# Chef Metal で夢見るコード（レシピ）によるクラスタ管理

どうも見習いシェフのかっぱです。

## 追記（1）

Chef Zero の記述に関して

* Cookbook の管理のみに機能を限定

と記載しておりましたが [@sawanoboly](https://twitter.com/sawanoboly) さんからご指摘頂き修正いたしました！

実際には Chef Zero は以下のような特徴があります。

* Chef Server と互換（Chef Server 11 と互換）
* 認証機構が無い
* 各種情報（cookbook 等）はメモリ上に展開されるので永続性が無い

[こちら](http://slides.com/urasoko/chef-zero#/)や[こちら](http://slides.com/urasoko/chef-zero#/)のスライドが解りやすいかと思います。

ということで @sawanoboly さん、ご指摘有難うございました！

## はじめに

[Chef Metal](https://github.com/opscode/chef-metal) という Chef を利用したクラスタ管理フレームワークを試してみたいと思います。この記事を書くにあたり以下を参考にさせて頂きました。

* [opscode/chef-metal](https://github.com/opscode/chef-metal)
* [Chef Metal 0.2 Release](http://www.getchef.com/blog/2014/03/04/chef-metal-0-2-release/)
* [#ChefConf 2014: John Keiser, "Chef Metal: Automate Your Whole Cluster In Chef"](https://www.youtube.com/watch?v=Yb8QdL30WgM)
* [Chef Metal](http://slides.com/jkeiser/chef-metal#/)
* [Chef MetalのGitHubページの翻訳をしてみました](http://qiita.com/TsuyoshiUshio@github/items/47457f7dcd4c9b866cd2)
* [Chef Metalの凄さを体験しよう](http://qiita.com/TsuyoshiUshio@github/items/180a54994df57a1d08d3)
* [Configuring the AWS CLI](http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html#d0e726)

本記事より詳細な内容は上記の参考を御覧ください。

## Chef Metal とは？

### どんなもの？

This library solves the problem of repeatably creating machines and infrastructures in Chef. It has a plugin model that lets you write bootstrappers for your favorite infrastructures, including VirtualBox, EC2, LXC, bare metal, and many more!

[README.md](https://github.com/opscode/chef-metal/blob/master/README.md) の冒頭にあるように Chef Metal は以下のような特徴があります。

* Chef で繰り返しマシンやインフラを構築する際の問題を解決する
* プラグインモデルを採用
* VirtualBox や EC2 や LXC やベアメタル等の環境に応じた bootstrapper を作って利用することが出来る

### machine リソースと Driver

Chef Metal の重要な要素として machine リソースと Driver があります。

#### machine リソース

machine リソースは Chef Metal の基本単位でマシンに何を適用するか（例えば recipe を適用、tag をつける等...）を宣言する為に利用します。以下のように記載します。

machine 'mario' do  
 recipe 'postgresql'  
 recipe 'mydb'  
 tag 'mydb\_master'  
end  
  
num\_webservers = 1  
  
1.upto(num\_webservers) do |i|  
 machine "luigi#{i}" do  
 recipe 'apache'  
 recipe 'mywebapp'  
 end  
end

通常 machine リソースは上記のようにアプリケーションのトポロジ毎に宣言して、独立した OS とプロビジョニングに依存しないファイルに設定を行います。また、num\_servers の数によって Web サーバーの数を動的に変化させることが出来るようです。そして、それがコードで出来てしまうといのは素晴らしいことですよね。

#### Drivers

Drivers handle the real work of getting those abstract definitions into real, physical form. They handle the following tasks, idempotently (you can run the resource again and again and it will only create the machine once--though it may notice things are wrong and fix them!):

上記のように書かれていますので、ざっくりと意訳すると以下のような内容かと思います。

* 構築するマシン（物理マシンや仮想マシン）の定義を抽象化する
* 冪等性を維持（何度実行しても一度だけマシンが作成される）

また、Drivers 以下のような処理を行います。

* クラウド環境から仮想マシンの作成、ベアメタルなマシンの取得
* SSH や WinRM その他の方法でマシンに接続する
* Bootstraping と Chef を適用して収束

