Vagrant / Ansibleで作る開発環境

ヤポドゥ ハンズオン勉強会 Vol.1

ANSIBLE

自己紹介

・田村 亮
・@launcher_test
・フリーランスエンジニア (一人親方)
・インフラ系

フリーランスになる前後(2012年くらい)からハンズオン勉強会を開催したかった。

もともとコカコー○の自販機補充員とか配管工とか佐○急便の仕分けとか警備員とか色々やってました。

ヤポドゥ勉強会

- ハンズオン形式でインフラ系技術の習得を目指す。
- 現場で使える技術を学ぶ
- とりあえず動かせるレベルが目標
- 1 ～ 3ヶ月毎に開催予定
- Github https://github.com/yapodu

※ヤポドゥとはラサ語(チベット語)で「良い」の意味

Infrastructure as Codeとは

- いままで手順書や個人の感覚で行ってたインフラのセットアップや構成変更をコードで管理し自動実行する。

- 手順書があるとはいえ人作業で差分が生まれる可能性

  => コードベースで実行されるため実施内容に差分はない,冪等性がある

- その設定がいつ誰が設定・承認したのか

  => コードをGit等で管理すれば明確に

- 20 ～ 30台もコンソール立ち上げて手順書コピペするのが面倒くさい

  => 対象ホストを指定し自動実行

更にその先には

「単にサーバー構成変更を自動化しましょうという話ではなく、インフラをすべてソフトウェアとして、コードで扱うことでアプリケーション開発で行われてきたいろいろな "ワークフロー" をインフラ作業の世界にも導入しましょうね」

必読

Infrastructure as Code
http://d.hatena.ne.jp/naoya/20131215/1387090668

Vagrant とは

- Hashicorp社が作成している仮想マシンを動かす仮想化ソフトのラッパー
- 今回はVirtualBoxが対象ですがawsやkvmも操作できます。
- とっても便利。

Hashicorp社もInfrastructure as Codeを含む構成管理ツールterraformをリリースしている
http://www.terraform.io/intro/index.html

