Serverspecに見る技術トレンドを生み出すヒント

自己紹介

-  宮下 剛輔
- http://mizzy.org/
- mizzy@github
- gosukenator@twitter
- フリーランスエンジニア/Serverspec Operations

アジェンダ

- 最近のインフラ系技術トレンド
- **Serverspec**とは
- **Serverspec**が生まれた背景
- **Serverspec**の認知度
- **Serverspec**の成功要因
- まとめ

最近のインフラ関連技術トレンド振り返り

注: OSからミドルウェア
といったソフトウェアレイヤーを
指す言葉としてのインフラです

インフラ関連技術トレンド

- サーバ構成管理ツール
  - **Chef**人気
    - 今年になって**4**月,**5**月と立て続けに書籍発刊
    - **US**では**2014**年に**Puppet**のシェアを超えたらしい
  - **Ansible**
    - **Chef**へのカウンター

インフラ関連技術トレンド（つづき）

- 仮想化関連ツール
  - Vagrant
  - Docker
- テスト駆動インフラとインフラCI
  - Test Kitchen
  - Serverspec

インフラ関連技術トレンド（つづき）

- Immutable Infrastructure
  - Trash Your Servers and Burn Your Code: Immutable Infrastructure and Disposable Components
  - WEB+DB PRESS Vol.81

テスト駆動インフラ
インフラCI

Webアプリケーション開発のためのプログラミング技術情報誌
FOR ALL WEB APPLICATION DEVELOPERS ウェブDBプレス

新人さん大歓迎!
祝! 80号

WEB+DB PRESS

10年先も役立つ!!

vol. 80 2014

Web技術入門

●TCP/IP ●HTTP ●URL ●HTML

Infrastructure as Codeがもたらすワークフローの変革

テスト駆動インフラ

はじめての

コマンド操作も怖くない><

Mac開発環境

効率的に知識を得て、成果に結び付ける

エンジニアの学び方

スタンフォード大学の無料オンライン授業

Courseraで計算機科学を学ぼう

新連載 モダンなPHP開発環境

テスト駆動インフラ/インフラCI

- アプリケーション開発の手法をインフラ構築に応用
- @riywoくんが2012年にインフラCIについて書いてる
  - http://blog.riywo.com/2012/05/27/145310
- 自分自身構想は2007年ぐらいからあった
  - 「テスト駆動サーバ構築」という言葉を考えていた
  - http://d.hatena.ne.jp/dayflower/20070405/1175782564#c

テスト駆動インフラ/インフラCI

- 以前から概念はあったとはいえ「インフラをテスト可能にする」という世界を広げたのはServerspec
- 単にテスト可能にするだけではなく、GitHub Flowによりインフラ変更をプルリクエストし、CIサーバにより継続的にテストする、という世界を開いた

Takuto Wada
@t_wada

Follow

Github Flow を使ったインフラ変更: インフラ変更が PR になり、 CI サーバによって serverspec でテストされ、テストが通ったら安心してマージ可能になるフローの紹介。インフラ変更自体も PR & CI 可能になっている世界。やばい。 #devsumiB

10:31 AM - 13 Feb 2014

8 RETWEETS **9** FAVORITES

Serverspec

Serverspecとは

- 2013年3月末にリリース
- RSpecでサーバの状態をテスト
- 本質はサーバの状態を記述したコードをテスト
  - PuppetマニフェストやChefレシピなど
- インフラコードの開発やリファクタリングを効率良く行うためのツール

Serverspecによるテストコード

```
describe package('nginx') do
  it { should be installed }
end

describe service('nginx') do
  it { should be_enabled }
  it { should be_running }
end
```

Serverspecのスペル

- 正
  - Serverspec/serverspec/SERVERSPEC
- 誤
  - ServerSpec/Server Spec

Serverspecはなぜ生まれたのか？

テストツール開発のきっかけ

- 2006年にPuppetを使い始めた
- サーバ構築はPuppetによって自動化できた
- じゃあ次は構築後のチェックを自動化したい
  - Excelチェックシートでの目視確認やめたい

