CoreOS-Vagrant-for-Windows-7

Documentation

**Table of Contents**

[1 Background 2](#_Toc428874325)

[1.1 Strongly Suggested Before You Start 2](#_Toc428874326)

[2 Running CoreOS on Vagrant 3](#_Toc428874327)

[2.1 Install Vagrant 3](#_Toc428874328)

[2.2 Install VirtualBox 3](#_Toc428874329)

[2.3 Clone Vagrant Repo 4](#_Toc428874330)

[2.4 Starting a Cluster 4](#_Toc428874331)

[2.4.1 user-data 4](#_Toc428874332)

[2.4.2 config.rb 4](#_Toc428874333)

[2.4.3 Vagrantfile 4](#_Toc428874334)

[2.4.4 Repeatable Launching of coreos-vagrant 5](#_Toc428874335)

[2.4.5 Bring up the Cluster 6](#_Toc428874336)

[2.4.6 Logging into the Guest Machines 6](#_Toc428874337)

[2.4.7 Find the insecure\_private\_key 6](#_Toc428874338)

[2.4.7.1 Configure PuTTY to Use SSH Key 7](#_Toc428874339)

[2.4.7.2 Connect to core-01 9](#_Toc428874340)

[2.4.7.3 Configure SSH Client and Connect to core-02 and core-03 10](#_Toc428874341)

[2.4.7.4 Information on the IP Addresses 11](#_Toc428874342)

[2.4.8 Review1: 12](#_Toc428874343)

[2.5 Shared Folders 13](#_Toc428874344)

[2.5.1 Notes on the VagrantFile Shared Folder Configuration Options 13](#_Toc428874345)

[2.5.2 RSYNC: 15](#_Toc428874346)

[2.5.2.1.1 Installing CYGWIN 15](#_Toc428874347)

[2.5.3 Set up for shared folders. 15](#_Toc428874348)

[2.5.3.1 Hacking the helper.rb File 16](#_Toc428874349)

[2.5.4 Testing the Changes 16](#_Toc428874350)

[2.5.5 Review2: 17](#_Toc428874351)

[2.6 Getting Files from the Guest VM 18](#_Toc428874352)

[3 Research Links Used 19](#_Toc428874353)

# Background

Being frustrated trying to get a quick working environment for CoreOS and was intrigued by the claim that using Vagrant would allow CoreOS run on a laptop initiated this foray. The problem is that most of the documentation out there is focused on the Linux / Apple communities and those of us coming from the Windows community are quite a bit behind the curve in familiarization and support. We are all on the bleeding edge of this technology so we will take quite a few bumps and more so being the redheaded step child.

Note: At least a month of playing with CoreOS, Docker and RancherOS on a VMWare ESX 5.5 and Hyper-V 2008R2 that was upgraded to Hyper-V 2012R2 has been conducted before this venture into coreos-vagrant. Having to play DEV-OPS and have less time to explore CoreOS / Docker is a source of frustration. CoreOS-Vagrant working on Windows 7 was not straight forward and much time and effort in researching a path through the set up prevents the Docker playground time.

This document is to assist other early adopters to join the fun of getting CoreOS clusters working with some Docker applications on existing equipment. Hopefully you will find this document useful in aiding your path to a creating a cluster on your Windows laptop that wasn’t downloaded from Microsoft ☺

Read all of this information because it is required. Skips may be suggested where possible. Assume that you select default install paths for all packages.

## Strongly Suggested Before You Start

You have a 64 bit processor. Were never successful getting CoreOS to run under 32bit Ubuntu 14 OS.

You have a GitHub account <https://github.com/join> and have the GitHub desktop installed on your machine <https://desktop.github.com/> which will allow you to perform the steps below.

Have Putty (or other SSH client) installed on your machine and be familiar with PuTTYgen and the PuTTY client. However, if you have a preferred SSH client that you use and know how to install SSH keys then use that. There is no restriction on the SSH client.

# Running CoreOS on Vagrant

The guide provided by CoreOS is very helpful. There are only a few branches suggested to avoid the blind allies found.

## Install Vagrant

Navigate to this link and follow the steps to install Vagrant and Virtual Box

<https://coreos.com/os/docs/latest/booting-on-vagrant.html>

Here is Vagrant: <http://www.vagrantup.com/downloads.html>

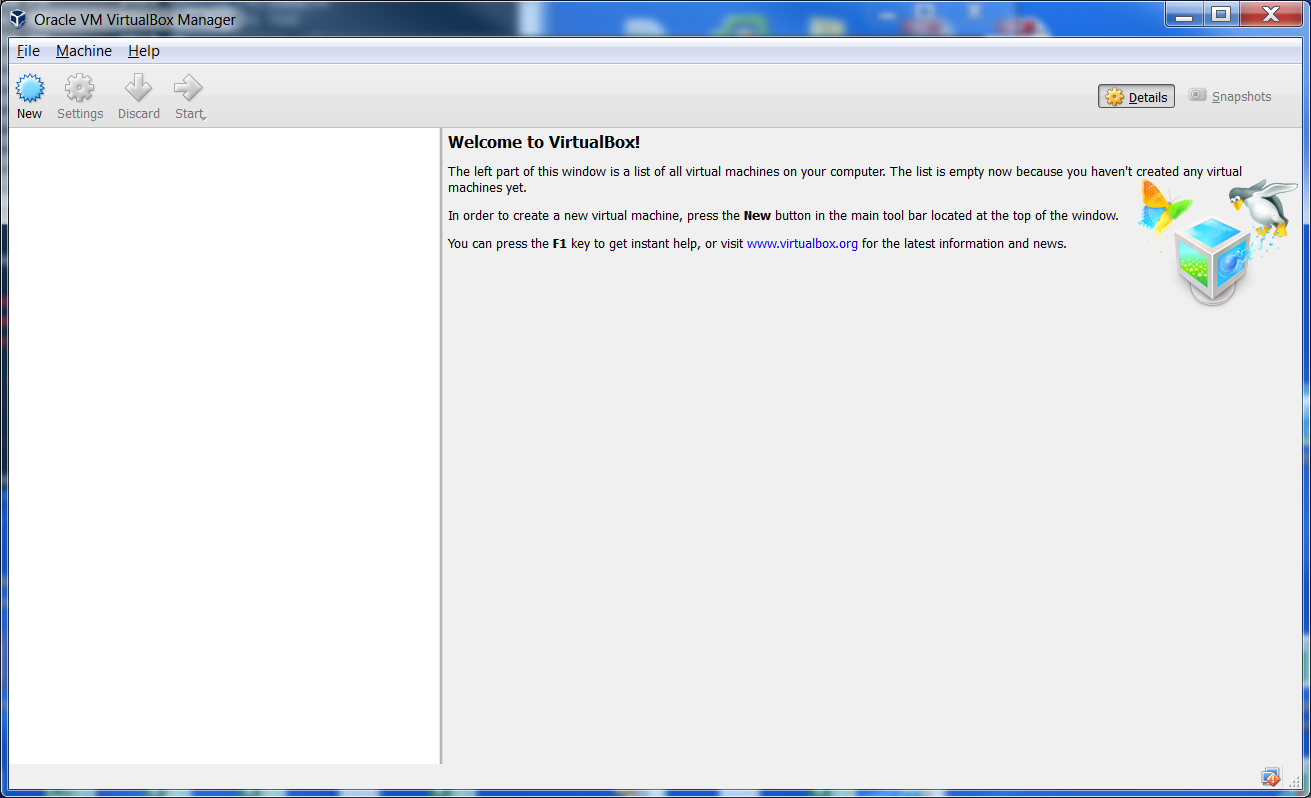
Install Vagrant and reboot.

