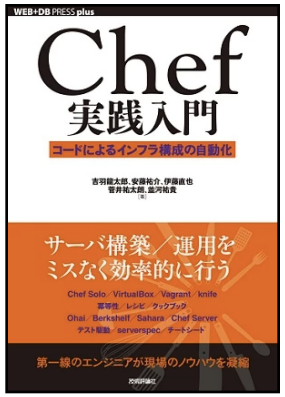
Chef ベーシックトレーニング

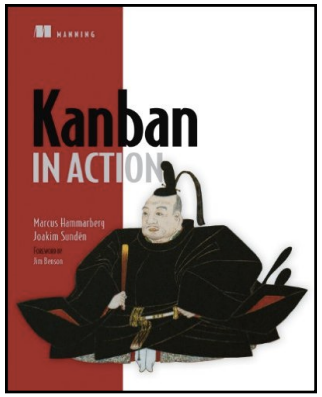
2016年4月

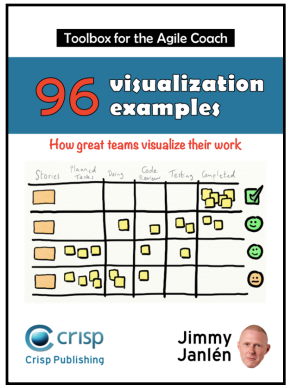
RYUZEE.COM

RYUZEE.COMについて

- アジャイル開発/DevOps/クラウドに関する従量課金型コンサルティングサービスを提供
- http://www.ryuzee.com @ryuzee







本トレーニングの対象 / 想定

- [対象] インフラ構築自動化の経験があまりない方
- [対象] とりあえず簡単にChefがどんなものか触って確認したい方
- [対象外] ガッツリとハンズオンで色々試したい方 (Chef社のサイトなどを参照して自習してください)
- [想定] 所要時間を2-3時間に設定したコンテンツになっており、全てはカバーしていません。より詳細を学習したい場合はChef社のサイトや関連書籍をご利用ください

更新履歴

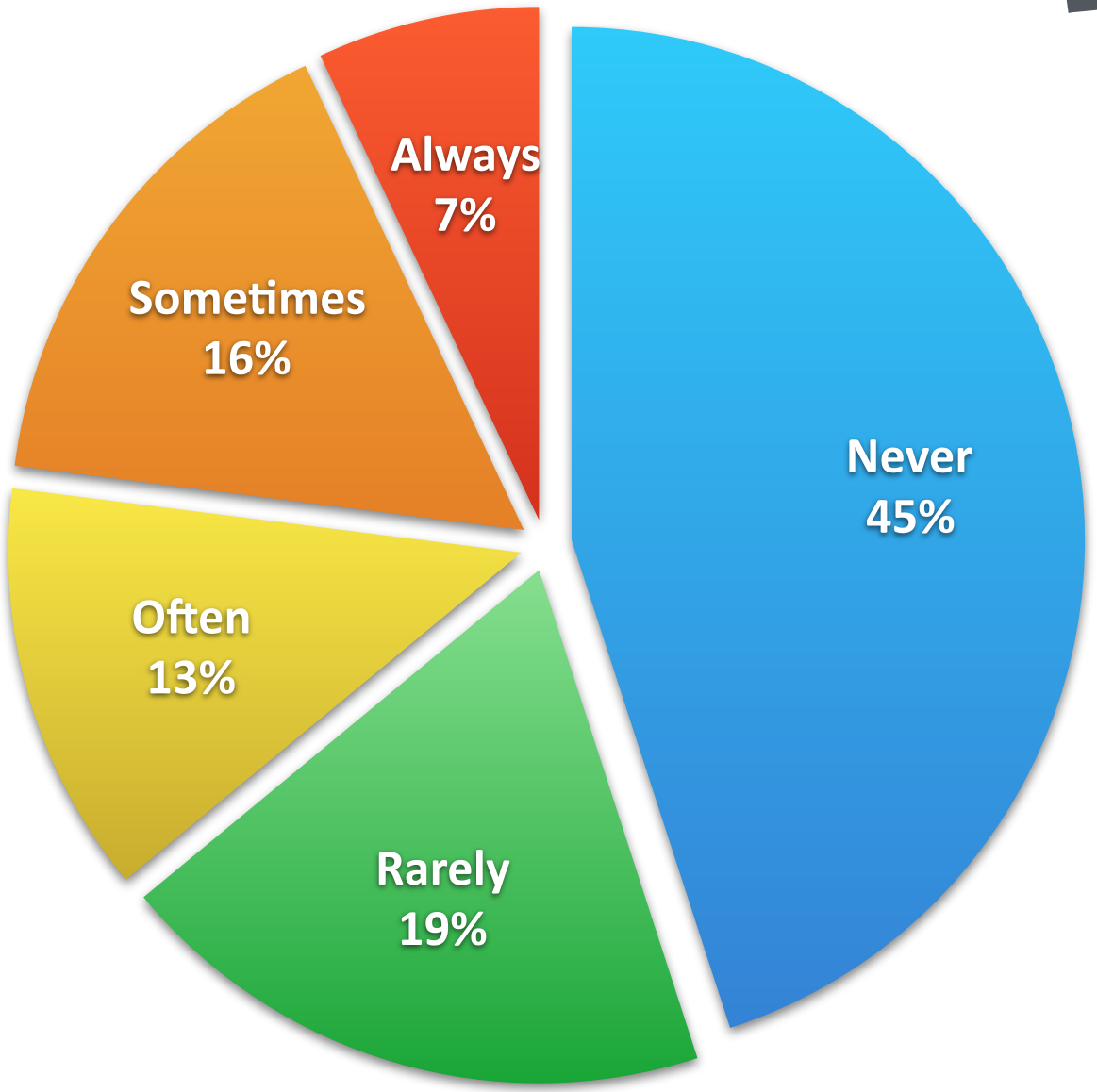
- 2016/4/9 Attribute/Role/Environmentの説明を追加
- 2016/4/3 Vagrant1.8系でスナップショット機能がついたためSaharaのインストールを廃止
- 2016/4/1 初版作成

