

5 OpenStack Computeの インストールの自動化

西田 潤

目次

5.1 Puppetを使ったOpenStackの配置ツール

5.2 VirtualBox, Vagrant, Chefを使った
OpenStack Computeのインストール

5章の概要

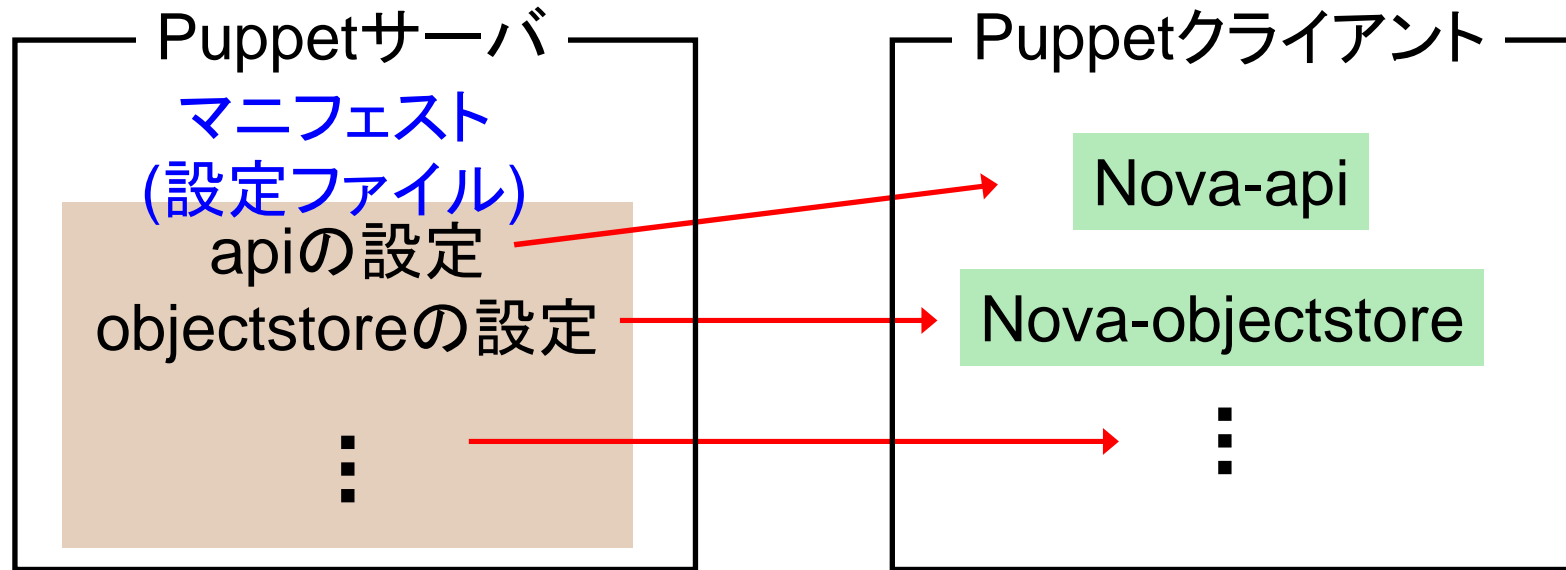
インストールの自動化により、以下の2つの利点を得られる

- (1) 大規模なクラウド環境を効果的で再現可能な方式で構築可能
- (2) 連続的な統合やテストが可能

OpenStack Computeを自動的に構築する方法は以下の2通り

- (1) Puppet(基盤管理プラットフォーム)を使用する方法
- (2) Chef(基盤管理フレームワーク)とVagrant(仮想化された環境を構築して配布するためのツール)を使用する方法