## Install VirtualBox

Need to install VirtualBox from Oracle to allow hypervisor support under Windows 7.

* **VirtualBox platform packages**. The binaries are released under the terms of the GPL version 2.
  + **VirtualBox 5.0.2 for Windows hosts** [x86/amd64](http://download.virtualbox.org/virtualbox/5.0.2/VirtualBox-5.0.2-102096-Win.exe)

Here is Virtual Box: <http://download.virtualbox.org/virtualbox/5.0.2/VirtualBox-5.0.2-102096-Win.exe>

Download and install VirtualBox, install and start it!

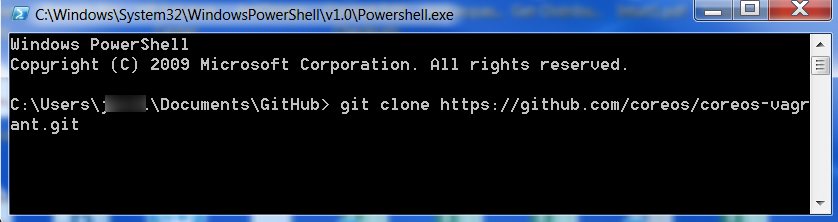
Virtual Box Manager

Nothing is there so on with the steps from Running CoreOS on Vagrant.

## Clone Vagrant Repo

<https://coreos.com/os/docs/latest/booting-on-vagrant.html>

See: Clone the Vagrant Repo



NOTE: you may want to open a GitHub 🡪 Git Shell to make this work. Why the GitHub Desktop was installed.

## Starting a Cluster

Move on to the Staring a Cluster section.

### user-data

Copy user-data.sample to user-data

The user-data file contains the #cloud-config file for the CoreOS machine.

Optional: Suggest you comment out the etcd: section as this has been depreciated. It is not required to comment out the etcd: section. Configure the etcd2: section of the user-data “#cloud-config”.

To support the configuration you use <https://discovery.etcd.io/new?size=3> to get a cluster token. As it says, you need to do this every time you destroy your environment! {Will investigate getting this automated as it should be one of the items that can be done. Code is provided but does not behave as expected.}

### config.rb

Copy config.rb.sample to config.rb

Change $num\_instances=1 to $num\_instances=3

### Vagrantfile

Suggest using the stable channel. Why would you want to introduce more variables when you are just getting things to work. For now there are no changes required for this file.

We now are ready to start up the machines for the first time.

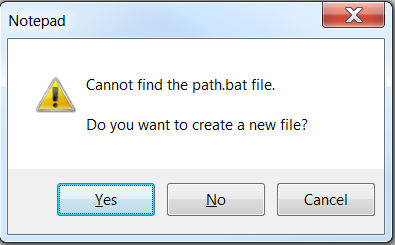
Not everything is configured but you need to go through the following exercise before moving forward anyway.

### Repeatable Launching of coreos-vagrant

You will always want to start a cmd window in your coreos-vagrant directory. Suggest creating a link on the desktop where Target = “C:\Windows\System32\cmd.exe” and Start In = “C:\Users\<username>\Documents\GitHub\coreos-vagrant” where you cloned the coreos-vagrant repository. This will take you to that directory every time.

Make sure the path for virtual box is available!

Create a nice utility to make sure everything is available for you to run vagrant.

At the command prompt in coreos-vagrant type “notepad path.bat”

Create the file when prompted. Click Yes.

In the empty file cut and paste the following and then save the file.

PATH = %PATH%, C:\Program Files\Oracle\VirtualBox;

Now when you launch your shortcut to the coreos-vagrant command line environment you want to invoke the PATH.BAT as the first thing you do to allow access to the VirtualBox program.