既に以下のような Driver が用意されています。

* [Fog を介した EC2 や DigitalOcean や OpenStack](https://github.com/opscode/chef-metal-fog)
* [Vagrant を介した VirtualBox や VMware](https://github.com/opscode/chef-metal-vagrant)
* コンテナとして　[LXC](https://github.com/opscode/chef-metal-lxc) や [Docker](https://github.com/opscode/chef-metal-docker)
* ベアメタルでは [SSH](https://github.com/double-z/chef-metal-ssh) ドライバを利用する

### Chef Metal が目指すもの〜ホストの状態からクラスタの状態管理へ〜

ここまでの個人的な見解ですが Chef Metral はホストの状態を管理する Chef レシピで複数のホスト（クラスタ）の状態を管理することを目指したツールではないかと考えています。

## Chef Metal と関連ツールのインストール

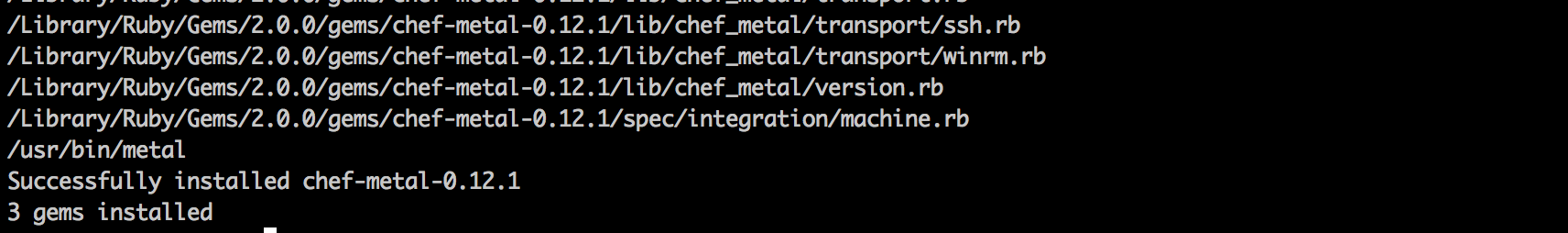
うんちくはこのあたりにして Chef Metal を触ってみたいと思います。まずは [chef-metal](https://github.com/opscode/chef-metal) と [chef-metal-fog](https://github.com/opscode/chef-metal-fog) をインストールします。

### chef-metal

コアである [chef-metal](https://github.com/opscode/chef-metal) をインストールします。

sudo gem install chef-metal --no-ri --no-rdoc -V

以下のようにインストール成功です。



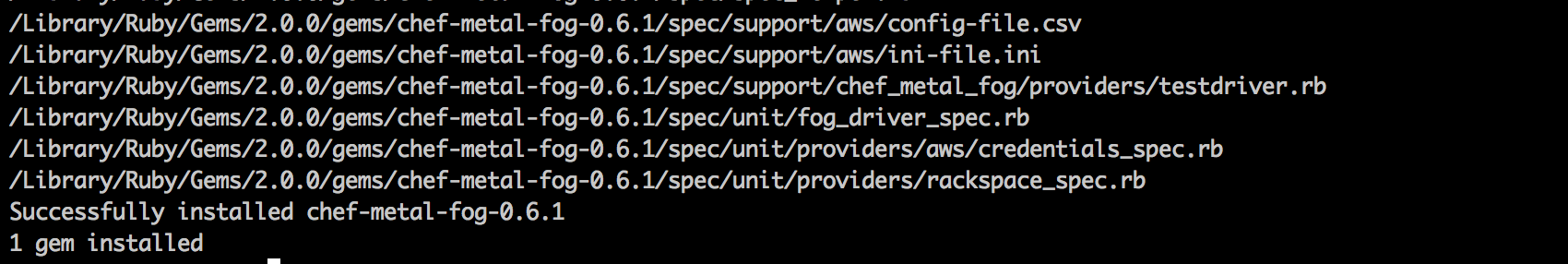
2014062101.png

### chef-metal-fog

次に EC2 や DigitalOcean と連携する為の [chef-metal-fog](https://github.com/opscode/chef-metal-fog) をインストールします。

sudo gem install chef-metal-fog --no-ri --no-rdoc -V

以下のようにインストール成功です。



2014062102.png

## EC2 インスタンスを利用して Chef Metral で Web サーバークラスタを構築してみる

[こちら](https://github.com/opscode/chef-metal/blob/master/README.md#fog-ec2-openstack-and-friends)を参考にして EC2 インスタンスと Chef Metral で Web サーバークラスタを構築してみたいと思います。（と言っても Apache をインストールするだけですが...）