Assurer

- 2007年につくったPerl製サーバテストフレームワーク
- プラグインで拡張可能
  - Plaggerの影響
  - Filter, Format, Notify, Publish, Test
- make testでテストを実行/TAP形式で結果出力
  - CPAN AuthorなのでPerlでのテストのやり方を踏襲

Assurerの敗因

- Serverspecと違い振る舞いをテスト
  - 状態のテストよりも要件が複雑
- 欲張って色々詰め込みすぎて更に複雑になった
  - フェーズ分け/プラガブル/分散処理/シェル
- コンセプトがServerspecほど明確ではなかった
- 開発者である自分すら使わないものになった

Assurer以後

- 時々思い返すことはあったけど積極的に何かしようともしなかった
- 現場から若干距離を置いていて、Puppetなどもそんなに触らなくなった

またPuppetさわりはじめた

- 自分がかつて書いたPuppetマニフェストがレガシーコードになっていた
- リファクタリングしたくなった
- リファクタリングするなら当然テストが必要だしCIもやりたい

インフラテストの前準備

- インフラCIやろうと思って、まずはCIを実行する環境を整えた
- LXCコンテナでテストするために、コンテナをさくっとつくれるPuppetマニフェスト書いた
  - ちょうど同じタイミングでDockerが出てきて無用の長物になった

テストツール探し

- 環境は目処がついたので次はテストツール
- まずは既存のものでよさげなものがないか探した
  - **rspec-puppet**ぐらいしか**Puppet**で使えるものはなさそう
  - **Chef**周辺の方が充実
    - **ChefSpec, Test Kitchen, Cucumber-Chef, minitest-chef-handler**

自前でつくることにした

- ただし**rspec-puppet**はきちんとモジュール化されていないと使えない
- 自分のマニフェストはモジュール化されてないレベルのレガシーさ
- **rspec-puppet**は実際のサーバの状態をテストするわけじゃない
- じゃあ自分で作るか、ってことに

Serverspec誕生

- 当時の同僚(@hiboma)がLXCのコンテナをテストするコードをRSpecで書いていた
- よし、それパクって汎用的に使えるようしよう
- プロトタイプを一日でつくり、翌日にはrubygems.orgで公開

初期のServerspec

- 最初はRedHat系Linuxのみ対応、SSHバックエンドのみ。Execバックエンドはなかった。
- @tkmrさんが複数OS対応とDebian対応してくれた
- @raphinkさんがバックエンド切り替え対応してくれた
- こんな感じで色んな人を巻き込んだ

Serverspecその後

- Star: 738, Fork: 149, Contributors: 69
- 対応OS増えたり、サポートする機能も増えたり
- 現在v2.0に向けた作業中
  - より拡張しやすくするためのリファクタリングがメイン

本当に広く
使われてるの？

Serverspecの認知度

- Black Duck Open Source Rookies of the Year 2013
  - Docker, InfluxDB, Appium等と並んで受賞
- Thought Works Technology Radar 2014
  - Provisioning Testingの項目にServerspecの文字が出てくる
- rubygems.orgでのダウンロード数約30万
  - Chefが約400万

serverspec 1.9.0

RSpec tests for your servers configured by Puppet, Chef or anything else

```
INSTALL > gem install serverspec
```


Authors

Gosuke Miyashita

302,205 total downloads

7,555 for this version

Owners

Licenses

MIT

雑誌における言及

- WEB+DB PRESS Vol.75
- WEB+DB PRESS Vol.76
- WEB+DB PRESS Vol.80
- WEB+DB PRESS Vol.81

Webアプリケーション開発のためのプログラミング技術情報誌
FOR ALL WEB APPLICATION DEVELOPERS ウェブDBプレス

WEB+DB PRESS

vol.76
2013

実践UIデザイン
ユーザの満足度を高める設計、実装、検証方法

Web決済入門
PayPal、WebPay、
iOS/Androidアプリ内決済

D3.jsで、見やすく・多彩に・お手軽に
作って学ぶ
データ可視化

Gradle……進化したJavaのビルド
serverspec……テスト駆動インフラ構築
Fluentd＋FnordMetric……データの収集とグラフ化

書籍における言及

- Test-Driven Infrastructure with Chef, 2nd Edition
- Chef活用ガイド
- チーム開発実践入門
- Chef実践入門
- パーフェクトRuby on Rails

