Chef-Solo

What is chef?

Chef is a Ruby DSL (domain specific language) for configuring GNU/Linux (or BSD) machines (Windows is not well supported), it has 2 flavors, "Chef Server" and "Chef Solo", in this

document I'm talking about **Chef SOLO** because it's easier to get started with - and works well as a complement to Rails apps.

Why use chef?

- Easy to deploy future servers
- Quickly create a test server
- Use other peoples cookbooks to quickly get software setup
- Get software setup a standard way across many of your servers
- Allows an easy upgrade path
- Allows you to deploy anywhere quickly, allowing you to change service providers quick and easy.
- Easily change your chef scripts for the next server you have to setup
- Changing between distros and even OS's is made easier (EG: Linux to BSD)

Server setup

Complementing technologies

Virtualbox setup with Ubuntu Server 10.04.3 64bit LTS & Network bridging
apt cacher or **squid** for caching your deb files (at work I use squid)

VirtualBox server setup

Using 'Ubuntu Server 10.04.3 64bit LTS' I'm going to install this within a virtualbox 3.2.12 and giving it an internal IP. This will be my test production server.

I'm using ubuntu 10.04.3 LTS server as this is exactly what I get at MammothVPS and for reasons described in my google doc: "distros"

1. Install Ubuntu into VirtualBox

Download from http://mirror.internode.on.net/pub/ubuntu/releases/10.04/

Install Ubuntu from an ISO into a virtualbox image. I used dynamic storage.

Select OpenSSH server within installation

2. Setup requirements

Configure your user for passwordless sudo by adding to your /etc/sudoers file:

```
map7 ALL=(ALL) NOPASSWD: ALL
```

3. Configure virtualbox image as bridged network.

http://www.virtualbox.org/manual/ch06.html

Using VirtualBox 3.2.12 on Ubuntu Desktop 10.04.3 LTS 64bit all I had to do is:
- Open Virtualbox
- Click on my server
- Click settings
- Click Network
- Select 'Bridged Adapter' from the 'Attached to' drop down box
- In the 'Name' drop down box select your hosts active network adapter. (on LTSP it's eth3)
- Click OK

Now you can boot your virtualbox server and you should have an internal IP.

4. Setup apt cache (Optional)

Squid (FreeBSD 7.x)

Ref: http://itkia.com/using-squid-to-cache-apt-updates-for-debian-and-ubuntu/

This is handy if you already run squid, but it would be overkill otherwise.

At work we run a transparent cache so the changes were simple. (Make a backup before editing).

```
# cd /usr/local/etc/squid
# cp squid.conf squid.conf.bak
```

Edit squid.conf and check that you have these lines:

```
cache_replacement_policy heap LFUDA

maximum_object_size 100 MB

refresh_pattern Packages\.bz2$  0     20%   4320 refresh-ims
refresh_pattern Sources\.bz2$   0     20%   4320 refresh-ims
refresh_pattern Release\.gpg$   0     20%   4320 refresh-ims
refresh_pattern Release$         0    20%   4320 refresh-ims
refresh_pattern .      0    20%    4320 refresh-ims
```

Save the file and restart squid

```
# /usr/local/etc/rc.d/squid restart
```

Now when you install debs and you are testing your setup over and over again it will only download the debs once.

apt-cacher

This is written in perl and has a bug. I've put the work around in this document but there must be something better than this apt-cacher.

Install it via apt-get

```
$ sudo apt-get install apt-cacher apache2
```

Make it start at boot time

```
$ sudo vi /etc/default/apt-cacher
```

Change Autostart to 1

Restart services

```
$ sudo service apache2 restart
$ sudo service apt-cacher start
```

Go to http://localhost/apt-cacher and follow these instructions:

Usage: edit your /etc/apt/sources.list so all your HTTP sources are prepended with the address

of your apt-cacher machine and the port, like this:
```
deb http://example.debian.org/debian unstable main contrib non-free
```
becomes
```
deb http://hal9000:3142/example.debian.org/debian unstable main contrib non-free
```

Import existing cache using symlinks (-s)
```
$ sudo /usr/share/apt-cacher/apt-cacher-import.pl -s /var/cache/apt/archives
```

BUG: apt-cacher does not want to update on a 64bit machine.
Ref: https://bugs.launchpad.net/ubuntu/+source/apt-cacher/+bug/778019

Download the following asm.zip file
http://ubuntuforums.org/attachment.php?attachmentid=199786&d=1313006417

Unzip this into /usr/lib/perl/5.10.1

Now update your apt-get
```
$ sudo apt-get update
```

VirtualBox Restart Script

In Virtualbox when testing out your script make use of the snapshots feature and take plenty of snapshots along the way. You can take snapshots whilst the server is running to make it faster to drop back to a running system.