参考

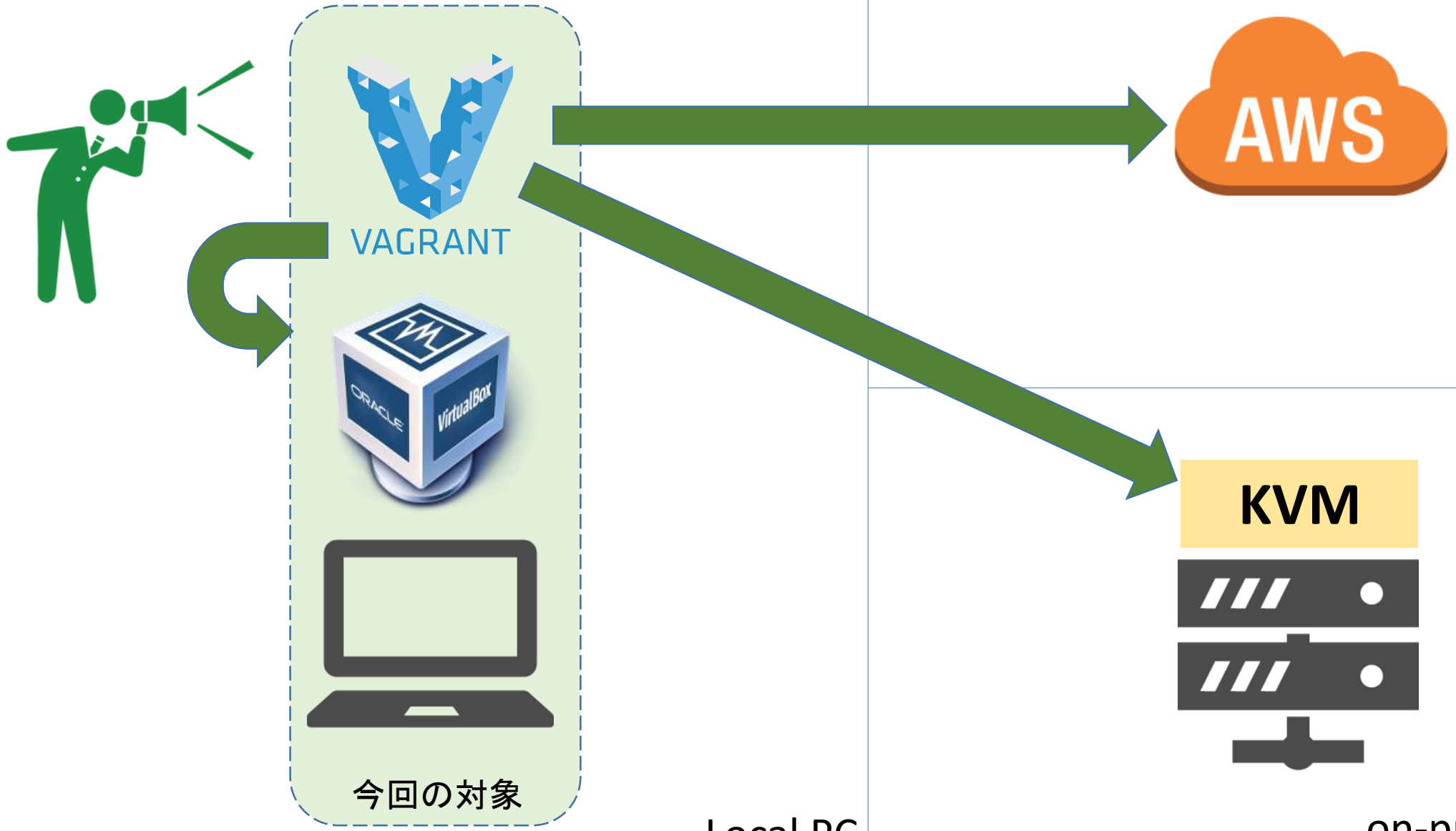
aws: はじめてのvagrant-aws

http://qiita.com/asam316/items/d5768afee6797ea8b4e2

kvm: KVM用仮想マシンをVagrantで手軽に作る

http://knowledge.sakura.ad.jp/tech/2535/

Vagrant イメージ図



作業前確認

- Windows の人は ssh が必要になります。なければmsysgit、もしくは Github for Windows をインストールして下さい。
- Windowsのログイン中のユーザーディレクトリ名が2バイト文字の方

  ※コマンドプロンプトから echo %USERPROFILE% で確認

  1.C:¥vagrant_user_dir を作成

  2.環境変数 VAGRANT_HOME を上記フォルダに指定して下さい。
- VirtualBox 4.3.8 / Vagrant 1.6.0 未満の方はアップデートインストールで最新版をインストールして下さい。

  ※ ヘルプ -> VirtualBox について / vagrant –version で確認

VirtualBox / Vagrant インストール

- VirtualBox

http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html#vbox

- Vagrant

https://www.vagrantup.com/downloads.html

仮想環境のベースとなるBoxファイルを準備

・Virtualboxに仮想マシンイメージをvagrantで差し込む

URL:www.vagrantbox.es

CentOS 6.5 x86_64のVirtualBoxのURLをコピー

「COPY」ボタンを押して、URLをコピーしておき以下を実行。

例：vagrant box add dev-host-01 "コピーしたBOX定義URL"

```
$ mkdir  -p /c/vagrant/centos6 ; cd /c/vagrant/centos6
$ vagrant box add dev-host-01 https://github.com/2creatives/vagrant-centos/releases/download/v6.5.3/centos65-x86_64-20140116.box
```

※時間がかかるため休止にならないよう注意

or

```
$ vagrant box add ‘用意したboxファイルパス’
$ vagrant init dev-host-01
$ ls -l Vagrantfile
```

Vagrantfile の編集

・Vagrantfile：vagrantで操作する仮想機の設定を行うファイル

・CPU,メモリ,ローカルPCとのNW設定,コマンド実行等を制御

```
$ cp -p Vagrantfile Vagrantfile.`date +%Y%m%d`
$ vim Vagrantfile
~~~編集箇所
  config.vm.network “forwarded_port”, guest: 3000, host: 13000 ●←コメント解除 & Port番号変更
  config.vm.network ″public_network″ ●←コメント解除
  config.vm.provider ″virtualbox″ do |vb| ●←コメント解除
    vb.memory = ″1024″ ●←コメント解除  vagrantのバージョンにより vb.customize～の記述になっている場合もある。同じくコメントアウト
  end ●←コメント解除
~~~
```