### AWS の credential ファイルを用意する

[こちら](http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html#d0e726)を参考に AWS 環境にアクセスする為の資格情報を下記のように設定します。

vi ~/.aws/config

以下のように設定します。

[default]  
aws\_access\_key\_id=AKIAIOSFODNN7EXAMPLE  
aws\_secret\_access\_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY  
region=us-east-1

例えば、複数のキーペアを以下のように管理している場合...

[default]  
aws\_access\_key\_id=AKIAIOSFODNN7EXAMPLE  
aws\_secret\_access\_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY  
region=us-east-1  
  
[default hogehuga]  
aws\_access\_key\_id=AKxxxxxxxxxxxEXAMPLE  
aws\_secret\_access\_key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxEXAMPLEKEY  
region=us-east-1

fog:AWS:hogehuga と指定することでキーペアを切り替えることが出来ます。

### Chef リポジトリっぽい階層を作る

とりあえず最低限 cookbooks というディレクトリを作っておきましょう。

mkdir -p ~/path/to/chef-metal-test/cookbooks

### cookbook っぽい階層作る

test という名前の cookbook を作ったつもりで cookbooks 以下に test というディレクトリと recipes というディレクトリを作っておきます。

mkdir -p ~/pato/to/chef-metal-test/cookbooks/test/recipes

#### metadata.rb

俺は cookbook だぜ！とアピールする為に metadata.rb を以下のように作成します。

vim ~/pato/to/chef-metal-test/cookbooks/test/metadata.rb

metadata.rb ファイルの中身は以下のような内容で記載します。

name 'test'  
version '0.0.1'

#### レシピを書く（simple.rb）

machine リソースを含むレシピを以下のように作成します。

vi ~/pato/to/chef-metal-test/cookbooks/test/recipes/simple.rb

以下のような内容を記載します。

require 'chef\_metal\_fog'  
  
with\_machine\_options :ssh\_username => 'root',  
 :bootstrap\_options => {  
 :key\_name => 'key',  
 :private\_key\_path => '~/.ssh/key.pem',  
 :image\_id => 'ami-abc1234',  
 :flavor\_id => 't1.micro',  
 :security\_group\_ids => 'sg-abc1234'  
}  
  
num\_webservers = 3  
  
1.upto(num\_webservers) do |i|  
 machine "webserver-#{i}" do  
 recipe 'test::apache'  
 end  
end

このレシピでは以下のような内容でクラスタが収束します。

* インスタンスは 3 台（AMI は ami-abc1234）
* インスタンスタイプは t1.micro
* 起動したインスタンスに apache というレシピを適用する