DevOpsとは何か?

現在の状況

- ビジネスの変化がどんどん早くなっている
- ITがビジネスの鍵に
- 良いソフトウェアを顧客に届けることはビジネスにとって必須
- 一方、特にトラディショナルな企業や大企業では、デリバリの速度はとても遅く、間違いやすい方法でやっている
- これはビジネス機会や売上の損失に繋がる
- 他にも問題がおこるかもしれない
- ITがボトルネックになることがよくある

ムダ



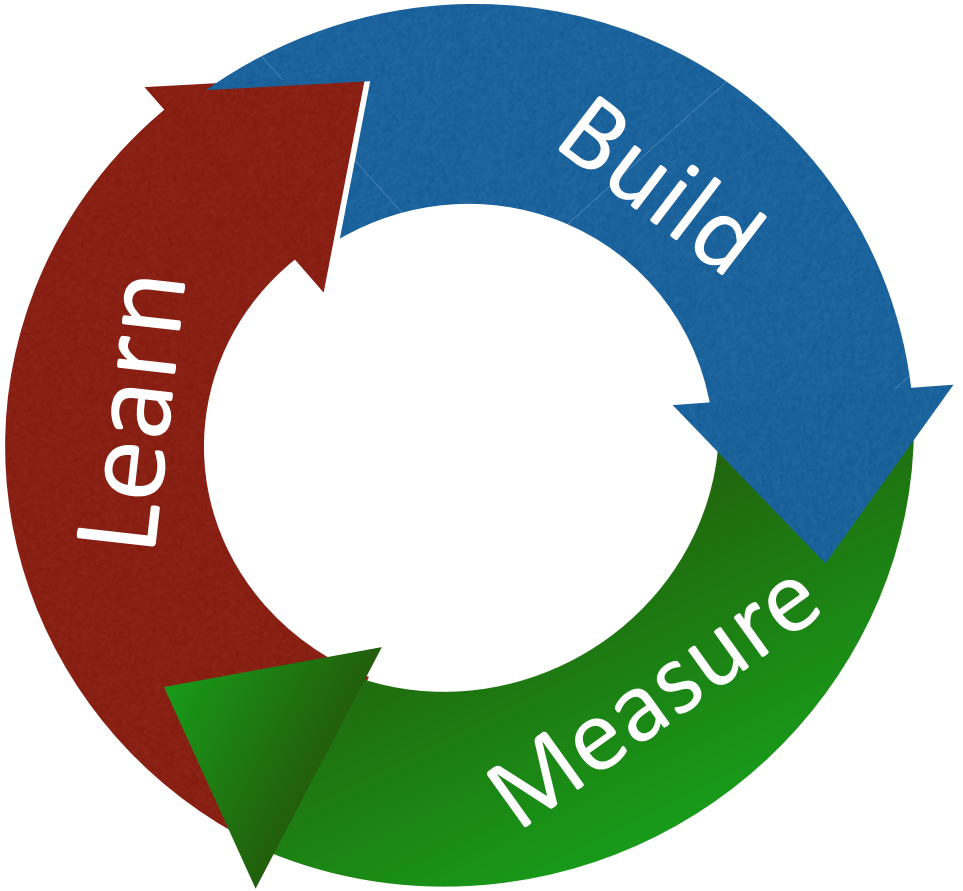
- ソフトウェアの機能のうち64%は、ほとんど、もしくは全く使われていない (Standish Group “Chaos” report 2002).
- 自分のパソコンにプリインストールされているソフトウェアを考えればイメージがつくはず

7つのムダ

- 作りすぎのムダ
- 手待ちのムダ
- 運搬のムダ
- 加工のムダ
- 在庫のムダ
- 動作のムダ
- 不良をつくるムダ

何が正しいのか、未来に何がおこるのかを事前に知るのは難しい。
早い変化に対応できる能力があることが重要。

素早いフィードバックサイクルをつくる



DevとOpsのコンフリクト

- 役割や責任の違い (誰がそれを決めたのか、思い込みでは？)
- 「それは自分の仕事ではありません」 (そんなこと言うのは誰だ？)
- サイロ化
- **これらがオーバーヘッドをつくりだしビジネスの速度を遅くする**

