CoreOS-Vagrant-for-Windows-7

Documentation

James W. Schott II

**Table of Contents**

[1 Background 3](#_Toc429032407)

[1.1 Strongly Suggested Before You Start 3](#_Toc429032408)

[1.1.1 Packages Used 3](#_Toc429032409)

[1.1.2 Host System Tested 3](#_Toc429032410)

[2 Running CoreOS on Vagrant 4](#_Toc429032411)

[2.1 Install Vagrant 4](#_Toc429032412)

[2.2 Install VirtualBox 4](#_Toc429032413)

[2.3 Clone Vagrant Repo 5](#_Toc429032414)

[2.4 Starting a Cluster 5](#_Toc429032415)

[2.4.1 user-data 5](#_Toc429032416)

[2.4.2 config.rb 5](#_Toc429032417)

[2.4.3 Vagrantfile 5](#_Toc429032418)

[2.4.4 Repeatable Launching of coreos-vagrant 6](#_Toc429032419)

[2.4.5 Bring up the Cluster 7](#_Toc429032420)

[2.4.6 Logging into the Guest Machines 7](#_Toc429032421)

[2.4.7 Find the insecure\_private\_key 7](#_Toc429032422)

[2.4.7.1 Configure PuTTY to Use SSH Key 8](#_Toc429032423)

[2.4.7.2 Connect to core-01 10](#_Toc429032424)

[2.4.7.3 Configure SSH Client and Connect to core-02 and core-03 11](#_Toc429032425)

[2.4.7.4 Information on the IP Addresses 12](#_Toc429032426)

[2.4.8 Review1: 13](#_Toc429032427)

[2.5 Shared Folders 14](#_Toc429032428)

[2.5.1 Notes on the VagrantFile Shared Folder Configuration Options 14](#_Toc429032429)

[2.5.2 RSYNC: 16](#_Toc429032430)

[2.5.2.1.1 Installing CYGWIN 16](#_Toc429032431)

[2.5.3 Set up for shared folders. 16](#_Toc429032432)

[2.5.3.1 Hacking the helper.rb File 17](#_Toc429032433)

[2.5.4 Testing the Changes 17](#_Toc429032434)

[2.5.5 Review2: 18](#_Toc429032435)

[2.6 Getting Files from the Guest VM 19](#_Toc429032436)

[2.6.1 Description of Guest to Host Folder Synchronization using rsync 19](#_Toc429032437)

[2.6.1.1 Setting up Folders for Guest Specific Folder Sharing 20](#_Toc429032438)

[2.6.1.2 Modifications to Vagrantfile 21](#_Toc429032439)

[2.6.1.3 Test the Guest Shared Folders (Push) 21](#_Toc429032440)

[2.6.1.4 The rsyncvmshare.bat 22](#_Toc429032441)

[2.6.1.5 Solving the Guest Username Challenge 23](#_Toc429032442)

[2.6.1.6 Test the Guest Shared Folder (Pull) 24](#_Toc429032443)

[2.6.2 Review 3: 24](#_Toc429032444)

[2.7 Recreate the Cluster 25](#_Toc429032445)

[2.7.1 Removing the Old Host SSH Signatures 25](#_Toc429032446)

[2.7.2 Bring up a Fresh Cluster 27](#_Toc429032447)

[2.7.3 Exercise for the Fresh Cluster 27](#_Toc429032448)

[2.7.3.1 Setting up the Guest Machines for Our Exercise 27](#_Toc429032449)

[2.7.4 Exercise for Fresh Cluster Shared Folders 28](#_Toc429032450)

[2.7.5 Review 4: 30](#_Toc429032451)

[2.8 Known Issues 31](#_Toc429032452)

[3 Research Links Used 32](#_Toc429032453)

# 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.

This document does not contain all setup steps. References to other works are suggested in place and a list of sources is provided at the end for your own research and interpretation.

### Packages Used

To accomplish this install the following packages have been used.

* Vagrant
* VirtualBox
* GitHub Desktop
* PuTTY
* Cygwin (MinGW is an alternative
* but it was not used here)

### Host System Tested

Dell Precision M6800, Windows 7 Pro, 6.1.7601 Service Pack 1 Build 7601, x64-based PC, Intel® Core™ i7-4900MQ CPU @ 2.80GHz, 2801 Mhz, 4 Core(s), 8 Logical Processor(s), BIOS Dell Inc. A09, 6/26/14, 16.0 G Physical Memory