Vagrantfile diff

```
$ diff Vagrantfile Vagrantfile.`date +%Y%m%d`
25c25
<   config.vm.network "forwarded_port", guest: 3000, host: 13000
---
>   # config.vm.network "forwarded_port", guest: 80, host: 8080
46c46
<   config.vm.provider "virtualbox" do |vb|
---
>   # config.vm.provider "virtualbox" do |vb|
51,52c51,52
<     vb.memory = "1024"
<   end
---
>   #   vb.memory = "1024"
>   # end
```

Vagrant 起動と確認

差し込んだBoxイメージをVagrantfile で起動する

VirtualBox を起動して仮想マシンが起動してないことを確認

```
$ vagrant up
※Windowsはファイアーウォールブロック許可ポップがでたら許可
```

VirtualBox を起動して仮想マシンが起動していることを確認

Vagrant の ssh でログインしip確認とローカルssh（ansible用のおまじない）

```
$ vagrant ssh
```

```
$ ip addr
$ ssh vagrant@localhost
$ sudo init 0
```

Vagrant Snapshotプラグイン

VirtualBoxのイメージをスナップショットで管理できるようになる。

サンドボックプラグインの share が存在するが、今回はvbox-snapshot を導入。

```
$ vagrant plugin install vagrant-vbox-snapshot
$ vagrant plugin list
vagrant-vbox-snapshot (0.0.8)
```

仮想マシンが停止してることを確認しsnapshot取得

```
$ vagrant global-status
id       name    provider   state    directory
-----------------------------------------------------------------

d8ec367  default virtualbox running c:/vagrant/centos6
$ vagrant snapshot take first
```