#### レシピを書く（apache.rb）

machine リソースで起動させたインスタンスに適用するレシピ recipe 'test::apache' で利用するレシピを以下のように作成しました。

vi ~/pato/to/chef-metal-test/cookbooks/test/recipes/apache.rb

内容は以下のように非常にシンプルです。

package "httpd" do  
 action :install  
end

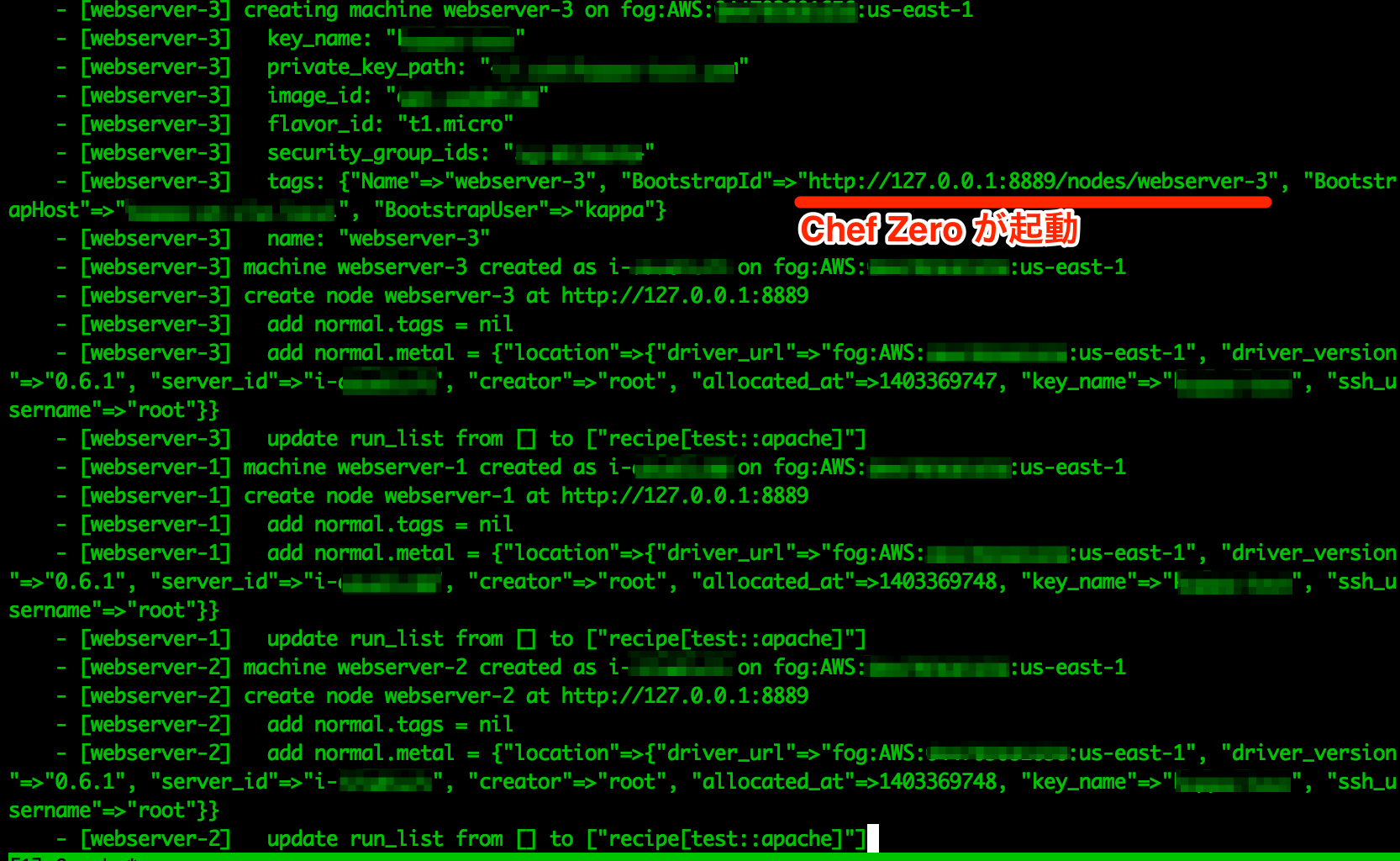
### Chef Metal を実行する

「Chef Metal を実行」という言い方が正しいかは悩みどころですが...以下のように chef-client を実行します。

cd ~/pato/to/chef-metal-test/  
chef-client -z -r "recipe[test::simple]"

