[連貓也會的Rails佈署]使用Chef-Solo一鍵安裝機器

gogojimmy

June 2, 2013

Contents

Intro

這大概會是連續800篇的連載文章，當然就是循序漸進的從安裝一台server到安裝50000台server這麼複雜，首先我想先從最基本的裝機器開始說，因為在以前他一直是我的軟肋。

首先要來跟大家說一個秘密，就是我從來沒有一次就從安裝機器到 cap deploy一氣呵成安裝好的紀錄，總是在每次改東改西後，滿懷著希望輸入`cap deploy`後，機器跑阿跑的，然後吐給我一個Rollback告訴我哪裡出問題，然後我就摸摸鼻子改了改重來，就這樣修修改改到Deploy成功，這樣的步驟不知道陪了我幾個夜晚。

什麼？你說我沒有把記錄抄下來？你犯傻的嗎怎麼可能沒有，但有時候科學不能解釋所有事情，莫非定律告訴我們如果事情可能會出錯，那麼他一定會出錯，但今天我要告訴你，這一切都將成為過去，一切的故事，都是從與Chef開始，當我跟他交上朋友後，我的Deploy再也沒出過問題，所有關於機器的問題都交給他幫我處理，今天我就要介紹我這位朋友給你認識。

關於Chef

如果你英文跟我一樣頂呱呱的話，你應該知道Chef就是大廚的意思，Chef有兩種，分別是Chef-Solo跟Chef-Server，Chef-Solo顧名思義基本上就是單機作業，專屬於你個人的機器上，依照你的指示去管理你的機器，，這位大廚依照食譜集(cookbooks)裡的各種食譜(recipe)去幫你管理菜色(你的機器所需要的套件)，你需要做的就是告訴這位大廚你想要什麼食譜(套件)，這些食譜裡各種菜色的樣式(套件的設定)想要怎樣，而Chef-Server則是一個真正的大大大大廚，他需要自己生活在一台機器裡，因為他是個大大大大廚，跟Chef-Solo一樣他管理著所有的食譜，管理著旗下所有的機器(node)，你說跟Chef-Solo有什麼不同，當你的機器數量是動態不固定的時候你就知道他的好處了，因為今天要跟大家介紹的是Chef-Solo這位朋友，關於Chef-Server的故事我們下次再提，你只要知道我們今天請來的這位大廚，我們要怎麼去跟他溝通來幫我們管機器就好了。

跟Capistrano有什麼不同？

Capistrano其實簡單來說是將你設定好的指令透過ssh來執行，因此基本上只要機器能讓你ssh進去，而又有權限的情況下其實你也可以用Capistrano來做到與Chef-Solo一樣的功能，例如capistrano-recipes這個gem就是在做這件事，但這樣會有一個情況是程式的佈署設定很容易與機器的設定混在一起，日子久了當你用的東西越來越多時就很容易變亂，這樣當進行下一個專案的時候又要開始重寫Capistrano設定檔，因此這樣目標就變得很清楚了，我們希望能將佈署時做的事情與機器設定兩件事情分開，在這個前提下，Chef就是你的好幫手，因此我們並沒有捨棄Capistrano，而是將裝機器的事情交給了另一位專業人士，在設定上的模式會是這樣的：

1. 用Capistrano在機器上安裝好Chef-Solo

2. 將食譜上傳到機器
3. 用Capistrano執行Chef-Solo

在這樣的模式下，我們只要將機器環境的設定集中在cookbooks就好了，capistrano只要專心做佈署程式碼上去時該做的事情就好，讓大家各司其職，在這樣的結構下，你的設定便會變得乾淨許多，你甚至可以把cookbooks獨立出一個repository讓以後的專案都可以重複利用，反正每次裝機器都是裝那些東西，就開始使用Chef，不要再浪費生命裝機器了