# 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>

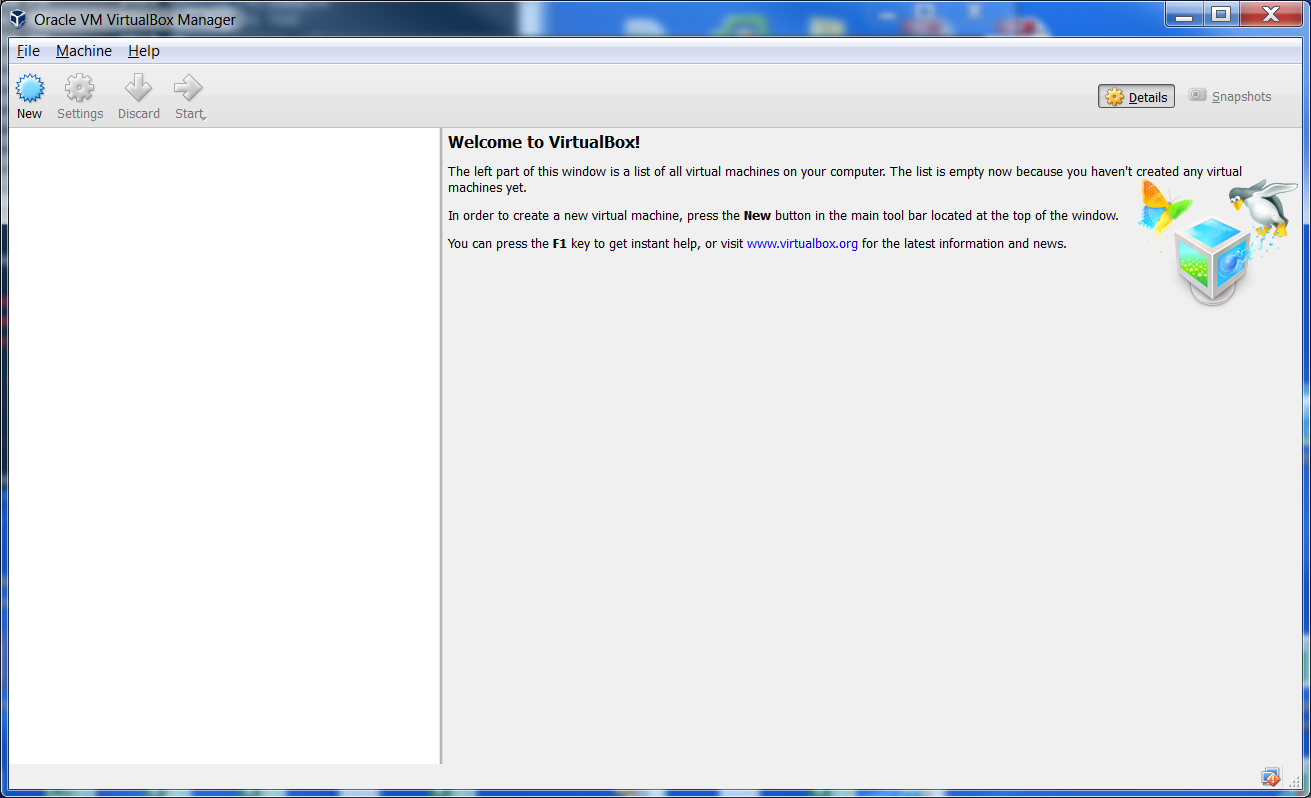
Install Vagrant and reboot.

## 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!

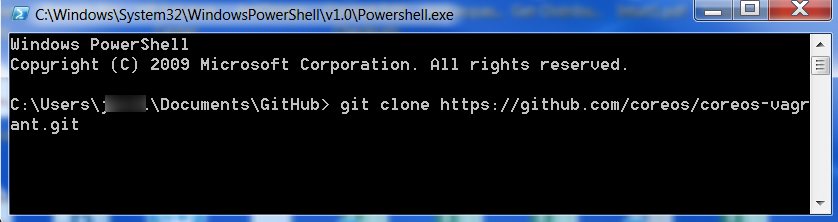
Virtual Box Manager

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. This is 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. Alternatively copy user-data.jschott2.endstate 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”. This is equivalent to using user-data.jschott2.endstate as your user-data file.

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 work. Code is provided but did not behave as expected with minimal attempts.}

### config.rb

Copy config.rb.sample to config.rb or alternatively use config.rb.jschott2.endstate as config.rb that have this change already.

Change $num\_instances=1 to $num\_instances=3 ***Be sure to save your changes!***

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

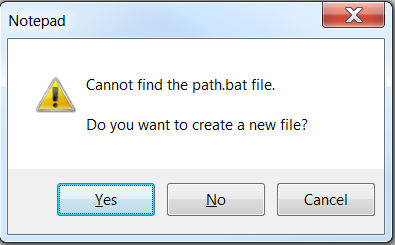
Not everything is configured but you need to go through the following exercise before moving forward anyway.