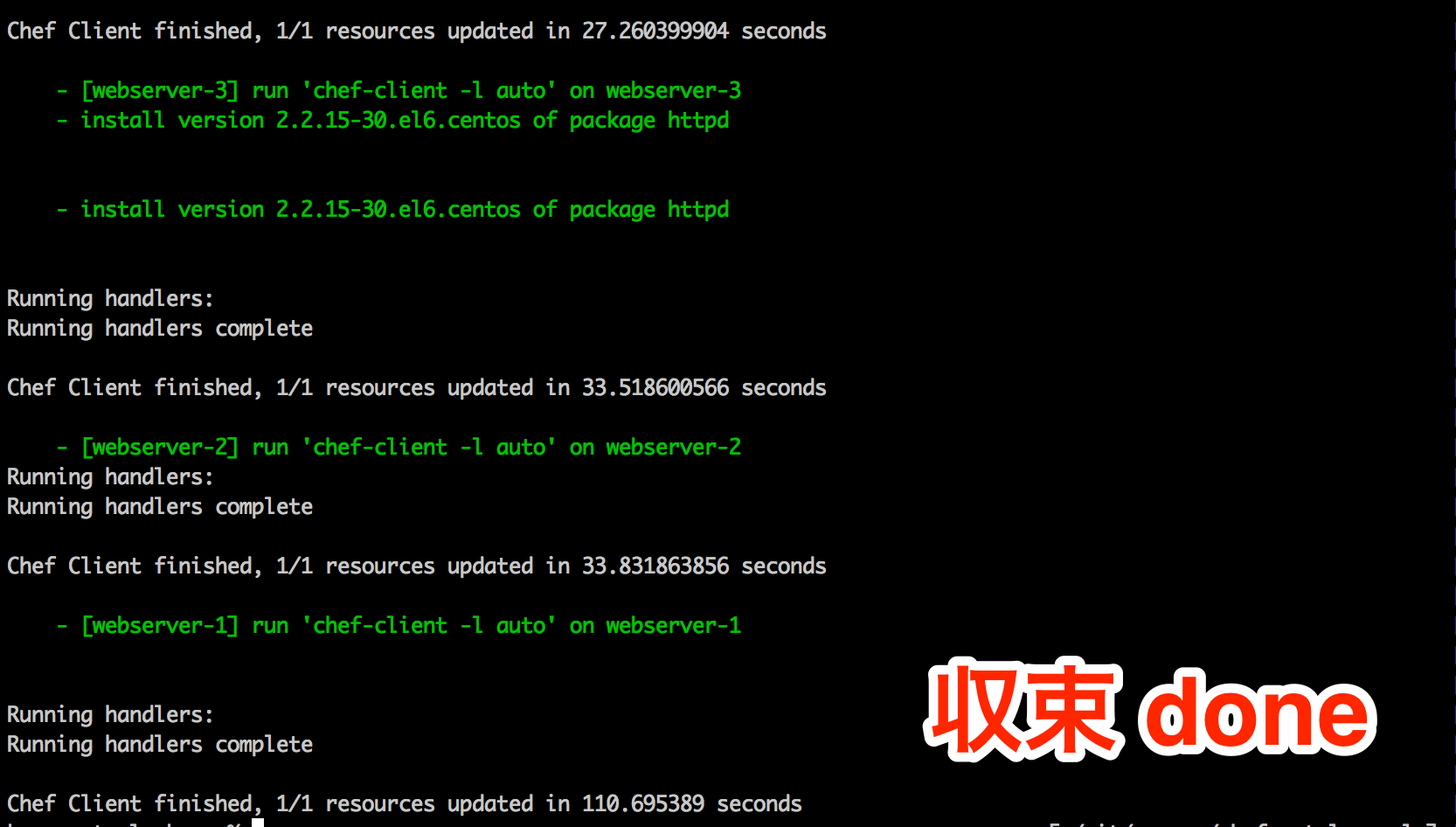
ちなみに -z オプションは [Chef Zero](https://github.com/opscode/chef-zero) と呼ばれる簡易的な Chef Server で認証機構が無いのとデータが保されないという制限があるもののでお手軽に Chef Server の機能が利用可能です。 [README](https://github.com/opscode/chef-zero/blob/master/README.md) や[こちら](http://www.creationline.com/lab/2749)を御覧ください。