5.1 Puppetを使ったOpenStackの構築



<Puppet>

クライアント・サーバ型のシステム自動管理ツール

<deployment-tool (配置ツール)>

- (1) Puppetをインストールする
- (2) OpenStackの環境の設定などを行う

前準備

(1) ネットワーク

- (A) サーバはインターネットに接続できなければならない
- (B) Puppetサーバは名前でnovaコンポーネントサーバにアクセス可能

注意

- * ユーザ名を変更すべきだが、全てのnovaコンポーネントサーバにで、`~/DeploymentTool/conf/deploy.conf` の (`ssh_user='user'`)のユーザ名を同じ名前にする必要がある
- * 本章では例として、ユーザ名を[nii]とする

(2) パーミッション

- (A) 構築とサービスの供給のためにルートユーザパーミッションを保有する必要がある

前準備2

(3) ソフトウェア

- (A) 名前でPuppetサーバにアクセスするためにインストールサーバを設定
- (B) nova-volumeサービスのVolumeManagerのデフォルト設定を変更しないならば, LVMを設定
- (C) Python 2.6以上をインストール
- (D) 現在実装されているアーキテクチャではnova-api, nova-objectstore, euca2oolsは1つのサーバにおく

(4) OS

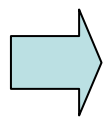
- (A) Ubuntu 10.04 or 10.10 をインストール

テスト済の環境

このdeploymentツールは以下の設定でテスト済み

- (1) Nova-computeコンポーネントは複数のサーバにインストール
- (2) OS: Ubuntu10.04 or Ubuntu10.10
- (3) Multiple network modes (VLAN Mode, Flat Mode)

現段階ではテストがまだ充分でない



今後の発展のため、以下にあなたの環境で起こった問題を知らせてください!

<https://answers.launchpad.net/nova-deployment-tool>

配置ツールの概要

Nova-deploymentツールを使い, Novaのインストール, テスト, アンインストールを行う

deploy.pyは詳細設定にPuppetを使用する

前準備で説明した環境以外で実行したい場合は,
Deployment-tool内のdeploy.confを編集する必要あり

以降のスライドでは, 以下の項目を説明

- (1) OpenStack環境の構築手順
- (2) nova-deployment-tool内のdeploy.pyを実行

deploymentツールのコマンド

(1) インストールするコマンド

```
$ python deploy.py install
```

(2) インストールの成功を確認するコマンド(テスト)

```
$ python deploy.py test
```

(3) novaコンポーネントをアンインストールするコマンド

```
$ python deploy.py uninstall  
$ python deploy.py all = python deploy.py  uninstall; python deploy.py  
install; python deploy.py test
```

以降のスライドでPuppetを使ったOpenStackの構築を行う手順を説明

deploymentツールのインストール

Novaコンポーネントサーバで
OpenStack Compute PPA(Personal Package Archive)を使用

コマンド例

```
$ sudo apt-get update
```

```
$ sudo apt-get install python-software-properties -y
```

```
$ sudo add-apt-repository ppa:nova-core/release
```

```
$ sudo apt-get update
```

ユーザをsudoersに登録する

全てのコンポーネントをユーザで実行できるようにする

➡ sudoersのファイルを編集し, ユーザを
novaコンポーネントサービスのsudoerとして登録

```
$ sudo visudo
```

visudoの最後の行に以下の記述を追加して, 保存する.

```
nii ALL=(ALL) NOPASSWD:ALL
```

```
nova ALL=(ALL) NOPASSWD:ALL
```

SSHの設定

パブリックキーとプライベートキーを生成して
SSHが動作するように設定

Nova-deploymentツールはパスワードの入力なしに
novaコンポーネントサービスに接続するために必要

Nova-deploymentツールを動作させるサーバの
パブリックキーとプライベートキーを生成

```
$ ssh-keygen -t rsa -N '' -f ~/.ssh/id_rsa
```

SSHの設定2

生成したパブリックキーを
novaコンポーネントサービスにコピー(登録)

```
$ ssh-copy-id nii@(each nova component server name)
```