安裝Chef-Solo

這邊會以Vagrant作為示範，如果你不知道怎麼使用Vagrant，可以參考鍵人雉作使用Vagrant練習環境佈署，讓我們`vagrant up`一台什麼東西都還沒有裝的VM，並且裝上Chef-Solo，Chef-Solo需要安裝Ruby環境以及一些有的沒有的，你可以使用我的gist來安裝，先登入我們的VM後切換到root：

```
$ vagrant up
$ vagrant ssh
$ sudo su
$ curl -L https://gist.github.com/gogojimmy/5523985/raw/
   b9d777bc380ee791c2f4534e9261b4b99289ed9f/bootstrap-chef-solo.sh | sh
```

安裝好後，執行`chef-solo -v`應該可以看到chef-solo已安裝成功並印出版本號，接下來我們要開始寫我們的食譜了

Cookbook

如同剛剛所說，Cookbook是個食譜集，裡面包含了我們想要的所有菜色，我們先來建立我們第一道菜，就來安裝Nginx吧，先建立出chef的資料夾

注意：以下操作都是在VM上執行

```
mkdir -p /etc/chef/coobooks/nginx/recipes
```

`/etc/chef`這個結構是固定的，不能亂放其他地方....

注意到了嗎，不管是哪道菜色的資料結構其實是有規則的，在cookbooks的資料夾下面都是{套件名稱}/{recipes}，而在每個recipes的目錄下都會有個`default.rb`的設定檔，這個設定檔就是讓我們去設定怎麼安裝及啟動套件的地方，現在就讓我們建立‘default.rb’這支檔案並寫下對nginx的設定：

```
package 'nginx'
```

`package`這個方法是讓Chef-Solo去呼叫你作業系統的套件管理指令進行安裝，例如以Ubuntu來說就是`apt`、Fedora的話就是`yum`，因此當我們下了`package 'nginx'`之後，chef就會根據你的作業系統去搜尋套件進行安裝，nginx的食譜寫好後，再來就是要告訴chef要去哪邊找食譜來安裝，我們在這邊就要建立一個json格式的設定檔來完成，這個設定檔主要就是針對這台機器的設定，告訴chef說這台機器要安裝哪些套件，機器在chef的概念裡叫做`node`，因此我們在根目錄下建立一個node.json的檔案，裡面寫好告訴Chef要安裝哪些套件的設定

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

`run_list`這個key指向的array就是告訴Chef要去執行哪些recipe，我們剛剛新增了Nginx的recipe因此這邊理所當然的就加上去了，如果你還有很多其他的recipe就是去擴充這個array就好了，最後一件要做的事情，就是建立一支`solo.rb`的檔案，這個檔案的目的要告訴Chef說關於哪些套件要進行安裝的`node.json`檔案以及cookbooks在哪邊的設定，因此我們一樣在chef的根目錄下建立這支檔案

```
cookbook_path File.expand_path("../cookbooks", __FILE__)
json_attribs File.expand_path("../node.json", __FILE__)
```

最後我們執行chef-solo來看看

```
$ chef-solo solo.rb
Starting Chef Client, version 11.4.0
Compiling Cookbooks...
Converging 1 resources
Recipe: nginx::default
  * package[nginx] action install
    - install version 1.1.19-1ubuntu0.2 of package nginx

Chef Client finished, 1 resources updated
root@vagrant-ubuntu-precise-64:/etc/chef# which nginx
/usr/sbin/nginx
```

噹噹！我們成功的做好一個nginx的recipe，並且讓chef安裝成功，往後你每次再執行的時候，Chef都會檢查如果該套件已經安裝過的話就略過安裝，簡單的說，就是我們已經確保了`run_list`裡面所設定的東西在你每次執行後都會安裝好了！這裡介紹的是package最基本的用法，詳細其實package還支援了很多其他的功能，像是你可以用直接用`version`設定版本，用`opions`來指定安裝時的選項像是`--with-http_ssl_module`，也可以指定`action`，`action`預設動作為`install`，你可以指定為像是`upgrade`、`remove`之類的，詳細的設定請參考Opscode文件中的說明

