Mục lục

[**A.** **Vagrant** 2](#_Toc472529809)

[**1.** **Các khái niệm cơ bản** 2](#_Toc472529810)

[**1.1.** **Vagrant là gì ?** 2](#_Toc472529811)

[**1.2.** **Lợi ích của việc sử dụng Vagrant** 3](#_Toc472529812)

[**1.3.** **VirtualBox** 4](#_Toc472529813)

[**2.** **Cài đặt** 5](#_Toc472529814)

[**2.1.** **Cài đặt VM VirtualBox** 6](#_Toc472529815)

[**2.2.** **Cài đặt Git Bash** 7](#_Toc472529816)

[**2.3.** **Cài đặt Vagrant** 9](#_Toc472529817)

[**2.4.** **Cài đặt PuTTy** 14](#_Toc472529818)

[**2.5.** **Cài đặt WinSCP** 14](#_Toc472529819)

[**2.6.** **Làm việc với Vagrant** 25](#_Toc472529820)

[**2.7.** **Config file Vagrantfile** 37](#_Toc472529821)

[**2.8.** **Lỗi phát sinh khi sử dụng vagrant** 38](#_Toc472529822)

[**3.** **Tổng kết** 42](#_Toc472529823)

[**B.** **Ansible** 43](#_Toc472529824)

[**1.** **Khái niệm cơ bản** 43](#_Toc472529825)

[**2.** **Cài đặt Ansible trong Vagrant** 43](#_Toc472529826)

[**C.** **Áp dụng** 43](#_Toc472529827)

**STEP2 : Cài đặt môi trường với Vagrant, tìm hiểu Gem và các khái niệm liên quan(24/8-28/8)**

1. **Vagrant**

Từ hôm nay mình sẽ học cách sử dụng Vagrant. Vậy vagrant là gì ? Nó dùng để làm gì ? Mình sẽ dùng nó để học Rails như thế nào?

Về cách cài đặt Vagrant trên mạng có rất nhiều trang hướng dẫn. Mình sẽ ghi chú rõ ràng về việc cài đặt cũng như sử dụng nó.

1. **Các khái niệm cơ bản**
   1. **Vagrant là gì ?**

Đây là bài viết khá chi tiết và cụ thể về Vagrant trên viblo blog

Tham khảo : <https://viblo.asia/dinhhoanglong91/posts/1l0rvmDQGyqA>

Vấn đề tồn tại: Trong 1 team sẽ xảy ra trường hợp mỗi 1 lập trình viên làm việc trên 1 hệ điều hành khác nhau. Fan cuồng Windows cài hệ điều hành Windows để làm việc. Fan của những chú chim cánh cụt thì cài các bản phân phối của Linux. Fan quả táo thì cài MacOsX. Làm trong 1 nhóm mà lại không chung môi trường làm việc đôi khi sẽ xảy ra trường hợp mà chỉ những người dùng Windowsw bị mà mấy ông dùng Linux với Unix không bị. Lúc đó lại mất thời gian đi dò bug, đổ tại Windows lởm, blablo.... Mệt mỏi vô cùng.

Thật may mắn khi Vagrant của công ty HashiCorp ra đời, nó đã giúp giải quyết bài toán hóc búa trên. Chỉ với vài cài đặt cả nhóm đã có thể làm việc trên cùng 1 môi trường.

Đơn giản nó được giải thích như sau: Các bạn vẫn cứ code trên hệ điều hành mà bạn thích còn việc biên dịch hay thông dịch code thì cứ để máy ảo lo và Vagrant sẽ làm công việc đồng bộ dữ liệu của bạn với môi trường máy ảo. Chiếc máy ảo này sẽ được share cho các thành viên trong nhóm.

Vagrant là 1 công cụ xây dựng và quản lý các máy ảo, có thể chạy trên Ubuntu, MacOS và cả Windows. Vagrant có thể quản lý nhiều loại máy ảo khác nhau, tự động cài đặt phần mềm cho máy tính mà không quan tâm đang sử dụng OS hay distribution nào.

* 1. **Lợi ích của việc sử dụng Vagrant**

**+ Với Developer**

Tất cả các phần mềm cũng như config khác đều được thực hiện bởi người tạo ra Vagrantfile, tất cả các thành viên sẽ có cùng 1 môi trường phát triển. Không phân biệt bạn dùng Mac, Ubuntu hay Windows, bạn sẽ có cùng môi trường phát triển với các thành viên khác và thậm chí với cả production server. Điều này giảm thiểu được thời gian cài đặt phần mềm, giảm thiểu các bug chỉ xảy ra ở 1 môi trường nhất định

**+ Với Operations Engineer**

Bạn có thể setup cùng lúc nhiều VM networks 1 cách đơn giản: Nếu chỉ dùng VirtualBox đơn thuần, bạn cần setup lần lượt cho từng máy trong cả hệ thống server. Còn với vagrant, bạn chỉ cần 1 file config và có thể build/edit cùng 1 lúc nhiều máy 1 cách đơn giản.

Quản lý source 1 cách đơn giản: Bằng việc đẩy tất cả setup vào 1 textfile và quản lý bới subversion control (git chẳng hạn), bạn sẽ dễ dàng kiểm soát các thiết lập của mình. Trong trường hợp có lỗi cũng dễ dàng check thay đổi hoặc revert về trạng thái không lỗi.

Đa nền tảng: Với hệ thống boxes đa dạng và phong phú, bạn dễ dàng thử với nhiều OS khác nhau cũng như distribution khác nhau. Ngoài ra, với việc sử dụng các provision tool, bạn dễ dàng cài đặt phần mềm trên nhiều dòng OS khác nhau.

**+ Với Design**

Bạn không còn phải cố gắng thiết lập website chạy trên máy mình 1 cách khó khăn với hàng tá intruction, bạn cũng không cần phải nhìn xem design của mình có chạy ổn trên máy của developer không, điều bạn cần chỉ là cài đặt vagrant, và vagrant up sẽ làm phần còn lại cho bạn. Sau đó, bạn có thể quay lại với công việc của mình, design!

->Mấy cái này toàn lý thuyết, có nói cũng chả hiểu gì, thôi cứ bắt tay vô làm rồi không hiểu thì hỏi thầy google nhỉ ☺

* 1. **VirtualBox**

Sơ lược qua về VirsualBox

Tham khảo: <http://forum.bkav.com.vn/forum/bkav-forum/bkav-contest/29597-huong-dan-tao-may-ao-bang-virtualbox-complete>

Đây là 1 bài viết rất kĩ về VirtualBox.

1. **Kiến thức cơ bản về VirtualBox**

VirsualBox (viết tắt là VB) là gì? VirtualBox (Oracle VM VirtualBox) là một phần mềm ảo hóa miễn phí. Trước kia nó thuộc về Microsystem nhưng bây giờ VirtualBox đã thuộc sở hữu của Oracle. VirtualBox có thể chạy được trên nhiều hệ điều hành khác nhau, bao gồm Linux (Ubuntu), OS X, Windows,...

1. **Ảo hóa trên VirtualBox là như thế nào?**

Đầu tiên VirtualBox được cài vào máy tính, máy tính cài đặt VirtualBox có thể chạy hệ điều hành Linux, OS X, Windows, …

Thứ hai, phần mềm này sẽ tận dụng phần cứng của máy tính hiện tại của bạn để cho phép VirtualBox chạy nhiều hệ điều hành tại cùng một thời điểm. Bạn có thể chạy bao nhiêu máy ảo cũng được miễn là phần cứng của bạn cho phép, quan trọng nhất là RAM và dung lượng trống trên ổ cứng.

VD: bạn có thể cài đặt và chạy hệ điều hành Windows và OS X trên Linux hoặc chạy Linux trên Windows hoặc Mac, …

1. **Thuật ngữ thường xuyên được sử dụng trong VirtualBox**

Có 1 số thuật ngữ quan trong sử dụng trong VirtualBox mà bạn nên biết và nên nhớ vì mình sẽ sử dụng nhiều trong tài liệu này:

+ Host operating system(host OS): hay còn gọi là host, đây là hệ điều hành chạy trên máy thật(máy tính vật lý), là nơi VirtualBox được cài đặt(chính là hệ điều hành mà máy tính của ta đang sử dụng).

+ Guest operating system(guest OS): hay còn gọi là guest, đây là hệ điều hành chạy bên trong máy ảo.

+ Virtual machine(VM): Còn gọi là máy ảo, đây là nơi chạy guest OS. Nói dễ hiểu hơn thì VM là cửa sổ bạn thấy khi guest OS chạy.

+ Guest Additions: Đây là những gói phần mềm đặc biệt, được cài đặt bên trong VM để cải thiện hiệu năng của guest OS và thêm một số tính năng, ví dụ như tự động điều chỉnh độ phân giải màn hình, Shared Folder (chia sẻ thư mục), Shared clipboard (copy text giữa hai máy, …), Drag’n’Drop (kéo thả file giữa 2 máy), ….

Nội dung được tham khảo tại: <http://ubunsys.com/ky-thuat/gioi-thieu-virtualbox-oracle-vm-virtualbox/13.06.2015/>

1. **Cài đặt**

Vagrant được viết bằng Ruby, đây là 1 repository thuộc hang top trending của Ruby trên Github, bạn có thể xem source tại trang:

<https://github.com/mitchellh/vagrant>

Bạn cũng không cần phải biết Ruby hay có sẵn Ruby trong máy mới có thể cài đặt được , mà chỉ cần tham khảo 1 số bài viết hướng dẫn cài đặt mình cung cấp sau đây.

Đây là một số site mình tham khảo để cài đặt Vagrant:

+ <https://c6h0st.wordpress.com/2015/06/13/tutorial-huong-dan-cai-dat-va-su-dung-vagrant/>

+ <http://fsd14.com/post/160-huong-dan-cai-dat-vagrant-tao-moi-truong-lam-viec-chung>

+ <http://iam.hoangstark.com/laravel/2014/11/27/php-composer-homestead-tren-windows.html>

Tuy nhiêm, để sử dụng Vagrant và trở thành master Vagrant thì bạn cần có kiến thức về Ruby.

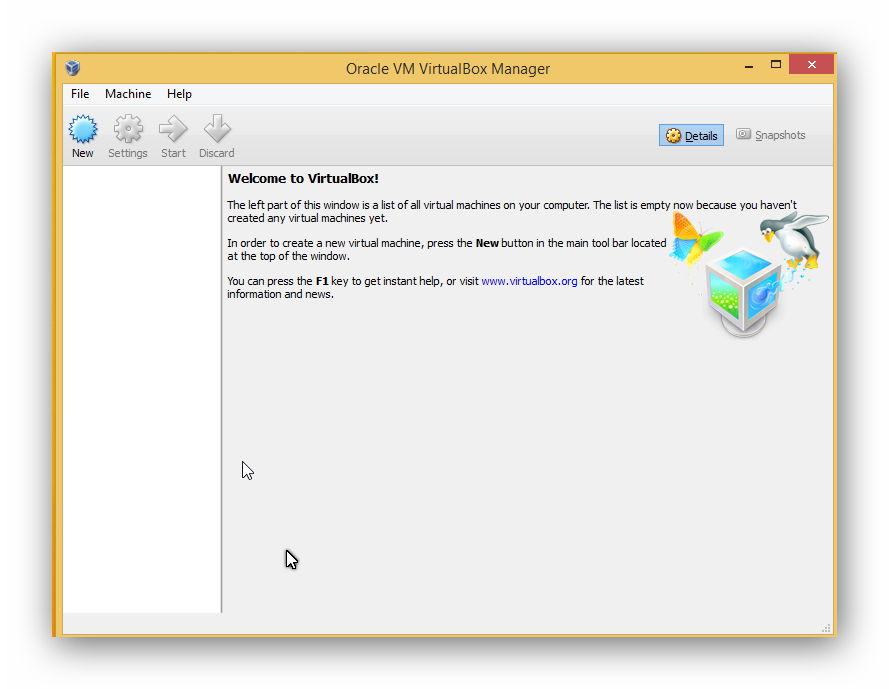
Nào, bây giờ mình hãy bắt tay vào cài đặt nhỉ

* 1. **Cài đặt VM VirtualBox**

Trước khi cài Vagrant, bạn nên cài VirtualBox để có thể tạo ra máy ảo trên máy tính của bạn. Bạn có thể tải phần mềm VirtualBox về và cài đặt vào máy như một phần mềm bình thường, yên tâm đây là phần mềm hoàn toàn miễn phí.

Link down: <https://www.virtualbox.org/wiki/Downloads>

Do mình đang sử dụng OS Windows7 64bit nên mình sẽ down bản Virtual Box 5.0.2 for Windows host.(Mới nhất tính đến thời điểm hiện tại ngày 24/08/2015)

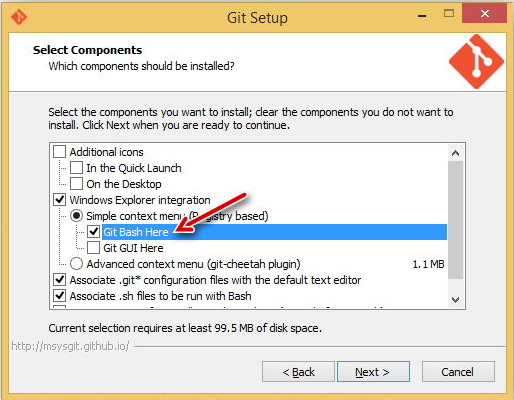


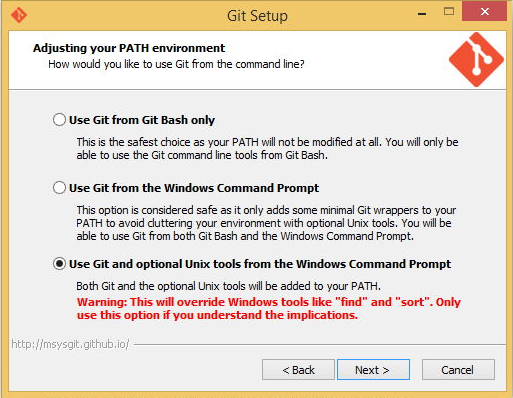
**Update**: Hiện tại VirtualBox đã release đến version 5.1 (tính đến ngày 2016/08/28) , tuy nhiên sau 1 quá trình sử dụng thì mình vẫn khuyên các bạn nên sử dụng phiên bản 5.0.x thay vì dùng 5.1. Phiển bản 5.1 sẽ gây ra 1 số lỗi khó chịu khi làm việc với vagrant, đúng hơn là với các boxes của vagrant.

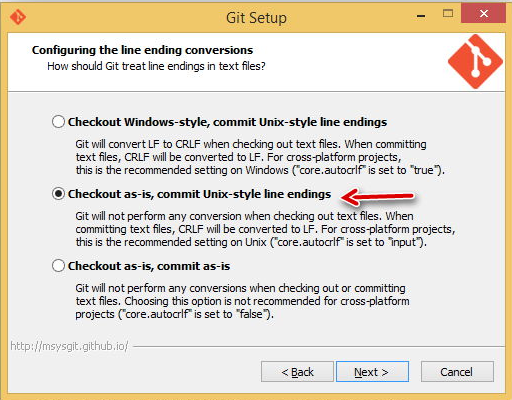
* 1. **Cài đặt Git Bash**

Tại sao lại có Git ở đây? Bởi vì chúng ta sẽ sử dụng và điều khiển Vagrant qua bằng lệnh linux nên nếu dùng Windows thì sẽ hơi bất tiện khi dùng cái ứng dụng cmd có sẵn. Do vậy chúng ta cần một phần mềm để hỗ trợ giả lập môi trường UNIX trên Windows và công cụ Git có hỗ trợ cái này, mặc khác Vagrant rất có ích trong việc dùng chung với Git nên cứ cài sẵn để sau này cần thì dùng chung luôn.