When testing I found it easier to restore the previous snapshot within a script.
```
#!/bin/bash
# Reset and restart my server as a headless server.
#
# Control my Virtualbox machine
VBoxManage controlvm Ubuntu poweroff
VBoxManage snapshot Ubuntu restorecurrent
VBoxManage startvm --type headless Ubuntu
```

Development box setup

Requirements

Password ssh

Make sure you can access your server without a password
```
$ ssh-copy-id <user>@<host>
```

Chef-solo setup

Refer to my config: Dropbox/chef-solo & also on github.

With chef-solo you have some bootstrapping files (deploy.sh & install.sh), some config files (solo.rb & solo.json) and you have your cookbooks with recipes on how to install software.

You should be able to run your chef scripts over and over again, keep this in mind when building your recipes.

Bootstrapping

Ref: opscode's troubleshooting and technical FAQ and opscode's bootstrap chef-solo guide

Ruby must be installed through RVM as a system wide install as the ruby packages are too old and RVM installed under the users account has permission problems when mixed with chef.

deploy.sh

This script is ran on the development box and copies your scripts, config and cookbook files over to the new server.

```
#!/bin/bash
# Copies files to the server and runs install.

#play 'bork.wav' 2>/dev/null &

#host="${1:-map7@192.168.200.161}"
host="${1}"
json="${2}"

if [ -z "$host" ]; then
      echo "Usage: ./deploy.sh [user@host] [json]"
      echo "\nEG: ./deploy.sh fred@192.168.1.1 web_server.json"
      echo "If no json is given it will default to solo.json"
      exit
fi

if [ -z "$json" ]; then
      json="solo.json"
fi

# The host key might change when we instantiate a new VM, so
# we remove (-R) the old host key from known_hosts
echo 'keygen'
ssh-keygen -R "${host#*@}" 2> /dev/null

echo 'copy chef dir & run install.sh'
```

```
tar cj . | ssh -o 'StrictHostKeyChecking no' "$host" "
sudo rm -rf ~/chef &&
mkdir ~/chef &&
cd ~/chef &&
tar xj &&
sudo bash install.sh $json"
```

install.sh

This file installs just enough to get chef up and running on the server.

```
#!/bin/bash

json="${1}"

logfile="/root/chef-solo.log"

# This runs as root on the server
chef_binary="/usr/local/rvm/gems/ruby-1.9.2-p290/bin/chef-solo"


# Are we on a vanilla system?
if ! test -f "$chef_binary"; then

       export DEBIAN_FRONTEND=noninteractive

       # Upgrade headlessly (this is only safe-ish on vanilla systems)
       apt-get update -o Acquire::http::No-Cache=True
       apt-get -o Dpkg::Options::="--force-confnew" \
       --force-yes -fuy dist-upgrade

       # Install RVM as root (System-wide install)
       aptitude install -y curl git-core bzip2 build-essential zlib1g-dev libssl-dev

       # Note system-wide installs are not in the RVM main version
       # bash < <(curl -s https://rvm.beginrescueend.com/install/rvm)
       bash <( curl -L https://github.com/wayneeseguin/rvm/raw/1.3.0/contrib/install-
system-wide ) --version '1.3.0'
       (cat <<'EOP'
[[ -s "/usr/local/rvm/scripts/rvm" ]] && source "/usr/local/rvm/scripts/rvm" # This
loads RVM into a shell session.
EOP
       ) > /etc/profile.d/rvm.sh

       # Install Ruby using RVM
       [[ -s "/usr/local/rvm/scripts/rvm" ]] && source "/usr/local/rvm/scripts/rvm"
       rvm install 1.9.2-p290
       rvm use 1.9.2-p290 --default

       # Upgrade rubygems (Latest version 1.8.10 has a lot of problems)
       # Use 'bundle exec' in front of everything
       # 1.7.2 - is suggested but I've found issues with this as well.
       # 1.6.2 - is what I use on my dev box and comes with rvm install
       # gem update --system

       # Install chef
       gem install --no-rdoc --no-ri chef --version 0.10.0
fi
```

```
# Run chef-solo on server
[[ -s "/usr/local/rvm/scripts/rvm" ]] && source "/usr/local/rvm/scripts/rvm"
"$chef_binary" --config solo.rb --json-attributes "$json"
```

Config files

solo.rb

This sets paths and log levels for chef.

```
root = File.absolute_path(File.dirname(__FILE__))

file_cache_path root
cookbook_path root + '/cookbooks'
role_path root + "/roles"
log_level :info
log_location STDOUT
```

solo.json

This file lists which recipes and roles you are using. Roles are optional and explained later. All you need to know here is this file is where you tell chef to start setting up services on your server.

```
{
      "run_list": [
      "role[web_server]",
      "recipe[pais::default]"
      ]
}
```

Running chef-solo

To run chef-solo on a new server you can type the following

```
$ ./deploy.sh <server>
```

Once you have finished then you can ssh into your server and check that chef has installed your packages.

```
$ ssh <server>
$ rails -v
```

Cookbooks

There are cookbooks which are full of recipes to install and setup such things as apache2. They handle all the little configuration details which are common in setups.