運用Chef安裝套件

你知道我知道裝機器不是把套件裝起來就結束了，以剛剛的Nginx為例，你裝完以後至少也要讓他啟動吧，我們在原先nginx的設定檔中加入下面這幾行：

```
package 'nginx'

service 'nginx' do
  supports [:status, :restart, :reload]

  action :start
end
```

在你使用`package`這個resource安裝完nginx的同時，套件本身也幫你增加了一份script到`/etc/init.d/`底下，一般來說我們這時候可以去執行這個script來啟動nginx，但這樣就太遜了，在Chef裡面我們使用`service`這個resource並且指定為nginx的service，預設情況下Chef會去找與你指定service同樣name的process來確定這個服務是否存活，但我們這邊可以使用`supports`來告訴Chef他可以使用`status`這個指令來取代，並且另外支援`restart`與`reload`的指令，最後用`action`來指定其動作是`start`，因此當我們再次執行我們的`solo.rb`，就會有不同的變化了：

```
$ chef-solo solo.rb
Starting Chef Client, version 11.4.0
Compiling Cookbooks...
Converging 2 resources
Recipe: nginx::default
  * package[nginx] action install (up to date)
```

```
    * service[nginx] action start
      - start service service[nginx]

  Chef Client finished, 1 resources updated
```

現在在我們安裝好Nginx的同時，我們也一併啟動了nginx，如果你現在開啟你的瀏覽器來連線到你的VM位置的話，你就會看到你的Nginx已經跑起來了！更多關於Service的設定你可以參考Opscode文件中的說明

運用Chef安裝deploy使用者

Chef本身提供了user這個resource讓你可以用來管理使用者，為了有架構的管理我們的cookbook，我們現在在cookbooks底下新增一個user的cookbook，與剛剛Nginx的部份一樣我們在cookbook裡新增一個default的資料夾，並且在裡面新增default.rb的檔案用來讓我們設定user

```
user 'gogojimmy' do
  password "$1$OxPMgmAb$bLlr2pIPTetowdIPOuCw20"
  gid "admin"
  home "/home/gogojimmy"
  supports manage_home: true
end
```

在這邊我使用user這個resource建立一個名為gogojimmy的user，指定為admin的group、指定家目錄，並且提昇家目錄管理權限，其中password欄位不是直接輸入密碼而是使用openssl加密後的hash字串，再怎麼說設定檔會放在版本控制系統中，加個密總是比較安全，你可以參考Opscode的官方說明做進一步的了解，在這邊產生密碼你可以使用openssl這套工具：

```
openssl -l 'password'
```

再來我們要做的事情就是把我們剛剛做好的recipe加入到node.json的run_list中：

```
{
  "run_list": ["recipe[nginx]", "recipe[user]"]
}
```

再次執行後chef-solo你就會看到chef幫你建立好剛剛你所設定的user了：

```
$ chef-solo solo.rb
Starting Chef Client, version 11.4.0
Compiling Cookbooks...
Converging 3 resources
Recipe: nginx::default
  * package[nginx] action install (up to date)
  * service[nginx] action start (up to date)
Recipe: user::default
  * user[gogojimmy] action create
    - create user user[gogojimmy]
```

其實像是user的名稱寫死在recipe中也不太對勁，recipe重點是執行resouce的動作，因此我們其實可以把像是使用者名稱，資料夾位置等一些設定直接在我們的node.json檔案中寫好，讓recipe去存取就可以了，概念上你可以想像就像是Rails的i18n，現在讓我們把user的設定部分搬到node.json中，記得放在nginx的前面，Chef會按照run_list的順序執行，因為等等我們在Nginx的設定部分會需要到這個user，所以我們必須讓他在Nginx前面執行：

```
{
  "run_list": ["recipe[user]", "recipe[nginx]"],
  "user": {
    "name": "gogojimmy",
    "password": "$1$OxPMgmAb$bLlr2pIPTetowdIPOuCw20"
  }
}
```

而原先recipe的部份我們也要做修改：

```
user node[:user][:name] do
  password node[:user][:password]
  gid "admin"
  home "/home/#{node[:user][:name]}"
  supports manage_home: true
end
```