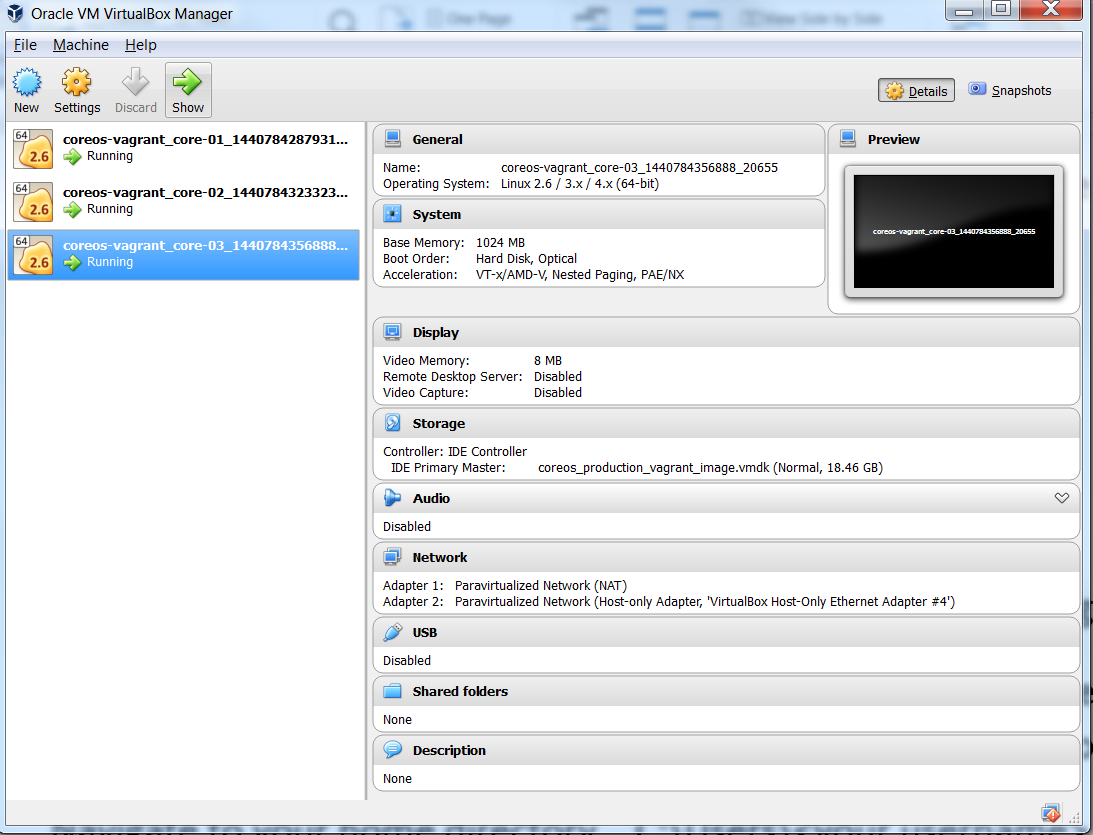
PATH.BAT

Make sure the path for your ssh client is available if you are not using PuTTY (any “YDS” comments are welcome to let us know how to get PuTTY to work from the command line. Did not find it easy and abandoned it since using the nice interface going through the PuTTY client works fine.

You may want to add your ssh client path to the PATH.BAT file.

### Bring up the Cluster

Make sure that you have VirtualBox running.

vagrant up

VirtualBox Manager with Initial VMs Running

This should successfully bring up three CoreOS VMs under Vagrant.

### Logging into the Guest Machines

If you are using the PuTTY client then you need to perform the following steps **BEFORE YOU CAN SSH INTO THE MACHINES**! You probably may need some of this for other SSH clients too. Skip at your own risk.

### Find the insecure\_private\_key

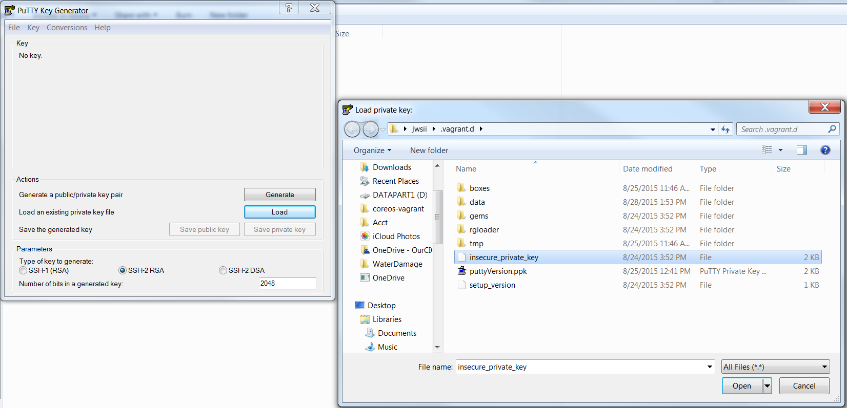
Navigate to your home directory. C:\Users\<username>

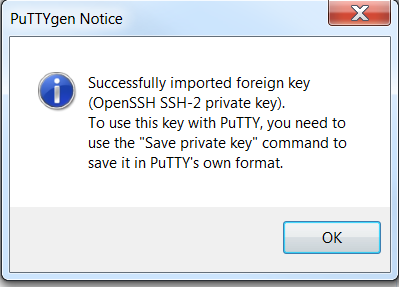
From the cmd prompt you can perform a “cd %HOMEPATH%” to change to your home directory.

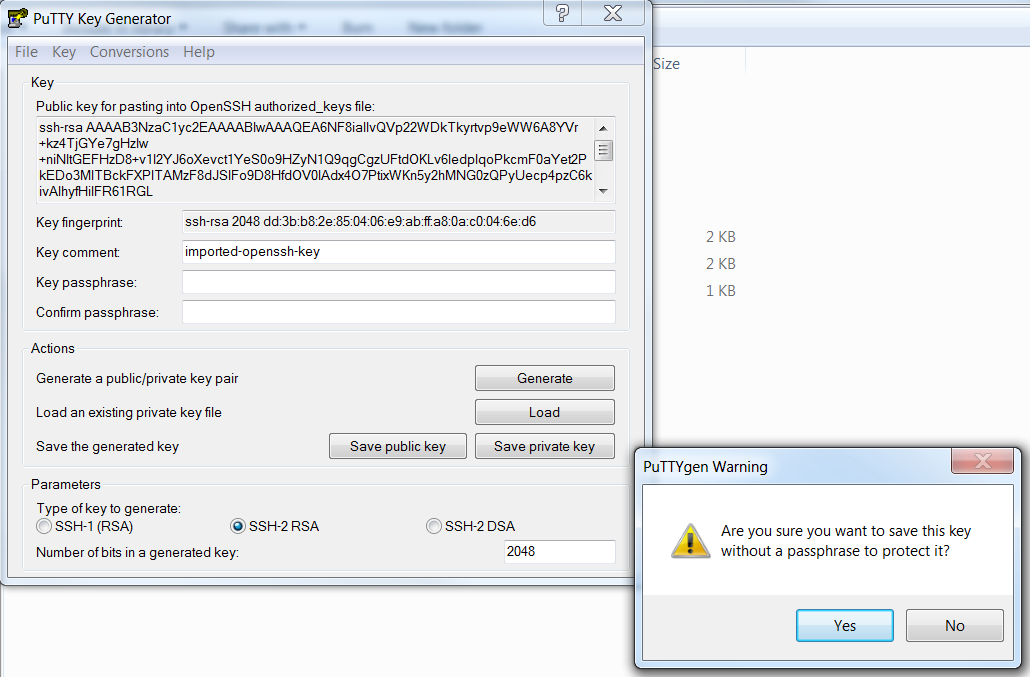
Under this directory find a folder “.vagrant.d” that will contain the insecure\_private\_key needed.