- OpsはDevのように考える
- DevはOpsのように考える

Werner Vogels, CTO, amazon.com

You build it, You run it

DevOpsが目指すところ

- DevOpsが意図するのは、文化とツールを活用して、ビジネス上の結果を達成すること、ビジネスのアジリティをあげること、そしてリスクを減らすこと。

DevOpsの5つの側面

- Culture (文化)
- Lean (リーン)
- **Automation (自動化)**
- Measurement (測定)
- Sharing (共有)

Infrastructure as Codeがなぜ必要なのか？

手作業によるプロビジョニングの課題

- 対象サーバが増えるとそれだけプロビジョニングの時間が増える
- 手順書やチェックリストがメンテナンスされない・最新に保たれない
- 手順書を使って手作業しても間違える
- ドキュメントは他のプロジェクトで再利用しにくい

Infrastructure as Codeの利点

- プロビジョニングにかかる時間は台数が増えてもほぼ一定
- コード＝手順書となるのでコードだけを最新に保てばよい
- コードは書いたとおりに動作する。同じコードを使えば同じサーバができあがる（ように書く）
- コードはCIツールを使って継続的にテストできる
- 再利用性が高い

どのようにインフラ構築を自動化するか

自動化のいくつかの方法

- シェルスクリプト
- Capistranoなどのデプロイツール
- Chef / Puppet / Ansibleなどのプロビジョニングツール

シェルスクリプト

```
#!/bin/sh
yum install -y httpd httpd-devel php php-mbstring php-pdo php-mysql mysql-server
/sbin/chkconfig --level 2345 httpd on
/sbin/chkconfig --level 2345 mysqld on
/etc/rc.d/init.d/mysqld start
/etc/rc.d/init.d/httpd start
```

- 一番単純なやり方
- ただし制御構文が多数含まれていると保守しにくくなるという課題がある

デプロイツール(Capistrano)

```
task :install_amp, roles => :web do
  run <<-CMD
    sudo yum install -y httpd httpd-devel php php-mbstring php-pdo php-mysql mysql-server &&
    sudo /sbin/chkconfig --level 2345 httpd on &&
    sudo /sbin/chkconfig --level 2345 mysqld on &&
    sudo /etc/rc.d/init.d/mysqld start &&
    sudo /etc/rc.d/init.d/httpd start
  CMD
end
```

- Capistranoやデプロイツールは通常アプリケーションのデプロイ用に作られている
- インフラ構築に利用することはもちろん可能だが、シェルスクリプトを埋め込むような形では大した利点なし

プロビジョニングツール(Chef)

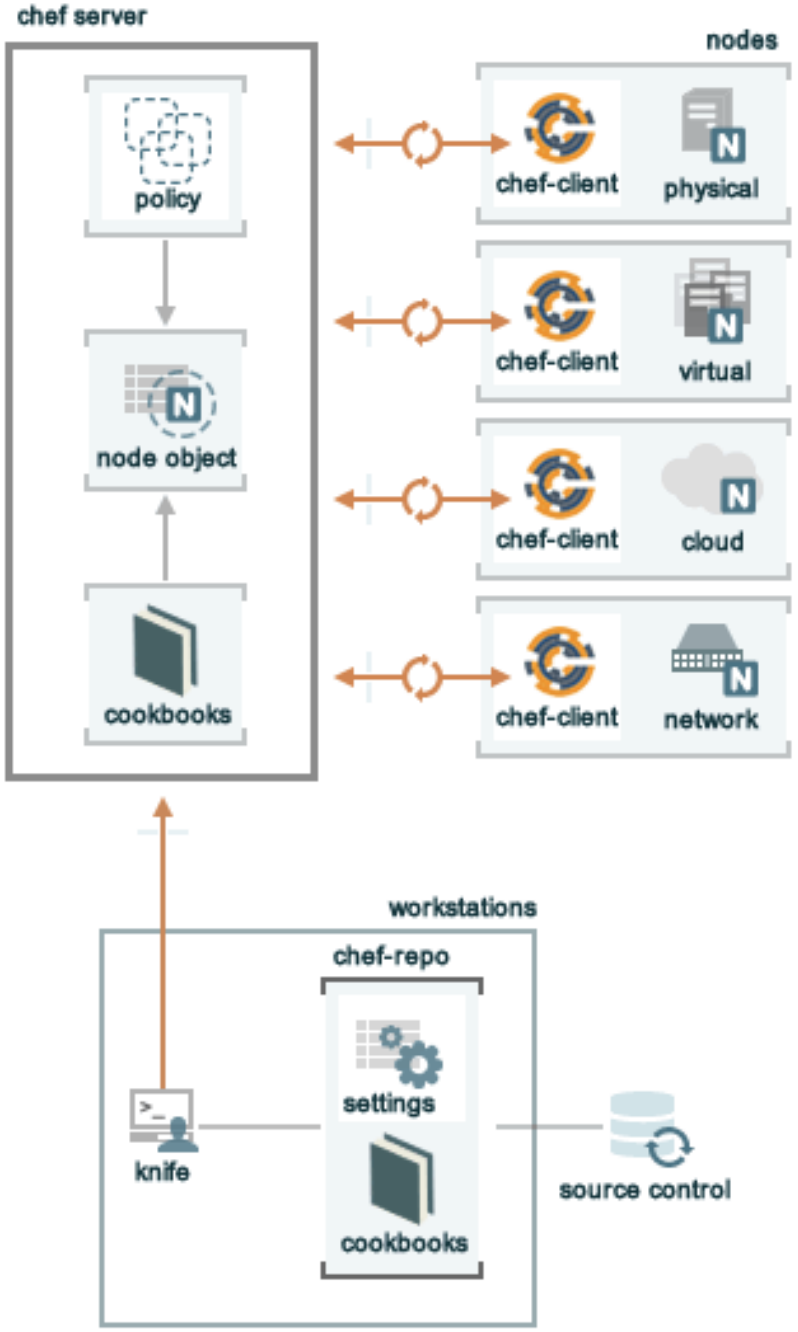
```
%w{httpd httpd-devel php php-mbstring php-pdo php-mysql mysql-server}.each do |p|
  package p do
    action :install
  end
end

service "httpd" do
  action [:enable, :restart]
  supports :status => true, :start => true, :stop => true, :restart => true
end

service "mysqld" do
  action [:enable, :restart]
  supports :status => true, :start => true, :stop => true, :restart => true
end
```