### 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.

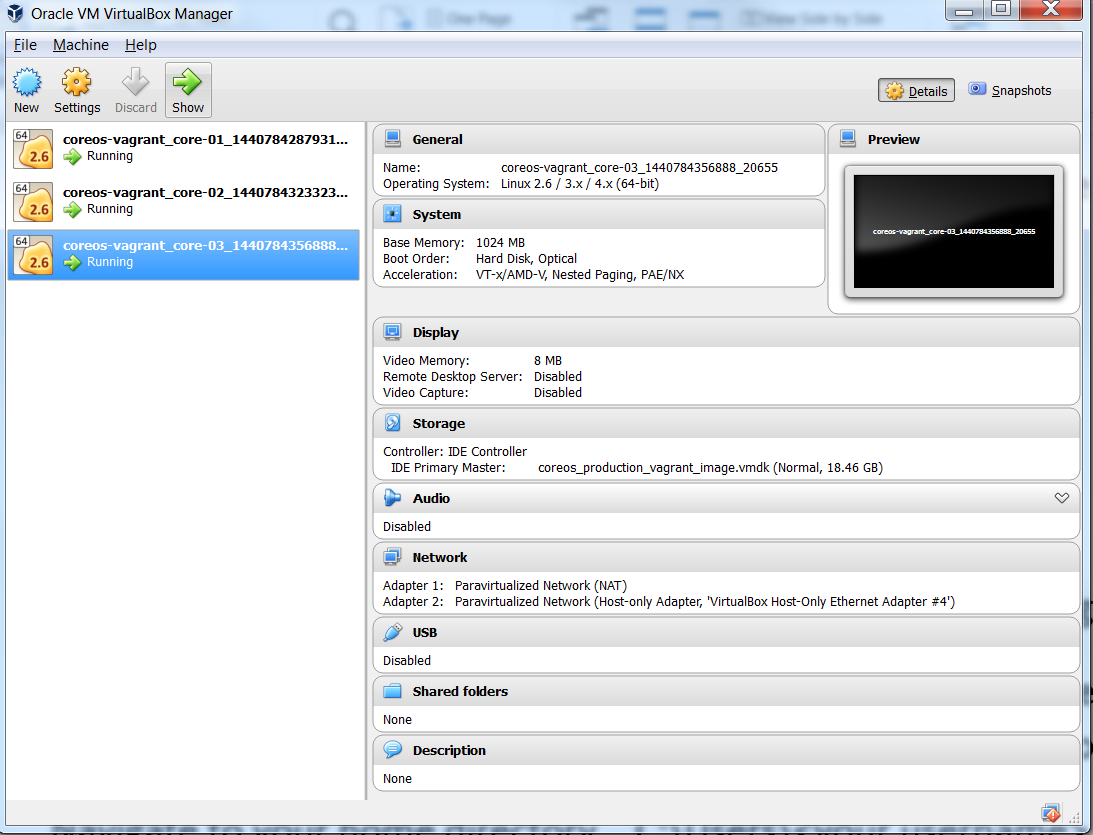
PATH.BAT

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

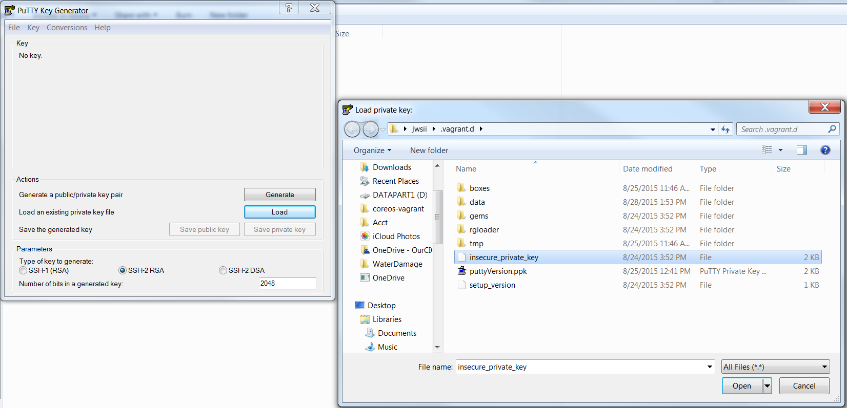
Navigate to your home directory. C:\Users\<username>

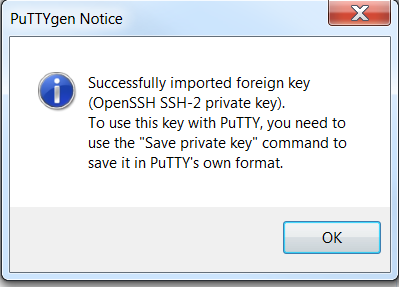
From the cmd prompt you can perform a “cd %HOMEPATH%” to change to your home directory.