#### Configure PuTTY to Use SSH Key

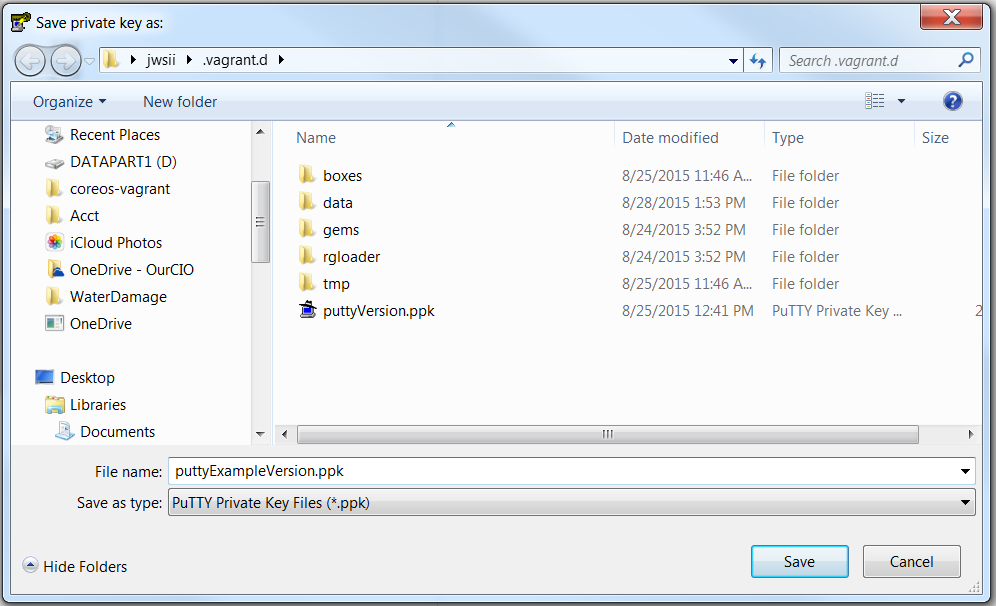
Using PuTTYgen Load an existing private key file.



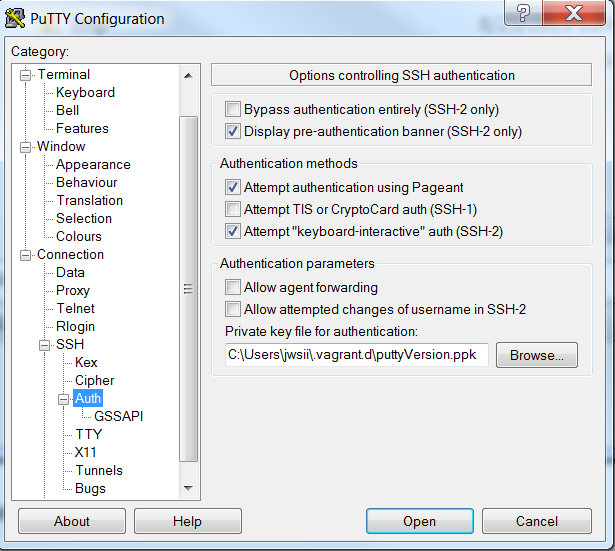
Note: Change the file type to All Files the insecure\_private\_key file can be found.

You are presented a reassuring message box that the SSH-2 key was imported. Click OK.

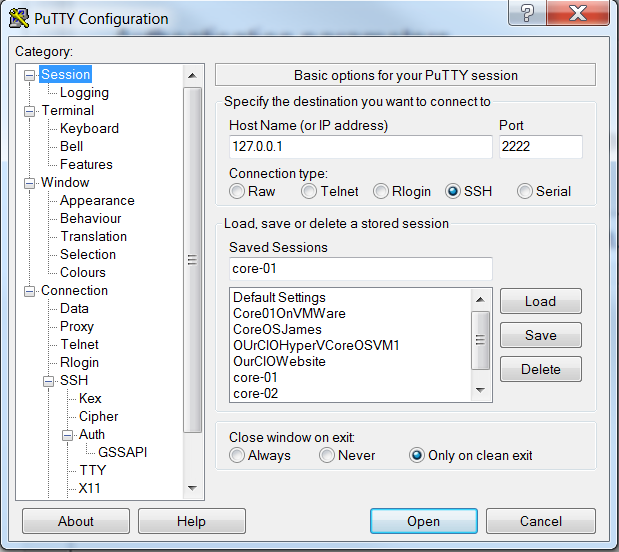
You are prompted with a warning about not having a passphrase. Click Yes



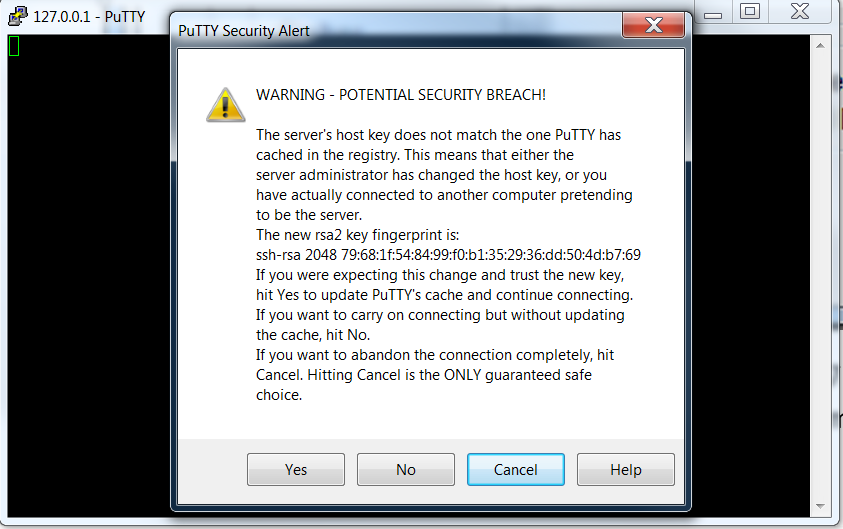
Suggest saving to the .vagrant.d folder with a “putty” prefix. Showing a “puttyExample” prefix for the documentation. Save the PuTTY private key file.