O'REILLY®

2nd Edition

Test-Driven Infrastructure with Chef

BRING BEHAVIOR-DRIVEN DEVELOPMENT
TO INFRASTRUCTURE AS CODE

Stephen Nelson-Smith

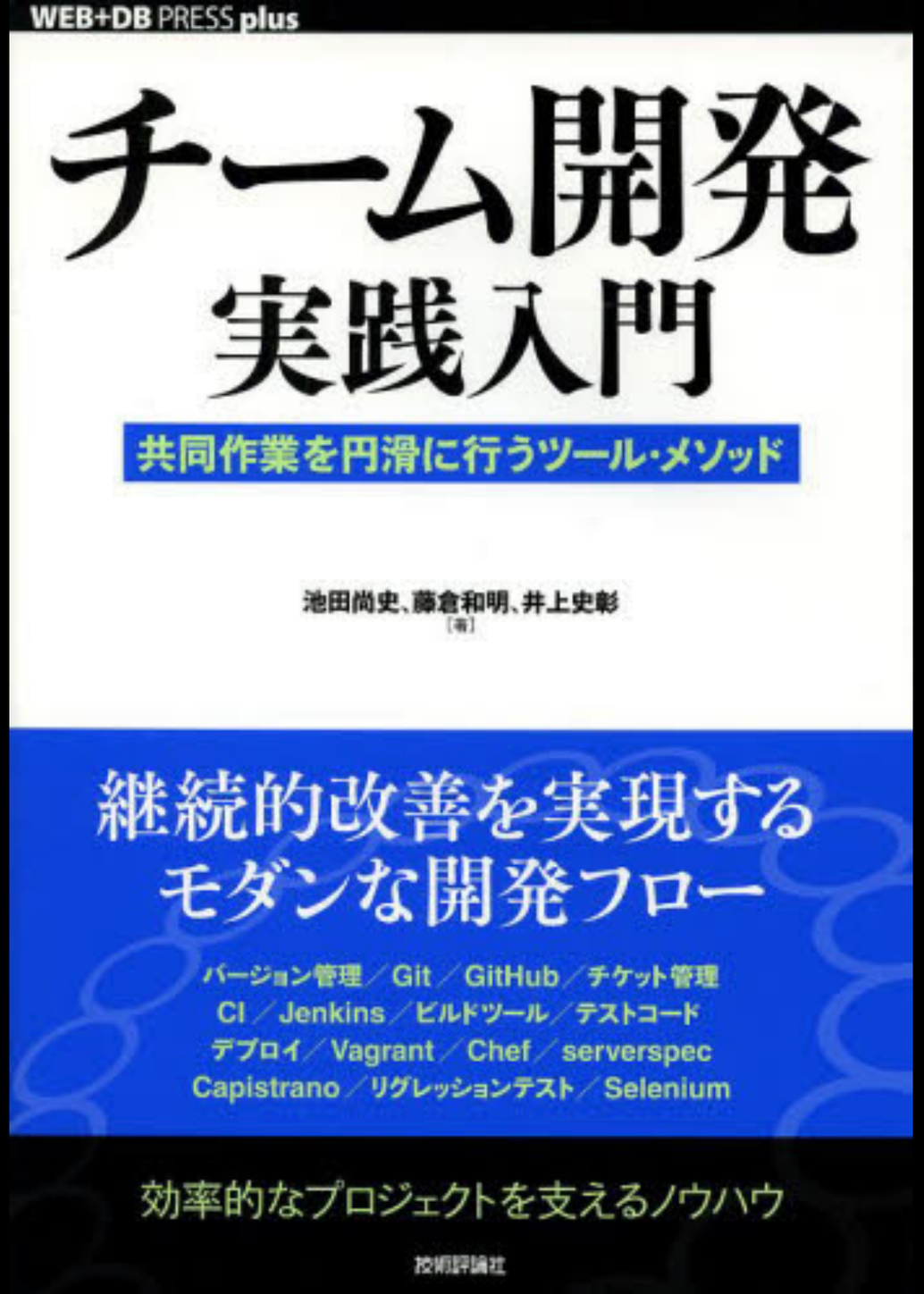
Chef 活用ガイド

コードではじめる構成管理

澤登亨彦、樋口大輔 著
クリエーションライン株式会社 監修

Infrastructure as Codeを実践しよう！

ASCII


WEB+DB PRESS plus
チーム開発
実践入門
共同作業を円滑に行うツール・メソッド
池田尚史、藤倉和明、井上史彰
[著]
継続的改善を実現する
モダンな開発フロー
バージョン管理／Git／GitHub／チケット管理
CI／Jenkins／ビルドツール／テストコード
デプロイ／Vagrant／Chef／serverspec
Capistrano／リグレッションテスト／Selenium
効率的なプロジェクトを支えるノウハウ
技術評論社

WEB+DB PRESS plus

Chef 実践入門

コードによるインフラ構成の自動化

吉羽龍太郎、安藤祐介、伊藤直也
菅井祐太朗、並河祐貴
［著］

サーバ構築／運用を
ミスなく効率的に行う

Chef Solo／VirtualBox／Vagrant／knife
冪等性／レシピ／クックブック
Ohai／Berkshelf／Sahara／Chef Server
テスト駆動／serverspec／チートシート