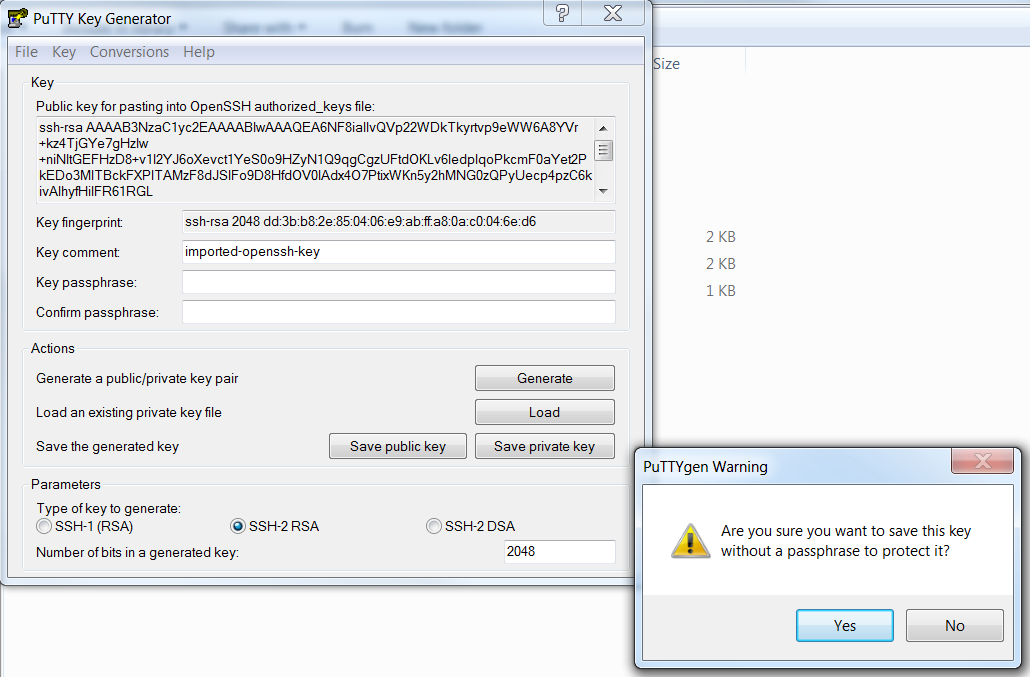
Under this directory find a folder “.vagrant.d” that will contain the insecure\_private\_key needed.

#### Configure PuTTY to Use SSH Key

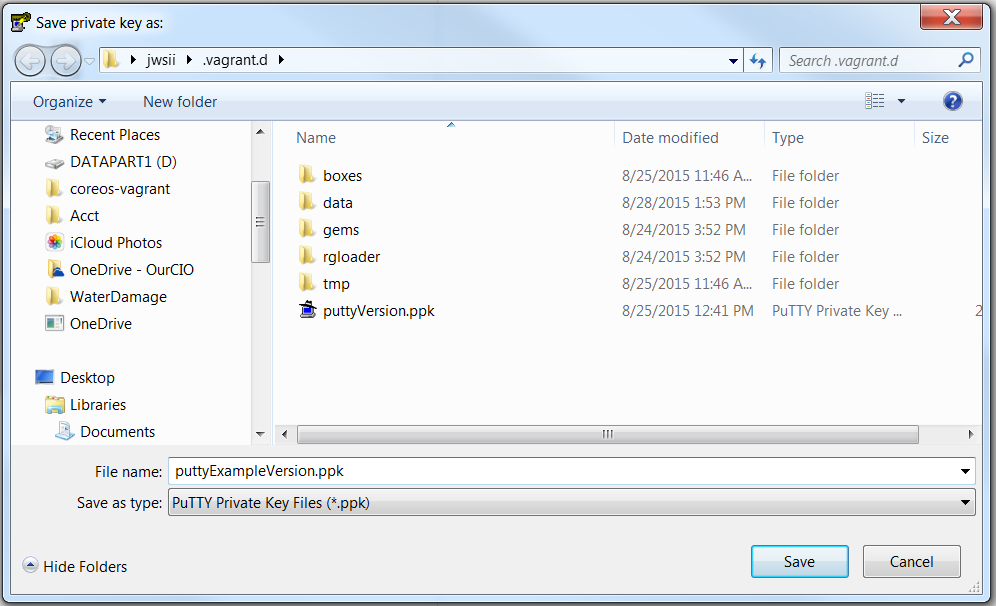
Using PuTTYgen Load an existing private key file.



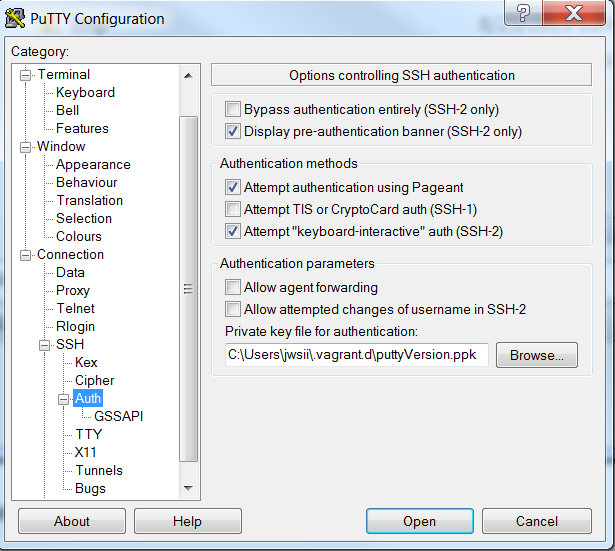
Note: Change the file type to All Files the insecure\_private\_key file can be found.