プロビジョニングツールの比較

Chef	Ansible	Puppet
DSL (**Rubyベース)**	DSL	DSL
Client / Server (Agent)	Agentless	Client / Server (Agent)
knife, berkshelf, foodcriticなどの多数の関連ツールあり	管理用のツールがすくない	かなり前に開発されたもので既に非主流といえる
DSL(Rubyベース)の学習が必要	単純で学習すべきことは少ない	独自DSLを学習する必要あり

- Rubyで書かれた多くのインフラ関連のツールがあるため、インフラエンジニアでもRubyを書けるようになっておくと良い(Go言語も主流化しつつある)

Chefのアーキテクチャ



アーキテクチャ

- 基本的には **Client / Server** 型
- 全ての情報はChef Serverに保存される。全てのノードはChef Clientを経由してChef Serverにアクセスし、それによってCookbookやAttributeなどを取得し適用する

基本的な用語 #1

- **Chef Server** => Chef Serverはハブのような形で、Cookbookやポリシーなどを保存するとともに、各ノードのメタデータも保存する
- **ノード** => ノードとは物理や仮想やクラウドやネットワーク機器のマシンを指す。Chefで管理するにはChef Clientのインストールが必要
- **Chef Client** => ノードにインストールするツールで、サービスとして動作させることもコマンドラインツールとして動作させることも可能

基本的な用語 #2

- **Cookbook** => Cookbookはプロビジョニングに必要なファイル一式をまとめたもので、レシピ(Recipe)、アトリビュート(Attributes)、テンプレート(Templates)、拡張などを含む。この単位で配布したり管理する
- **レシピ** => 対象のノードを設定するためのDSL(Rubyベース)のコード。1つのCookbookには複数のレシピを含むことができる

注意: Chef Solo

- Chefには、Chef Soloと呼ばれる非Client/Server型のモードもあります (ありました)
- が現時点では非推奨となっており今後廃止される可能性があります
- Chef ServerなしでChefを実行したい場合は、**Chef Local Mode** を使うことになります。そのためのツールとしてknife-zeroが利用可能です
- 多くのWebサイトや書籍にChef Soloの記述がありますが、上記について認識しておくようにしてください

ハンズオンの環境

環境の全体像

ホスト名: development

IPアドレス: 192.168.33.10

OS: Ubuntu 14.04
Chef-DK / Docker / vi / vim / emacs

この環境にログインして作業

Development

ホスト名: node01

IPアドレス: 192.168.33.200

OS: Ubuntu 14.04
Chef Clientインストール済み

この環境に適用される

Node01

Vagrant: HashiCorp提供のオープンソースの開発環境管理ツール

VirtualBox: Oracle提供のオープンソースの仮想化ツール

自分の端末

VirtualBox

search...
Login Preferences

Welcome to VirtualBox.org!

- About
- Screenshots
- Downloads
- Documentation
  - End-user docs
  - Technical docs
- Contribute
- Community

VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. See "About VirtualBox" for an introduction.

Presently, VirtualBox runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD.

VirtualBox is being actively developed with frequent releases and has an ever growing list of features, supported guest operating systems and platforms it runs on. VirtualBox is a community effort backed by a dedicated company: everyone is encouraged to contribute while Oracle ensures the product always meets professional quality criteria.

Download VirtualBox 5.0

News Flash

- **New March 4th, 2016**
  VirtualBox 5.0.16 released!
  Oracle today released a 5.0 maintenance release which improves stability and fixes regressions. See the Changelog for details.
- **New January 19th, 2016**
  VirtualBox 4.3.36 released!
  Oracle today released maintenance releases which improve stability and fixes regressions. See the Changelog for details.
- **New July 9th, 2015**
  VirtualBox 5.0 released!
  Read the official press release for details.
- **Important February, 2015**
  We're hiring!
  Looking for a new challenge? We're looking for generic product developers (Russia).

