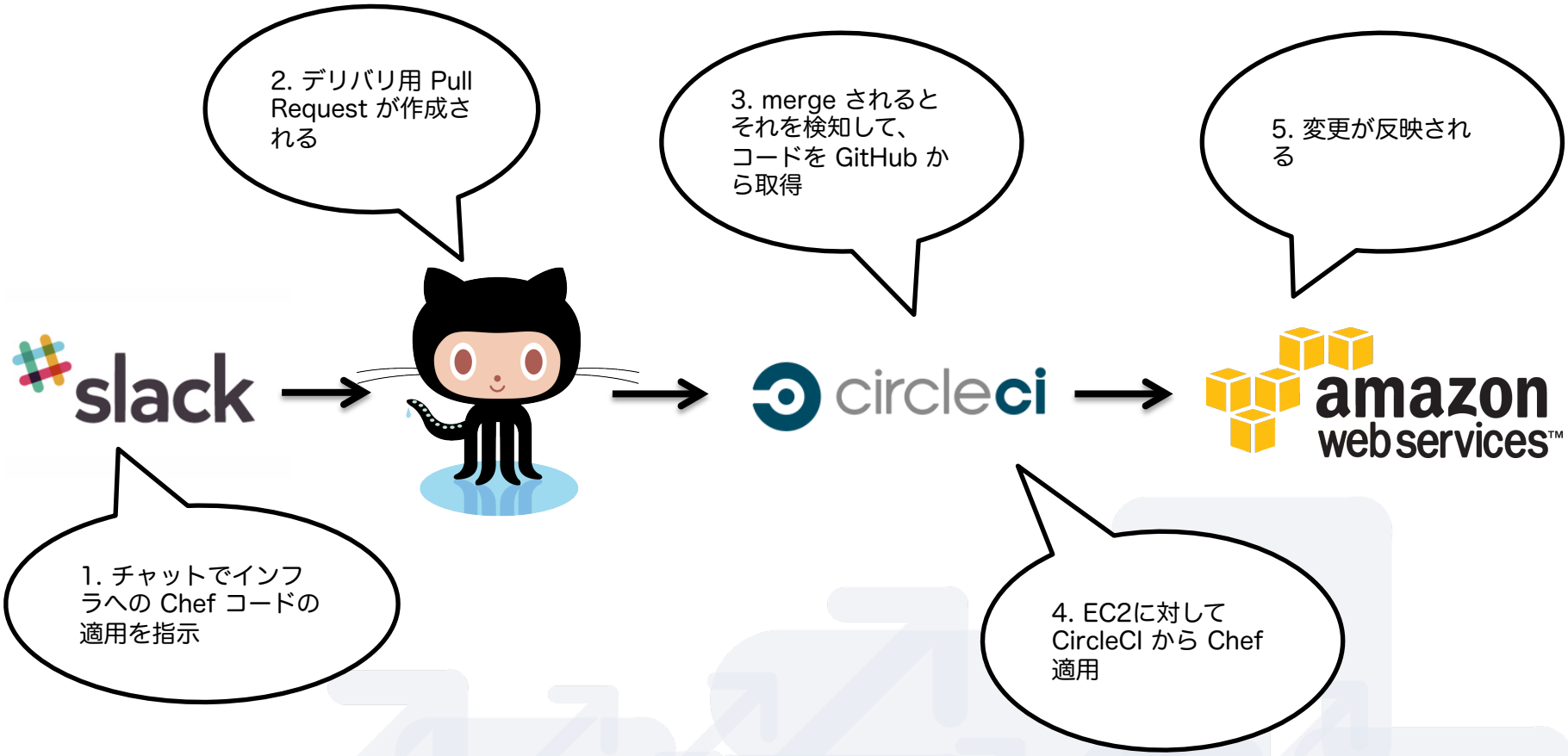
* as a Service を駆使して業務を構築する (イマココ)

- 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。)