設置ツールのソースコードをダウンロードし、
圧縮ファイルの解凍する

```
$ wget http://launchpad.net/nova-deployment-tool/trunk/bexar  
/+download/novadeployment-tool.tgz
```

```
$ tar xzvf nova-deployment-tool.tgz
```

deploymentツールの設定

このままでnova-deploymentツールを実行できるか？

➡ このままでは実行できない可能性大！

計算機の環境などによって設定ファイルの変更が必要

例：インストールはシングルサーバかマルチサーバか

ファイルを展開し， conf/deploy.confを編集

deploy.confで使われている定義1

(1) nova_api

nova-apiコンポーネントがインストールされているサーバ

(2) nova_objectstore

nova-objectstoreコンポーネントがインストールされているサーバ

(3) nova_compute

nova-computeコンポーネントがインストールされているサーバ

(4) nova_scheduler

nova-schedulerコンポーネントがインストールされているサーバ

deploy.confで使われている定義2

(5) nova_network

nova-networkコンポーネントがインストールされているサーバ

(6) nova_volume

nova-volumeコンポーネントがインストールされているサーバ

(7) euca2ools

テストシーケンスが動いているサーバ

(8) mysql

mysqlがインストールされているサーバ

deploy.confで使われている定義3

(9) puppet_server

puppetがインストールされているサーバ

(10) libvirt_type

仮想化タイプ

(11) network_interface

nova-computeコンポーネントで使われている
ネットワークインターフェース

(12) ssh_user

novaコンポーネントのSSHで使われているユーザネーム

現在実装されているアーキテクチャでは, nova-api,
nova-objectstore, euca2oolsを1つのサーバにおく

設定の例

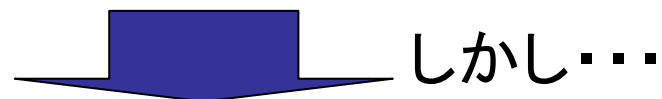
```
<begin ~/DeploymentTool/conf/deploy.conf>  
nova_api=ubuntu1  
nova_objectstore=ubuntu1  
nova_compute=ubuntu3  
nova_scheduler=ubuntu4  
nova_network=ubuntu5  
nova_volume=ubuntu6  
euca2ools=ubuntu1  
mysql=ubuntu1  
puppet_server=ubuntu7  
libvirt_type=kvm  
network_manager=nova.network.manager.VlanManager  
network_interface=eth0  
ssh_user=nii  
<end ~/DeploymentTool/conf/deploy.conf>
```

- * multiple novaコンポーネントにしたければ
nova_compute=ubuntu3, ubuntu8とすることができる

5.2 VirtualBox, Vagrant, Chefを使った OpenStack computeのインストール

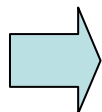
多くの依存関係を持つ分散処理システムの統合テストは
現在の大きな課題

理想的にはPXE bootをベースにしたOSで、
システムを完璧にインストールしたい



しかし...

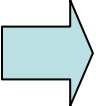
残念ながら全ての計算機が多くのハードウェアを持ってはいない



仮想マシンでテストを行うことはできる

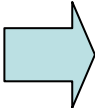
VirtualBox と VagrantでOpenStack Compute (Nova)
をインストールする手順を示す

Virtual Box のインストール

Virtual Box とは  オラクルが開発した仮想化ソフトウェア

コマンドラインで操作可 Mac/Linux/Windowsで動作

以下でインストールするバージョン

 VirtualBox 4.0 とvagrant prerelease

以下3つのOSでのVirtualBoxとvagrantのインストール法を示す

(1) OSX (2) Ubuntu Maverick (3) Ubuntu Lucid

必要に応じて,gemやgitなどをインストールしながら, コマンドを実行

OSX

Virtual Boxのインストール手順

```
$ curl -O http://download.virtualbox.org/virtualbox/4.0.2/  
$ VirtualBox-4.0.2-69518-OSX.dmg  
$ open VirtualBox-4.0.2-69518-OSX.dmg
```

vagrantのインストール手順

```
$ sudo gem update --system  
$ sudo gem install vagrant --pre
```