More information...

Hot picks:

- Pre-built virtual machines for developers at Oracle Tech Network
- **Hyperbox** Open-source Virtual Infrastructure Manager project site
- **phpVirtualBox** AJAX web interface project site
- **IQEmu** automated Windows VM creation, application integration http://mirage335-site

ORACLE

Contact – Privacy policy – Terms

https://www.virtualbox.org/
よりダウンロードしインストール

https://www.vagrantup.com/
よりダウンロードしインストール。
既にインストール済みの場合は1.8以上で
あることを確認し必要なら更新する

Development environments made easy.

Create and configure lightweight, reproducible, and portable development environments.

DOWNLOAD GET STARTED

Vagrantの基本

- **Vagrantは開発環境の構築・管理ツールの中でもっとも有名なオープンソース製品。HashiCorpが中心となって開発が進んでいる**

- VagrantではVirtualBoxの仮想マシン、Dockerのコンテナ、AzureやAWSの仮想サーバなどを扱える。Vagrantがそれぞれの差を吸収していると考えると分かりやすい

- Vagrantfileが環境を定義したもの。同一のVagrantfileからは同一の環境ができる。チームで共用すれば全員同じ環境が手に入るので開発する上でメリットが非常に大きい

Vagrantfileの例

```
Vagrant.configure(2) do |config|

  config.vm.define :development do |development|
    development.vm.box = 'ubuntu-14.04.4-chef-training-development-kit'
    development.vm.hostname = 'development'
    development.vm.network 'private_network', ip: '192.168.33.10'
  end

  config.vm.define :node01 do |node01|
    node01.vm.box = 'ubuntu-14.04.4-chef-training-node'
    node01.vm.hostname = 'node01'
    node01.vm.network 'private_network', ip: '192.168.33.200'
  end
end
```

http://bit.ly/224TbdH
よりダウンロード可能

Vagrantの組み込みコマンド例

boot virtual machines
vagrant up

boot specified machines
vagrant up development

login to the specific machine
vagrant ssh development

reboot machines
vagrant reload [machine name]

stop machines
vagrant halt [machine name]

dispose machines
vagrant destroy [machine name]

add box as a template
vagrant box add box_name box_url

install plugin
vagrant plugin install plugin_name

Boxの追加

- ターミナルまたはコマンドプロンプトで以下のようにboxを追加します

```
vagrant box add ubuntu-14.04.4-chef-training-development-kit http://bit.ly/1W4FWtV
vagrant box add ubuntu-14.04.4-chef-training-node http://bit.ly/1PQjMEl
```

全ての準備が終わったら…

```
vagrant up
```

- Vagrantfileを配置したディレクトリにて、上記コマンドをターミナルまたはコマンドプロンプトにて実行
- 仮想マシンが2台立ち上がるはず
- 起動に失敗した場合はbox名のタイポなどエラー出力を確認します

最初のCookbookを作ってみる

nginxのインストールを自動化する

- nginx [engine x] は(言うまでもなく)、HTTPおよびリバースプロキシサーバのプロダクト。Apache HTTP Serverよりも性能面で優位
- これからChef経由でnginxをインストールします
- まずは、”development”環境にログインします。ターミナル(OS X)またはコマンドプロンプト(Windows)で “**vagrant ssh development**”と入力します

開発環境へのログイン

- 以下のようなメッセージが表示されることを確認してください
- これでUbuntuの仮想マシン内の開発環境にいることになります
- ユーザー名はvagrantでカレントディレクトリは/home/vagrant

```
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Fri Apr  1 20:54:29 2016 from 10.0.2.2
vagrant@development:~$
```

gitの設定

- ハンズオンを始める前に以下のようにgitの設定をします

```
git config --global user.name “Sushi Taro”
git config --global user.email taro@example.com
```

Repositoryの作成

- 最初にCookbookやノード情報を保存するためのRepositoryを作成します。以下のコマンドを実行します

```
chef generate repo chef-repo
cd chef-repo
```

コマンドの実行結果

```
Installing Cookbook Gems:
Compiling Cookbooks...
Recipe: code_generator::repo
  * directory[/home/vagrant/chef-repo] action create (up to date)
  * template[/home/vagrant/chef-repo/LICENSE] action create_if_missing (up to date)
  * cookbook_file[/home/vagrant/chef-repo/.chef-repo.txt] action create_if_missing (up to date)
  * cookbook_file[/home/vagrant/chef-repo/README.md] action create_if_missing (up to date)
(snip)
 Recipe: code_generator::repo
  * cookbook_file[/home/vagrant/chef-repo/cookbooks/README.md] action create_if_missing (up to date)
  * execute[initialize-git] action run
    - execute git init .
  * template[/home/vagrant/chef-repo/.gitignore] action create_if_missing
    - create new file /home/vagrant/chef-repo/.gitignore
    - update content in file /home/vagrant/chef-repo/.gitignore from none to 3523c4
    (diff output suppressed by config)
```