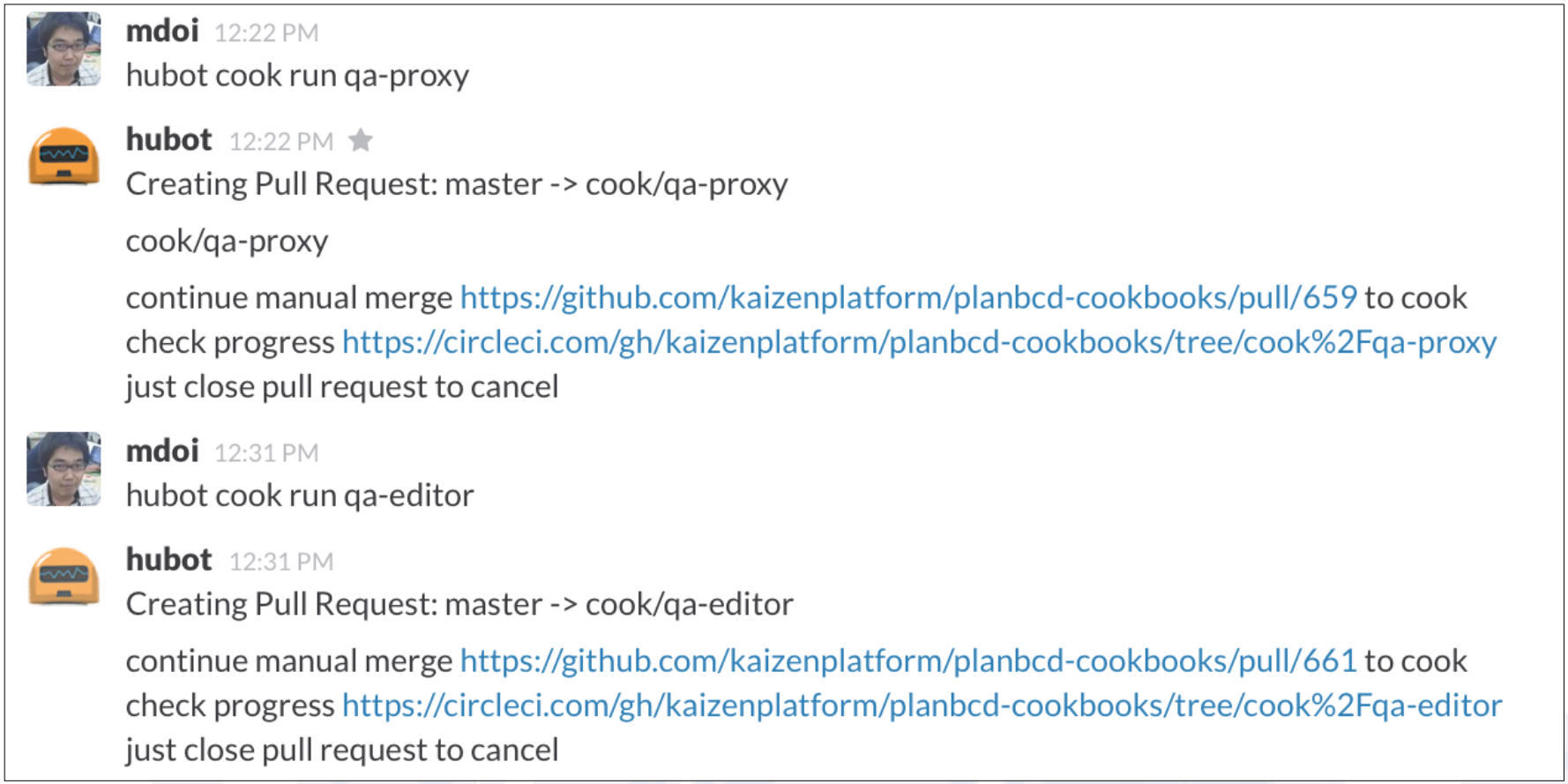
どんな風に使っているか
(ユースケース)