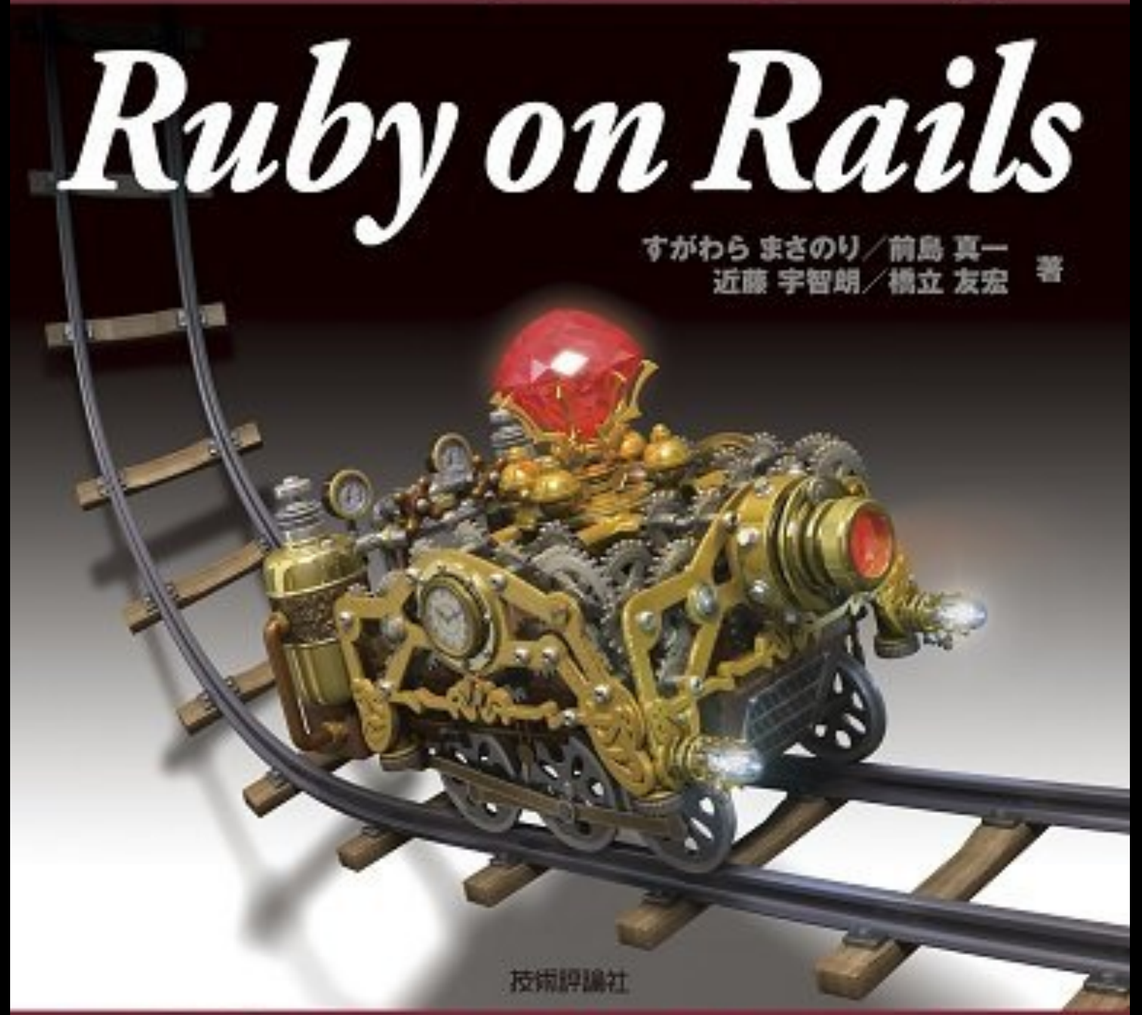
第一線のエンジニアが現場のノウハウを凝縮

技術評論社

パーフェクト

Ruby on Rails

すがわら まさのり／前島 真一
近藤 宇智朗／橋立 友宏 著



技術評論社

Railsプログラマ必読
Rails 4.1系対応！

Railsのセオリーを徹底解説

Railsに関する基本情報からテスト、インフラ・運用、設計や拡張法まで完全網羅

MVC／Asset Pipeline／モデル設計／DevOps／Rails Engine

海外での反応

Results for **from:adamhjk serverspec** Save

Top / All

Adam Jacob @adamhjk · Feb 20

@fnichol that's true. between chefspec and **serverspec**, rspec covers all my things.

Castro, San Francisco

View conversation Reply Retweet Favorite More

Adam Jacob @adamhjk · Feb 20

@Twirrim unit and functional testing of infrastructure code is great. **serverspec** output is a great monitor.

Castro, San Francisco

View conversation Reply Retweet Favorite More

Adam Jacob @adamhjk · Feb 20

No matter that I dislike the implied test subject syntax in **Serverspec**: it is completely awesome to use.

Castro, San Francisco

Expand Reply Retweeted Favorite More

Adam Jacob @adamhjk · Feb 12

@drnic test-kitchen and **serverspec** are totally copacetic. test-kitchen is the harness, **serverspec** is the test tool

Mission Dolores, San Francisco

View conversation Reply Retweet Favorite More

Adam Jacob @adamhjk · Feb 12

@drnic like, I love what **serverspec** (gist.github.com/adamhjk/8947573) does - but the implicit tests drive me crazy

Mission Dolores, San Francisco

View conversation Reply Retweet Favorite More

INFRASTRUKTUR MIT QUALITÄT

Peter Roßbach
Systemarchitekt, Infra-Coder, Trainer, Softwareentwickler, Apache Tomcat Committer

Andreas Schmidt
Infra-Coder, System Engineer, Softwareentwickler

Security-Tests mit serverspec

12 JUNI 2014

Sichere Systeme zu bauen und zu betreiben ist eine kontinuierliche Herausforderung. Ein erster Ansatz ist, Sicherheitsaspekte zu spezifizieren und testbar zu machen. Serverspec besitzt eine Reihe von nützlichen Eigenschaften, um dem Ziel näher zu kommen.

Durch die deklarative rspec-Syntax ist es möglich, Infrastrukturspezifikationen zu verfassen, die sicherheitsrelevante Aspekte, wie etwa Berechtigungen, beschreiben. Und das ganze wird natürlich auf Knopfdruck testbar.

You should_not ...