ディレクトリ構造

- “tree -L 2”コマンドを実行すると右図のようなディレクトリ構成が表示されればOKです
- この中にcookbooksディレクトリがあることが分かります。ここにcookbookを保存します

```
vagrant@development:~/chef-repo$ tree -L 2
.
|-- chefignore
|-- cookbooks
|   |-- example
|   `-- README.md
|-- data_bags
|   |-- example
|   `-- README.md
|-- environments
|   |-- example.json
|   `-- README.md
|-- LICENSE
|-- README.md
`-- roles
    |-- example.json
    `-- README.md

6 directories, 9 files
```

nginx cookbookの作成

- “**knife cookbook create nginx -o ./cookbooks/**“ コマンドを実行してください。これでcookbookのディレクトリや必要なファイルを自動で生成します
- 右図のような構造になっていることを確認してください

```
vagrant@development:~/chef-repo$ tree -F 2 ./cookbooks/nginx/
2 [error opening dir]
./cookbooks/nginx/
|-- attributes/
|-- CHANGELOG.md
|-- definitions/
|-- files/
|   `-- default/
|-- libraries/
|-- metadata.rb
|-- providers/
|-- README.md
|-- recipes/
|   `-- default.rb
|-- resources/
`-- templates/
    `-- default/

10 directories, 4 files
```

cookbookの実装

- ./cookbooks/nginx/recipes/default.rbを好きなエディタで開き、右図のテキストを入力します
- これは**nginxのパッケージをインストールし、サービスを有効化・起動することを意味しています**

```
package 'nginx' do
  action :install
end

service 'nginx' do
  action [ :enable, :start ]
end
```

Chef Clientの設定

```
knife zero bootstrap 192.168.33.200 -x vagrant --sudo --ssh-password vagrant
```

- このコマンドによって設定対象のノードにChef Clientをインストールしつつ必要な情報を取得して保存します
- コマンド実行後、“nodes”と“clients”ディレクトリが/home/vagrant/chef-repo/ 以下に作成されていることを確認してください

対象ノードの登録確認

```
knife node list -z
```

- 上記のコマンドを実行して“node01”が表示されればOKです
- 複数のノードを扱っている場合はノード数分表示されます

```
knife node show node01 -z
```

- 上記のコマンドを実行して“node01”の詳細情報を表示できます

run_listの設定

```
knife node run_list add node01 'recipe[nginx]' -z
```

- このコマンドは、node01という名前のノードにnginxクックブックのdefaultのレシピを適用するように設定することを意味しています。
- このコマンドによって“nodes/node01.json”のファイルが更新されます。ファイルの末尾のrun_listに追加されていることを確認してください
- 同時に複数のレシピを適用することももちろんできます

VagrantでSnapshotの取得

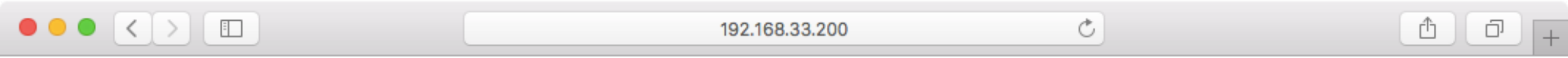
```
vagrant snapshot save node01 node01_001
```

- 上記のコマンドをホスト側(仮想マシン側ではない)で実行します
- これはvagrantの組み込みコマンドであるshanpshotを使って、現在の仮想マシンの断面を保存しています。これによってあとから変更を簡単に戻せるようになります
- 待ち時間を短縮するために、その他いろいろなツールを活用するように普段から工夫していきましょう

対象ノードに変更を適用

```
knife zero converge 'name:node01' -x vagrant --sudo -a knife_zero.host --ssh-password vagrant
```

- 上記のコマンドを実行すると、node01にnginxのレシピが適用されます。Chefでは、このことを**ノードが指定した状態に収束(Converge)した**といいます
- ブラウザで右記のURLにアクセスしてください http://192.168.33.200



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

ここまで大丈夫ですか？

- ここまで正常に動作したのであれば、作成したファイルをバージョン管理します

```
git add *
git commit -m “initial commit”
```

- もしうまくいかない場合はエラーが出力されているはずなので内容を確認します

Template #1

- templateというリソースを使うことで、ファイルを生成して配置することができます。
- index.htmlを変更してみます。“**index.html.erb**”というファイルを“**cookbooks/nginx/templates/default/**”に配置してください。内容は右記の通りです

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
</head>
<body>
<h1>Welcome to Chef Training</h1>
<p><%= node.name %></p>
</body>
</html>
```

Template #2

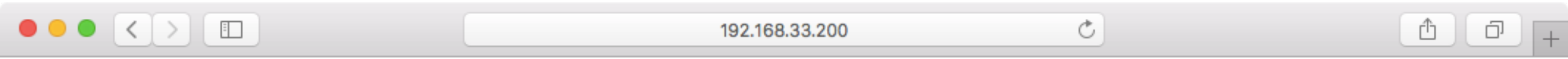
- 次にレシピを修正します
- cookbooks/nginx/recipes/default.rbを開き、右図のハイライトした箇所を追加します
- 収束させる前に、一度**“vagrant snapshot restore node01 node01_001”**として対象ノードの状態を元に戻しておきましょう