Close PuTTYgen and open the PuTTY client.

Under Connection 🡪 SSH 🡪 Auth browse to the puttyVersion of the private key just saved for use as the “Private key for authentication”.

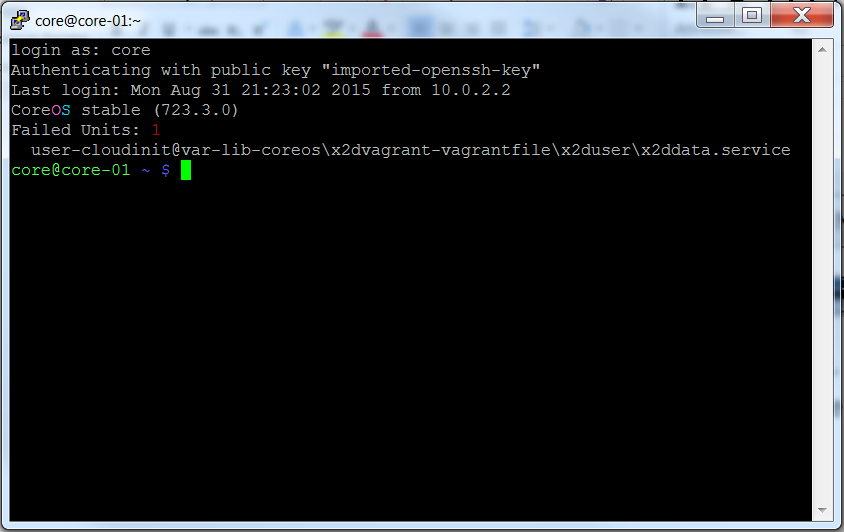
Shown above are the session connection basic options. 127.0.0.1:2222 is the IP and port that core-01 was generated. We strongly suggest that you save this information as core-01 so you can use it later.

#### Connect to core-01

After you save it, then open the connection.

You get a security warning from PuTTY since it knows nothing about the server which you are trying to connect.

Clicking No allows you to proceed without saving. Clicking Yes will save the server fingerprint in your PuTTY cache. You can just as easily read the material on the screen. “As with every security option, abstinence is the only guarantee.”



Logging in as core will give you the information above.

#### Configure SSH Client and Connect to core-02 and core-03

Similarly for core-02 and core-03 you can configure and save PuTTY connections.

The information is available as you bring up the machines but is listed here for your convenience.

Open the PuTTY client. Load the core-01 configuration. Change the port from 2222 to 2200 and save as core-02. Since the PuTTY version of the insecure\_private\_key was already configured in core-01 it comes along. Change the port from 2200 to 2201 and save as core-03. Now we have 3 PuTTY sessions configured for use.

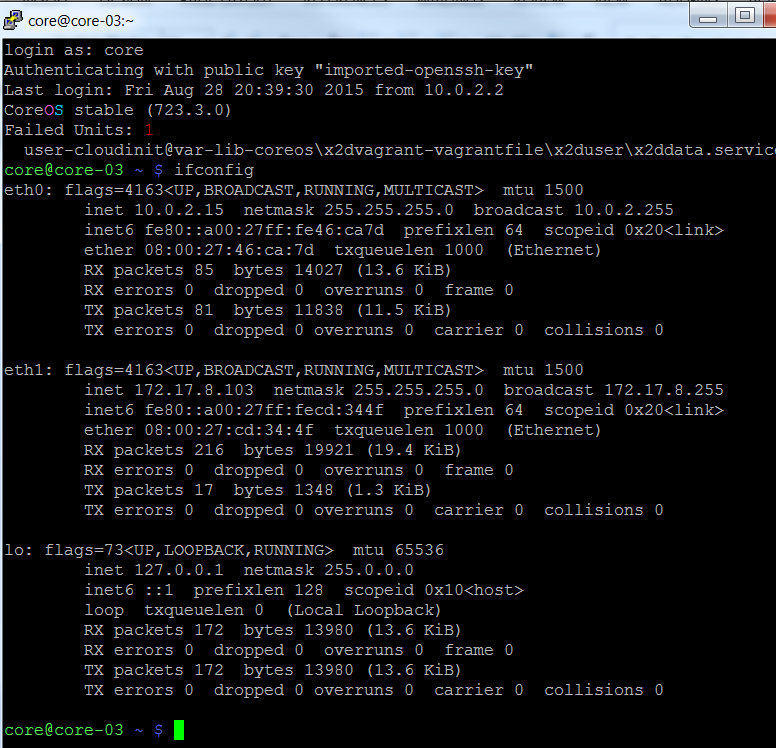
Log into each server to verify you have things configured properly.

#### Information on the IP Addresses

Informational you can skip to Review:

Find the IP address that was assigned to the VM and you can ssh on that ip using port 22

172.17.8.101:22, 172.17.8.102:22, 172.17.8.103:22 (need to use the insecure\_private\_key and core username). See the example of getting IP addresses from core-03 shown below:



We see our local host loopback IP 127.0.0.1, the virtual network IP 172.17.8.103. We also see an entry for eth0: that is the “NIC” card for the VM. This is the same on all of the VM machines and provides a path to the internet.

### Review1:

We have installed Vagrant and VirtualBox on the laptop.

Used Git to clone the vagrant repository.

Copied user-data.sample to user-data and config.rb.sample to config.rb and left the Vagrant file alone for the time being.

Got a cluster key and opened the user-data file. Modified the discovery element with the new key.

discovery: <https://discovery.etcd.io/cad301173cde933bec00f6c2f167a789>

Of course this token will not work for you.

Opened the config.rb file and changed $num\_instances=1 to $num\_instances=3.

Created a desktop shortcut and batch file to allow us to quickly configure a working environment to launch coreos-vagrant cluster.

We brought up a cluster of 3 CoreOS machines

Configured the SSH client to use the insecure\_private\_key

Successfully logged into the Guest machines using the core username and the insecure\_private\_key supplied using PuTTY or some other SSH client

.