Vagrant Snapshotプラグイン オプション

スナップショットを取得する

```
$ vagrant snapshot take‘スナップショット名’
```

スナップショットリスト表示

```
$ vagrant snapshot list
```

スナップショット移動

```
$ vagrant snapshot go‘スナップショット名’
```

スナップショット削除

```
$ vagrant snapshot delete‘スナップショット名’
```

Vagrant ユーザー鍵転送

vagrantユーザーの秘密鍵をlinux上に転送しておきます。

Ansibleは実行時にsshログインするので該当ユーザーの鍵指定またはパスワード入力が必要、今回は鍵を使用する。

※本来はlocalhostでsshログインは不要

仮想マシンを起動しssh鍵の確認と転送

```
$ vagrant up
$ vagrant ssh-config
~~~

IdentityFile
c:/vagrant/centos6/.vagrant/machines/default/virtualbox/private_key
~~~

$ scp -P 2222 ¥
/c/vagrant/centos6/.vagrant/machines/default/virtualbox/private_key ¥
 vagrant@127.0.0.1:~/.ssh/
~~~
```

Ansible とは

- 構成管理ツール / オーケストレーションツール※
- 類似のものとしてChef,Puppet,Itamae等がある。
- 実績としはChefが豊富なはず。
- もともと2年以上Chefでやってました。

ANSIBLE

※オーケストレーションツールの定義は色々だが複数の対象機に以下が行えると思えば、とりあえずはよろしいかと。

・構成管理
・ソースコードデプロイやミドル再読み込み
・他システムとの連携(例：Github,Jenkins等)

Ansible の前にChef 概要

- 構成管理ツール として一番有名だと思う。
- 乱暴に言うと
  大規模ならChef Server/Chef Client
  小規模ならChef-soloだった。

CHEF™

2014年 秋口まで