再次執行chef-solo：

```
$ chef-solo solo.rb
Starting Chef Client, version 11.4.0
Compiling Cookbooks...
Converging 3 resources
Recipe: nginx::default
  * package[nginx] action install (up to date)
  * service[nginx] action start (up to date)
Recipe: user::default
  * user[gogojimmy] action create (up to date)
Chef Client finished, 0 resources updated
```

一樣ok！只是因為我們剛剛已經新增過這個user了，因此他就不會在另外新增一個。

善用template整理設定

我們安裝的套件常常也需要我們編寫另外的設定檔，例如MySQL的my.cnf，或是針對網站本身的Nginx設定檔*.conf，在Chef中我們可以使用template來建立一份我們需要的設定檔，template是一個erb的檔案，意味著你可以在這個template中用變數及運算，例如我們現在先來新增一個Nginx的設定檔，我們在nginx的資料夾下新增一個templates的資料夾，templates下你要建立一個根據node作業系統的版本的資料夾用來存放針對不同作業系統的template，如果Chef找不到對應的作業系統時他就會去找有沒有default的template，因此我們在這邊新增default的資料夾就好，並且在default資料夾下新增一個nginx.conf.erb的檔案

```
server {
  server_name 33.33.33.10;
  root /home/<%= node[:user][:name] %>/demo;
}
```

我們很簡單的設定nginx建立一個host在user底下的demo站台的設定檔，再來我們回到剛剛Nginx的recipe中去告訴Chef使用這個template：

```
template "/etc/nginx/sites-enabled/nginx.conf" do
  source 'nginx.conf.erb'
  notifies :restart, 'service[nginx]', :immediately
end
```

我們使用template這個resource，後面告訴Chef這個檔案要被存在哪個地方，以上面為例就是/etc/nginx/sites-available/nginx.conf，最後使用source的指令告訴Chef要去哪邊找這支template來用，notifies這個指令是一個callback，用來告訴Chef當這邊執行完畢的時候要去呼叫resource執行動作，並且指派等級是immediately，預設等級會是delayed，會在Chef執行完所有的工作後才執行，現在萬事具備，我們再來只要在user底下建立我們需要的demo資料夾以及一個簡單的 'index.html '檔案就可以了，在這邊我們也可以使用Chef來完成，為求迅速我直接將這部分寫在nginx的recipe中，現在這個reicipe的全貌會是這樣：

```
package 'nginx'

service 'nginx' do
  supports [:status, :restart, :reload]
  action :start
end

directory "/home/#{node[:user][:name]}/demo" do
  owner node[:user][:name]
end

file "/home/#{node[:user][:name]}/demo/index.html" do
  owner node[:user][:name]
  content "<h1>Hello gogojimmy!</h1>"
end

template "/etc/nginx/sites-enabled/nginx.conf" do
  source 'nginx.conf.erb'
  notifies :restart, 'service[nginx]', :immediately
end
```

這邊不多做解釋了，原理都是一樣的，再來我們就直接給他執行下去：

```
$ chef-solo solo.rb
Starting Chef Client, version 11.4.0
Compiling Cookbooks...
Converging 6 resources
Recipe: nginx::default
  * package[nginx] action install (up to date)
  * service[nginx] action start (up to date)
  * template[/etc/nginx/sites-available/nginx.conf] action create (up to
      date)
  * directory[/home/gogojimmy/demo] action create
    - create new directory /home/gogojimmy/demo
    - change owner from '' to 'gogojimmy'
server {
```

```
  * file[/home/gogojimmy/demo/index.html] action create
    - create new file /home/gogojimmy/demo/index.html with content
       checksum 375690
        --- /tmp/chef-tempfile20130530-5037-1gsrh3h     2013-05-30
           16:11:00.741313182 +0000
        +++ /tmp/chef-diff20130530-5037-h4yu39     2013-05-30
           16:11:00.741313182 +0000
        @@ -0,0 +1 @@
        +<h1>Hello World!</h1>

Recipe: user::default
  * user[gogojimmy] action create (up to date)
Chef Client finished, 2 resources updated
```