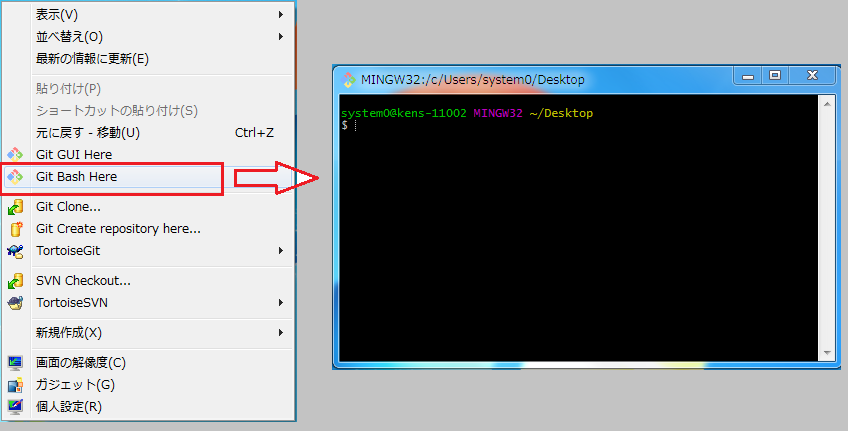
Đầu tiên bạn truy cập vào: <http://git-scm.com/download/win> để tải Git dành cho Windows phiên bản mới nhất. Sau đó chạy file vừa tải về để cài đặt và chọn tùy chọn như hình dưới.







Sau khi cài xong, bạn có thể thấy máy tính mình có thêm một ứng dụng tên là Git Bash và Git GUI khi vào menu Start trên máy tính, chúng ta chỉ cần sử dụng Git Bash trong khuôn khổ bài viết này. Mở lên thì bạn sẽ có cửa sổ nhập lệnh như thế này:



Tạm thời để đó đi, chúng ta cần cài thêm Vagrant nữa.

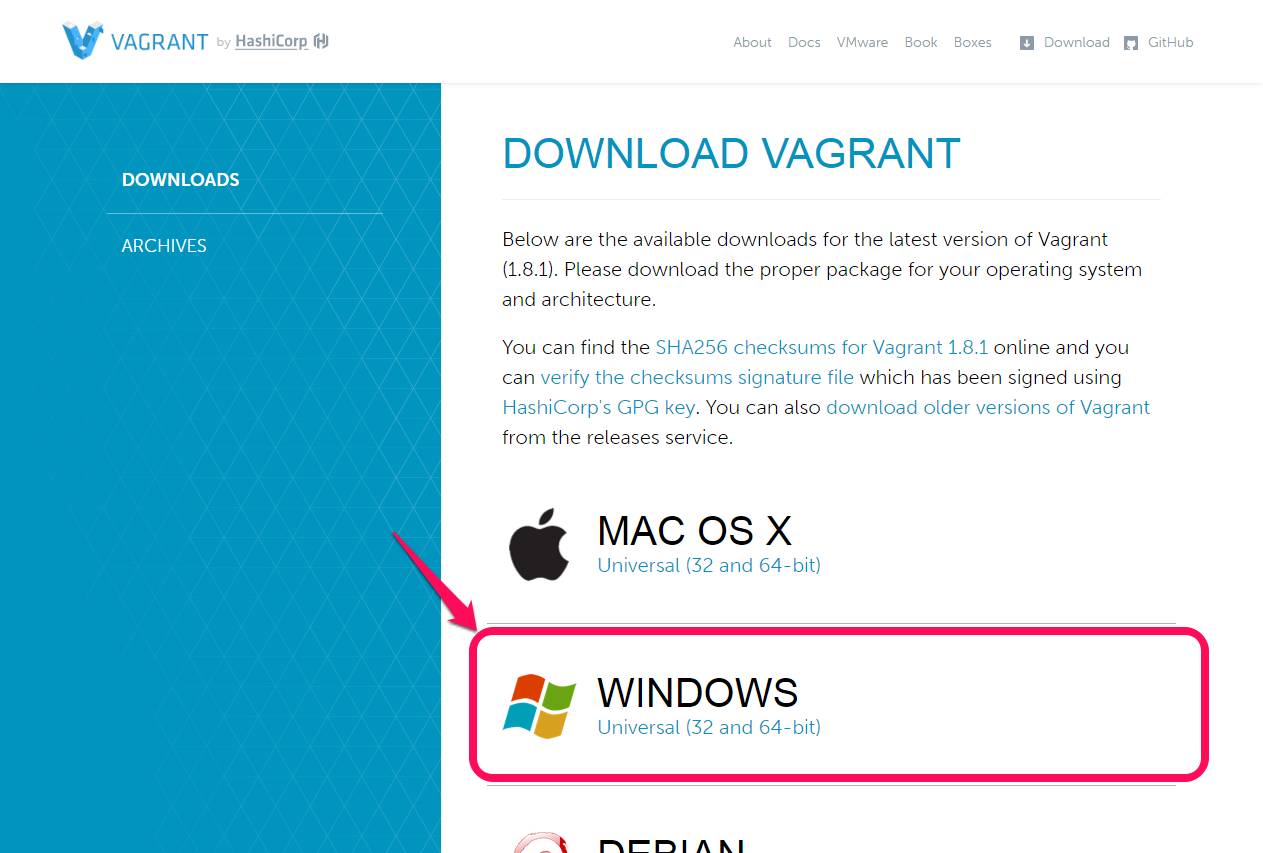
* 1. **Cài đặt Vagrant**

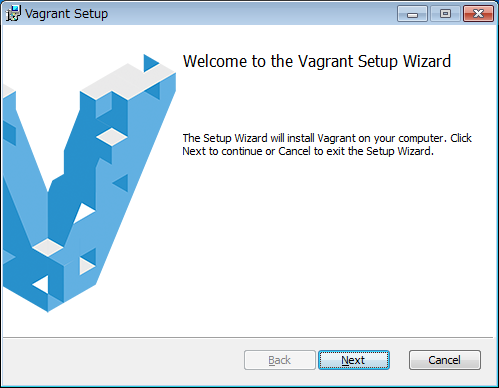
Dowload: <https://www.vagrantup.com/downloads.html>

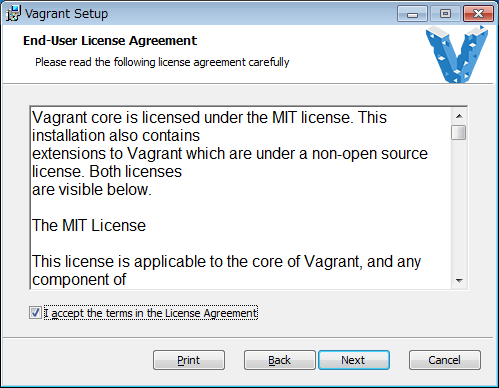
Chọn phiên bản phù hợp với OS đang sử dụng, ở đây mình dùng Windows 7 64 bit.

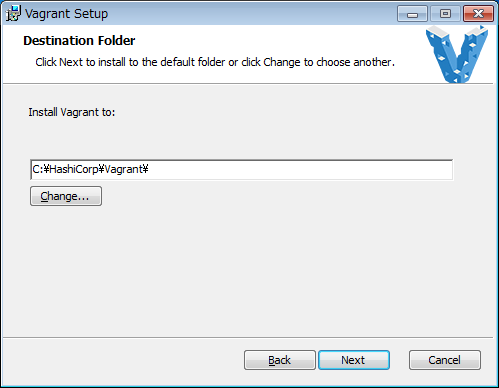
Vagrant hiện tại mình sử dụng là bản: vagrant\_1.7.4 (mới nhất tính đến ngày 24/08/2015)

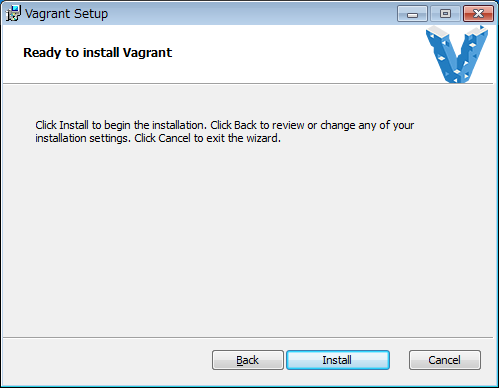
Các bước cài đặt

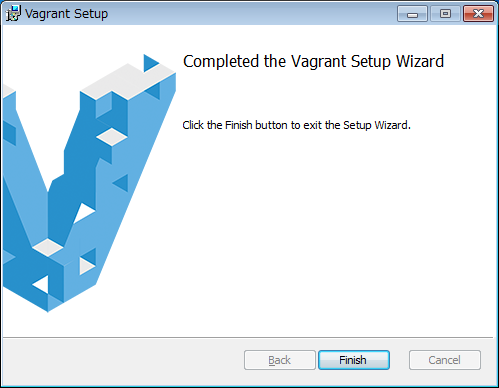






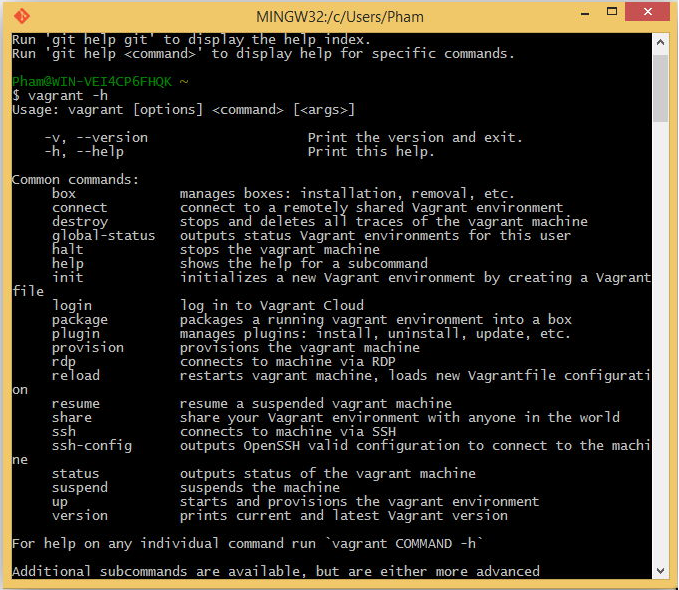






Sau khi cài đặt xong nhớ restart máy.Bây giờ bạn mở git bash ra để kiểm tra xem vagrant đã cài đăt thành công hay chưa

>vagrant –h



Một số **lệnh Vagrant cơ bản cần phải nhớ**

Trong khi dùng Vagrant, có thể bạn sẽ sử dụng rất nhiều một số lệnh dưới đây nên tốt nhất là hãy nhớ chức năng của nó:

Trong vagrant thì mọi cú pháp đều phải bắt đầu với “vagrant”

* vagrant box add tên\_box // Download máy ảo
* vagrant init // Cài đặt vagrant ở folder hiện tại
* vagrant box list – Xem danh sách các box.
* vagrant up // Khởi động máy ảo
* vagrant halt // Tắt máy ảo
* vagarnt provision // Cập nhật cấu hình từ file Vagrantfile cho máy ảo khi nó đang chạy
* vagrant reload: Restart và tải lại các thiết lập trong file Vagrantfile của máy ảo, khi đổi nội dung file đó bạn phải sử dụng lệnh vagrant halt trước để tắt máy ảo, sau đó sử dụng lệnh reload này để nạp lại cấu hình.
* vagrant ssh // Truy cập vào máy ảo qua ssh. Máy thật lúc này đóng vai trò là 1 client.
* vagrant destroy tên\_box: Cho máy ảo về vườn(Xóa máy ảo)
* vagrant suspend: Cho máy ảo tạm nghỉ.
* vagrant login: Đăng nhập vào hệ thống Vagrant Cloud.
* vagrant share --ssh: Chia sẻ máy ảo của bạn cho người khác truy cập, bạn phải gõ lệnh vagrant login trước khi dùng tính năng này.

Tham khảo tại:

+ <http://thachpham.com/hosting-domain/may-chu/tao-may-ao-tren-may-tinh-voi-vagrant-va-virtualbox.html>

* 1. **Cài đặt PuTTy**

PuTTy là một tool được sử dụng trong việc tạo kết nối đến server thông qua ssh.

<http://lamthenao.com/internet/huong-dan-su-dung-putty-de-truy-cap-va-cai-dat-server.html>

<https://vinahost.vn/ac/knowledgebase/84/Hng-dn-ssh-vao-server-bng-chng-trinh-Putty.html>

* 1. **Cài đặt WinSCP**

WinSCP là ứng dụng truyền tải file đầu cuối dành cho hệ điều hành Windows với giao thức SSH để truyền tải file một cách an toàn. WinSCP còn hỗ trợ giao thức SFTP, FTP, và cả SCP.

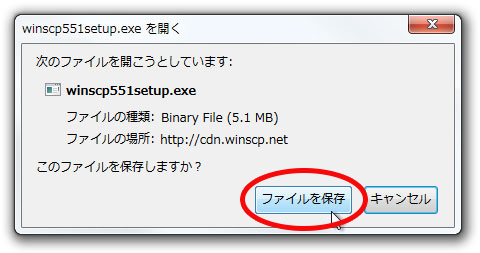
Chương trình thường được dùng để truyền file giữa máy nội bộ và một máy riêng biệt từ xa thông qua giao thức bảo mật cao. Bạn hãy tải về dùng thử WinSCP để chia sẽ file một cách an toàn.

WinSCP sẽ được sử dụng sau khi tạo guestOS trên VirtualBox với vagrant. WinSCP sẽ làm vai trò kết nối giữa hostOS với guestOS và có thể trực tiếp thao tác thông qua truyền nhận file giữa 2 OS. Giả sử bây giờ bạn làm việc ở Windows OS, guestOS là CentOS , vậy làm sao để bạn push source code lên môi trường guest để thực hiện việc test. WinSCP sẽ hỗ trợ bạn việc đó. Mình sẽ nói kĩ hơn về việc kết nối và sử dụng WinSCP khi thao tác tạo 1 ứng dụng thực tế.