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

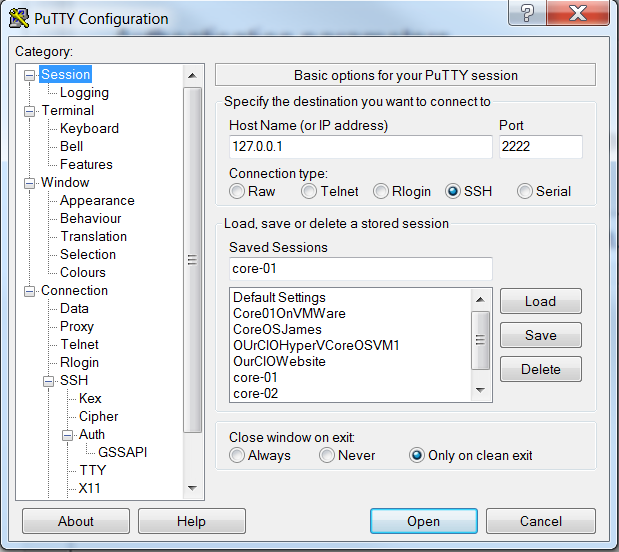
You are prompted with a warning about not having a passphrase. Click Yes



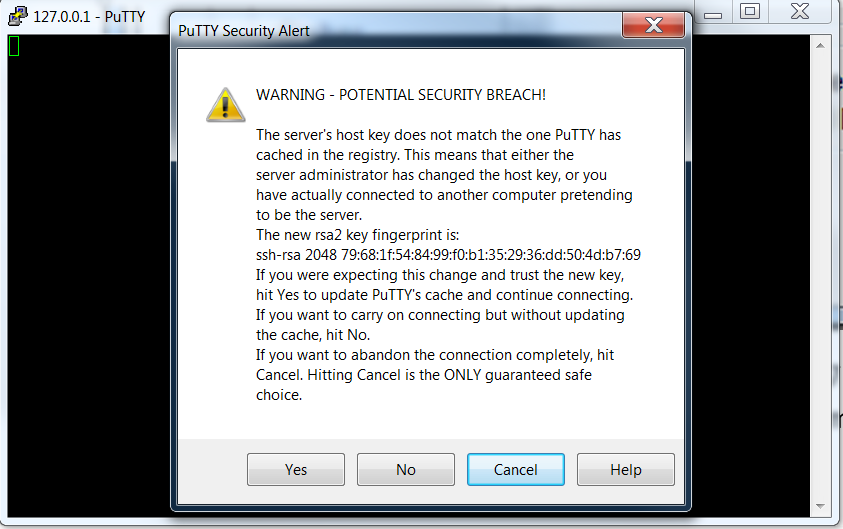
Suggest saving to the .vagrant.d folder with a “putty” prefix. Showing a “puttyExample” prefix for the documentation. Save the PuTTY private key file.

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”.

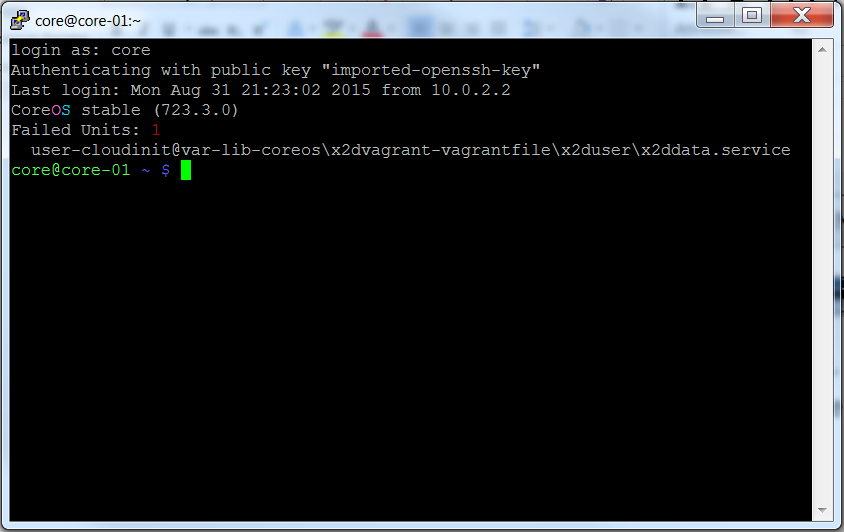
Shown above are the session connection basic options. 127.0.0.1:2222 is the IP and port that core-01 was generated. We strongly suggest that you save this information as core-01 so you can use it later.