- 2012年 これからはインフラのコード化が必須だ　=> Chef を学ぶ
- 2013年 中規模以上ならServr/Client小規模案件なら Chef-solo で組むように
- 2014年 Chef-solo導入支援の案件をやった！
- 2014年秋 案件としても実績のあるVagrant / Chef-soloで勉強会を開催しようと準備。

要約：
chef-solo はもう終わり

http://dqn.sakusakutto.jp/2014/09/chef-solo_zero_local_mode.html


忘れろ。
なにかほかの
楽しいこと
考えるんだ。

と、言うのは大げさ

- chef-soloは現在でも使用できるし、代替としてchef-zeroが存在している。
- そもそもchefはserver/clientで運用することが正義な部分がある。
- 決して今までのノウハウが無になったわけではない。
- 個人的には指定がなければ今後もChef Server/Clientで構築する予定

じゃ、なんで今回はChefではなくAnsibleなのか。

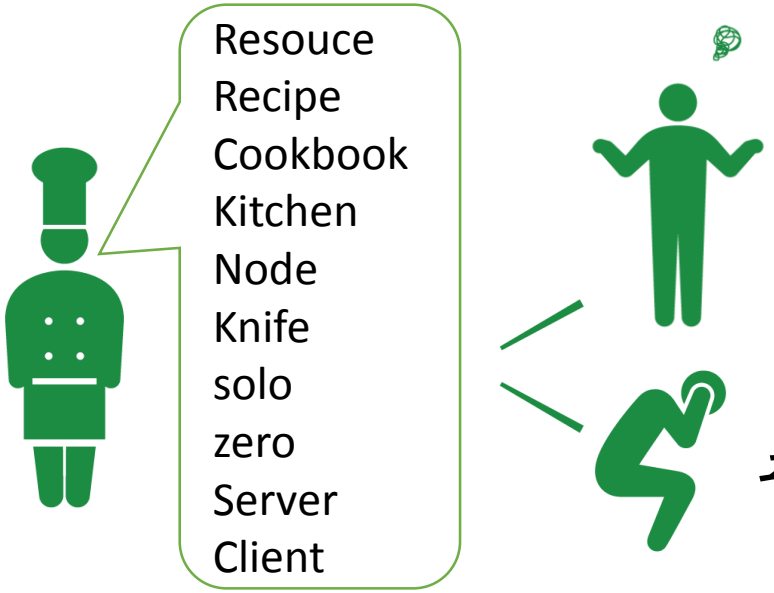
Ansibleの利点

- クライアントレス sshが通ればよい

  ※パッケージインストール等の要root権限が必要な際は事前にsudo可のユーザーで実行またはrootパス入力
- シンプル chefに比べて登場人物(独自用語含む)が少ない
- YAMLでシンプルに記述できる

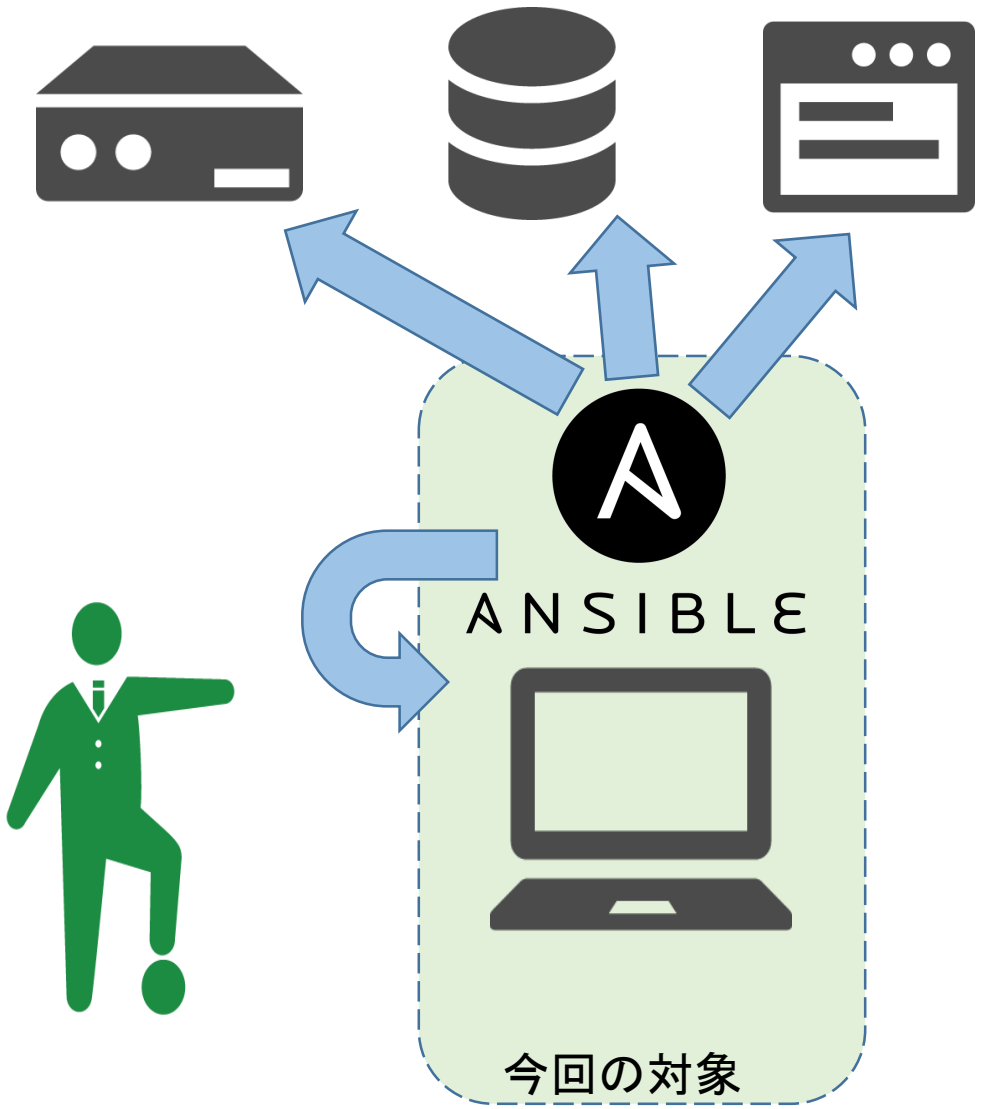
  ※chefの記述は個人的には好き

今回のようなケースにピッタリ

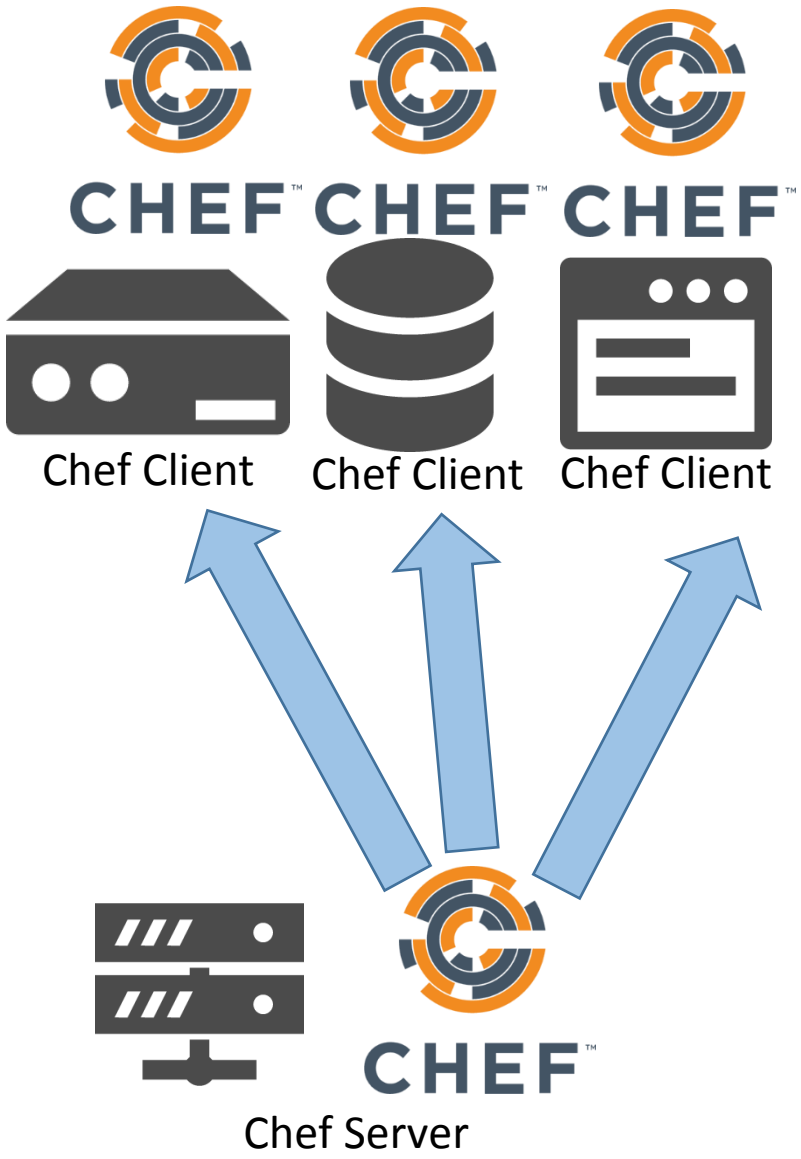


そこまで多くない

構成管理 イメージ図



Ansible



Chef Server / Client

Chef-zero

Chef-solo

Ansible yum インストール

・ssh ログイン

vagrant ssh じゃなくても ssh –p 2222 vagrant@localhost でいける

各自好きなターミナルで接続して下さい。

port : 2222 / user : vagrant / pass : vagrant

・ログイン後アップロードした鍵の権限変更