Eine praktische Eigenschaft ist dabei, Eigenschaften ausdrücken, die eben nicht vorliegen sollten. Dazu bietet rspec das Schlüsselwort `should_not` an, wie z.B. in

```
describe file '/etc/ssh/sshd_config' do
  its(:content) { should_not contain(/PermitRootLogin Yes/) }
end
```

Security-Tests beziehen sich natürlich nicht nur auf Eigenschaften von Dateien, sondern auf viel mehr, beispielsweise:

- Pakete: Manche Pakete sollten besser nicht installiert sein (Beispiel: `sendmail`)
- Dienste: Service sollten nicht laufen, und auch nicht enabled sein. Andere sollten auf jeden Fall laufen (z.B. iptables)
- Die Dateiberechtigungen betreffen vor allem World-Rechte und Ausführbarkeit. Die sollten stark eingeschränkt sein.
- Dateien sollten nur den Nutzer gehören, die sie zur Ausführung benötigen.
- Manche User sollten keine Shell haben.
- (NFS-)Mounts, die nur als Dateiablage dienen, sollten auch nicht ausführbar sein.
- Services die Ports öffnen, sollten dies nur auf bestimmten IPs tun.
- Konfigurationsdateien für Services wie bspw. Apache sollten sichere Einstellungen beinhalten, z.B. aktuelle SSL Cipher Suites
- Die Zertifikate und Schlüssel müssen die richtigen sein, z.B. auch mit hoher Schlüsselstärke
- ... und vieles mehr.

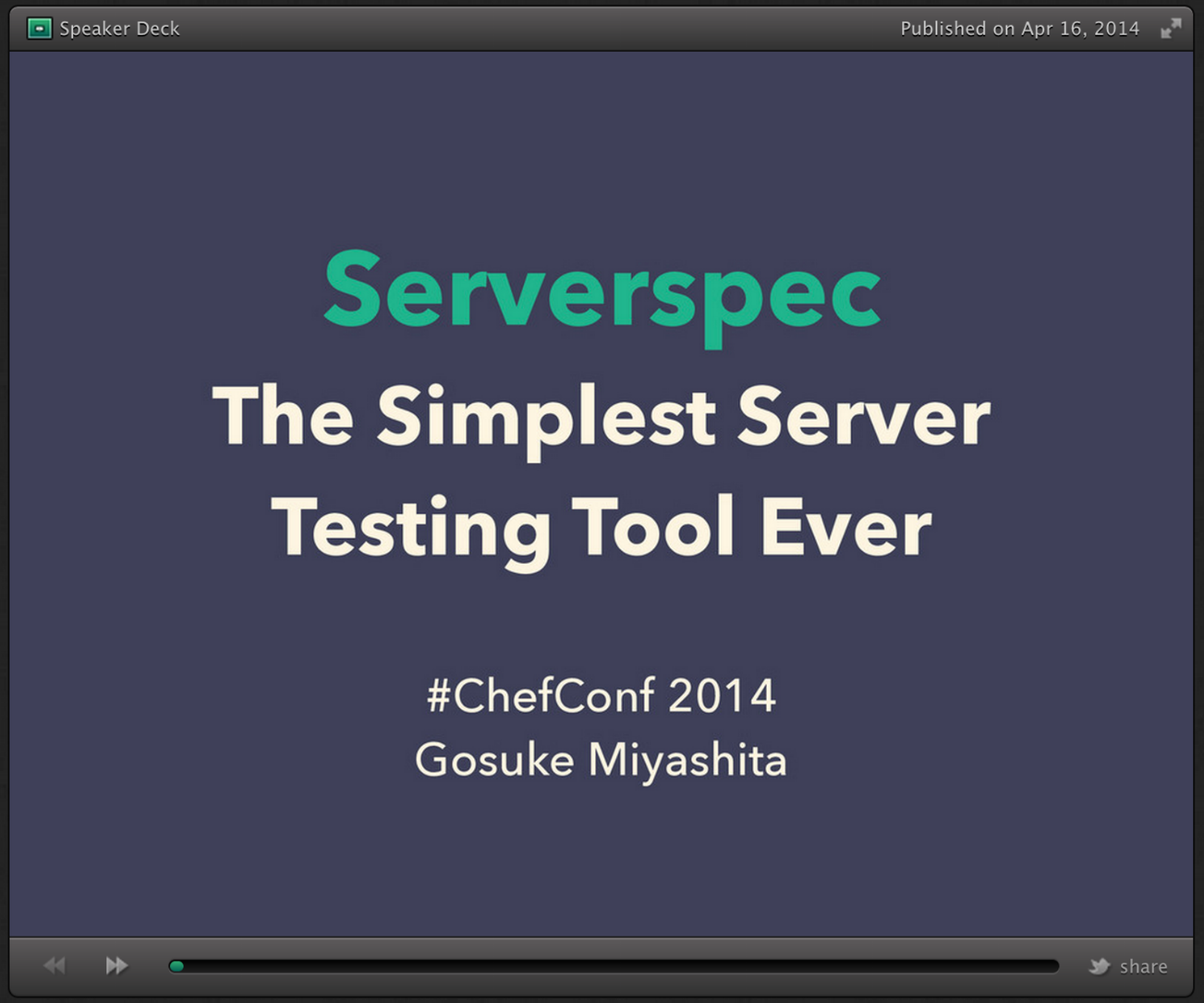
Das lässt sich ganz gut mit serverspec beschreiben.