#### 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.

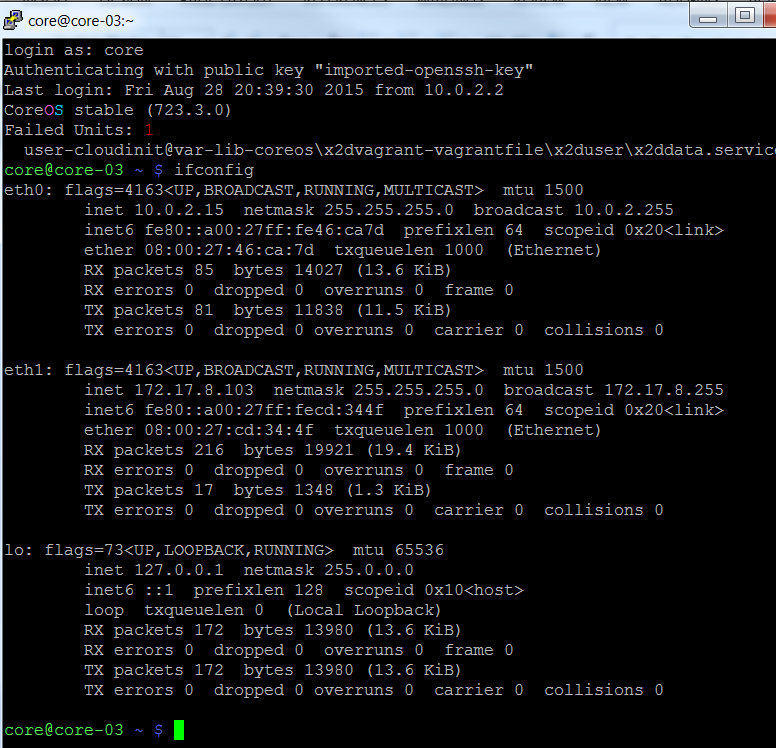
Log into each server to verify you have things configured properly.

#### 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. Could have used user-data.jschott2.endstate and config.rb.jschott2.endstate.

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

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 and saved the change; if not using config.rb.jschott2.endstate as the config.rb file.

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"

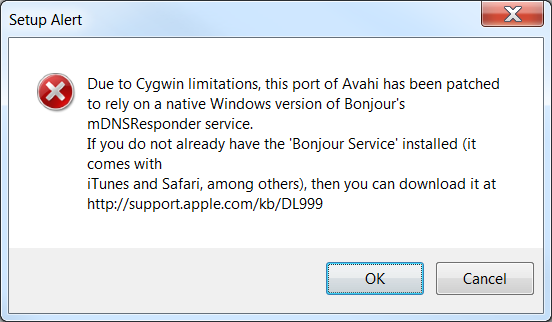
#Yea! rsync works but requires install of a package. I used cygwin.

### 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. Did not test with a 32 bit processor.)

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 copy Vagrantfile.jschott2 to Vagrantfile which contains the changes.)

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

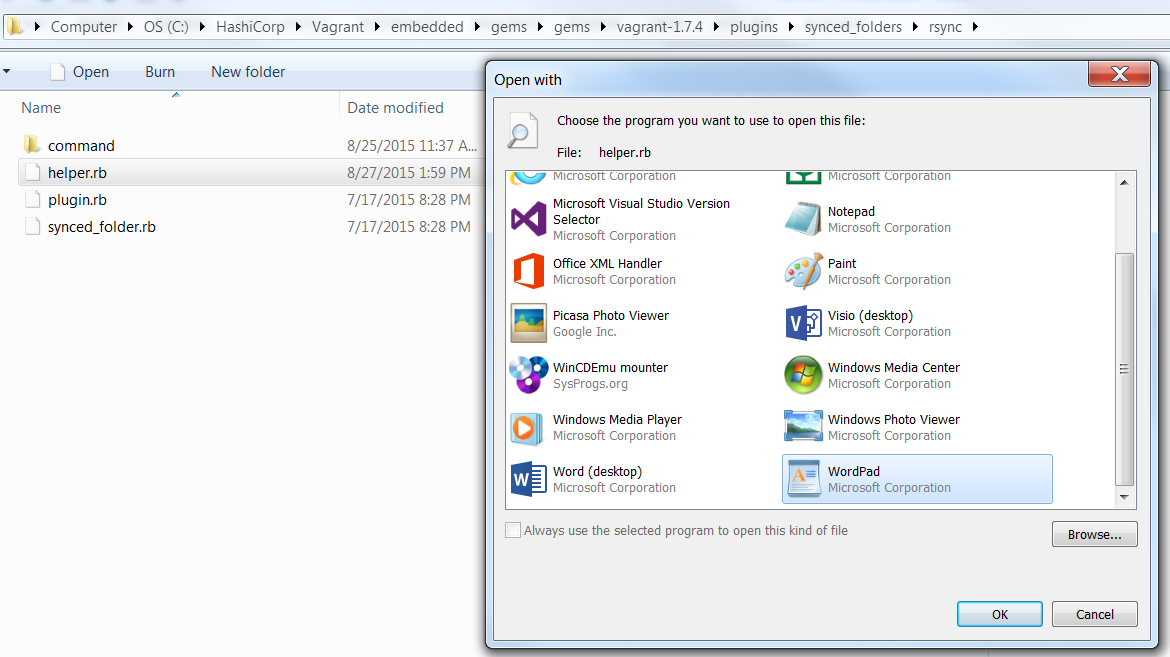
Save your Vagrantfile.