## Shared Folders

We now digress from the documentation on Share Folder Setup completely. We agree that having shared folders would make it nice to get code and Dockerfiles into CoreOS. Unfortunately Windows 7 does not support the snippet supplied in <https://coreos.com/os/docs/latest/booting-on-vagrant.html> section on Shared Folder Setup.

config.vm.synced\_folder ".", "/home/core/share", id: "core", :nfs => true, :mount\_options => ['nolock,vers=3,udp']

### Notes on the VagrantFile Shared Folder Configuration Options

So let’s look at the options. (You can skip to RSYNC: since this is just information and not instructions.)

In the Vagrant file supplied you will find the notes on the various options.

############################################################################

# This was found as an option to open a file share but it is for a single

# linux machine instance. nfs is also a non windows type

############################################################################

#config.vm.network "private\_network", ip: "172.17.8.150"

#config.vm.synced\_folder ".", "/home/core/share", id: "core", :nfs => false, :mount\_options => ['nolock,vers=3,udp']

############################################################################

# uses nfs does not work with windows

# Uncomment below to enable NFS for sharing the host machine into the coreos-vagrant VM.

# config.vm.synced\_folder "shared", "/home/core/share", id: "core", :nfs => true, :mount\_options => ['nolock,vers=3,udp']

##############################################################################

#config.vm.synced\_folder "./shared", "/home/core/share", type: "nfs"

############################################################################

# Windows users: NFS folders DO NOT WORK on Windows hosts.

# Vagrant will ignore your request for NFS synced folders on Windows

############################################################################

# nfs does not work with windows

############################################################################

# This option is not in the Vagrant documentation

############################################################################

#config.vm.synced\_folder "./shared", "/home/core/share", type: "docker"

# docker is not documented ?

############################################################################

# using the VirtualBox provider, then VirtualBox shared folders are the

# default synced folder type.

# These synced folders use the VirtualBox shared folder system to sync

# file changes from the guest to the host and vice versa.

############################################################################

# config.vm.synced\_folder "./shared", "/home/core/share", type: "virtualbox"

# virtualbox does not work

############################################################################

# Windows only! SMB is currently only supported when the host machine is Windows.

# The guest machine can be Windows or Linux

############################################################################

#config.vm.synced\_folder "./shared", "/home/core/share", type: "smb"

# config.vm.synced\_folder "shared", "/home/core/share", type: "smb" :mount\_options => ['nolock,vers=3,udp']

# smb does not work

############################################################################

# On Windows, rsync installed with Cygwin or MinGW will be detected by Vagrant

# and works well.

############################################################################

# config.vm.synced\_folder "./shared", "/home/core/share", type: "rsync"

# Results: "rsync: change\_dir: "..." failed: No such file or directory (2)"

#####################################################################################

# config.vm.synced\_folder "c:/Users/mylaptop/Documents/GitHub/coreos-vagrant/shared", "/home/core/share", type: "rsync"

# Results: "the host path of the shared folder is missing: "...""

######################################################################################

# config.vm.synced\_folder "/cygwin64/home/jwsii/shared/", "/home/core/share/", type: "rsync"

##########################################################################################

# This configures a sync folder from the host to the guest.

##########################################################################################

# config.vm.synced\_folder "/cygwin64/home/jwsii/shared", "/home/core/share2/", type: "rsync"

config.vm.synced\_folder "./shared", "/home/core/share2/", type: "rsync"

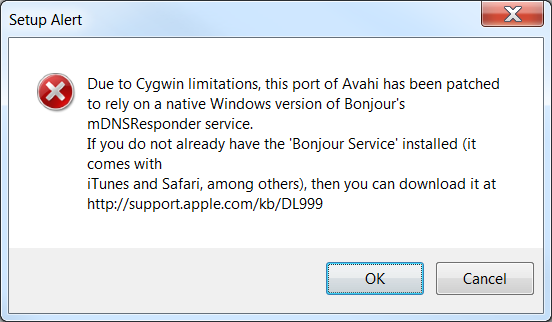
#Yea! rsync works but requires install of a package. I used cygwin.

### RSYNC:

We find that the only working option at this time is rsync. But rsync requires a Linux API so that it will work. Took the suggestions from <http://www.howtogeek.com/175008/the-non-beginners-guide-to-syncing-data-with-rsync/> The Non-Beginner’s Guide to Syncing Data with Rsync.

##### Installing CYGWIN

Follow the instructions on how to install Cygwin minimal packages. Used lug.mtu.edu as the download site.

Note on selecting packages. Search for rsync then click the Net install. Search for vim and then find the editor. Search for SSH and then find openssh.

Download and install Bonjour if you don’t have it.

Once this is installed you are ready for the next steps.

Add a path to your cygwin64 directory. (It is assumed you are using 64 bit Cygwin since CoreOS does not work with 32 bit processors.)

From the coreos-vagrant command prompt type “notepad path.bat” which launches the batch file for editing. Cut and paste the following to the end of the line. We are going to add a path to wordpad.exe too. This comes in handy in a minute for those that like to use command line commands.

C:\cygwin64\bin;C:\Program Files\Windows NT\Accessories

Results in:

PATH=%PATH%,C:\Program Files\Oracle\VirtualBox;C:\cygwin64\bin;C:\Program Files\Windows NT\Accessories

### Set up for shared folders.

In your coreos-vagrant directory create a subdirectory “shared”.

Navigate to the shared subdirectory and create a new file “ItWORKS.txt”.

In the folder sync section of the Vagrantfile add this line of code. (Or use the Vagrantfile supplied in this repository.)

config.vm.synced\_folder "./shared", "/home/core/share/", type: "rsync"