Testing server infrastructure with #serverspec

by Andreas Schmidt

Published April 10, 2014 in Technology

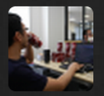
Andreas Sch...
2 Presentations



Serverspec: The Simplest Server Testing Tool Ever

by Gosuke Miyashita

Published April 16, 2014 in Technology



Gosuke Miya...
7 Presentations


TOKYO
Adam Jacob

Follow

Serverspec and Sensu - Testing and Monitoring collide

6,779 views

by m_richardson
on Apr 17, 2014

Lightning talk from Sydney DevOps Meetup

Related | More

Sensu @ Yelp!: A Guided Tour
715 views

Open Source Monitoring Tools
10194 views

Sense and Sensu-bility: Painless Metrics And Monitoring In The Cloud with Sensu
8316 views

Sensu at nycdevops Meetup
1690 views

OSDC2014: Testing Server Infrastructure with #serverspec
182 views

Puppet meetup testing
422 views

Testing Your Automation Code (Vagrant Version)
189 views

What is Test Kitchen
93 views

Sensu
7862 views

なぜServerspecは成功したのか？

能動的な要因

別領域で成功しているプラクティスを持ち込む

- ソフトウェア開発の世界のプラクティス
  - テスト駆動開発
  - 継続的インテグレーション
- 同じ用語使うとイメージもしやすい
  - テスト駆動インフラ/インフラCI

名前重要

- **Server + RSpec**という直球な名前
- 名前から何をするものなのかが類推しやすい
- 覚えやすい

英語ドキュメント

- 日本国外でも広く認知してもらおうと思ったら、これは大前提
- **README**や**serverspec.org**などは最初から英語で書いた

メジャーなキーワード/プロダクトにのっかる

- **Puppet**や**Chef**といったワードを散りばめる
  - 実際に関連するツールだし
- **Puppet**や**Chef**の中の人に拾われて、彼らがツイートすることで広まった感

ドキュメントのわかりやすさ

- 英語でドキュメントを書いて、メジャーなワード散りばめてリーチャビリティ高めるだけではまだ不十分
- ざっと斜め読みしただけで何ができるツールなのかすぐにわかるようにするのが重要
- すぐにわからないと、興味を失って離脱され、二度と戻らないかもしれない

ツール自体のシンプルさ

- ドキュメントをわかりやすくするためには、ツールそのものがシンプルでわかりやすい必要がある
- **Serverspec**はテストのみに特化した
  - 他のツールにある構成管理ツールとの連携や**VM**の操作等は省いた
- ゆえにシンプルでわかりやすい

シンプルだけど拡張しやすい

- 基本的な機能はあまり増やさない方針
- ただしプログラムを書いて拡張しやすくしてる
- 他のツールとも組み合わせやすい
- **UNIX哲学**
  - **Keep It Simple, Stupid**