実行すると以下のような表示となります...。



2014062201.png

しばし...ビールでも飲んで待ちましょう。



2014062202.png

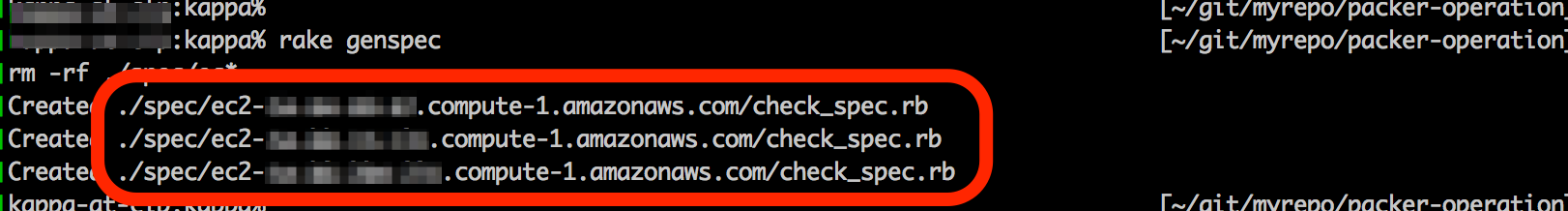
収束が完了しました。エラーもなく終了したようです。

### Serverspec を使った確認

ちゃんと Apache がインストールされたかは Serverspec を使って確認しましょう。

#### AWS SDK for Ruby + Rake で spec ファイルを用意

AWS SDK for Ruby + Rake を使って spec ファイルを用意します。



2014062203.png

AWS SDK for Ruby を使えば EC2 につけられた tag を指定してインスタンスの情報を取得することが出来るので、Rake と組み合わせて複数の Spec ファイルを一気に作成してしまいましょう。