#### 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 of working with Windows but using Notepad is even worse.

Add the following code to the helper.rb file or copy either the helper.rb.jschott2.endstate or helper.rb.jschott2.endstate.clean to this directory and rename it helper.rb to get the same results.

if Vagrant::Util::Platform.windows?

# rsync for Windows expects cygwin style paths, always.

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

hostpath = "/cygdrive" + hostpath

end

Save your changes!

### 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 on and off. 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.

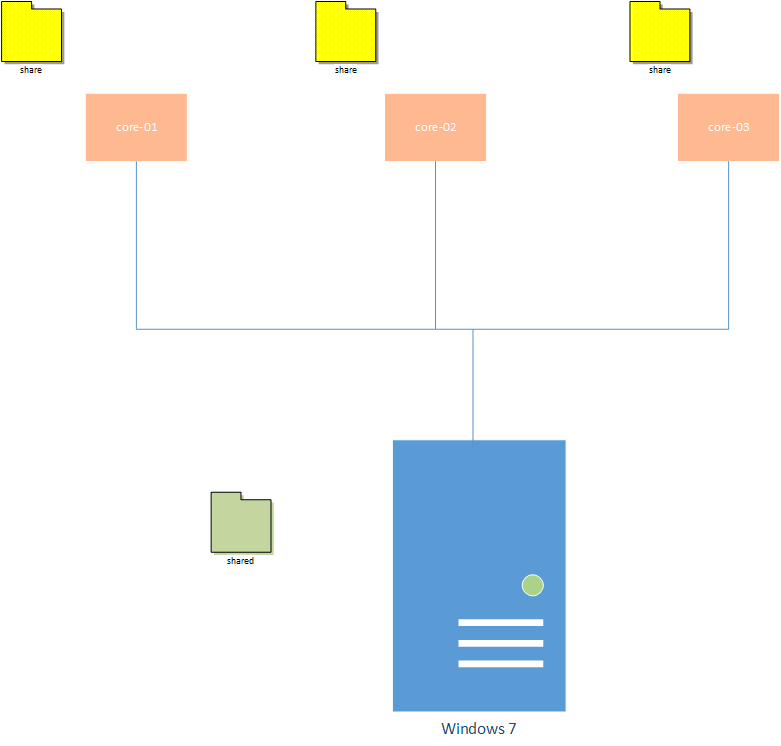
Getting files from the host to the guest is very helpful in configuring the CoreOS / Docker environments. For well-formed CoreOS / Docker configurations this may be all you need. The immediate need to get files back from the Guest machines becomes evident when you trouble shoot a problem on the Guest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

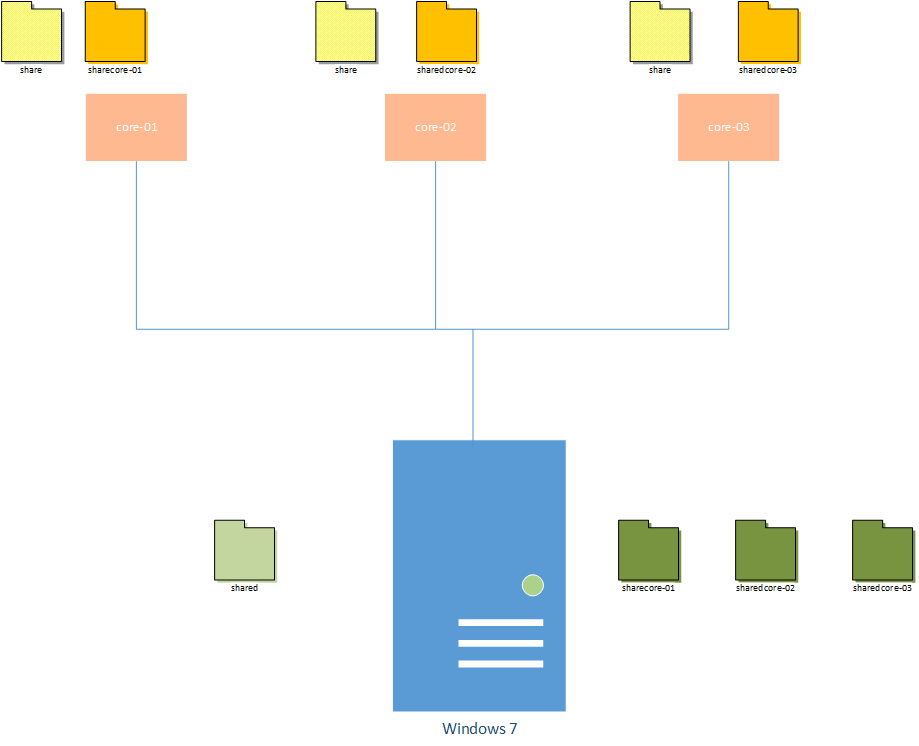
Copy and paste may work for some cases but creating shell scripts are not that forgiving in our Windows 7 coreos-vagrant environment. The nasty /r placed at the end of Windows files cause the shell scripts to fail. Also we will not have execute attributes on the files which will have to be fixed. So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