Cài đặt:

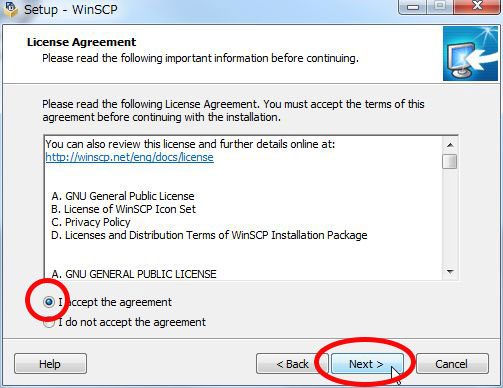
<https://vinafix.vn/threads/huong-dan-su-dung-winscp-quan-ly-file-don-gian-qua-sftp.3285/>

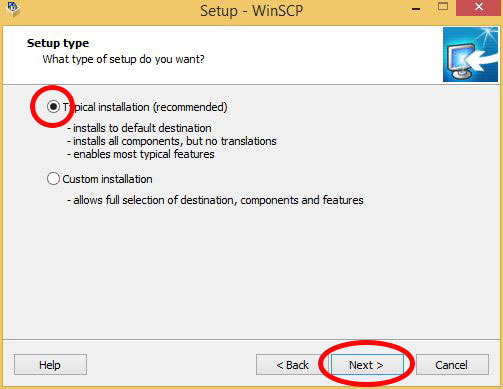


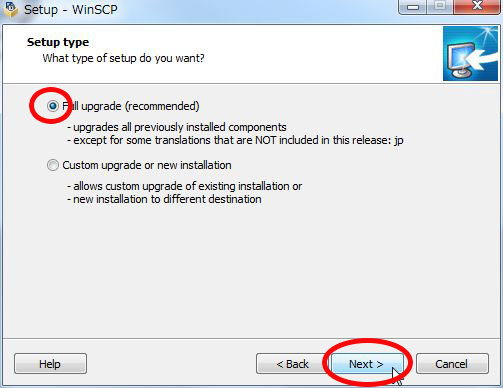


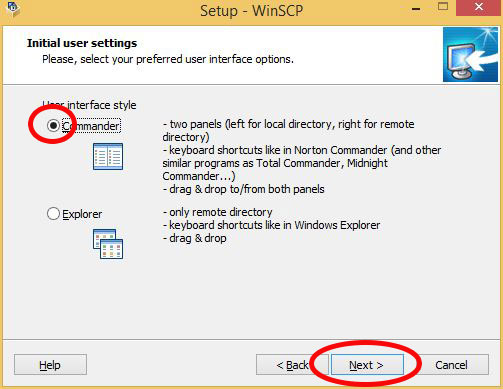


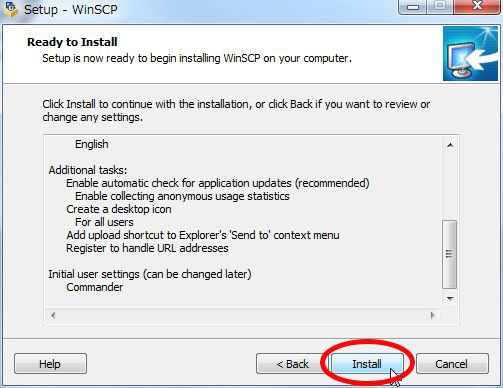










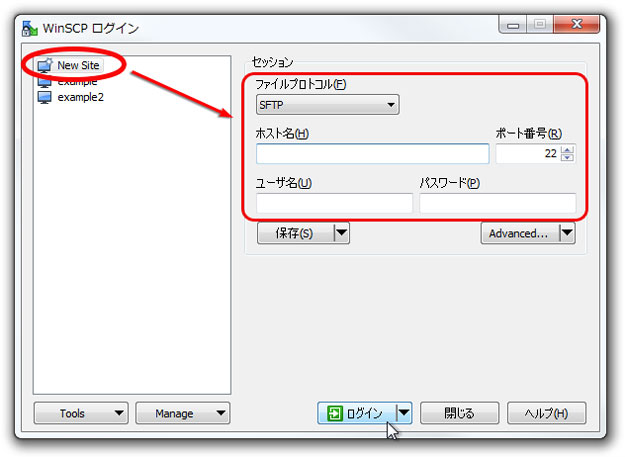




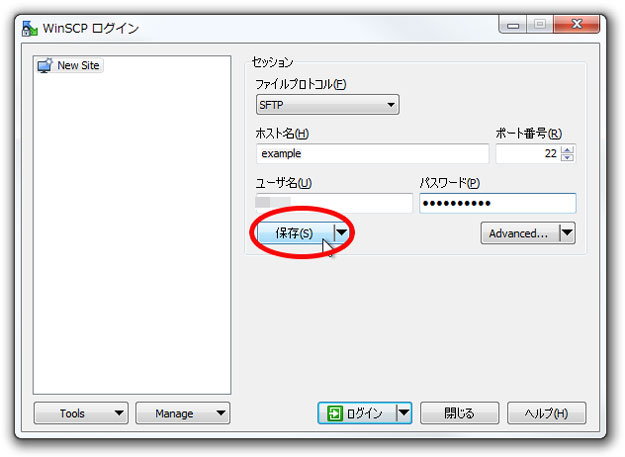
Cách sử dụng đã được viết rất kĩ trong site này:

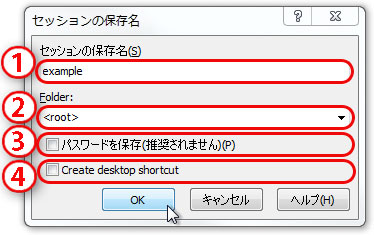
+ <http://synclogue-navi.com/winscp5-manual>

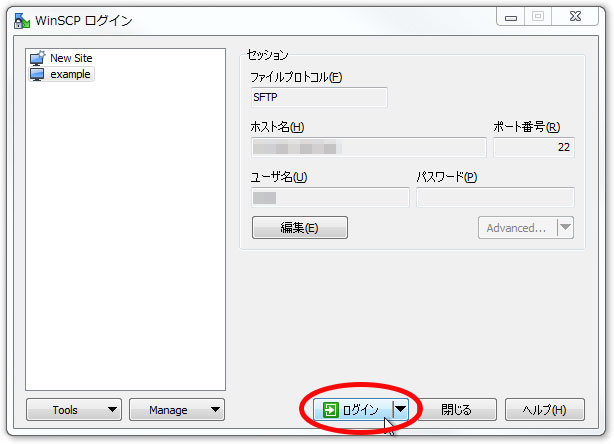
Tạo 1 session mới



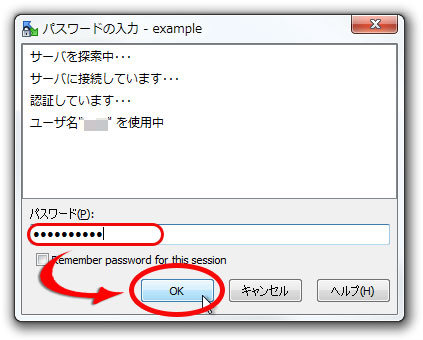


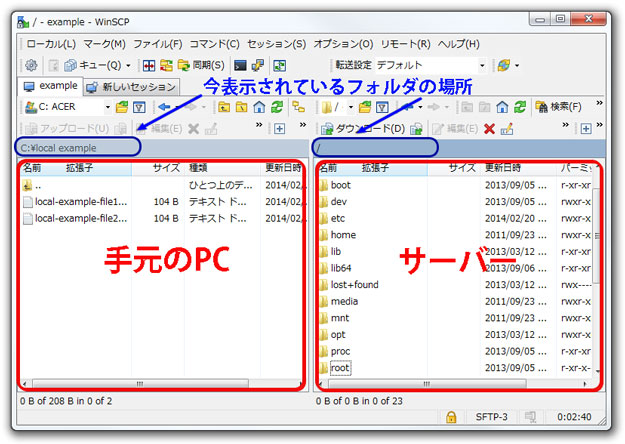


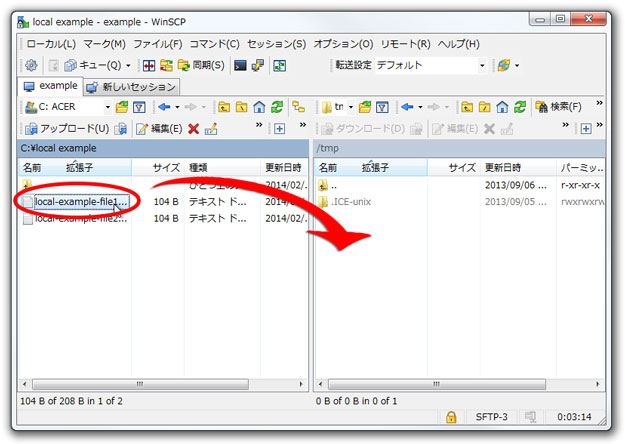


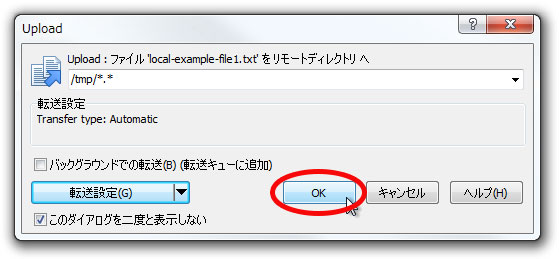


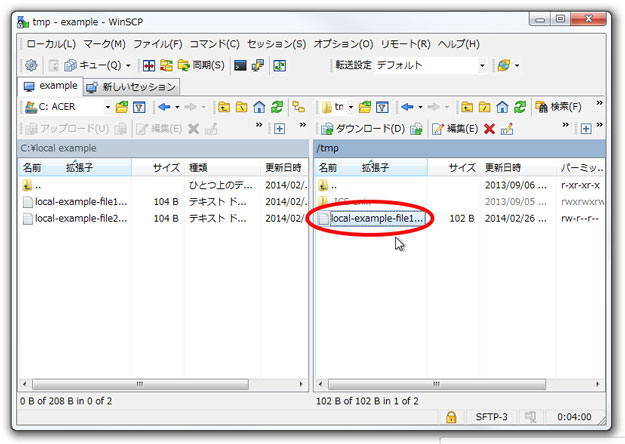












* 1. **Làm việc với Vagrant**

================================================================

Và đây là ghi chép đầy đủ nhất về việc cài đặt

Tham khảo:

<http://qiita.com/yamagh/items/61e540765fa4d2152dc4>

<http://qiita.com/Gin326M/items/01c8aef45f9a55b9b028>

Trước khi sử dụng, mình cần nói sơ qua về các quy trình để có một máy chủ ảo nhằm tránh bỡ ngỡ cho nhiều bạn.

Trong Vagrant, nó có một khái niệm bạn cần biết rõ đó là Box. Box có nghĩa là một gói hệ điều hành và có nhiều box đã cài sẵn một số ứng dụng cần thiết, chẳng hạn như box CentOS 6.5 32bit, CentOS 6.5 64bit, Ubuntu 12.04 có sẵn LAMP,…

Khi dùng Vagrant, bạn sẽ cần tạo một thư mục riêng cho mỗi máy ảo và mỗi thư mục này sẽ chứa các thiết lập cho 1 máy ảo. Bởi vì khi khởi động, bạn sẽ cần truy cập vào thư mục của máy ảo cần khởi động và sử dụng lệnh “vagrant up” để khởi động nó lên.

Cũng nên nói lại rằng, tất cả các thao tác ở Vagrant đều thông qua lệnh. Nếu bạn dùng Windows thì bật phần mềm Git Bash của Git để thao tác thông qua giao thức SSH, còn Linux thì dùng Terminal.Khi cài đặt Git nếu bạn chọn thao tác qua cmd thì có thể sử dụng cmd , nếu ko sẽ thao tác qua Gitbash(sẽ được viết rõ ở phần tiếp theo)

**B0: Tạo folder làm việc riêng**

Thực ra bước này cũng ko hẳn là cần thiết nhưng để tránh trường hợp sau này có phiền phức – vì môi trường windows đúng là 1 môi trường khá tồi dành cho việc phát triển ứng dụng. Vì vậy mọi hoạt động mình sẽ gói gọn nằm trong folder này

+ Tạo folder làm việc:

>mkdir work #Tạo folder work , mình đặt tại Desktop , còn tùy sở thích và thuận tiện, ai đặt đâu cũng được, nhưng khuyến khích đặt càng gần với root(thư mục ổ đĩa) càng tốt, tránh trường hợp đường dẫn thư mục quá dài và lỗi không đáng có xảy ra

>cd work # Đi đến thư mục vừa tạo

**B1: Nạp box cho Vagrant**

Khi cài đặt Vagrant thì nó sẽ ko có sẵn các box, mà mình cần nạp nó về máy của mình. Mình quyết định sử dụng gói CentOS7.0

Và cài đặt vẫn tương tự với lệnh:

Cách 1

>vagrant box add {tên\_box} # Do boxes này đã được định nghĩa sẵn nên không cần gắn them đường dẫn

VD: box\_name = chef/centos-7.0 #=> vagrant box add chef/centos-7.0

Box\_name được lấy từ: <https://atlas.hashicorp.com/boxes/search>

Cách 2

>vagrant box add {tên\_box} {url}

VD:

# Sử dụng box được cung cấp trên mạng

vagrant box add CentOS7 https://dl.dropboxusercontent.com/seunrskm/centos-7.0-x86\_64.box

# Sử dụng box được tạo nội bộ nên đường dẫn chính là path tới thư mục chứa box

vagrant box add centos7 c:/centos/box/centos7.box

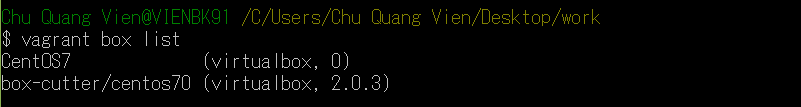
Kết quả

|  |
| --- |
| $box> Adding box 'CentOS7' (v0) for provider:  $box> Downloading: https://dl.dropboxusercontent.com/s/w3lbekm7eunrskm/centos-7.0- x86\_64.box  $box> Successfully added box 'CentOS7' (v0) for 'virtualbox'! |



# Check số lượng box đang tồn tại trong vagrant

>vagrant box list



**B2: Khởi tạo máy ảo**

Bây giờ để tạo máy ảo mới , ta tạo 1 thư mục riêng cho nó. Khuyến khích đặt tên thư mục có tên cùng với tên máy ảo cho dễ nhận biết

>mkdir CentOS7 # Tên thư mục do mình tự đặt nhưng đặt cùng cho dễ nhớ , có thể tạo bằng tay nếu ko thích dùng lệnh , như vậy thư mục CentOS7 sẽ nằm trong thư mục /work mình tạo ở bước B0

Và ta sẽ đi vào thư mục vừa tạo đó

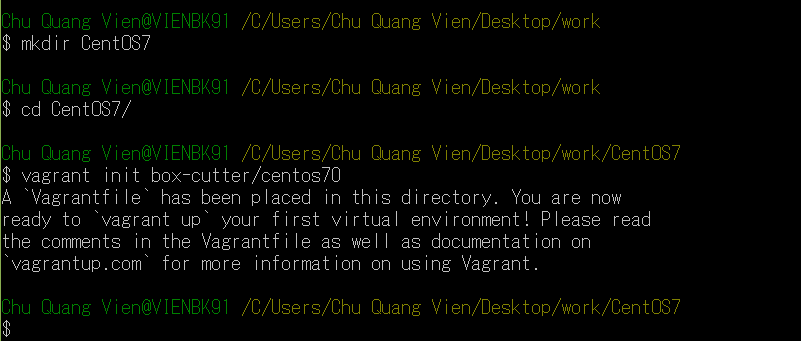
>cd CentOS7 # Đi vào trong thư mục vừa tạo

Tại thư mục đó ta khởi tạo máy ảo với box cần sử dụng bằng lệnh

>vagrant init {tên\_box} # Ở đây mình chỉ có 1 box là CentOS7

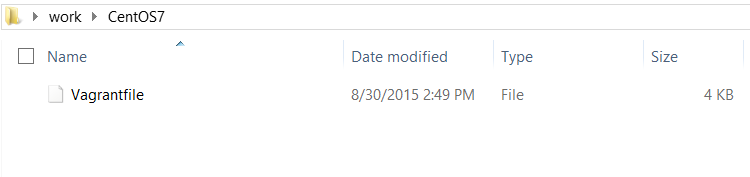
Sau khi xong bước này, nó sẽ hiện lên thông báo:

|  |
| --- |
| A `Vagrantfile` has been placed in this directory. You are now  ready to `vagrant up` your first virtual environment! Please read  the comments in the Vagrantfile as well as documentation on  `vagrantup.com` for more information on using Vagrant. |



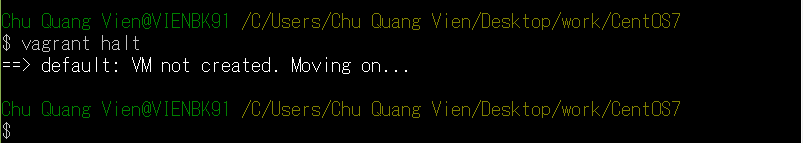
**Ý nghĩa của thông báo**: Thông báo này có nghĩa là bên trong thư mục máy ảo(Ở đây chính là thư mục CentOS7) nó đã đặt File Vagrantfile(file này chính là file cấu hình việc khởi tạo máy ảo) vào đó rồi.　Hãy mở file đó lên và config để có thể sử dụng nhiều chức năng hơn trước khi khởi động máy ảo với lệnh “vagrant up”.