インフラ構成の継続的デリバリー



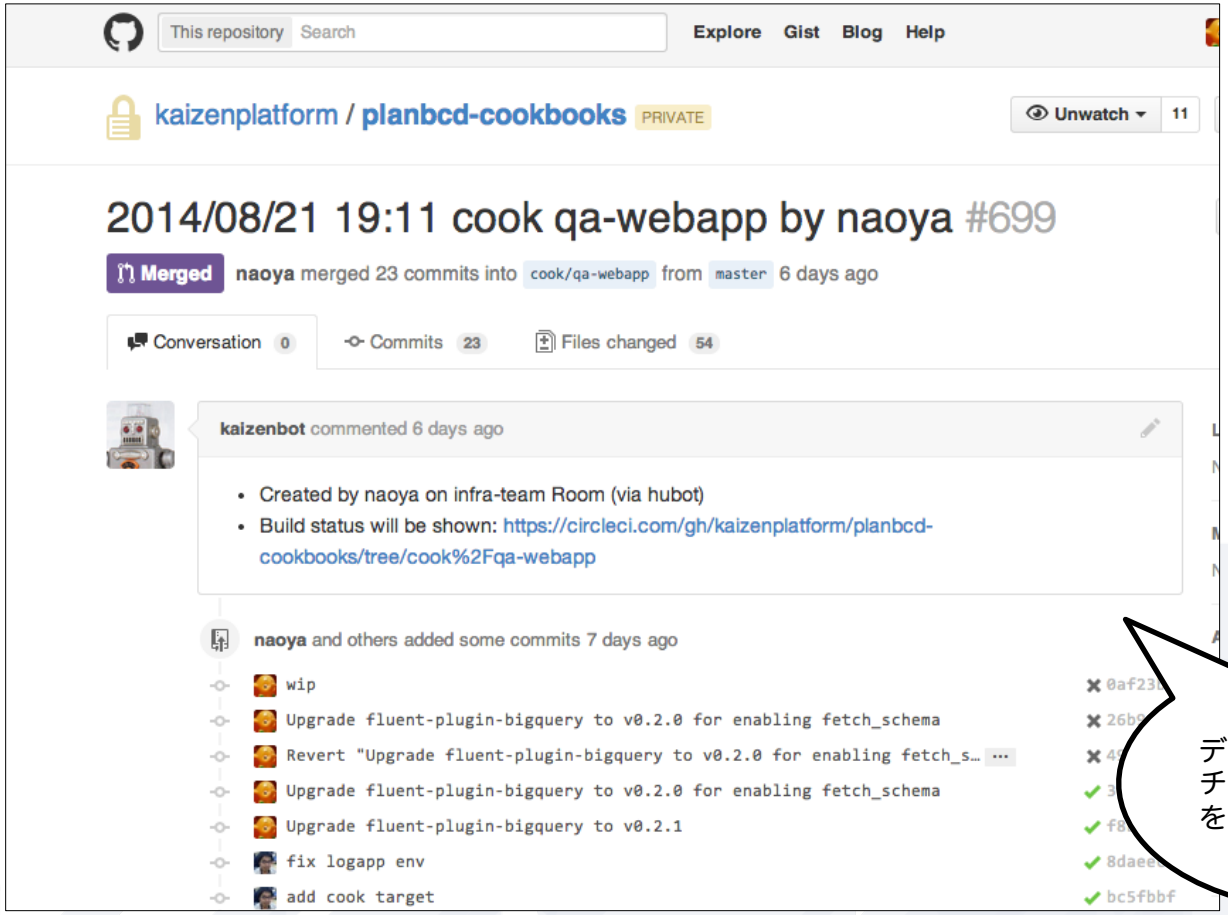
KAIZEN PLATFORM

1. チャットで指示



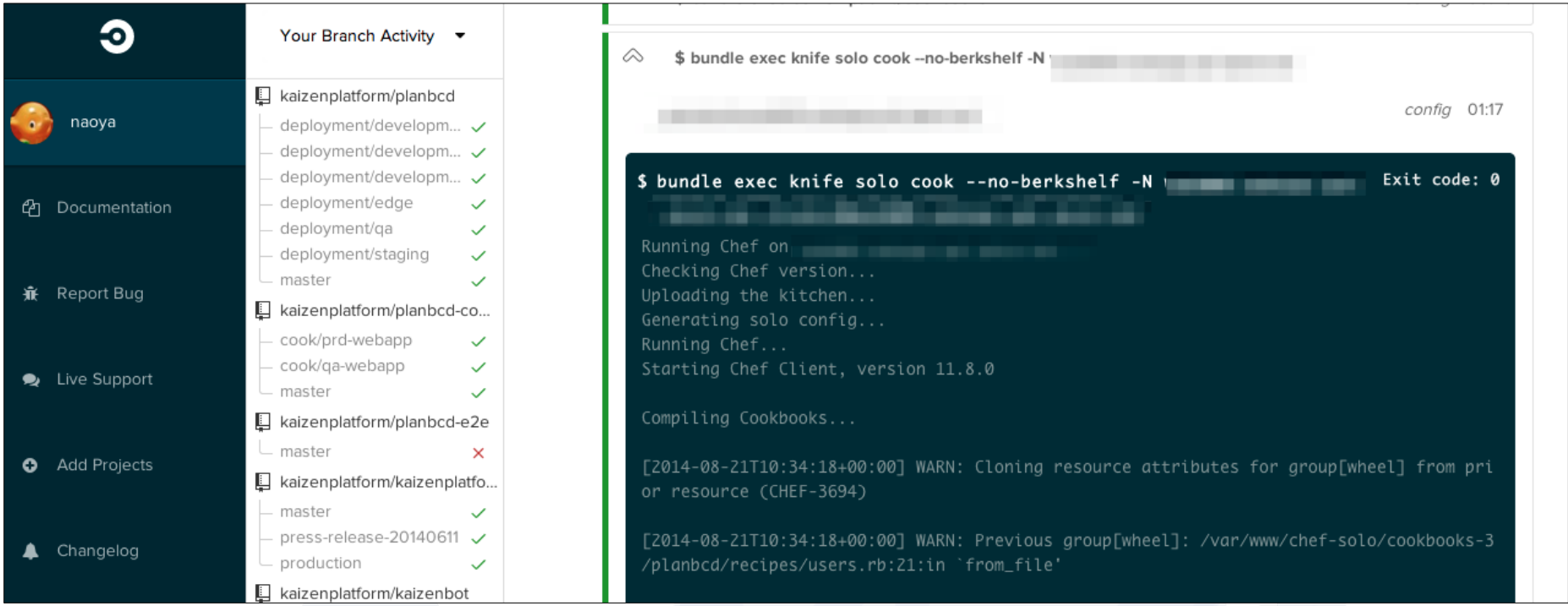
KAIZEN PLATFORM

2. デリバリー用 Pull Reqestが作られる



KAIZEN PLATFORM

3. CircleCI が検知して実行



KAIZEN PLATFORM

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

ほかにもこんなの

- 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

KAIZEN PLATFORM

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)