Ubuntu Maverick

Virtual Boxのインストール手順

```
$ wget -q http://download.virtualbox.org/virtualbox/debian/oracle_vbox.asc -O- |  
sudo apt-key add -  
$ echo "deb http://download.virtualbox.org/virtualbox/debian maverick contrib" |  
sudo tee /etc/apt/sources.list.d/virtualbox.list  
$ sudo apt-get update  
$ sudo apt-get install -y virtualbox-4.0
```

vagrantのインストール手順

```
$ sudo gem install vagrant --pre  
$ sudo ln -s /var/lib/gems/1.8/bin/vagrant /usr/local/bin/vagrant
```

Ubuntu Lucid

Virtual Boxのインストール手順

```
$ wget -q http://download.virtualbox.org/virtualbox/debian/oracle_vbox.asc -O- |  
sudo apt-key add -  
$ echo "deb http://download.virtualbox.org/virtualbox/debian lucid contrib" |  
sudo tee /etc/apt/sources.list.d/virtualbox.list  
$ sudo apt-get update  
$ sudo apt-get install -y virtualbox-4.0
```

vagrantのインストール手順

```
$ wget http://production.cf.rubygems.org/rubygems/rubygems-1.3.6.zip  
$ sudo apt-get install -y unzip  
$ unzip rubygems-1.3.6.zip  
$ cd rubygems-1.3.6  
$ sudo ruby setup.rb  
$ sudo gem1.8 install vagrant --pre
```

OpenStack Computeの インストールに向けて

(1) Chef Recipesを取得する

```
$ cd ~  
$ git clone http://github.com/ansolabs/openstack-cookbooks.git
```

(2) ディレクトリのセットアップ

```
$ mkdir aptcache  
$ mkdir chef  
$ cd chef
```

Vagrantがあれば,
以下のようなものが見える

ここではchef-soloを使い,
OpenStack Computeをインストールしていく

(3) chef-solo vagrant fileを取得

```
$ curl -o Vagrantfile  
https://gist.github.com/raw/786945/solo.rb
```

(a) chef-solo
(b) chef-server
(c) puppet

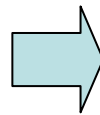
Vagrant Instance内での OpenStack Computeの動作1

(1) OpenStack Computeをインストールし動作させる

```
$ vagrant up
```

(2) 3~10分で, vagrant instanceが動作する

VagrantからMAC addressesについて
問題があるというエラーがでた場合



もう一度vagrantを
立ち上げよう！

```
$ vagrant ssh
```

Vagrant Instance内での OpenStack Computeの動作2

(3) Instanceを動作させ、それにつなげる

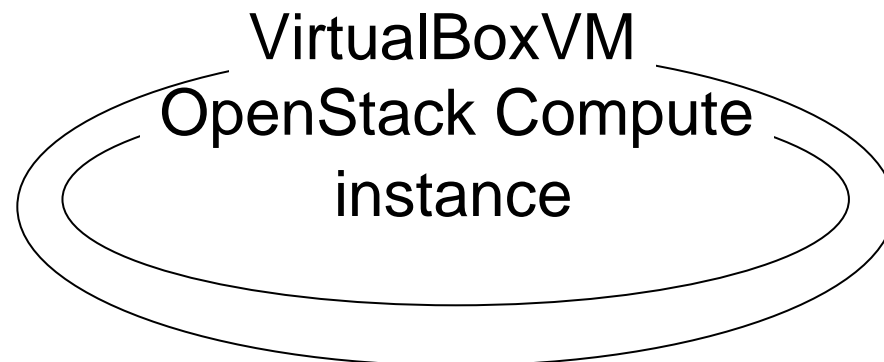
```
$ ./vagrant/novarc  
$ euca-add-keypair test > test.pem  
$ chmod 600 test.pem  
$ euca-run-instances -t m1.tiny -k test ami-tty  
    wait for boot (euca-describe-instances should report running)  
$ ssh -i test.pem root@10.0.0.3
```

(4) VMsが起動する

(5) 終了後, システムを
壊すことができる

```
$ vagrant destroy
```

(6) 再び起動したければ,
pemファイルとnovarcを
取り除いておく必要あり



```
$ vagrant destroy  
$ rm *.pem novarc
```