接下來在你host機器上的瀏覽器重新整理剛剛的網頁，你應該就能看到我們的結果了！至此為止，你已經完整的設定了一台機器，接下來你可以複製這些cookbooks下來，重新開一台新的VM然後灌好Chef-Solo，接著把複製下來的cookbooks丟上去後執行，Chef-Solo就會一氣呵成的幫你完成你剛剛做的那些作業，這輩子再也不用擔心裝機器的步驟了。

搭配Knife讓Chef更好用

即使Chef是這麼好用的工具，在使用上還是會有些讓人覺得麻煩的地方，例如說明明就是一個固定的資料結構但你得一直新增資料夾，在你想要安裝機器前你還必須手動先安裝Chef-Solo，剛剛所有的操作都在VM上，實務上如果我們使用VPS的時候，一直與VPS溝通耗時又耗力，為了解決這些問題，你就必須給Chef一把Knife，Knife是Opscode推出的一套command line的Chef管理工具，你可以在你的workingstation的機器使用Knife來管理VPS上的Chef，你甚至不用手動去幫機器安裝Chef-Solo，讓我們看看Knife可以做什麼，在我們的workingstation的機器上，讓我們先安裝好knife，針對我們所使用的是Chef-Solo，因此我們要安裝的是Knife-Solo：

```
$ gem install knife-solo
```

裝好了`knife-solo`以後，讓我們使用`knife solo init`來開啟一個新專案

```
$ knife solo init knife-solo-demo
$ cd knife-solo-demo
$ tree
.├──
 cookbooks├──
 data_bags├──
 nodes├──
 roles├──
 site-cookbooks└──
 solo.rb
```

沒錯！只要一個指令他就幫你把Chef基本的專案結構建立完成了！再來我們進行下一步之前我們先產生一個chef的設定檔，基本上這個設定檔可有可無，但是如果沒有的話每次你打指令的時候他會一直叫警告實在很惱人，因此我們還是先來產生一個設定檔：

```
$ knife configure -r . --defaults
```

讓我們準備一台全新的VM，建立好後請指定好他的IP位置，不知道怎麼建立跟設定VM的請參考鍵人矬作使用Vagrant練習環境佈署，再來就是厲害的地方了：

```
$ knife solo bootstrap vagrant@33.33.33.10
Bootstrapping Chef...

  % Total    % Received % Xferd  Average Speed   Time    Time     Time
     Current
                                 Dload  Upload   Total   Spent    Left
                                    Speed
100  6510  100  6510    0     0   2649      0  0:00:02  0:00:02 --:--:--
     4389
Downloading Chef  for ubuntu...
Installing Chef
Selecting previously unselected package chef.
(Reading database ... 62744 files and directories currently installed.)
Unpacking chef (from .../tmp.scDzMEID/chef__amd64.deb) ...
Setting up chef (11.4.4-2.ubuntu.11.04) ...
Thank you for installing Chef!
Generating node config 'nodes/33.33.33.10.json'...
vagrant@33.33.33.10's password:
Starting Chef Client, version 11.4.0
Compiling Cookbooks...
Converging 0 resources
Chef Client finished, 0 resources updated
```

看到沒有！`knife solo bootstrap`這個指令直接幫我們連到VM去安裝好Chef並且還上傳了我們的VM，其實`knife solo bootstrap`是`knife solo prepare`與`knife solo cook`兩個指令的組合，前面那個指令會幫我們的機器下載安裝Chef，而後面的指令是幫我們上傳本地端的cookbooks以及執行`chef-solo`，所以這個指令也只會用在第一次，再來以後我們都只要`cook`、`cook`、`cook`就對了！

再來讓我們跟剛剛一樣建立一個Nginx的設定吧，這次我們不用再慢慢的手工打造，讓我們直接用knife幫我們建立：

```
$ knife cookbook create nginx
** Creating cookbook nginx
** Creating README for cookbook: nginx
** Creating CHANGELOG for cookbook: nginx
** Creating metadata for cookbook: nginx
```