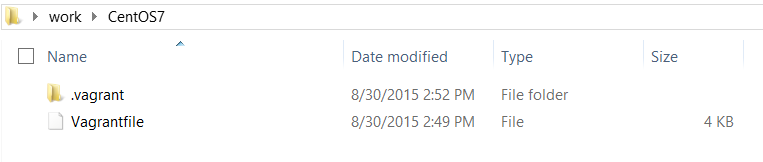
Cấu trúc thư mục sinh ra:



**Khuyến khích**: Nếu chưa biết setup môi trường ra sao thì tạm thời hãy cứ comment toàn bộ như ban đầu, và sau khi đã cài đặt xong. Tắt máy ảo bằng lệnh

>vagrant halt





Sau đó mới thay đổi nội dung file Vagrantfile này. Nếu thay đổi sớm, mà ko biết cách config thì sẽ tạo ra nhiều lỗi không lường trước. Nên tạm thời hãy cứ để nguyên file như vậy.

Bình thường thì ko cần phải sửa đổi gì, tuy nhiên nếu bạn dùng máy ảo để làm máy chủ webservice thì bạn cần config một chút. Trong file Vagrant này, các thiết lập đã được ghi mẫu sẵn và ta chỉ việc bỏ dấu # comment đi hoặc có thể coppy lại.

Bây giờ ta sẽ đi giải thích 1 số config cần chú ý.

// Tên box sau khi bạn chạy vagrant init(tên box được add và init)

config.vm.box = "chef/centos-7.0"

// Có đồng bộ dữ liệu máy thật với máy ảo tự động không. Tôi khuyến cáo là có và tốt nhất là comment cái dòng này lại vì mặc định nó là `true`

config.vm.box\_check\_update = false

// Truy cập vào máy ảo thông qua port. Trong trường hợp này là port 8080

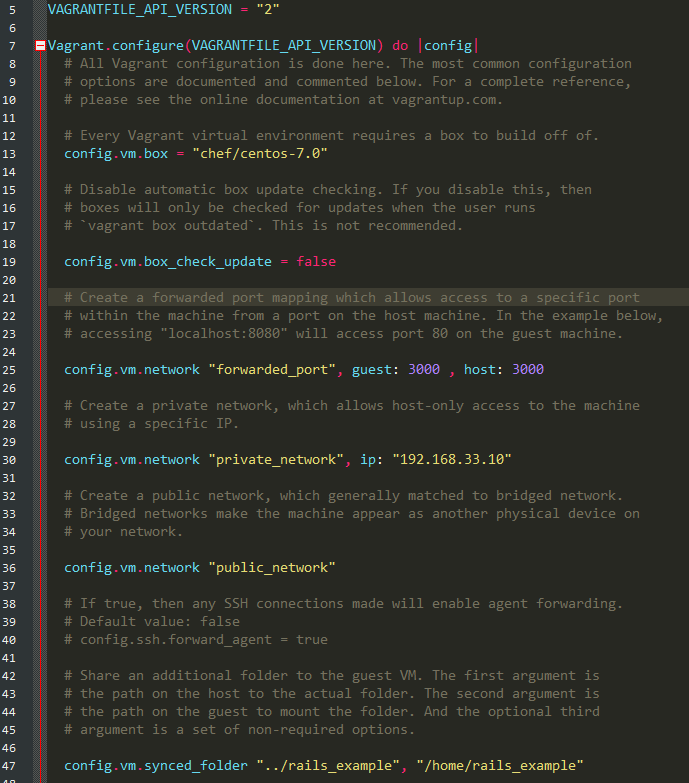
config.vm.network "forwarded\_port", guest: 80, host: 8080

// Truy cập vào máy ảo thông qua địa chỉ IP. Khuyến cáo để mặc định.

config.vm.network "private\_network", ip: "192.168.33.10"

// Đồng bộ hóa dữ liệu thư mục

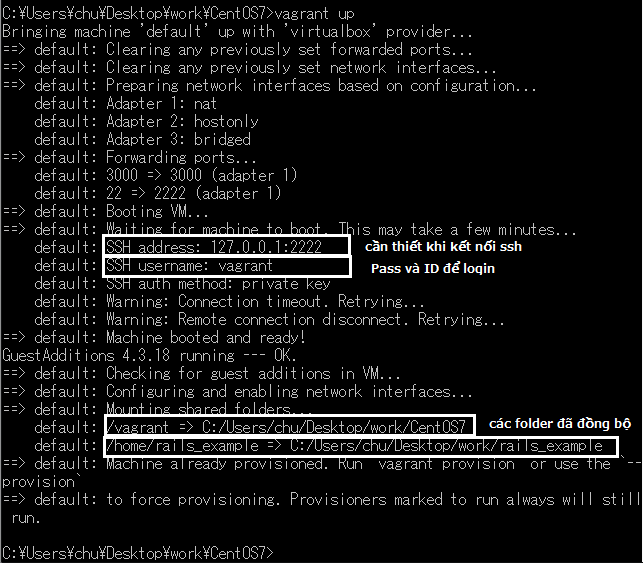
config.vm.synced\_folder "../data", "/vagrant\_data"



Tuy nhiên phần sau ta sẽ giải thích rõ hơn về việc config này. Tiếp tục công việc nào.

>vagrant up # Khởi động máy ảo

Kết quả:

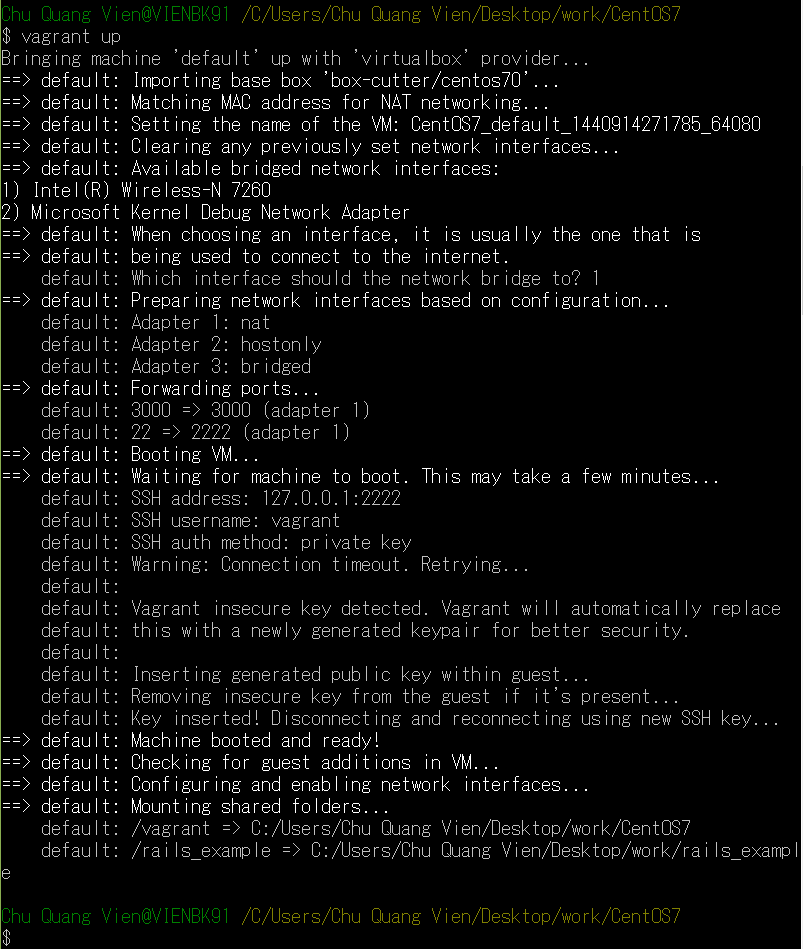


Tuy nhiên khi setup tại nhà thì xuất hiện tình huống

<http://blog.monochromegane.com/blog/2013/05/22/vagrant-specify-network-interface/>

<http://www.d-wood.com/blog/2014/06/13_6344.html>

Đánh số 1 vào



Nhưng sau đó ko sử dụng command prompt được nữa mà phải chuyển qua Gitbash dung thì mới vào được guest thông qua ssh (tốt nhất ngay từ đầu ta nên dùng git bash để đỡ mất công chuyển qua chuyển lại )

Username và pass được sử dụng trong trường hợp ta kết nối với ssh bằng các phần mềm khác như là Gitbash , Tera Term,..

**B3: Thiết lập môi trường máy ảo**

Trước khi đi tiếp, mình nên tìm hiểu qua ssh là gì?

SSH (tiếng Anh: Secure Shell) là một giao thức mạng dùng để thiết lập kết nối mạng một cách bảo mật. SSH hoạt động ở lớp trên trong mô hình phân lớp TCP/IP. Các công cụ SSH (như là OpenSSH, PuTTy,…) cung cấp cho người dùng cách thức để thiết lập kết nối mạng được mã hoá để tạo một kênh kết nối riêng tư.

Tham khảo về ssh:

+ <http://tuyetdoi.com/ssh-la-gi-cach-su-dung-ssh/>

+ <https://vi.wikipedia.org/wiki/SSH>

+ <http://wikilinux.vn/chuoi-bai-ve-ssh-phan-1-ssh-su-dung-cap-khoa-cong-khai-bi-mat/>

Nếu sử dung **Tera Term** thì có thể tham khảo bài sau:

+ <http://qiita.com/hiroyasu55/items/11a4c996b0c62450940f> (Thiết lập port,ssh,..)

+ <http://webkaru.net/linux/tera-term-install/> (Cài đặt)

Tổng quan về Tera Term thì nên tham khảo tại đây:

+ <http://webkaru.net/linux/ssh/> (Đầy đủ hình ảnh)

Tera Term thì bọn Nhật nó chuộng, nhưng VN thì chuộng Git hơn.

Nếu sử dụng **puTTY** thì tham khảo bài viết:

+ <https://c6h0st.wordpress.com/2015/06/13/tutorial-huong-dan-cai-dat-va-su-dung-vagrant/> (Trong bài viết này đã viết rất kĩ về việc cài đặt Git cũng như config biến môi trường)

Tuy nhiên putty được khuyến cáo là ko nên sử dụng vì sẽ ko thể thực hiện được việc share vagrant với các host khác

Nếu sử dụng **Git** có thể tham khảo tại:

+ <http://thachpham.com/hosting-domain/may-chu/tao-may-ao-tren-may-tinh-voi-vagrant-va-virtualbox.html> (Cài đặt)

+ <https://c6h0st.wordpress.com/2015/06/13/tutorial-huong-dan-cai-dat-va-su-dung-vagrant/> (Config biến môi trường)

Ở đây mình sử dụng **git**.

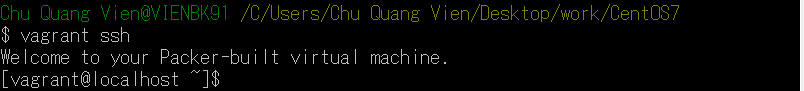
Sau khi đã cài đặt git xong, hãy bật cmd và tiếp tục công việc(Với điều kiện đã chọn sử dụng cmd để làm việc như là Gitbash , nếu ko thì phải sử dụng Gitbash). Và khi cài đặt Git thì cần chú ý: Để dùng được git thì sau khi cài đặt mình phải thêm vào biến môi trường “;C:\Program Files (x86)\Git\bin;” . Chú ý là dấu ; dùng để ngăn cách các khai báo biến môi trường với nhau.

“C:\HashiCorp\Vagrant\bin” sẽ được thêm vào biến môi trường sau khi cài đặt xong vagrant. (Hãy mở đường dẫn PATH trong thiết lập biến môi trường để check lại điều này)

+ Mở cmd và truy cập ssh (tại folder của nó , ở đây là CentOS7)

Truy cập máy ảo thông qua giao thức SSH

>vagrant ssh



**=>Từ h ta sẽ thao tác với ssh**

Quá trình cài đặt và làm việc với CentOS , tham khảo các site sau:

+ <http://shgam.hatenadiary.jp/entry/2013/10/13/175515>

+ <http://qiita.com/yamagh/items/61e540765fa4d2152dc4>

+ http://vietonrails.com/201411-cai-dat-virtualbox-%EF%BC%8B-vagrant-%EF%BC%8B-ruby-rails4-window7/

Phân tích bản chất của việc sử dụng Vagrant:

==================================================================

**Nhầm lẫn**:

Thực sự ban đầu mình không hiểu ý nghĩa của cụm từ “Đồng bộ các môi trường phát triển” trong công dụng của Vagrant. Mình nghĩ là mình sẽ cài đặt tất cả mọi thứ trên hostOS(máy mình) rồi phát triển, sau đó chuyển nó vào thư mục vagrant đã tạo ra để Vagrant đồng bộ nó cho giống với môi trường của các OS khác (theo kiểu end of line trong Windows là \r\n ,sẽ được vagrant chuyển thành \n trong CentOS). Sau đó mới hiểu rằng,khi mình sử dụng php,mysql,apache cài trên máy tính mình, lúc ấy code có thể chạy được nhưng khi code đó dùng trên centos lại không hoạt động do phiên bản php,mysql,apache trên máy mình không giống trên các os khác, môi trường cũng khác nên config cũng không giống nhau,… Vậy thì việc mình cài đặt mọi thứ trên host rồi làm việc là vô nghĩa.Như vậy vagrant chẳng có ý nghĩa gì cả.