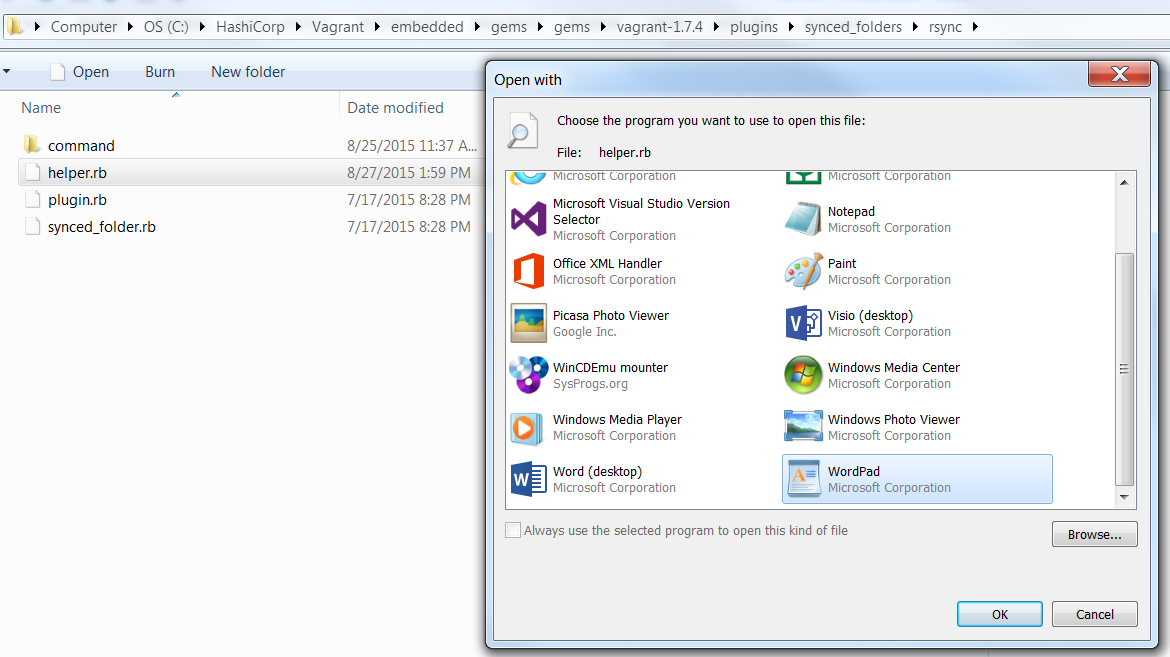
Save your Vagrantfile.

#### Hacking the helper.rb File

Now some hacking is in order.

Navigate to the “C:\HashiCorp\Vagrant\embedded\gems\gems\vagrant-1.7.4\plugins\synced\_folders\rsync\” folder to find the helper.rb file.

Open it using WORDPAD! This will preserve the formatting and give you a fighting chance.

DO NOT SAVE THIS IN TEXT FORMAT. A save to the file should be clean without any save as options. Windows will put a /r at the end of line and really hose up the ruby file. You have to type all of the entries without using cut and paste from outside of the file. If you cut and paste from outside of the file, WordPad believes you have changed the file attributes. Just another nicety.

Add the following code to the helper.rb file:

if Vagrant::Util::Platform.windows?

# rsync for Windows expects cygwin style paths, always.

hostpath = Vagrant::Util::Platform.cygwin\_path(hostpath)

hostpath = "/cygdrive" + hostpath

end

### Testing the Changes

Try it out on one of the vm instances.

vagrant reload core-01

This will reload the core-01 VM with the changes made to the Vagrant file (user-data and config.rb too!).

That was easy huh?

If you did not disconnect from the SSH connection before the vagrant reload command you notice that the connection terminated. Makes sense since the VM cycled and came up with a totally new VM. It just happens to have all of the same attributes as the previous core-01 for compute, memory and network. The disk has changed!

Log into the core-01 machine and look for the share directory under /home/core by typing “ls”.

Then type “ls share” and see if ItWORKS.txt is found.

Repeat for “vagrant reload core-02” and “vagrant reload core-03”.

We see that the configurations in the Vagrantfile created a new directory in the /home/core folder called “share”. The reload is actually instantiating a new copy of the CoreOS machine with the new configurations.

### Review2:

Optionally read about file sharing that does not work in Windows 7 and what does.

Installed rsync and Cygwin.

Created a shared folder under coreos-vagrant where we can place files to push to the guest machines.

Modified the batch file to find the Cygwin executables and find WordPad.

Changed the Vagrant file to share the folders.

Hacked the helper.rb file so that it will have the proper “/cygdrive” path.

Reloaded the guest machines with the configuration upgrade.

Logged in and found our ItWORKS.txt file we placed in the shared directory.

If that is all you want to do, have at it and don’t bother with the rest of this text.

## Getting Files from the Guest VM

Moving files from the guests to the host.

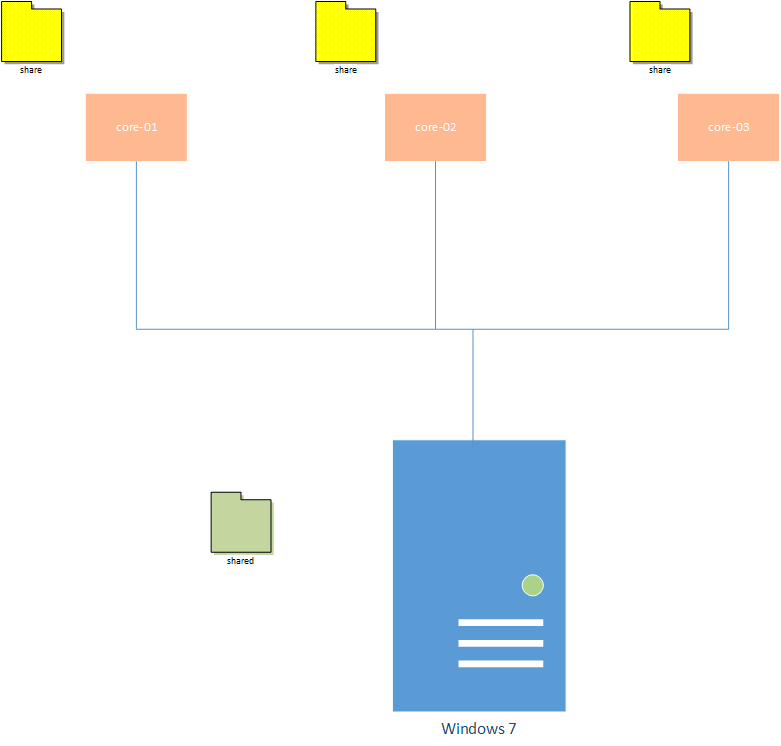
Getting files from the host to the guest is very helpful in configuring the CoreOS / Docker environments. For well-formed CoreOS / Docker configurations this may be all you need. The immediate need to get files back from the Guest machines becomes evident when you trouble shoot a problem on the Guest and have the solution in a file.

Remember your coreos-vagrant environment is ephemeral by design. You will often ”vagrant destroy” the environment to allow a fresh build of the machines. This supports the continuous improvement cycle in building and deploying clusters and applications. So addressing the problem of saving tested configurations is crucial.