就跟剛剛一樣，knife就會幫我們在cookbooks下面建立好nginx的cookbook的基本專案結構，並且附上`default.rb`的基本設定檔等著我們去使用了，現在讓我們把剛剛學到的建立user跟nginx的相關設定都給他複製上去，過程就不贅述了只是複製貼上的工作我相信你行的，唯一要注意的是原先我們使用的`node.json`這支檔案現在不在根目錄下，而是在`nodes`這個資料夾下幫我們建立了一個以你設定的VM ip位置為檔名的`json`檔，記得把你的`run_list`以及參數的設定寫在這支檔案中，寫完後我們再`cook`一次

```
$ knife solo cook vagrant@33.33.33.10
Checking Chef version...
[2013-05-31T19:03:31+00:00] INFO: *** Chef 10.12.0 ***
```

```
[2013-05-31T19:03:32+00:00] INFO: Setting the run_list to ["recipe[user]
   ", "recipe[nginx]"] from JSON
[2013-05-31T19:03:32+00:00] INFO: Run List is [recipe[user], recipe[
   nginx]]
[2013-05-31T19:03:32+00:00] INFO: Run List expands to [user, nginx]
[2013-05-31T19:03:32+00:00] INFO: Starting Chef Run for vagrant-ubuntu-
   quantal-64
[2013-05-31T19:03:32+00:00] INFO: Running start handlers
[2013-05-31T19:03:32+00:00] INFO: Start handlers complete.
[2013-05-31T19:03:32+00:00] INFO: Processing user[gogojimmy] action
   create (user::default line 10)
[2013-05-31T19:03:32+00:00] INFO: Processing package[nginx] action
   install (nginx::default line 10)
[2013-05-31T19:03:32+00:00] INFO: Processing service[nginx] action start
    (nginx::default line 12)
[2013-05-31T19:03:32+00:00] INFO: service[nginx] started
[2013-05-31T19:03:32+00:00] INFO: Processing directory[/home/gogojimmy/
   demo] action create (nginx::default line 17)
[2013-05-31T19:03:32+00:00] INFO: Processing file[/home/gogojimmy/demo/
   index.html] action create (nginx::default line 21)
[2013-05-31T19:03:32+00:00] INFO: Processing template[/etc/nginx/sites-
   enabled/nginx.conf] action create (nginx::default line 26)
[2013-05-31T19:03:32+00:00] INFO: Chef Run complete in 0.425268589
   seconds
[2013-05-31T19:03:32+00:00] INFO: Running report handlers
[2013-05-31T19:03:32+00:00] INFO: Report handlers complete
```