**Thông não:**

Sau 1 hồi được thông não, giờ mình mới hiểu cái Vagrant nó dùng để làm gì. Hóa ra mình chẳng cần tạo gì trên host(máy tính của mình) hết, mà mình sẽ khởi tạo toàn bộ môi trường(cài đặt apache,php,mysql,..) trên guest. Sau đó mình sẽ chỉ lấy source từ guest về và phát triển nó, dùng winscp để đồng bộ code trên máy mình với trên guest, và test trực tiếp xem trên guest có chạy không. Như vậy vagrant đã giải quyết vấn đề đầu tiên, làm việc ở OS nào không quan trọng, mọi thứ sẽ được biên dịch và thông dịch trong máy ảo.

Nhưng như vậy chỉ là mình thay vì chạy trên localhost thì mình chay trên guest với môi trường CentOS thôi, còn những người khác trong team thì sao ?

Trong team có nhiều người, mỗi người 1 máy tính, virtual host cài đặt cũng khác nhau,boxes down về rồi tự install các gói php,mysql,..trên đó cũng khác nhau theo từng máy. Không lẽ từng người tự test trên máy ảo của mình rồi merge code lại sau ư ? Như vậy thì việc test trên guest hay trên host của bản thân sẽ không có ý nghĩa với làm việc team.

Vagrant sẽ giải quyết điều đó.

Giả sử giờ 1 boxes đã đc cài đặt toàn bộ các gói cần thiết như apache,php,mysql,… đã được tạo ra.Chạy tốt và ổn định.

+ Bạn có thể chuyển thiết lập các máy ảo trong máy mình sang một máy tính khác (re-package), hoặc cho phép các thành viên cùng team với bạn truy cập vào thư mục riêng trên máy chủ để sửa/xem file mà họ không cần cài đặt bất cứ cái gì, miễn là dùng chung mạng LAN (Mạng nội bộ), thậm chí bạn có thể đưa máy chủ của bạn lên môi trường internet chỉ với vài dòng lệnh đơn giản.. Tiện lợi phải không nào ☺

Như vậy với vagrant thì team đã có thể cùng làm việc với 1 server test mà không hề mất phí thuê server hoặc là gặp cản trở trong quá trình phát triển với các OS khác nhau ☺

Lúc đó mới chuyển từ client engineer sang server engineer nên đúng là gà mờ, chẳng hiểu gì ☹

==================================================================

* 1. **Config file Vagrantfile**

================================================================

// Tên box sau khi bạn chạy vagrant init

config.vm.box = "chef/centos-7.0"

// Có đồng bộ dữ liệu máy thật với máy ảo tự động không. Tôi khuyến cáo là có và tốt nhất là comment cái dòng này lại vì mặc định nó là `true`

config.vm.box\_check\_update = false

// Truy cập vào máy ảo thông qua port. Trong trường hợp này là port 8080

config.vm.network "forwarded\_port", guest: 80, host: 8080

// Truy cập vào máy ảo thông qua địa chỉ IP. Khuyến cáo để mặc định.

config.vm.network "private\_network", ip: "192.168.33.10"

// Giúp máy thật và máy ảo có thể nhận được IP của nhau

config.vm.network "public\_network"

// Đồng bộ hóa dữ liệu thư mục

config.vm.synced\_folder "../data", "/vagrant\_data"

Ở đây mình sẽ nói rõ hơn về config đồng bộ thư mục giữa máy ảo và máy thật(sử dụng để di chuyển dữ liệu qua lại giữa máy thật và máy ảo).

Như đã nói trong cài đặt thì thư mục CentOS7 chính là thư mục máy ảo được đặc tại host(máy thật) , và nó cũng tương đương với thư mục /vagrant trên máy ảo và nó luôn ở trạng thái đã được đồng bộ .Nghĩa là có bất kỳ file nào được thêm vào thư mục /CentOS7 trên máy thật thì nó cũng được thêm vào folder /vagrant trong máy ảo.

Nên việc mình config sẽ là để config cho các folder ở ngoài(chứ ko phải các folder nằm trong thư mục CentOS7, vì trong đó lúc nào cũng đồng bộ rồi), thì trong đó đường dẫn đầu tiên là đường dẫn tới thư mục mình muốn chứa và phải được tạo sẵn trên host của mình(trên win). Còn đường dẫn tiếp theo là nơi mình làm việc với nó trên guest(nó sữ tự động sinh ra khi mình khởi động máy ảo theo file config – còn nếu có rồi thì ko cần sinh ra nữa)

Cái config mẫu:

config.vm.synced\_folder "../data", "/vagrant\_data"

Ở đây ta hiểu thế này nhé, tính từ thư mục máy ảo hiện tại trên host là CentOS7, thì :

../data : .. là chỉ bố của thư mục hiện tại(bố của CentOS7 , chính là thư mục work), /data nghĩa là sẽ tồn tại 1 thư mục có tên data (thưc mục trên host cần phải được tạo sẵn). Như vậy data và CentOS7 là 2 thư mục đồng cấp. Và mọi dữ liệu trong thư mục data này, sẽ được đồng bộ với thư mục vagrant\_data là thư mục con của “/” – cái “/” chính là ông tổ của mọi thằng trong linux, nó là cha của home, và cái thư mục vagrant\_data sẽ được tự động sinh ra khi vagrant up được thực thi.

Vậy mình sẽ sửa file config như sau:

config.vm.synced\_folder "../rails\_example", "/rails\_example"

Như vậy mình chỉ cần tạo 1 folder đồng cấp với CentOS7 tên là rails\_example và config như trên, thì trên guest sẽ tự động sinh ra 1 folder có tên rails\_example là con của thư mục “/” (đồng cấp với home/, là chú của /home/vagrant). Và mình sẽ tạo rails project tại đây, rồi đồng bộ nó với win và làm mọi việc trên win.

OK, vậy là mình đã khởi tạo 1 app rails trên guest.

À, còn phải vào file vagrant config lại port của guest và host nữa. Mình sử dụng cả 2 đều là cổng 3000

config.vm.network "forwarded\_port", guest: 3000 , host: 3000

* 1. **Lỗi phát sinh khi sử dụng vagrant**

Một số lỗi phát sinh trong quá trình sử dụng Vagrant

**Lỗi 1**: Khi mình cài box CentOS7 vào và xóa bằng tay đi, tuy nhiên lúc cài lại thì nó báo lỗi này nên ko add box vào được

|  |
| --- |
| The box you're attempting to add already exists. Remove it before  adding it again or add it with the `--force` flag. |

Thì mình có 1 số cách check xem mình đã thực sự remove box đó ra khỏi vagrant chưa như sau

>vagrant box list # check các box vẫn tồn tại trong vagrant

>vagrant global-status # Hiển thị danh sách tên box + id đang chạy, đang tắt hay đang làm gì đó

>vagrant global-status --prune # Xóa cache

>vagrant destroy box\_id # id được lấy từ lệnh trên

>vagrant box remove tên\_box # tên box được lấy trên lệnh box list

Như vậy là đã xóa box khỏi vagrant, để chặt chẽ hơn thì mình xóa luôn cả folder trên win và làm lại từ đầu. Nhớ là phải tắt vagrant đi sau khi làm xong mọi việc rồi mới add box mới thử nhé

>vagrant halt

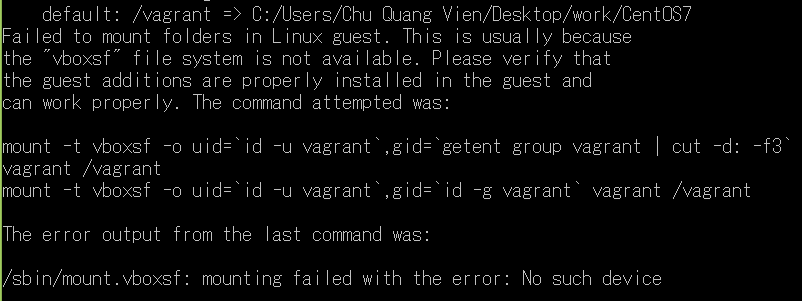
>vagrant box add {tên\_box} {url}

Ngoài ra có thể làm theo cách sau:

+ <http://qiita.com/takehiro_t59/items/63cc5dffe2f1b3f5bed4>

**Lỗi 2**: Sau khi cài đặt tất cả OK rồi, thì start vagrant lên, đợi nó load file Vagrantfile xong thì có lỗi phát sinh:

|  |
| --- |
| Failed to mount folders in Linux guest. This is usually because  the "vboxsf" file system is not available. Please verify that  the guest additions are properly installed in the guest and  can work properly. The command attempted was:  mount -t vboxsf -o uid=`id -u vagrant`,gid=`getent group vagrant | cut -d: -f3` vagrant /vagrant  mount -t vboxsf -o uid=`id -u vagrant`,gid=`id -g vagrant` vagrant /vagrant  The error output from the last command was:  /sbin/mount.vboxsf: mounting failed with the error: No such device |



Nguyên nhân và cách giải quyết:

+ <http://qiita.com/DQNEO/items/2375dd8002a831268cb5> (Nguyên nhân)

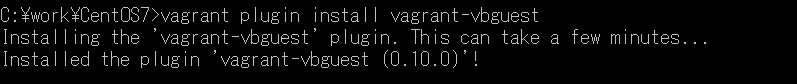
+<http://www.utano.jp/entry/2014/11/vagrant_guest_additions_update_error>

+ http://qiita.com/akippiko/items/278efedee35661634b85 (Giải quyết thành công)

>vagrant plugin install vagrant-vbguest

Installing the 'vagrant-vbguest' plugin. This can take a few minutes...

Installed the plugin 'vagrant-vbguest (0.10.0)'!



>vagrant vbguest --status

GuestAdditions 4.3.26 running --- OK.



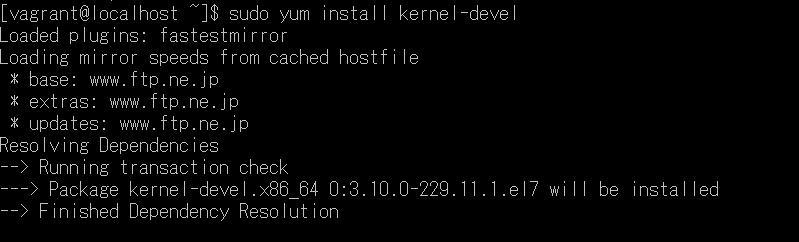
Tuy nhiên nếu nhận được thông báo status như thế này:



Thì có nghĩa là GuestAdditions vẫn chưa thể sử dụng được, mà ta cần cài đặt thêm kernel

Tham khảo: <http://qiita.com/DQNEO/items/2375dd8002a831268cb5>

>sudo yum install kernel-devel



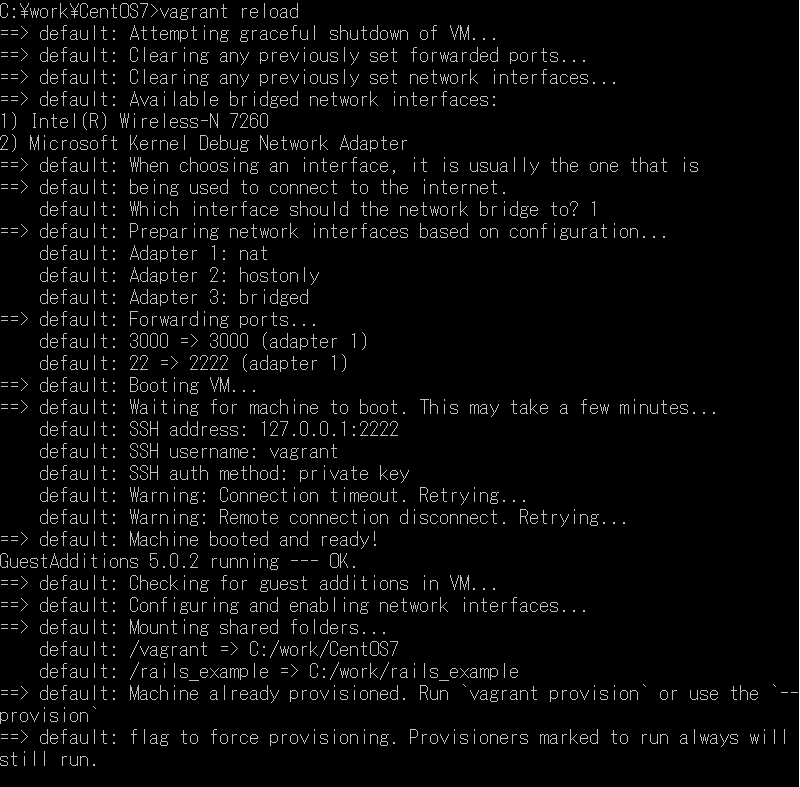
Sau đó thực hiện lại 1 lần nữa việc install plugin vagrant-vbguest

> vagrant plugin install vagrant-vbguest

> vagrant vbguest --status



>vagrant reload



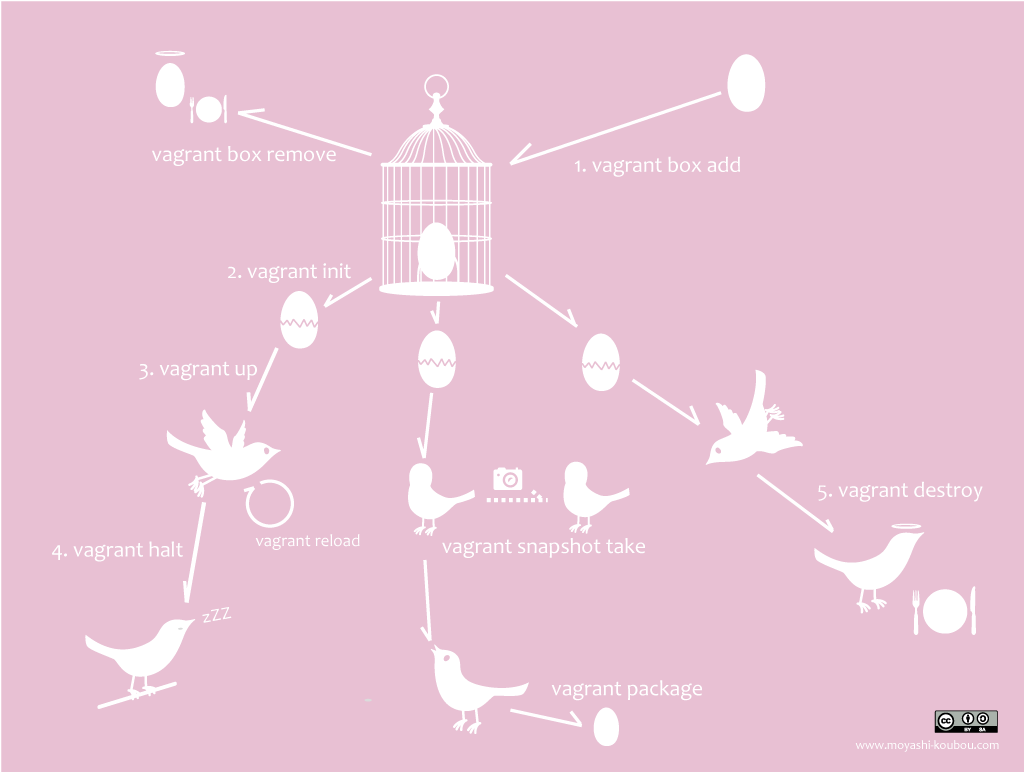
Tiếp theo mình sẽ tạo thử project rails, đồng bộ cũng như config file vagrant như thế nào.

