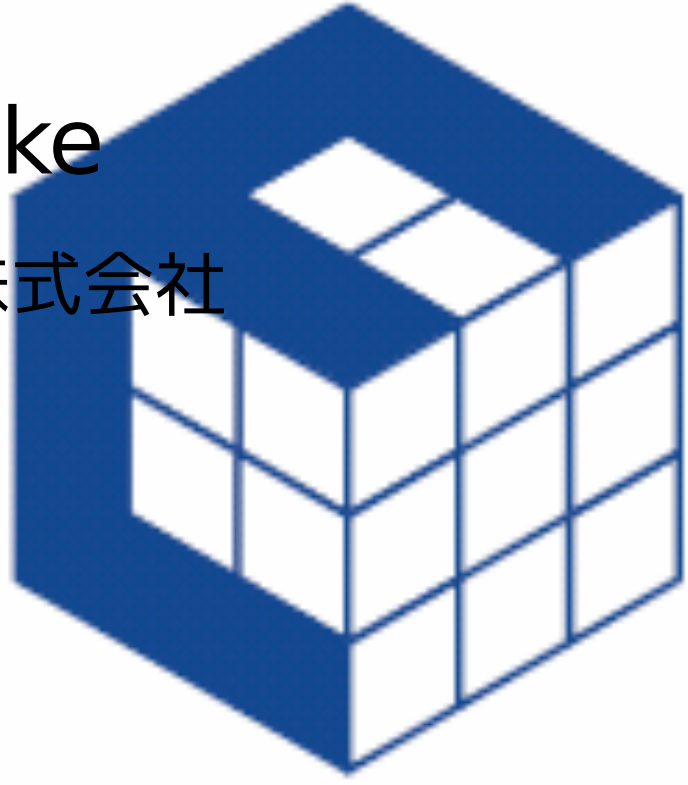
ChefとServerspecでテスト駆動インフラ開発

HIGUCHI Daisuke

クリエーションライン株式会社

2015/10/19



アジェンダ

- テスト駆動インフラ開発とは
- Excel手順書 Considered Harmful
- あるプロジェクトでのExcel手順書
- テスト駆動インフラ開発の実践
- コードの実例
- 感想・あるあるネタ
- まとめ

テスト駆動インフラ開発とは

テスト駆動開発とは

- まずテストを書く
- テストを満たす本体コードを書いて実行
- 本体コードを洗練していく

この手法を「インフラ」に適用すると...

- Serverspecのテストを書く
- テストを満たすChef Cookbookを書いて実行
- Chef Cookbookを洗練していく

Excel手順書 Considered Harmful

Excel手順書はなぜ「有害と考えられる(Considered Harmful)」か※多少誇張あり

- 更新されても差分が取れない → 要**人の目**
  - テキストなら diff が取れるのに
- 変更履歴が取れないのでファイル名に日付をつけたり「最新」とか「古い」とかいうディレクトリを作る、しかもしばしば更新されない → 要**新旧の判断基準**
  - テキストなら VCS に入れられるのに

あるプロジェクトでのExcel設定手順書

まずExcelで設定手順書を作る

2.システム情報
※デフォルトからの変更箇所は水色で表記
2.1.言語設定

(1)設定言語

項目	値	設定ファイル	設定値	備考
システム言語	日本語	/etc/sysconfig/i18n	LANG="ja_JP.UTF-8"	

(2)言語パッケージ

項目	値	パッケージ名	備考
追加言語パッケージ	日本語環境パッケージ	Japanese Support	yum groupinstall

これに従って**手動・目視で**サーバを設定する

- 書き間違えちゃった…
- コピペミスしちゃった…
- 面倒くさくなってきた… → **集中力低下による悪循環**

あるプロジェクトでのExcel確認手順書

次にExcelで確認手順書を作る

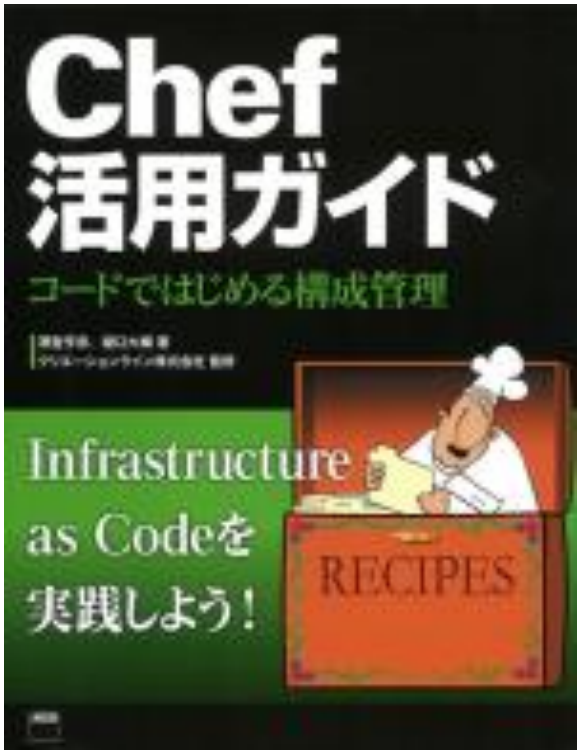
17	Bonding設定(Public)の有効化の確認	cat /proc/net/bonding/bond1	MII Status: up
18	Bonding設定(Private)の有効化の確認	cat /proc/net/bonding/bond0	MII Status: up
19	アップデートしたカーネルで起動しない設定	cat /etc/sysconfig/kernel	UPDATEDEFAULT=no
20	言語環境	echo $LANG	ja_JP.UTF-8
21	ブートローダーパスワード	grep password /boot/grub/grub.conf	空
22	SELINUX無効化	getenforce	Disabled
23	ランレベル	runlevel	5