這時候你在打開瀏覽器連線，你會發現已經成功上線了，一切都是那麼沒好，第一次做完的時候我興奮的立馬把VM殺掉直接重新跑`knife solo bootstrap`**，結果就會是這樣：**

```
$ knife solo bootstrap vagrant@33.33.33.10
Bootstrapping Chef...
Enter the password for vagrant@33.33.33.10:
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
     Current
                                 Dload  Upload   Total   Spent    Left
                                    Speed
100  6510  100  6510    0     0   6755      0 --:--:-- --:--:-- --:--:--
    10742
Downloading Chef  for ubuntu...
Installing Chef
(Reading database ... 63748 files and directories currently installed.)
Preparing to replace chef 10.12.0-2 (using .../tmp.EMBKgQMR/chef__amd64.
   deb) ...
 * Stopping chef-client chef-client
                                              [ OK ]
Unpacking replacement chef ...
```

```
dpkg: warning: unable to delete old directory '/var/log/chef': Directory
    not empty
dpkg: warning: unable to delete old directory '/etc/chef': Directory not
    empty
Setting up chef (11.4.4-2.ubuntu.11.04) ...
Thank you for installing Chef!
Processing triggers for man-db ...
Processing triggers for ureadahead ...
Starting Chef Client, version 11.4.4
Compiling Cookbooks...
Converging 6 resources
Recipe: user::default
  * user[gogojimmy] action create
    - create user user[gogojimmy]

Recipe: nginx::default
  * package[nginx] action install
    - install version 1.2.1-2.2ubuntu0.1 of package nginx

  * service[nginx] action start
    - start service service[nginx]

  * directory[/home/gogojimmy/demo] action create
    - create new directory /home/gogojimmy/demo
    - change owner from '' to 'gogojimmy'

  * file[/home/gogojimmy/demo/index.html] action create
    - create new file /home/gogojimmy/demo/index.html with content
       checksum 60ce4d
        --- /tmp/chef-tempfile20130531-3432-c8tneh     2013-05-31
           19:14:01.334694350 +0000
        +++ /tmp/chef-diff20130531-3432-1q1fx4z     2013-05-31
           19:14:01.334694350 +0000
        @@ -0,0 +1 @@
        +<h1>Hello gogojimmy!</h1>

  * template[/etc/nginx/sites-enabled/nginx.conf] action create
    - create template[/etc/nginx/sites-enabled/nginx.conf]
        --- /tmp/chef-tempfile20130531-3432-18v20jr     2013-05-31
           19:14:01.418694347 +0000
        +++ /tmp/chef-rendered-template20130531-3432-w0bdgr
           2013-05-31 19:14:01.414694348 +0000
        @@ -0,0 +1,4 @@
        +server {
        +  server_name 33.33.33.10;
        +  root /home/gogojimmy/demo;
```

```
          +}

    * service[nginx] action restart
      - restart service service[nginx]

  Chef Client finished, 7 resources updated
```

一鍵完成阿！！！！看到這你能不流淚嗎？想到過往裝機器而逝去的那些青春，我就不禁鼻酸啊(擦眼角

最後，其實Opscode本身的社群有非常豐富的資源，有很多其他人已經寫好的`cookbook`你可以直接下載使用，畢竟我們就是不喜歡造輪子，你可以在這裡找到這些cookbook，想像就像是Ruby的ruby-toolbox吧！再來在下篇要跟大家分享的是如何使用Chef搭配Capistrano進行佈署的策略，現在我把螢幕切還給你們，各位同學可以自己練習看看，有什麼不懂的，可以在下面提問。

Q&A

Q：Chef裡面那些Cookbooks啊..Recipe啊…這些名詞好容易搞混，可以解釋一下嗎？

A：關於這個問題，當初我在看文件的時候也覺得這些洋人裝什麼生活化，再怎麼生活化人家看我們還不是一堆宅宅嗎？Anyway我也還寫不出一套Chef，所以我們還是遷就一下洋人，不過我可以為這些名次跟你做些解釋：

- `node`：指的就是你要安裝的機器
- `workstation`：指的就是你用來編輯食譜的機器，也就是你用來上PTT跟Facebook跟寫程式的這台電腦
- `kitchen`：指的就是你的chef專案所在地，有`solo.rb`所在的地方
- `recipe`：指的就是你用Ruby DSL語法寫成用來指示`node`該怎麼進行安裝設定的文件
- `cookbook`：不只存放`recipe`，同時也存放了像是`template`、`attributes`等resource的地方，就是一個含有安裝步驟、設定、樣式的食譜集
- `resource`：一個抽象的概念，通常就是指“套件”、“使用者”這種跟平台是什麼無關的抽象名詞。
- `data bag`：就是你用來設定`node`一些參數的JSON檔

Q：既然`knife-solo`這麼方便，為什麼不一開始跟大家說這個就好了呢，幹嘛花那麼多時間在這上面，`mkdir`打多了也是很煩人的

A：難道你以為我打字有比較輕鬆嗎？我這都是為了讓你更能仔細了解Chef的檔案結構啊，如此用心良苦你是不是應該移動你的滑鼠去下面幫我點個廣告呢？

實用連結及參考資訊

- knife-solo
- chef-solo-basic