Nếu nó báo thiếu kernel thì tham khảo link 1 đã cung cấp

1. **Tổng kết**

Các bạn vẫn cứ code trên hệ điều hành mà bạn thích còn việc biên dịch hay thông dịch code thì cứ để máy ảo lo và Vagrant sẽ làm công việc đồng bộ dữ liệu của bạn với môi trường máy ảo. Chiếc máy ảo này sẽ được share cho các thành viên trong nhóm

Vagrant có tác dụng như 1 cầu nối giữa máy ảo được tạo ra bởi Virtualbox và máy thật các bạn đang làm việc. Trong quá trình làm việc bất kỳ thay đổi nào với code của bạn trên máy thật sẽ được đồng bộ lên máy ảo.



1. **Ansible**
2. **Khái niệm cơ bản**
3. **Cài đặt Ansible trong Vagrant**

<http://qiita.com/ak-ymst/items/07906a5fb3f5fb283386>

<http://blog.shin1x1.com/entry/ansible_local-provisioner-in-vagrant>

<http://knowledge.sakura.ad.jp/tech/2882/>

<http://progmemo.wp.xdomain.jp/archives/972>

<http://www.moyashi-koubou.com/blog/vagrant_ansible_windows/>

<http://blog.vccloud.vn/ansible-cai-dat-va-cau-hinh/>

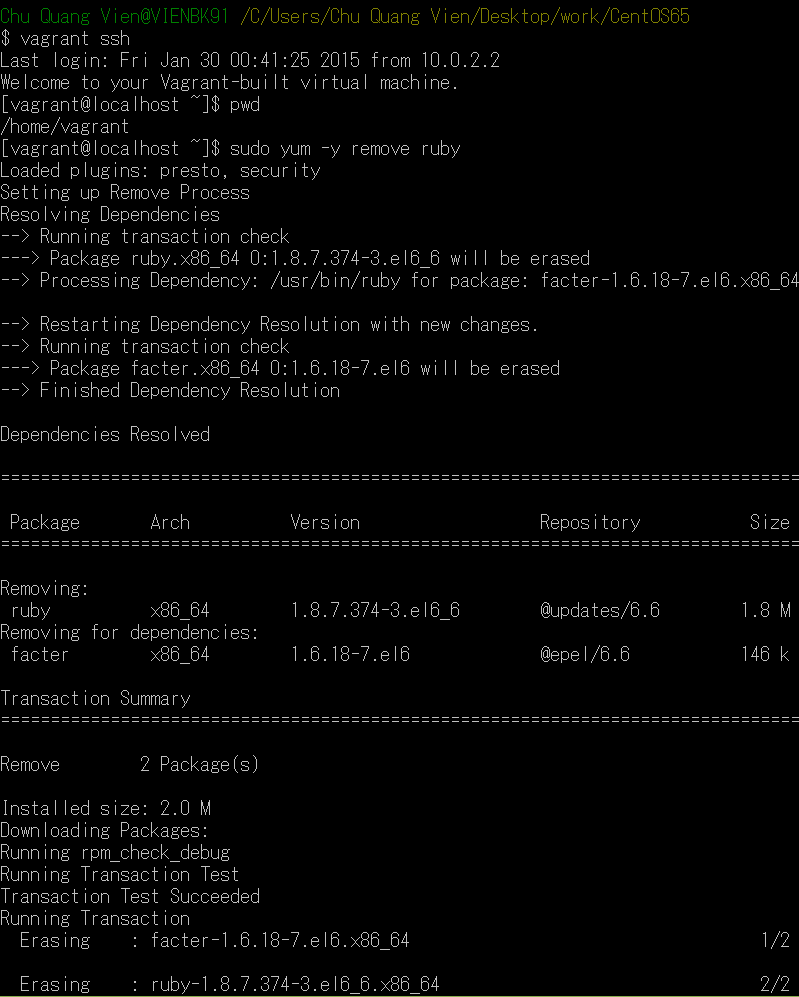
1. **Áp dụng**

Trong môi trường CentOS , Fedora , RHEL thì chúng đều sử dụng trình quản lý gói “yum”.

+ Loại bỏ các vesion cũ(Caí này dùng nếu bạn dùng trên Mac Yosemite hoặc Unix đã cài sẵn Ruby thì nên gỡ bỏ vì nó là phiên bản cũ, còn nếu dùng CentOS thì thường nếu chạy lệnh này nó sẽ báo ko tìm thấy thư mục Ruby :D Cơ bản là vì làm gì có mà tìm)

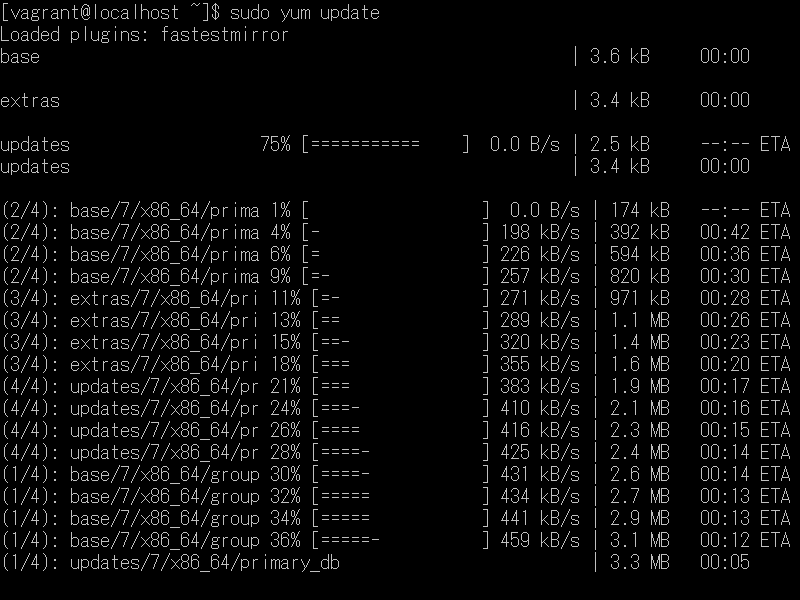
>sudo yum –y remove ruby # Lệnh xóa các phiên bản cũ của Ruby đã tồn tại trong OS

Kết quả:

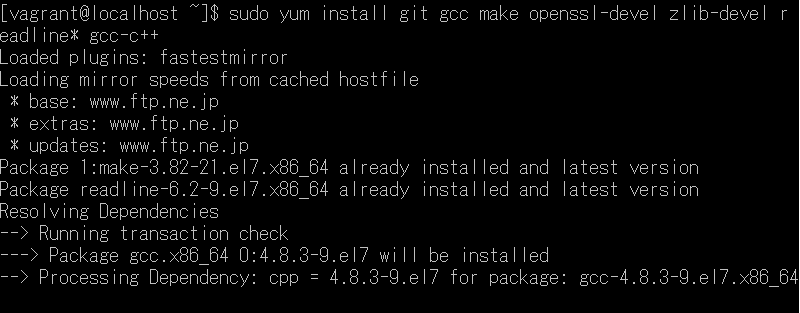


+ Cài đặt các pakage liên quan (Git, openssl , zlib , gcc-c++)

> sudo yum update



> sudo yum install git gcc make openssl-devel zlib-devel readline\* gcc-c++ # Cài đặt git



**+ Cài rbenv**

Sơ lược qua về các trình quản lý:

Ruby Version Manager (RVM) là một công cụ giúp bạn cài đặt, quản lý và làm việc với nhiều môi trường ruby. Thỉnh thoảng, có thể bạn sẽ gặp phải trường hợp mà một gem có thể hoạt động tốt với một vài phiên bản ruby nào đó, trong khi đó nó có thể không chạy chính xác trên những phiên bản còn lại của Ruby. Việc này đòi hỏi phải có một chương trình mà có thể giúp chúng ta quản lý các phiên bản này theo từng dự án. RVM là một trong những chương trình giúp chúng ta quản lý các phiên bản của Ruby.(Trích từ: <http://pingo.edu.vn/cac-buoc-cai-dat-cho-du-an-rails/> )

Rbenv cũng có chức năng tương tự như rvm, nó cho phép bạn quản lý nhiều phiên bản cài đặt của Ruby. Nó ko hỗ trợ cài đặt Ruby nhưng có thể dùng plugin “ruby-build” của nó để cài đặt Ruby.

Ngoài ra thì còn rất nhiều trình quản lý khác như là chruby , uru.　(Có thể tham khảo tại:

<https://www.ruby-lang.org/vi/documentation/installation/#chruby> )

Rbenv hay rvm đều là những “công cụ quản lý phiên bản ruby”. Việc sử dụng rbenv hay rvm sẽ giúp ta dễ dàng cài đặt 1 phiên bản Ruby cần sử dụng, đồng thời có thể thay đổi phiên bản ruby trên máy 1 cách nhanh chóng.rbenv được phát triển và duy trì bởi sstephenson trên trang dịch vụ lưu trữ Git repositories Github. Chúng ta sẽ sử dụng git để cài đặt rbenv.(Chú ý là 2 cái này chỉ được chọn 1 trong 2, dùng rvm thì thôi rbenv và ngược lại. Nếu cài song song thì sẽ phát sinh lỗi do cách quản lý Ruby của RVM khác với rbenv)

>git clone [https://github.com/sstephenson/rbenv.git ~/.rbenv](https://github.com/sstephenson/rbenv.git%20~/.rbenv)

Kết quả:

|  |
| --- |
| Cloning into '/home/vagrant/.rbenv'...  remote: Counting objects: 1889, done.  remote: Total 1889 (delta 0), reused 0 (delta 0)  Receiving objects: 100% (1889/1889), 298.61 KiB | 197.00 KiB/s, done.  Resolving deltas: 100% (1178/1178), done. |



**+ Thiết lập PATH**

Để có thể sử dụng câu lệnh rbenv mà không cần quan tâm tới đường dẫn tới thư mục ~/.rbenv thì ta có thể thêm đường dẫn đó vào file

~/.bash\_profile bằng cách :

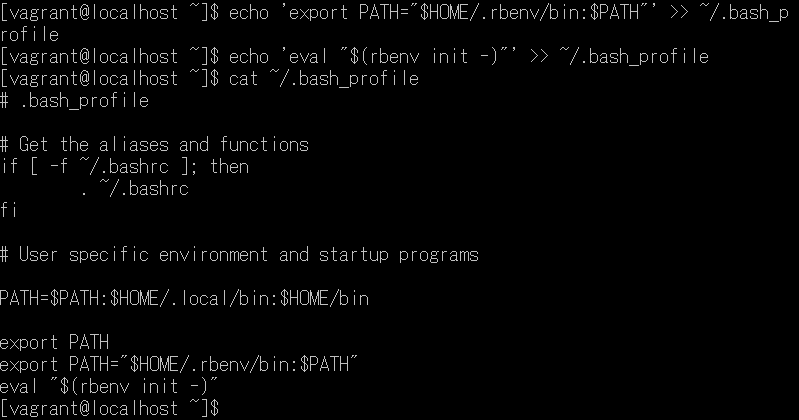
>echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bash\_profile # Thêm path vào trong file

>echo 'eval "$(rbenv init -)"' >> ~/.bash\_profile # Thêm rbenv init vào trong shell để có thể gọi rbenv ra bất cứ lúc nào

> cat ~/.bash\_profile

Kết quả:

|  |
| --- |
| # .bash\_profile  # Get the aliases and functions  if [ -f ~/.bashrc ]; then  . ~/.bashrc  fi  # User specific environment and startup programs  PATH=$PATH:$HOME/.local/bin:$HOME/bin  export PATH  export PATH="$HOME/.rbenv/bin:$PATH"  eval "$(rbenv init -)" |

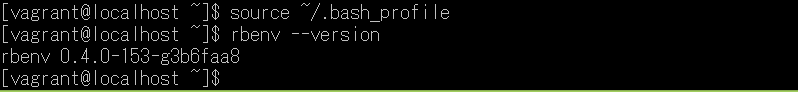


>source ~/.bash\_profile #reload bash\_profile

> rbenv --version # Kiểm tra version của rbenv vừa cài đặt, có kết quả nghĩa là cài đặt thành công~

Kết quả:

|  |
| --- |
| $rbenv 0.4.0-98-g13a474c |



+ Cài ruby

Để cài đặt một phiên bản mới của Ruby thêm dễ dàng hơn, mình khuyến khích bạn cài đặt plugin “ruby-build” của rbenv bằng cách:

>git clone https://github.com/sstephenson/ruby-build.git ~/.rbenv/plugins/ruby-build

Kết quả:

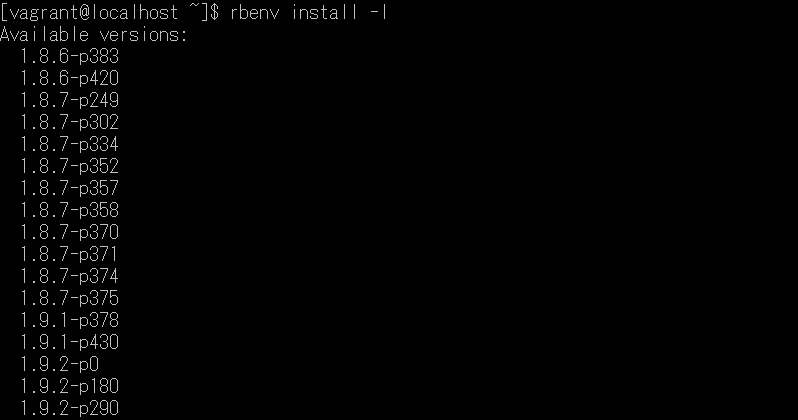
|  |
| --- |
| Cloning into '/home/vagrant/.rbenv/plugins/ruby-build'...  remote: Counting objects: 3786, done.  remote: Compressing objects: 100% (31/31), done.  remote: Total 3786 (delta 16), reused 2 (delta 0)  Receiving objects: 100% (3786/3786), 661.69 KiB | 361.00 KiB/s, done.  Resolving deltas: 100% (1859/1859), done. |



>rbenv install –l # Liệt kê các gói ruby có thể cài đặt

Kết quả

|  |
| --- |
| Available versions:  1.8.6-p383  1.8.6-p383  1.8.6-p420  .  省略  .  2.1.3  2.2.0-dev  2.2.0-dev  2.2.0-preview1  2.2.0-preview1  jruby-1.5.6  jruby-1.5.6 |



>rbenv install –v 2.2.3 # Chọn version mình muốn sử dụng. Ở đây mình chọn 2.2.3

Kết quả

|  |
| --- |
| Downloading ruby-2.2.3.tar.gz...  -> http://dqw8nmjcqpjn7.cloudfront.net/ 0818beb7b10ce9a058cd21d85cfe1dcd233e98b7342d32e9a5d4bebe98347f01  Installing ruby-2.2.3...  Installed ruby-2.2.3 to /home/vagrant/.rbenv/versions/2.2.3 |

Nếu thất bại sẽ cho kết quả

|  |
| --- |
| ...  BUILD FAILED  ... |

Và ko còn cách nào khác là search google để tìm cách sửa.