他ツールとの組み合わせ

- busser-serverspec
- beaker-rspec
- vagrant-serverspec
- sensu-community-plugins/plugins/serverspec

エージェントレス

- いくら便利でも、サーバに色々入れるのは嫌ですよね？
- **Serverspec**はテストを実行するマシンに**Ruby**があれば**OK**
- テスト対象のマシンには**SSH**でアクセスできれば**OK**
  - **Ruby**すら必要ない

使うまでの敷居を下げる

- ServerspecはRuby 1.8.7以降をサポート
- RedHat 6系のパッケージインストールで入るRubyに対応
  - 別途Rubyをダウンロードしてコンパイル、などする必要がない
- serverspec-initコマンド
  - コマンド一発でひな形を生成してすぐに試せる

自分一人でやりすぎない

- 自分が使う機能しか基本的には実装しない
  - シンプルに保つうえでも大事
- 欲しい機能があれば自分で実装してくれ、というスタンス
  - **OSS**なので
  - 結果として多くの人を巻き込めた

能動的要因まとめ

- 他領域のプラクティスを持ち込む
- 名前やドキュメント重要
- メジャーなプロダクトに乗っかる
- シンプルかつ拡張可能
- 使う敷居を下げる
- 他者を巻き込む

技術トレンド要因

サーバ構成管理ツールの成熟

- **Puppet**登場から**9**年
- **Chef**登場から**5**年
- みんながその次を見据えだした

Vagrantの普及

- テストを実行するためには手元でVMを動かせる方が捗る
- Vagrantにより手元でテストコードを書いて実行する環境がつくりやすくなった

Chef界隈でのテストの盛り上がり

- 元々Chef界隈の人たちはテストに対する意識が高かった
- ChefSpec, Test Kitchen, Cucumber-Chef, minitest-chef-handlerなどテスト系ツールはChef界隈が充実

JenkinsやCI as a Serviceの普及

- Jenkinsが開発の世界では当たり前のように使われている
- CI as a Serviceの普及
  - Travis, CircleCI, Drone, Werckerなど

GitHubの普及

- GitHubでインフラコードを管理
- プルリクエストベースで変更
- CI as a Serviceとの連携
- テスト駆動インフラ/CIと相性が良い

技術トレンド要因まとめ

- 構成管理+仮想化+テスト+**CI**というトレンド
- **GitHub**がこのトレンドを更に後押し
- 仮想化、構成管理、**CI**については定番がでてきた
- テストツールは決定版がなかった
- そこに**Serverspec**がはまった
- それによりこのトレンドを更に加速させることになった

まとめ

Serverspecの成功要因

- 他領域で成功しているプラクティスを持ち込む
  - テスト駆動開発/継続的インテグレーション
- 最初から普及を意識した活動
  - 名前、ドキュメント、メジャープロダクトにのっかる
- シンプルかつ拡張可能に
  - **UNIX哲学/Keep It Simple, Stupid**

Serverspecの成功要因

- 敢えて隙をつくって人を巻き込む
- トレンドにのっかる
  - 構成管理+仮想化+テスト+CI
  - テストツール以外の定番は既にあった
  - 隙間を埋めたのがServerspec

Serverspecの成功要因

- 影響力のある人にリーチする
  - 国内では伊藤直也氏のブログエントリや雑誌記事の影響が大きそう

Serverspecの成功要因

- 技術トレンドを生むとか考えない
  - 生もうと思って生めるものでもない
- 自分が欲しいもの、おもしろいと思うものを自分のためにつくる
  - 人類の技術発展ための無償開発、というスタンスではいいものはできない

まとめ

- 技術トレンドを生み出す近道はない
  - 少なくとも自分は知らない
- ただ自分が欲しいもの、おもしろいとおもうものをつくる/公開する
- その積み重ねが素晴らしいものを生むことにつながる
  - 自分がつくって公開したものは、小さなもの含めて100以上はある

Shut the fuck up and write some code

Openness is our driver for excellence

知之者不如好之者好之者不如樂之者