```
package 'nginx' do
  action :install
end

template 'index.html' do
  path '/usr/share/nginx/html/index.html'
  owner ‘root’
  group ‘root’
  mode 0644
end

service 'nginx' do
  action [ :enable, :start ]
end
```

対象ノードに変更を適用 (再)

```
knife zero converge 'name:node01' -x vagrant --sudo -a knife_zero.host --ssh-password vagrant
```

- 上記のコマンドを実行すると、node01にnginxのレシピが適用されます。Chefでは、このことを**ノードが指定した状態に収束(Converge)した**といいます
- ブラウザで右記のURLにアクセスしてください http://192.168.33.200



Welcome to Chef Training!!

node01

Resource

- ここまででクックブックの中で、“package”, “service”, “template”というキーワードを使ってきました。これらのキーワードのことをChefでは“**Resource**”と呼びます
- Chefには多くの組み込みResourceがあります(次ページ参照)
- これらのresourceを組み合わせることでcookbookを作成していきます

Resource (抜粋)

package	user	powershell_script	ifconfig
template	group	ruby_block	http_request
service	remote_file	cron	link
file	execute	git	log
directory	script	mount	chai

基本的な用語 #3 / さらに学習すべきこと

- **Attribute** => Attribute(s)とはプロビジョニングの際に使える変数のこと。たとえば、PHPの環境をつくるときにphp.iniの設定項目の値を環境によって変えたいとします。そのとき外部から値の変数を渡すことができます。
- **Role** => ロールを定義することができます。たとえばWebサーバロール、DBサーバロールといった形です。ロールの中には複数のレシピを含めたりAttributeを設定したりできます。ロールを使うことである用途のインフラを単純にロールを設定するだけで構築することができます。
- **Environment** => 例えば開発環境・ステージング環境・本番環境といった複数の環境を定義できます。EnvironmentごとにAttributeの値を変えることもできるので、接続先や通知先といったものを環境ごとに分けられます。

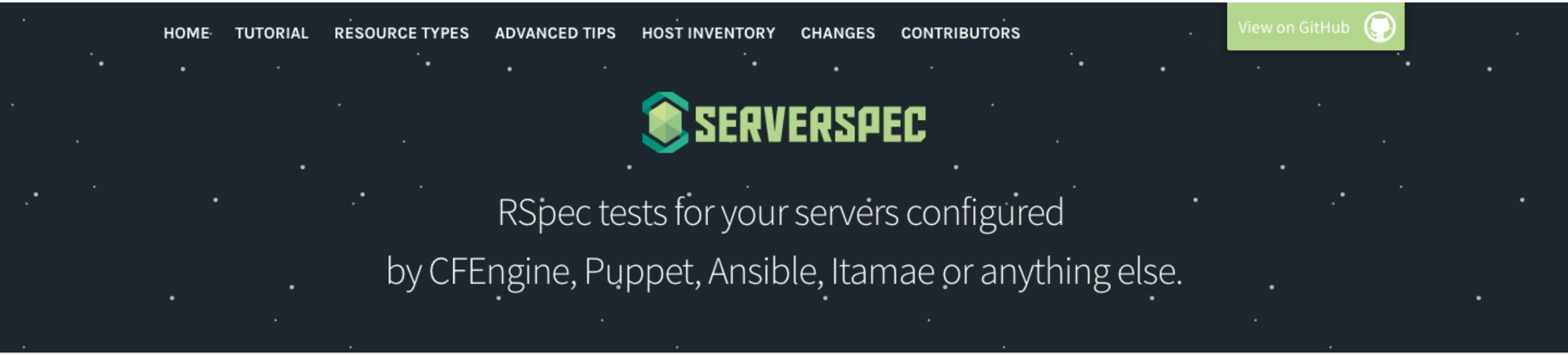
テスト自動化

なぜテスト自動化が大事なのか？

- リスク低減
- 手作業のプロセスを減らす
- いつでもどこでもデプロイ可能にする
- プロジェクトの可視性をあげる
- 信頼性をあげる

Test Kitchen

- Test Kitchenはcookbookを独立した環境で実行しテストするためのテストハーネス
- 詳しくは公式サイトを参照 http://kitchen.ci
- Test Kitchenは独立した環境(例えばVagrant, Docker, AWS, Azure…)を立ち上げ、指定したレシピを適用し結果を評価します
- IBats, shUnit2, RSpec, **Serverspec**などのテストフレームワークが利用できます



About V2

Serverspec/Specinfra v2 has been just released. See the document about v2.

http://serverspec.org/

About

With Serverspec, you can write RSpec tests for checking your servers are configured correctly.

Serverspec tests your servers' **actual state** by **executing command locally, via SSH, via WinRM, via Docker API and so on**. So you don't need to install any agent softwares on your servers and can use any configuration management tools, Puppet, Ansible, CFEngine, Itamae and so on.

But the true aim of Serverspec is to help refactoring infrastructure code.

Installation

準備

- Test Kitchenにはジェネレータが用意されています。以下のコマンドを実行してください

```
cd /home/vagrant/chef-repo/cookbooks/nginx
kitchen init -D kitchen-docker
```

- .kitchen.yml, chefignore, test/integration/defaultなどが作成されます

```
mkdir -p test/integration/default/serverspec/localhost
```

テストの作成

```
require 'spec_helper'

describe package("nginx") do
  it { should be_installed }
end

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

describe port(80) do
  it { should be_listening }
end

describe file('/usr/share/nginx/html/index.html')
do
  it { should be_file }
end
```

test/integration/default/serverspec/
localhost/default_spec.rb

test/integration/default/serverspec/
spec_helper.rb

```
require 'serverspec'

set :backend, :exec
```

テストの意図は何か?