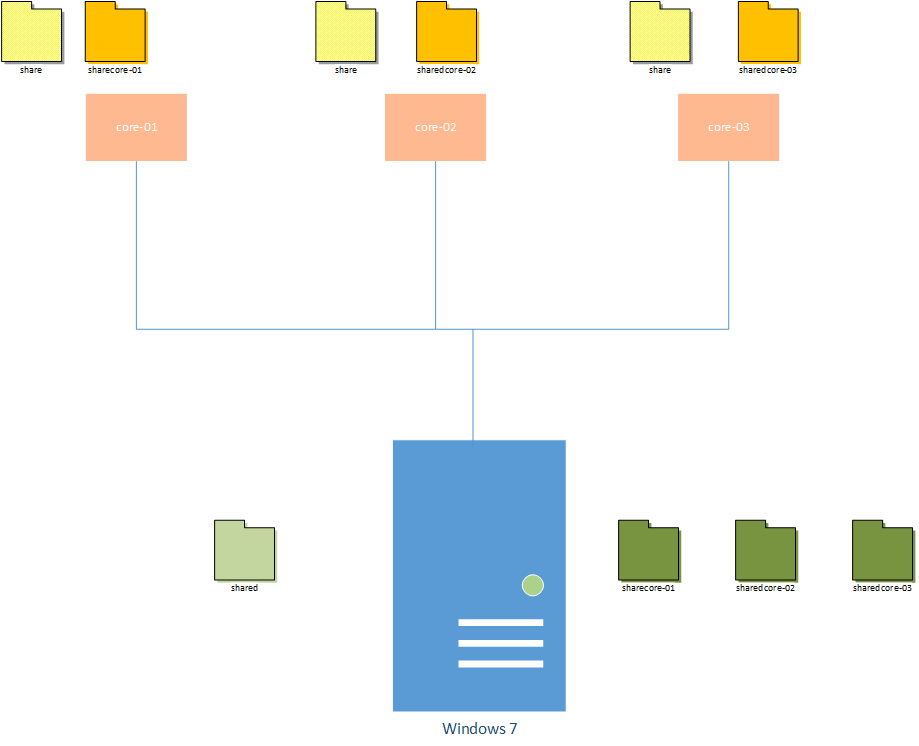
Copy and paste may work for some cases but creating shell scripts are not that forgiving in our Windows 7 coreos-vagrant environment. The nasty /r placed at the end of Windows files cause the shell scripts to fail. So we need a way to move these files back and forth after we create a working solution so they can be used in

There are some utilities like dos2unix that will convert your files.

### Description of Guest to Host Folder Synchronization using rsync

We are going to create methods to synchronize a folder on the guest with a folder on the host. Using rsync if we use the same folder on the host, each of the synchronizations from the guests would possibly overwrite the file.

Currently our configuration has a “shared” folder on the Windows 7 host with share folders on each of the Guests. It is easy to visualize that as we bring up 1, 2 and then 3 the contents of the Host “shared” folder is written to each Guest “Share” folder. If the Guests are then synchronized when we bring them down the contents would be written to the “shared” folder possibly overwriting the contents changed from one Guest by another.



Now consider a set of folders that correspond to the Guest VM exists along with the “shared” folder. Let the folders be created and update with contents from the Host on load and then find a way to update the Host-guest-specific folders from the Guest corresponding folders. That way we can create a fresh copy of the Guest OS with both universal (share) and specific (sharecore-0x) files for use. We can move work back from the Guests to the Host and preserve our hard work.

Simple enough right?

#### Setting up Folders for Guest Specific Folder Sharing

Launch your command prompt to coreos-vagrant. Issue the following commands by typing or cut and paste into the command window.

mkdir sharecore-01

mkdir sharecore-02

mkdir sharecore-03

#### Modifications to Vagrantfile

You can use the Vagrantfile.guestShare supplied in the repository or modify your own Vagrantfile with the following configuration. Place it after the configuration of the previous shared folder.

##########################################################################################

# Would like to do some things:

# sync a folder from the guest to the host

# have each guest folder be a unique name

##########################################################################################

config.vm.synced\_folder "./share" + vm\_name, "/home/core/" + vm\_name + "/", type: "rsync"

This takes care of configuring the folders on each Guest VM as we boot it up. The vm\_name variable is defined and set by the original Vagrantfile above.

…

(1..$num\_instances).each do |i|

config.vm.define vm\_name = "%s-%02d" % [$instance\_name\_prefix, i] do |config|

config.vm.hostname = vm\_name

…

Save the changes to your Vagrantfile.

##### Test the Guest Shared Folders

From the coreos-vagrant command prompt issue the following to test core-01 with these changes.

vagrant reload core-01

This should bring the Guest down gracefully and back up with the /home/core/core-01 folder on the Guest. Once it is reloaded and running, log into the Guest as core and “ls” to see the folder.

Optional exercise would be to create text files in your coreos-vagrant\

NOTES

Vagrant destroy

/\* We need to get rid of the previous ssh signatures for the old hosts \*/

/\* Assume you are doing this for all of your vagrant hosts \*/

Navigate to the .ssh directory

cd c:\sygwin64\home\<username on your laptop>\.ssh

del known\_hosts

/\* to get rid of selected hosts edit the file \*/

<<<<<<<<<<<<<<<<<<<<<<<

Log into core-03

Cd core-03

Touch coreo3stuff{1..3}

Log into core-02

Cd core-02

Touch core02stuff{1..2}

Log into core-01

Cd core-01

Touch core01stuff1

# Research Links Used

Attempt to document the work of others has been done. Apologies for any suggestions used and not credited. These were errors of omission and not done blatantly. Included by reference are subsequent downstream links from these sites.

<https://coreos.com/os/docs/latest/booting-on-vagrant.html>

<https://github.com/coreos/coreos-vagrant/issues/185>

<https://github.com/mitchellh/vagrant/issues/1827>

<https://hub.docker.com/r/yungsang/coreos-vboxguest/>

<http://www.howtogeek.com/175008/the-non-beginners-guide-to-syncing-data-with-rsync/>

<http://docs-v1.vagrantup.com/v1/docs/config/vm/share_folder.html>

<https://github.com/mitchellh/vagrant/issues/3086>

<https://github.com/mitchellh/vagrant/issues/3230>