>rbenv global 2.2.3 # Thiết lập Ruby 2.2.3 là phiên bản chung trên guest -> như vậy là Ruby 2.2.3 đã trở thành bản Ruby default trên máy

>ruby –v # Lấy version của ruby

Kết quả

|  |
| --- |
| ruby 2.2.3p242 (2014-09-19 revision 47630) [x86\_64-linux] |



Bổ sung nếu cần sử dụng: (tạm thời ko cần quan tâm)

==================================================================

Muốn xóa bỏ version Ruby mà mình ko muốn dùng nữa thì có thể kiểm tra các versions đang được sử dụng bằng lệnh

>rbenv versions

>~/.rbenv/versions # Nó sẽ trả về đường dẫn đến thư mục này

$-bash: /home/vagrant/.rbenv/versions/ : Is Directory # Kết quả

C1: Uninstall nó đi

>rbenv uninstall [-f|--force] <version> # force là xóa và bất chấp mọi thứ

C2: Xóa thư mục của nó đi

Và chỉ cần đi vào trong thư mục /version(thư mục cha của các ruby version) và dùng lệnh

>rm –rf tên\_thư\_mục\_muốn\_xóa # Ngoài ra có thể tìm đến thư mục của Version đó bằng lệnh >rbenv prefix tên\_version rồi xóa

# VD: >rm -rf ~/.rbenv/versions/1.9.3-p0

Sau khi xóa nhớ chạy lệnh

>rbenv rehash # để làm sạch tập tin nhị phân từ các version cũ để lại

Trong trường hợp chỉ muốn thiết lập phiên bản Ruby cho 1 thư mục nào đó thì có thể dùng lệnh sau

>cd thư\_mục # Đi đến thư mục muốn thiết lập phiên bản Ruby

>rbenv local 2.2.3 # Thiết lập Ruby cho folder này

Nếu muốn bỏ thiết lập phiên bản Ruby cho folder

>rbenv local --unset

==================================================================

**+ Cài đặt Bundler**

Sau khi cài đặt Ruby thành công, mặc định là bạn có thể chạy câu lệnh gem trên môi trường của bạn.

Một danh sách khá đầy đủ về các gem được phát triển bởi RubyGems có thể tìm thấy tại:

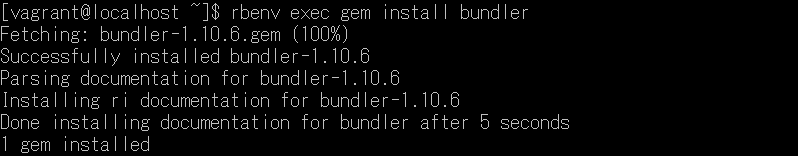
<https://rubygems.org/gems>.

Cách sử dụng đầy đủ của câu lệnh gem có thể tìm thấy tại: <http://guides.rubygems.org/command-reference/>

>rbenv exec gem install bundler # cài bundler

Kết quả:

|  |
| --- |
| Fetching: bundler-1.8.2.gem (100%)  Successfully installed bundler-1.8.2  Parsing documentation for bundler-1.8.2  Installing ri documentation for bundler-1.8.2  Done installing documentation for bundler after 3 seconds  1 gem installed |



>rbenv rehash

Ý nghĩa của câu lệnh trên: <http://dqn.sakusakutto.jp/2014/02/rbenv_rehash_what_it_does.html>

Bay giờ ta sẽ tìm hiểu qua về Gem , Bundler

Gem là gì ?

Khái niệm package (gói) trong UNIX có thể là khái niệm gần gũi nhất với gem. Đối với một ngôn ngữ lập trình như Ruby, việc xây dựng các ứng dụng kết nối các tiện ích như SQL, networking, ... là rất cần thiết. Để thực hiện việc này, chúng ta có thể dùng các gem của Ruby. Gem như một "plug-in" của Ruby, bạn chỉ cần cài đặt gem cần thiết vào máy (ví dụ gem mysql2), khi cần sử dụng, bạn require mysql2 và bạn có thể sử dụng các chức năng của mysql2 trong chương trình Ruby của mình.

=>Như vậy Gem ko phải 1 plugin mà là 1 chương trình quản lý các plugin(version , source , install, remove)

Khi install 1 gem bằng cách dùng lệnh gem install GEM\_NAME là ta cài thêm 1 gem vào thư mục gem của bạn,

VD: /home/vagrant/.rbenv/shims/gem/ruby-2.2.0/bin/

Bundler và cài đặt gem thông qua bundler

Mỗi dự án Rails có một “Gemfile”(file này được sinh ra cùng với dự án), file này được đọc bởi gem “bundle”. Bản thân bundle cũng là 1 gem, nên nó sẽ được cài đặt bằng lệnh “gem install bundle”. Tuy nhiên, sau khi bundle được cài đặt, nó sẽ đọc file Gemfile, cài đặt tất cả các gems được yêu cầu trong Gemfile, và tạo ra file Gemfile.lock. Khi code được đưa lên 1 repo (code này có kèm theo Gemfile.lock) và một developer khác download code về và chạy “bundle install”, nó sẽ dựa vào file Gemfile.lock để cài đặt chính xác các gems cùng với version tương ứng. Như vậy, bằng cách “đồng bộ” qua Gemfile.lock, môi trường làm việc của các developer trong cùng một dự án là giống nhau. Việc này tránh mỗi developer dùng cách “trực tiếp” để cài đặt nên các version khác nhau của 1 gem, gây mất đồng bộ. Nên tốt nhất là đặt tất cả các gem cần thiết cho dự án vào trong Gemfile. Công việc tạo “Gemfile” nằm ngoài bài này vì nó cần bạn có một số kiến thức để lựa các gems cho phù hợp với dự án.

Khi thêm dòng "gem "GEM\_NAME"" vào file Gemfile và sau đó chạy "bundle"(bằng lệnh bundle install) thực chất là ta cũng sẽ gọi tới lệnh gem install GEM\_NAME và cài vào thư mục tương tự, nếu đã cài gem rồi thì khi chạy bundle thì rails sẽ không cài lại mà chỉ check xem có đúng version không, nếu không mới cài lại.

Như vậy khác nhau giữa Gemfile và Gemfile.lock

|  |
| --- |
| Gemfile là nơi khai báo các plugin được sử dụng trong chương trình rails của bạn. Tất cả những gem được cài đặt sẽ được ghi vào Gemfile |
| Gemfile.lock là tập tin mà Bundler ghi lại các phiên bản chính xác đã được cài đặt, khi chạy lệnh bundle install sẽ check nội dung file Gemfile.lock để cài đặt đúng phiên bản đã được khai báo chứ ko phải cài đặt các phiên bản mới nhất như việc sử dụng lệnh gem install  Điều đó sẽ giúp các developer có thể đồng bộ chính xác các phiên bản đang sử dụng chứ ko phát sinh lỗi do sự chênh lệch về phiên bản khi cài đặt.  Mỗi ghi chạy bundler gemfile thì file này cũng tự động thay đổi. |

Ngoài ra có 1 điều rất thú vị đó là chỉ cần khai báo các gem muốn sử dụng trong gem file và sau đó chạy lệnh

>bundle install

Thì nó sẽ tiến hành cài đặt cho bạn gem này(thay vì gõ lệnh gem install tên\_gem). Như vậy là bạn có thể sử dụng gem mà ko cần quá nhiều thao tác với terminal

Bài viết hay về gemfile(link die rồi)

|  |
| --- |
| Cộng đồng Ruby (On Rails) là nơi hướng dẫn chia sẽ các kiến thức cơ bản. Với các thư viện bên ngoài thì giúp bạn sẽ nhanh chóng và dễ dàng thực hiện một số chức năng mà bạn phải mất cả hàng tháng trời (nhanh hay chậm còn tùy ^^) để viết được chúng. Các dễ dàng để ứng dụng Rails có thể sử dụng chúng là thông qua Gemfile. Bên cạnh Gemfile còn có dòng lệnh bundler giúp cài đặt các thư viện bên ngoài một cách dễ dàng.  #Hướng dẫn cơ bản  Đầu tiên bạn mở file Gemfile bạn thêm vào dòng:  gem 'font-awesome-sass'  Vì khi bạn tạo project thì đã có sẵn một số thư viện(gem) cơ bản của ứng dụng Rails. font-awesome-sass cung cấp cho bạn gem tạo các icon đẹp cho ứng dụng web (cái này mà viết chắc khó nhằn à). Sau khi thêm dòng này vào Gemfile, bạn mở command line mà chạy dòng lệnh bundle install để tiến hành cài đặt gem này. Như vậy bạn đã có thể sử dụng gem này.(Thay vì phải gõ lệnh gem install tên\_gem như trước)  #Xử lý các phiên bản  Với việc sử dụng gem như trên thì nó có thể sẽ lấy bất kỳ phiên bản nào và thường là phiên bản cuối cùng. Điều gì xảy ra nếu như bạn gửi code cho một người khác và khi người đó chạy lênh bundler install, nếu người đó có cài đặt hay chỉnh sửa gem mà cùng tên trước đó thì sẽ mất hết. Nhưng gem đã hỗ trợ những tính năng cho phép bạn khắc phục những trường hợp đó, ví dụ như sau:  gem 'rails', '4.0.1'  Khi bạn viết như vậy có nghĩa là bạn muốn sử dụng Rails ở version 4.0.1. Nếu một người khác chạy code ở trên máy của họ thì nó sẽ lấy phiên bản đúng là 4.0.1.  Nếu bạn không chắc nó có phải phiên bản mới nhất hay không thì bạn có thể thêm dấu ~> như sau :  gem 'rails', '~>4.0.1'  Giờ bạn có thể chạy bundle update hoặc bundle install trên máy khác. Nó sẽ vẫn sử dụng phiên bản 4.0.1 nhưng bên cạnh đó nó còn cài đặt cho bản phiên bản mới nhất ví dụ như 4.0.2.  Và cuối dùng là dấu >= sẽ yêu cầu bundler lấy phiên bản mới nhất, lớn hơn phiên bản bạn đưa vào. Ví dụ như:  gem 'rails', '>= 3.2'  Như vậy thì bundler sẽ lấy cho bản phiên bản từ 3.2.1 đến 4.1(mới nhất) nhưng như vậy sẽ có lỗi như nếu code của bạn không tương ứng với phiên bản của Rails. Vì thế bạn có thể giới hạn lại khoảng cách phiên bản và ở đây là cucumber từ 1.3.2 đến 1.3.9 như sau:  gem 'cucumber', '>=1.3.2', '<1.3.10'  #Thay đổi sources  Mặc định thì bundler sẽ lấy gem từ rubygem.org nhưng nếu muốn lấy gem từ nguồn khác thì bạn vẫn có thể lấy dễ dàng, ví dụ như sau:  source 'http://gems.github.com'  Bạn sử dụng source để chuyển từ RubyGems sang GitHub gem server. Và bạn cũng có thể kết hợp cả hai trong file Gemfile như sau:  source 'https://rubygems.org'  source 'http://gems.github.com'  Đơn giản phải không nào và đôi khi bạn cũng có thể viết như sau:  gem 'nokogiri', git: 'git://github.com/tenderlove/nokogiri.git', branch: '1.4'  Điều này nói bundler sẽ lấy phiên bản cuối cùng của nokogiri từ 1.4 trên GitHub.  #Lời kết  Như vậy thông qua bài viết trên bạn có thể hiểu hơn về cách sử dụng Gemfile trong ứng dụng Rails. Và cảm ơn các bạn đã theo dõi bài viết trên. |

Tại đây đã nói rất chi tiết về bundler:

+ <http://qiita.com/hazy_moon/items/ea6fdbc058256bbaafe0>

+ <http://xxxcaqui.hatenablog.com/entry/2013/02/11/013421>

+ <http://www.rubylife.jp/rails/ini/index2.html> (Giải thích rõ quá trình sinh ra và sử dụng của 2 file: gemfile và gemfile.lock)

Tạo gem bằng bundler

+ <https://viblo.asia/obito/posts/z3NVRkJLv9xn> (Tiếng Việt)

# Để hiểu ý nghĩa của rbenv rehash thì nên tham khảo bài viết sau

<http://dqn.sakusakutto.jp/2014/02/rbenv_rehash_what_it_does.html>

Phân tích rõ hơn về việc Gemfile và Gamefile.lock(dịch bài viết)

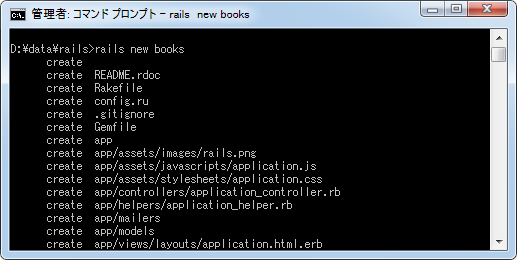
+ <http://www.rubylife.jp/rails/ini/index2.html>

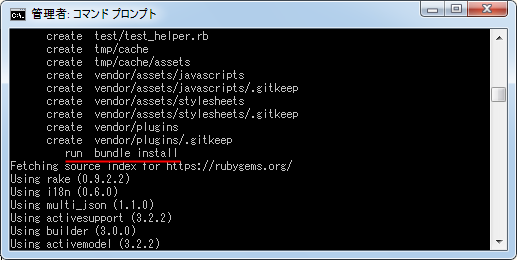
Gemfile được tạo ra cùng dự án, định nghĩa 1 số gem quan trọng đã được cài đặt trong quá trình khởi tạo

Khi thực hiện lệnh khởi tạo rails app

>rails new books # rails app có tên là books

thì nếu ko có khai báo gì thì nó sẽ tự động thực hiện những việc sau:





Nó đã thực hiện lệnh >run bundle install

Khi thực hiện khởi tạo này, đồng nghĩa nó đã thực hiện 4 việc sau:

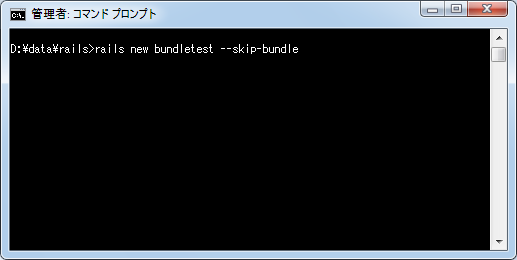
1. Khởi tạo những file quan trọng
2. Tạo ra Gemfile ghi lại những Gem pakage quan trong
3. Dựa vào Gemfile và cài đặt những Gem đó
4. Tạo ra Gemfile.lock để có thể sử dụng ở môi trường khác

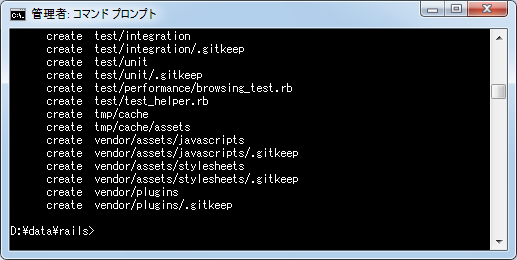
Và bây giờ ta thử tạo 1 rails app ko có bundler