これに従って**手動・目視で**サーバを確認する

- 確認が漏れてた…
- 見間違えちゃった…
- 面倒くさくなってきた… → **集中力低下による悪循環**

手動設定と確認の代わりにChefとServerspecで自動化しよう！

Chef活用ガイド



クリエーションライン株式会社 監修
澤登亨彦、樋口大輔 著

テスト駆動インフラ開発の実践

- Excel確認手順書からServerspecテストを起こし、設置直後のサーバにかける → すべて**失敗**
- Excel設定手順書からChef Cookbookを起こす
  - Test KitchenでServerspecテストをかけ、Chef Cookbookにバグがないことを確認
- Chef Cookbookをサーバに適用し、Serverspecテストをかける → すべて**成功**ならば、OK！
  - もし失敗したら、ServerspecテストとChef Cookbookのどちらか/両方を修正して擦り合せ
    - すべて**成功**になるまで繰り返す

Serverspec 実例(1/2)

```
shared_examples_for 'check /etc/sysconfig/i18n' do
  describe file( '/etc/sysconfig/i18n' ) do
    its( :content ) { should match %r|^LANG="ja_JP.UTF-8"$| }
  end
end
```

ファイル /etc/sysconfig/i18n が文字列
LANG="ja_JP.UTF-8" を含むか確認

- 自動ならファイルを間違えない！
- 自動なら文字列を間違えない！

Serverspec 実例(2/2)

```
shared_examples_for 'check cpu core num' do |cpu_cores|
  host_inventory['cpu']['total'].to_i.times do |i|
    describe host_inventory['cpu'][ i.to_s ]['cpu_cores'] do
      it { should eq cpu_cores }
    end
  end
end

describe 'check cpu core num' do
  it_should_behave_like 'check cpu core num', '4'
end
```

CPUのコア数を確認する(関数みたいにして引数(**cpu_cores**)も使える)

Chef Cookbook実例(1/3)

```
file '/etc/sysconfig/i18n' do
  owner 'root'
  group 'root'
  mode  '0644'
  content <<-_EOF_
LANG="ja_JP.UTF-8"
SYSFONT="latarcyrheb-sun16"
  _EOF_
end
```

ファイルの内容をベタ貼り

Chef Cookbook実例(2/3)

```
file '/etc/sudoers' do
  _file = Chef::Util::FileEdit.new( path )
  _file.search_file_replace_line(
    %q!^#\s*%wheel\s*ALL=\(ALL\)\s*ALL!,
    %Q!%wheel\tALL=(ALL)\tALL\n!
  )
  content _file.send( :editor ).lines.join
end
```

sed みたいな行置換

Chef Cookbook実例(3/3)

```
# attribute
default['service']['enabled'] = %w(
        nimbus
        zabbix-agent
)

# recipe
node['service']['enabled'].each do |s|
  service s do
    action [ :enable, :start ]
  end
end
```

引数を取ってのサービスのオンオフ

Test Kitchen 超いかす

- 楽々スクラップ＆ビルドとトライ＆エラー
- bonding設定などネットワークのテストもできる
- VM起動など待ち時間は慌てず騒がず**休憩時間**
- Chef Cookbookのテスト以外に、テスト環境そのものの立ち上げにも使える(スクラップしない)
- デフォルトのVagrant+VirtualBoxの代わりにDockerをバックエンドにもできる

kitchen-verifier-shell 超いかす

Chef CookbookのTest Kitchenでのテストは

- busser-serverspec (作:d-higuchi)より
- kitchen-verifier-shell (作:sawanoboly)で

busser-serverspecだとChef CookbookとServerspecの結びつきが強すぎる

- コードを一体化しないといけない
- TK内で両者を橋渡しするBusserは融通効かない

kitchen-verifier-shellなら両コードを分離可能

- シェル叩くだけなので仲介者のBusserが不要

よくあるテストの問題

人は**見たいものしか見ない**

- 入力テストOK → どんな入力でもOKの欠陥テスト
  - 正規表現マッチングはミスの宝庫
- ユニットテスト全OK → 結合した本番で失敗
  - あるユニットの動作が他ユニットに影響
- Test KitchenならOK → 本番で失敗
  - TKのネットワークと本番ネットワークの差異

OKになるようにテストを書き、**OKだからと安心**してしまう → 本末転倒、油断・慢心

まとめ

- 未来の楽のために、今は多少の苦労をしよう
  - Excel手順書ではなくMarkdownやreSTなど「よりマシンリーダブルでシンプルなフォーマット」
  - Chef CookbookやServerspec「習うより慣れろ」
- 本体もテストもコードは人間が書くから完全ではない
  - だからって最初からテストしないのは**論外**
  - 仕様・確認と実際の本体・テストのコードのカバレッジを保証する方法はないか？ → 検討課題

Any Questions?