```
$ chmod 600 ~/.ssh/private_key
```

・Ansible yum インストール

```
$ cd /tmp
$ sudo  rpm -ivh ¥
http://ftp.riken.jp/Linux/fedora/epel/6/x86_64/epel-release-6-8.noarch.rpm
$ sudo yum install -y ansible
```

git clone

・今回使用するAnsible関連ファイルを git clone

```
$ cd ~/ansible
$ sudo yum install git
$ git clone https://github.com/yapodu/vagrant-ansible
$ cd vagrant-ansible
```

playbook

・playbookは1ファイルに記述することもAnsibleが推奨するディレクトリへ配置することも可能

・今回はOS設定は1ファイル、Rails設定はディレクトリ配置で記述

yapodu-study.base.yml 概要

・hosts : 対象ホスト

・user : 実行ユーザー

・sudo : sudo で実行するか

・vars : palybook内変数

・task : 実行内容

OS設定

・セットアップ前確認

```
$ which vim
$ date
```

・syntax check実行

```
$ ansible-playbook -i hosts yapodu-study.base.yml --syntax-check
```

・task一覧表示

```
• $ ansible-playbook -i hosts yapodu-study.base.yml --list-tasks
```

・playbook実行

```
$ ansible-playbook -v -i hosts ¥
--private-key="~/.ssh/private_key" yapodu-study.base.yml
```

Playbook実行後確認

・yum update まで完了し確認

```
• $ which vim
• $ date
```

・タグ指定実行

```
$ ansible-playbook -v -i hosts ¥
--private-key="~/.ssh/private_key" yapodu-study.base.yml --tags=jst
```

・update後 タスク確認後再起動実施(snapshotを取得しておくと捗る)

```
$ ansible-playbook -v -i hosts ¥
--private-key="~/.ssh/private_key" yapodu-study.REBOOT.yml --step
```

Railsセットアップ

・まずはpalybook実行

```
$ cd ansible/vagrant-ansible
$ ansible-playbook -vvv -i hosts ¥
--private-key="~/.ssh/private_key" yapodu-study.dev.yml
```

・OS設定時のplaybookと異なりAnsibleのbest practicesに沿った配置

配置参考 : https://docs.ansible.com/playbooks_best_practices.html

```
$ vim -R yapodu-study.dev.yml
$ ls -l roles/mysql
$ vim -R roles/mysql/tasks/main.yml
$ ls -lR roles/mysql/
```

Rails確認

・アプリを配置、確認(発表者のみ)

単純なRails 画面を表示さたい場合は下記手順

```
$ rails new app01 -d mysql
$ cp -p app01/config/database.yml app01/config/database.yml.`date +%Y%m%d`
$ vim app01/config/database.yml
$ diff app01/config/database.yml app01/config/database.yml.`date +%Y%m%d`
<   password: changeme1234
>   password:
$ cp -p app01/Gemfile app01/Gemfile.`date +%Y%m%d`
$ diff  app01/Gemfile app01/Gemfile.`date +%Y%m%d`
< gem 'therubyracer',  platforms: :ruby
> # gem 'therubyracer',  platforms: :ruby
```

Rails確認

```
$ cd app01/
$ bundle install
$ bin/rake db:create db:migrate
$ rails server


$ cd app01
$ bin/rake db:create db:migrate
```

・ローカルPCのブラウザから

http://127.0.0.1:13000 へアクセス

Rails初期画面を確認

最後に

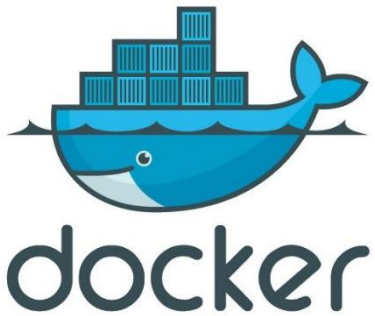
- 今回はあくまでも概要レベルなので、このハンズオンでの内容に磨きをかけて素晴らしい **Infrastructure as Code** を現場で実現して下さい。

- Vagrant を含む HashiCorp製品のハンズオントレーニングが日本で始まりました。

http://www.publickey1.jp/blog/15/vagrantterraform.html

1プロダクト 8万円 / 全プロダクト２５万円

- 今後のハンズオン勉強会予定
  - DockerとImmutable Infrastructure
    既存サービスをコンテナへ、運用監視フェーズまで
  - ServerSpec/CircleCIで実現するContinuous Integration
    継続的インテグレーションをインフラで
  - Google BigQueryとBigdata
    巨大データをGoogleのインフラで処理



ありがとうございました。