#### テスト

以下のように実行してテストを行います。

rspec -fd --color spec

以下のようにオールグリーンとなりました。

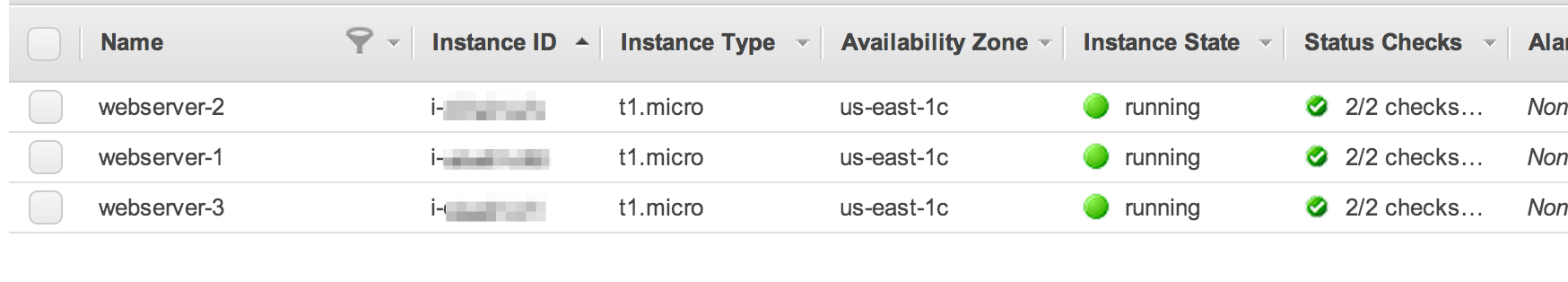


2014062204.png

SSH でログインして確認したら負けです。

#### 一応、マネジメントコンソールでも

マネジメントコンソールでインスタンスが起動しているかも確認しておきます。



2014062205.png

指定した台数のインスタンスも起動しており、超カンタンなクラスタが構築出来ました。

### 後片付けもしっかりと...

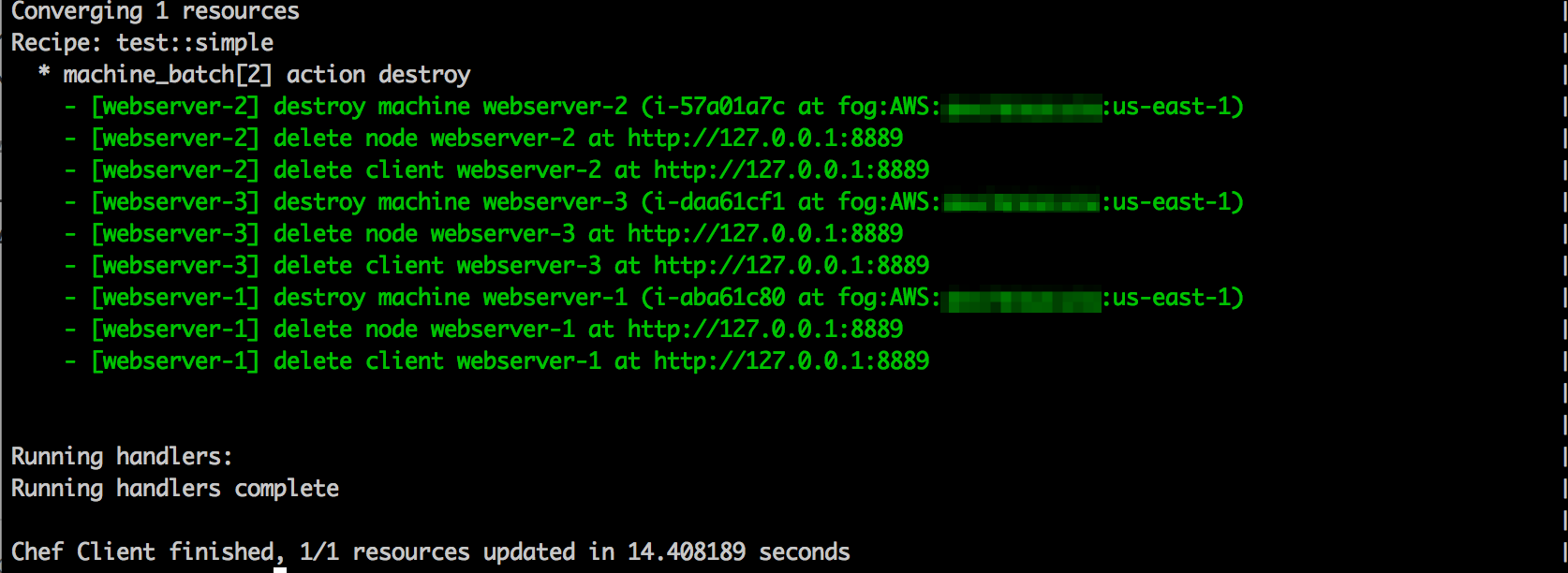
検証に使ったインスタンスも Chef Metal を使ってターミネートしましょう。

#### バルス！

バルスは :destroy の Action を利用します。

num\_webservers = 3  
  
1.upto(num\_webservers) do |i|  
 machine "webserver-#{i}" do  
 action :destroy  
 end  
end

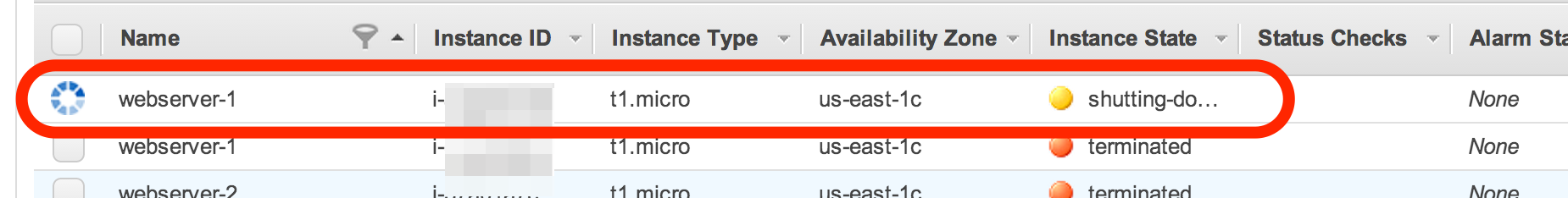
収束させると...



2014062207.png

#### ありがとうございました...

ちゃんとターミネートされています。



2014062206.png

お疲れ様でした。

## まとめ

約半日触ってみた感想...

### 面白い

* 従来は一台のインスタンスに対して Chef を適用していたが複数台まとめて適用というのは効率的
* 今回はちゃんと試しきれていないが machine リソースには stop や destroy 等の Action が用意されている
* [こちら](http://www.getchef.com/blog/2014/03/04/chef-metal-0-2-release/)にも書かれているようにリソースの状況に応じてホスト（インスタンス）の増減も可能なようだ
* Chef のレシピ（コード）でクラスタの状態を管理出来るというのは非常に素晴らしいと思う

### 残念なところ

* 現在も開発途上ということもあり情報量が少ないのが残念

引き続き、追っていきたいと思います。