KAIZEN PLATFORM

(自分的な) 注目キーワード

- microservices
- Immutable Infrastructure

microservices

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

MARTIN FOWLER

Intro Design Agile Refactoring NoSQL DSL Delivery About Me ThoughtWorks

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

FORM

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

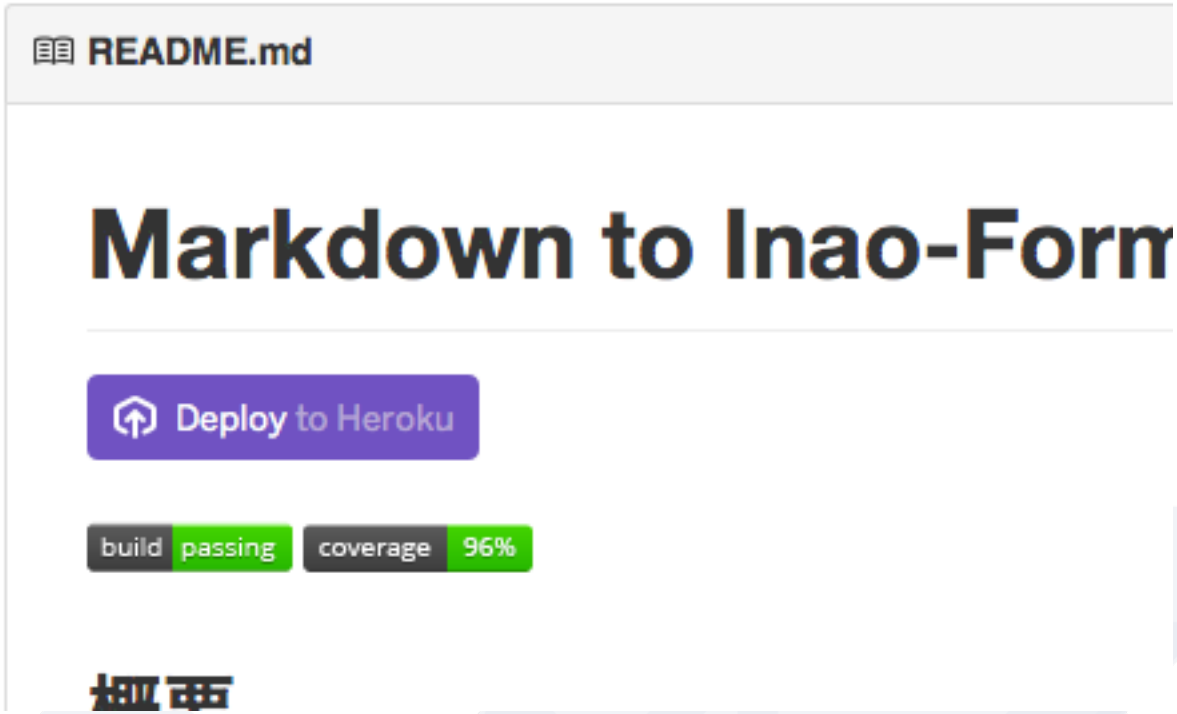
Immutable Infrastructure

- 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

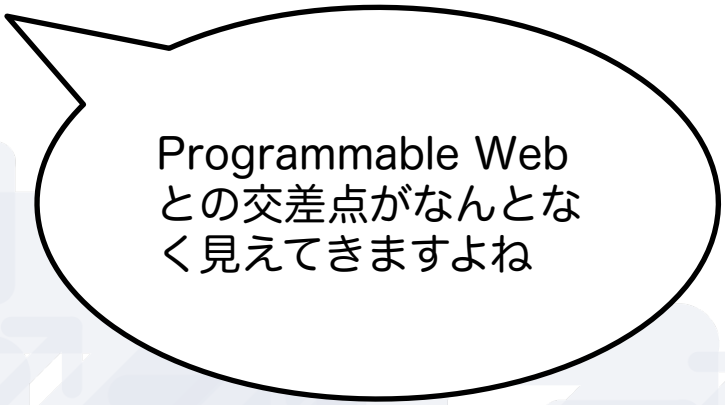


Heroku Button の仕組み

- heroku.com/deploy にリンクしてる
- 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 をある視点でみると

- 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

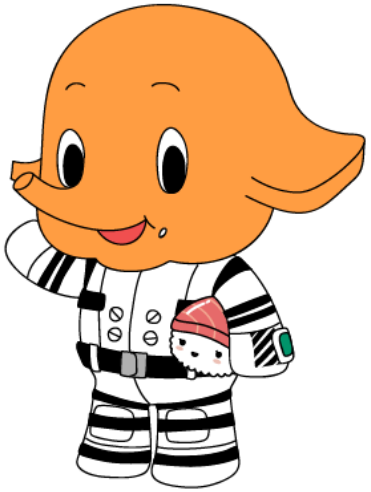


KAIZEN PLATFORM

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

KAIZEN PLATFORM