Khi tạo project rails mà ko muốn sử dụng bundler thì ta chỉ việc khai báo

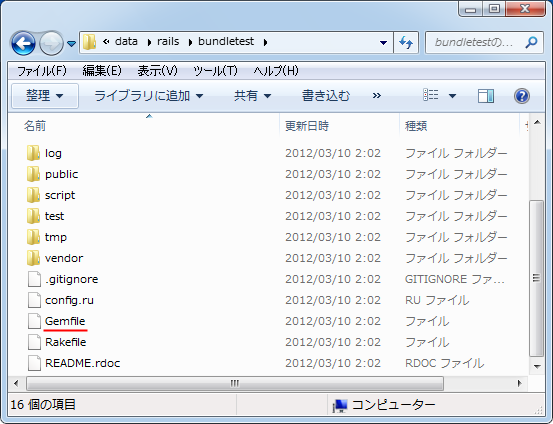
>rails new {tên\_project} --skip-bundle

Kết quả:





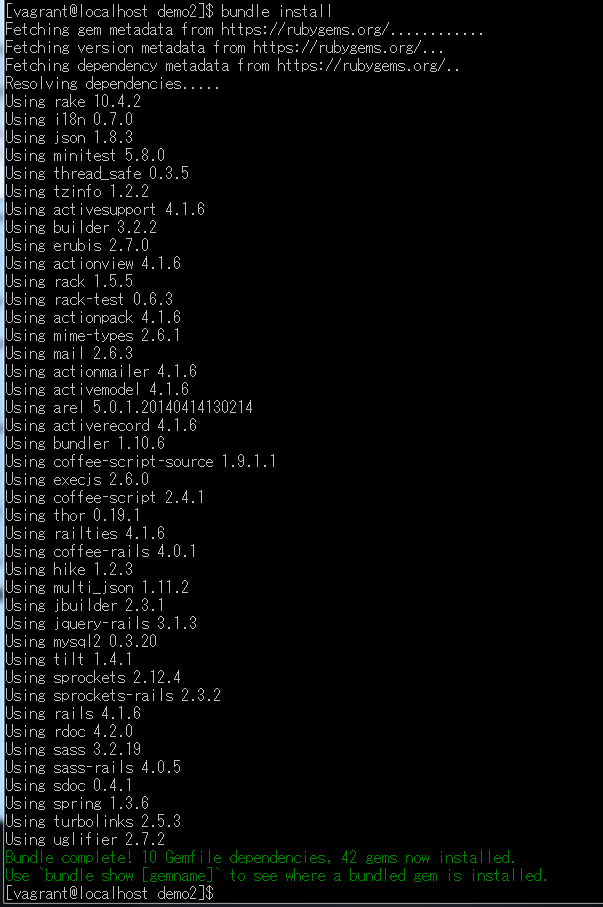
Như vây nghĩa là nó chỉ dừng lại ở công đoạn (1) là tạo các file quan trọng , còn nó ko thực hiện công việc “bundle install”. Tuy nhiên bây giờ ta hãy kiểm tra trong directory đã được đồng bộ với guest , ta sẽ thấy 1 file Gemfile đã được khởi tạo.



Mở file này ra ta sẽ thấy nó đã được khai báo các gem cần thiết, tuy nhiên thì các gem này vẫn chưa được cài đặt. Bây giờ ta sẽ đi vào bên trong project vừa tạo và thực hiện lệnh:

>bundle install

Thì sẽ thấy có 1 file Gemfile.lock được sinh ra và các gem cũng đã được cài đặt



Như vậy ta đã hiểu rõ hơn về nguyên lý cũng như cách làm việc của bundler khi cài đặt gem. Và nếu muốn cài gem ta chỉ cần khai báo nó trong Gemfile và thực hiện bundle install là được

+ Cài đặt Rails

> gem search -r rails | grep "^rails (" # check version rails tương thích

Đến đây nếu muốn cài đặt theo version vừa check thì chỉ cần

>gem install rails

Nếu không ta có thể cài đặt version cố định. Ở đây mình dùng version rails là 4.2.3 (vì bản ở trên lúc cài báo lỗi quái gì nên ko cài được)

>gem install rails --version 4.2.3



Cài đặt khá là mất tgian nên cần chờ đợi, sau khi cài đặt xong mình có thể check version bằng cách

>rails –v

C:\Users\Chu Quang Vien\Desktop\rails_version.PNG

Như vậy hiện tại mình sử dụng Ruby 2.2.2 và Rails 4.1.7 trên GuestOS là CentOS7.0.

Tham khảo:

+ <http://qiita.com/yamagh/items/61e540765fa4d2152dc4>

+ <http://qiita.com/Gin326M/items/01c8aef45f9a55b9b028>

+ <http://vietonrails.com/201411-cai-dat-virtualbox-%EF%BC%8B-vagrant-%EF%BC%8B-ruby-rails4-window7/>

==================================================================

=================================================================

Một số lỗi sẽ gặp phải trong quá trình cài đặt được mình memo ra đây.

**Lỗi 1**: Sau khi cài đặt xong thì khi chạy server rails thì nó lại tự exiting

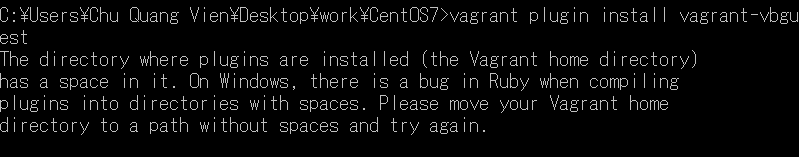
Thông báo lỗi

|  |
| --- |
| rails s  => Booting WEBrick  => Rails 4.0.4 application starting in development on http://0.0.0.0:3000  => Run `rails server -h` for more startup options  => Ctrl-C to shutdown server  A server is already running. Check /home/..name/rprojects/railsapp/tmp/pids/server.pid.  Exiting |

Và theo như hướng dẫn trong stackoverflow thì mình sẽ xóa file server.pid(theo đường dẫn) kia đi và khởi động lại server là sửa đc.Thành công !

**Lỗi 2:** Không thể tìm thấy đường dẫn đến thư mục home nên ko cài đặt plugin được

|  |
| --- |
| The directory where plugins are installed (the Vagrant home directory) has a space in it. On Windows, there is a bug in Ruby when compiling plugins into directories with spaces. Please move your Vagrant home directory to a path without spaces and try again |



Cách giải quyết:

+ <http://stackoverflow.com/questions/26102156/unable-to-install-plugin-hostmanager-in-vagrant-in-windows-8-1> (Nguyên nhân là do Ruby ko chấp nhận đường dẫn có dấu cách)

+ <http://qiita.com/taiko19xx/items/cdbe7e2946f7d5dfa237> (Giải quyết)

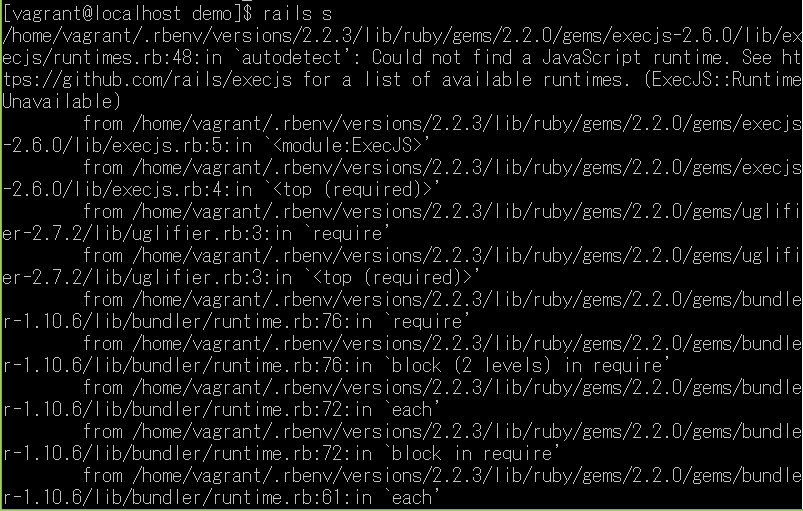
+ Tạo thêm 1 folder home bên trong C:¥HashiCorp¥Vagrant¥

+ Thêm biến môi trường VAGRANT\_HOME với đường dẫn C:¥HashiCorp¥Vagrant¥home

+ Chuyển folder về ổ C, tránh để bên trong các thư mục có đường dẫn chứa dấu cách(bên trên thì đường dẫn C:/Users/Chu Quang Vien/ có chứa dấu cách)



**Lỗi 3**: Start Server Rails thì lỗi Javascript runtime gì đó



|  |
| --- |
| …`autodetect': Could not find a JavaScript runtime. See https://github.com/sstephenson/execjs for a list of available runtimes. (ExecJS::RuntimeUnavailable) |

Lỗi là do thiếu nodejs

>sudo install nodejs # ko thực hiện được

+ Phương pháp fig: <http://succi.jp/blog/?p=1117>

$ cd .. # di chuyển về thư mục home của vagrant

$ git clone git://github.com/creationix/nvm.git ~/.nvm

$ source ~/.nvm/nvm.sh

$ nvm install v0.10.18

$ node --version # v0.10.18

Restart lại VM , tuy nhiên lần này vào thì vẫn bị lỗi rumtime của Rails do thiếu js @.@ Như vậy lần tiếp theo khởi động hệ thống mình vẫn phải làm công việc sau:

$ source ~/.nvm/nvm.sh

$ nvm install v0.10.18

$ node --version # v0.10.18

thì mới ko bị lỗi runtime của Java script. Mặc dù là đã chạy được app rails đầu tiên trên vagrant , nhưng nếu lỗi này tồn tại thì ko hài lòng chút nào

Ohh, làm theo hướng dẫn install javascript của link dưới đây thì sẽ fix được lỗi ở trên

+ <http://nekojarashi-daily-life.hatenablog.jp/entry/2015/07/09/071701> (Link này cung cấp cách giải quyết cho rất nhiều lỗi)

$ sudo yum -y install epel-release

$ sudo yum -y install nodejs npm

$ npm -g install bower

$ npm -g install coffee-script

$ npm -g install requirejs

$ npm -g install haml-coffee

**Ngày 26/08/2015**

Để tiếp tục việc học Rails và làm việc với Rails bằng cách sử dụng Vagrant để thiết lập môi trường thì trước hết ta sẽ đi nhắc lại 1 lần nữa kiến thức cơ bản.

**Các lệnh Linux cần nắm rõ**

ls : Liệt kê các thư mục hiện hành

ls –R : Liệt kê các thư mục con

ls /etc : Liệt kê thành phần bên trong thư mục /etc

ls –a :Liệt kê các thư mục kể cả ẩn

cd /etc : Di chuyển đến thư mục /etc

mkdir folder : Tạo thư mục folder

rm –rf demo : Xóa toàn bộ thư mục demo không rỗng

rmdir –v demo: Xóa thư mục demo rỗng

rmdir –vp demo/src : Xóa cả thư mục src và thư mục demo

pwd: Hiển thị đường dẫn đến thư mục hiện tại

cd.. : Quay về thư mục cha(thư mục cấp cap hơn)

cd : Chuyển về thư mục home( có thể dùng cd~)

cd - : Chuyển về thư mục trước(thư mục cha)

touch tên\_file: Tạo file mới

Trong unix thì (.) có nghĩa là thư mục hiện hành và (..) có nghĩa là cha của thư mục hiện hành

Trong linux , nếu dùng lệnh cd / ta sẽ được đưa đến thư mục gốc. Chính là cái thư mục “/” (pwd để xem đường dẫn).

Giờ mình sẽ đi phân tích thêm 1 chút về việc sử dụng file Vagrantfile.

Và ta bắt tay vào rails nào :D

Vậy là cài đặt môi trường Vagrant coi như đã xong, giờ mình sẽ test bằng cách tạo project Rails mới. Mình sẽ tạo các project trong thư mục rails\_example được sinh ra khi sửa file Vagrantfile.

>vagrant up

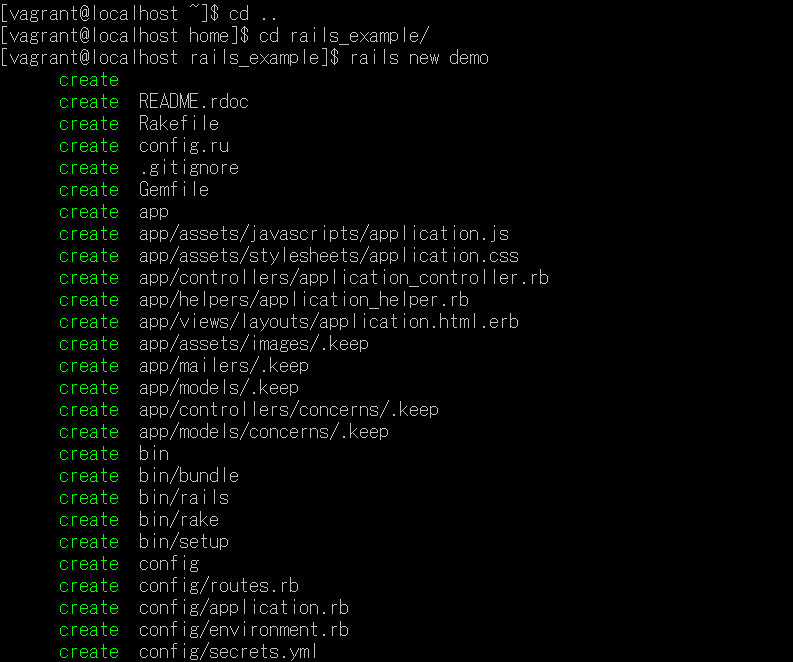
>vagrant ssh

>pwd # Kiểm tra đường dẫn

>cd .. # Về thư mục cha

/home/>cd rails\_example

/home/rails\_example>rails new demo



Nếu muốn tạo project rails theo version thì dùng lệnh

>rails \_x.x.x\_ new app\_name

Còn nếu ko thì dùng lệnh

>rails new app\_name # nó sẽ tự động cài với version default

Như vậy là ta đã tạo xong 1 project rails rồi

>cd app\_name # Vào trong thư mục rails app vừa tạo

>rails server (hoặc >rails s) # Khởi tạo server



Ra Win kiểm tra trong thư mục work/rails\_example/ có thêm folder demo không nhé :D

Bổ sung:

Cài đặt thêm 1 gem để hiển thị thư mục trong linux

>sudo yum –y install tree

#=> tree tên\_thư\_mục

Khi sử dụng máy tính cá nhân ở nhà, thì gặp 1 vấn đề đó là mạng ở nhà là mạng wifi, thì ko hiểu sao ko thể truy cập trực tiếp theo link: <http://localhost:3000> theo setup trong file config Vagrantfile.

Sau 1 hồi tìm hiểu thì mình làm thứ như sau:

Khởi động server bằng lệnh

>bundle exec rails s --bind=0.0.0.0

Hoặc

>rails s –b 0.0.0.0

Thì đã thành công, vì đây là cách chỉ định IP:0.0.0.0 thay vì để mặc định là 127.0.0.1

Tham khảo: <http://qiita.com/weakboson@github/items/9ef390679347460d6d77>

<https://teratail.com/questions/10530>