Here is a list of cookbooks with ratings and comments, try these first. I find them a little old and out of date though.
http://community.opscode.com/cookbooks

More up to date resource would be opscode's github:
https://github.com/opscode/cookbooks

I ended up forking opscode's cookbooks and fixing them for Ruby 1.9.2 / Rails 3.0
Ref to my git repo: https://github.com/map7/cookbooks

Creating your own cookbook

In your chef-solo directory on your development computer create a directory to store your own cookbook.

EG:

```
$ mkdir -p cookbooks/mick/recipes
```

You can call your cookbook anything you want, most people call it after the server name.

cookbooks/mick/recipes/default.rb

```
# --- Install packages we need ---
package 'sysstat'
package 'htop'

# --- Install Rails
gem_package "rails" do
  action :install
end
```

As you create your recipes you may want to refer to the chef Resources page it has all the commands you can use in the DSL.
http://wiki.opscode.com/display/chef/Resources

Add it to your run_list within solo.json

solo.json (with my cookbook)

```
{
      "run_list": ["recipe[mick::default]"]
}
```

Using the 3rd party cookbooks

Apache

Installing Apache2 cookbook in my chef-solo config

1. Clone opscode cookbooks repo

```
~/chef-solo-cookbooks % git clone https://github.com/map7/cookbooks.git
```

2. Copy (not link) the cookbooks you want from this repo to your chef-solo project.
Note: Don't try and embed cookbooks repository inside your chef-solo setup, it doesn't work!

3. Call the recipe in your solo.json file like so

```
{
      "run_list": ["recipe[apache]"]
}
```

Attributes

Ref: http://wiki.opscode.com/display/chef/Attributes

Some cookbooks have attributes in which you can customise the recipe a little.

```
{
      "apache": { "listen_ports": ["81", "8080"] },
      "postgresql": { "version": ["8.0"] },
      "run_list": [
            "recipe[apache]",
            "recipe[posgresql::server]"
      ]
}
```

Roles (Optional)

http://wiki.opscode.com/display/chef/Roles
You should set up roles, ie: a role could be 'Web server' or 'MythTV server' or 'FTP server', etc.

I want to setup a 'Rails server' role.

Each role must have a 'run_list' and each recipe/role within the run_list is run in defined order.

You can have roles in chef-solo eg: base.rb, web_server.rb or mythtv.rb

There are 4 different ways to setup your roles. I will be using the ruby DSL as it's the simpliest form.

roles/web_server.rb

```
name "web_server"
description "Builds a web server for Rails apps"
run_list( "recipe[apache2]", "recipe[passenger_apache2]" )
```

You can build up a heap of roles then call them in your json config file which you pass to the chef-solo through the use of -j. Here is an example:
solo.json

```
{"run_list": ["role[web_server]", "recipe[pais::default]"]}
```

As you can see you have heaps of freedom here to customise. If there is a duplicate recipe within your structure it will only be ran the first time.

Now I might build up a heap of roles and then I can create json config files. I can name the files whatever I want. I may choose to use server names.

Resources

Package website
http://wiki.opscode.com/display/chef/Chef+Solo

Chef-solo tutorials
http://jonathanotto.com/blog/chef-tutorial-in-minutes.html
http://architects.dzone.com/news/chef-hello-world
http://stackoverflow.com/questions/2011011/are-there-any-good-chef-chef-server-and-chef-client-tutorials-out-there

http://opinionated-programmer.com/2011/06/chef-solo-tutorial-managing-a-single-server-with-chef
http://pivotallabs.com/users/mkocher/blog/articles/1605-chef-solo-tugging-on-the-bootstraps

RVM in production pros & cons:
http://blog.zottmann.org/post/5873082674/a-few-notes-about-chef-ubuntu-10-04-ruby-1-9-2

Getting system-wide rvm install working:
http://stackoverflow.com/questions/5421800/rvm-system-wide-install-script-url-broken-what-is-replacement

Troubleshooting

passenger_apache2 recipe

> Using packages I get this error:
> FATAL: Chef::Exceptions::Exec: package[libcurl4-gnutls-dev] (passenger_apache2::default line 33) had an error:

apt-get -q -y install libcurl4-gnutls-dev=7.19.7-1ubuntu1
returned 100, expected 0

There was something wrong with the snapshots in Virtualbox had to revert back and try again

Hash Sum mismatch

Clear lists (this doesn't seem right to me).

Looks like there is a server inbetween or your ISP which is caching the files. This could be squid proxy.

Put this line in your script:
apt-get update -o Acquire::http::No-Cache=True

FATAL error with apt-get

This usually means you need to do an apt-get update. You may want to move this in your install.sh script so that it always does an apt-get update.