```
require 'spec_helper'

describe package("nginx") do
  it { should be_installed }
end

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

describe port(80) do
  it { should be_listening }
end

describe file("/usr/share/nginx/html/index.html")
do
  it { should be_file }
end
```

- nginxのパッケージがインストールされていること
- nginxのサービスが動作していること
- nginxのサービスがサーバ起動後に起動すること
- nginxがTCP80番ポートでListenしていること
- index.htmlが指定場所にあること

設定ファイルの編集とテスト実行

```
---
driver:
  name: docker

provisioner:
  name: chef_solo

platforms:
  - name: ubuntu-14.04

suites:
  - name: default
    run_list:
      - recipe[nginx::default]
    attributes:
```

- .kitchen.ymlを右記の内容に変更する
- これはUbuntu14のDockerマシンを起動してテストすることを意味します
- “**kitchen test**”と実行するとテストが始まります
- テストには数分時間がかかります

```
Fetching: rspec-mocks-3.4.1.gem (100%)
Fetching: rspec-3.4.0.gem (100%)
Fetching: multi_json-1.11.2.gem (100%)
Fetching: serverspec-2.31.1.gem (100%)
-----> serverspec installed (version 2.31.1)
       /opt/chef/embedded/bin/ruby -I/tmp/verifier/suites/serverspec -I/tmp/verifier/gems/gems/rspec-support-3.4.1/lib:/tmp/verifier/gems/g
pattern /tmp/verifier/suites/serverspec/\*\*/\*_spec.rb --color --format documentation --default-path /tmp/verifier/suites/serverspec

       Package "nginx"
         should be installed

       Service "nginx"
         should be enabled
         should be running

       Port "80"
         should be listening

       File "/usr/share/nginx/html/index.html"
         should be file

       Finished in 1.28 seconds (files took 0.81198 seconds to load)
       5 examples, 0 failures

       Finished verifying <default-ubuntu-1404> (0m13.20s).
-----> Destroying <default-ubuntu-1404>...
       UID                  PID                  PPID                 C                    STIME                TTY                  TIME
       root                 7220                 728                  0                    03:48                ?                    00:00:00
asswordAuthentication=yes -o UsePrivilegeSeparation=no -o PidFile=/tmp/sshd.pid
       root                 7240                 7220                 0                    03:48                ?                    00:00:00
       root                 7914                 7220                 0                    03:48                ?                    00:00:00
       www-data             7915                 7914                 0                    03:48                ?                    00:00:00
       www-data             7916                 7914                 0                    03:48                ?                    00:00:00
       www-data             7917                 7914                 0                    03:48                ?                    00:00:00
       www-data             7918                 7914                 0                    03:48                ?                    00:00:00
       2c172be4204d611f13da8136bfa43b07862fcecc1b481f143f2ac7066d379e2b
```

テスト結果はこのような形になります

Jenkinsを使ってテストを実行する

- CookbookのテストをJenkinsに実施させることはもちろん可能です
- 興味があれば試してみてください
  - なおJDK8 (JDK7はNG)とJenkinsのインストールは以下のように実行します

```
sudo add-apt-repository ppa:openjdk-r/ppa
sudo apt-get update
sudo apt-get install openjdk-8-jdk
wget http://pkg.jenkins-ci.org/debian-stable/binary/jenkins_1.642.4_all.deb
sudo dpkg -i jenkins_1.642.4_all.deb
```

よいCookbookを書くには？

コミュニティCookbook

- Chefには大きなエコシステムがあり、以下のようなCookbookがコミュニティからリリースされています。詳細は右記参照 https://supermarket.chef.io/

mysql	nginx	apache2	postgresql
java	git	apt	yum
php	build-essential	nodejs	mongodb
ntp	jenkins	database	python
docker	tomcat	rabbitmq	elasticsearch

Berkshelf : 依存関係の管理

- 特にコミュニティCookbookにおいては、ほかのCookbookに依存していることがあります
- この課題を解決するために使うのがBerkshelfです。詳細は公式サイトを確認してください http://berkshelf.com/
- Berkshelfは他の言語のパッケージマネージャ、たとえばcomposer(PHP), bundler(Ruby) and npm(Nodejs)のChef版と考えてください

コードをクリーンに保つ

- Cookbookは手順書と同等になります。したがって可読性や保守性は非常に重要です(もちろん継続的インテグレーションも重要です)
- たとえば、Foodcriticを使うことで例えばRubyのRubocopのようなCookbookの静的解析が可能です
- 単一責務の原則にしたがってCookbookを小さいサイズに保つようにしてください

質問?