There are some utilities like dos2unix that will convert your files. You still need to fix execute attribute!

### 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?

Remember when you issue a “vagrant <up | reload> (core-0<1 | 2 | 3>)” command, EVERY FOLDER ON HOST is rsynced to the corresponding folder on the Guest. This is very important in your expectations of the consequences of the commands. These commands refresh the VM environment with what is currently configured in your coreos-vagrant folder. Contents of shared folders, user-data, config.rb and Vagrantfile all contribute to this refresh.

#### 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.jschott2.endstate or Vagrantfile.jschott2.enstate.clean 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 (Push)

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\sharecore-0x folders and push them up with “vagrant reload core-0x” command.

We now have a repeatable method of getting the contents of “coreos-vagrant\shared” and “coreos-vagrant\sharecore-0x” to the “core-0x” Guest in folders “/home/core/share” and “/home/core/core-0x” respectively.

#### The rsyncvmshare.bat

In the repository you will find a bat file “rsyncvmshare.bat” that contains commands for rsync to move files from folders on the Guest to corresponding folders on the Host.  ***Place the “rsyncvmshare.bat” file in the coreos-vagrant directory.*** We will deconstruct the bat file from the inside out (not top down) starting with the command to invoke rsync.

NOTE: ***You will not be able to issue these commands from the command line until later!***

rsync -a %obj\_vm\_username%@%obj\_vm\_name%:/home/core/%obj\_vm\_name%/ share%obj\_vm\_name%

We will substitute rsyncuser for %obj\_vm\_username% and core-01 for %obj\_vm\_name% which gives us the command:

rsync –a rsyncuser@core-01:/home/core/core-01/ sharecore-01

This then tells rsync that we want to “-a” recursively copy files archiving the attributes from “core-01” vm folder “/home/core/core-01/” to the folder here on the Host named “sharecore-01”. The twist is this “rsyncuser” that we used. There is no “rsyncuser” on the “core-01” machine. The only user is “core” but if we use that user we are prompted for a password which we don’t have!

A quick note before we solve the problem of users on the Guest with a password we know. The syntax of the rsyncvmshare.bat file invocation is shown below.

rsyncvmshare <vmname> <username> (<BeforeAfterSwitch>)

Using this bat file we can set up commands to quickly copy the contents of the Guest shared folders back to the host. The optional “BeforeAfterSwitch” will allow you to display the contents in the Host folder before and after the command is performed. This allows you to see the changes to the directory. To pass in the values in the command example above with the option to display the directory contents before and after we would use the following command line.

rsyncvmshare core-01 rsyncuser y

To just run the command with no display of the directory contents you would use the following.

rsyncvmshare core-01 rsyncuser

To remember how to use the command you can get help by providing a “/h” | “/?” | “-h” | “—help” switch as the first parameter to the batch file.

rsyncvmshare -h

Given is the UNIX version of requesting help which you will need to familiarize yourself. The “/h” and “/?” switches are for those old school people who automatically try the DOS help switches before reading anything. This gives you information on how to use the batch file. It is strongly encouraged to read the file contents as a working example for creating some of your own scripts.

#### Solving the Guest Username Challenge

We need to create users on the host for use. There are two scenarios where we could use a separate username. First we have the need for a username and password using rsync. Secondly it would be interesting to log into the Guest machines as a separate user to explore. We will demonstrate a method to accomplish both of these desires.

In the repository you will find a UNIX shell script “user\_up.sh”. Copy this to the “shared” folder on the Host. This shell script will be the basis for creating Guest users. With the shell script in the “shared” folder we can reload the Guest and go through an example of setting up a user.

vagrant reload core-01

Log into the core-01 machine as core. From the command line issue the following command.

./share/user\_up.sh rsyncuser

This will create a user “rsyncuser” with a password “Password1!” and for those of you that want to push on you can be mostly successful.

The “user\_up.sh” shell script syntax

./share/user\_up.sh (-h | --help)

Provides help on the shell script usage which follows below.

Syntax for invoking a user setup.

user\_up.sh <username> (<password>)

By just passing a username as we did in the example above we get a user set up with the default password of “Password1!”. That is not always what we desire and the optional password parameter allows us to invoke the shell script with both a username and password.

./share/user\_up.sh foo bar

This command creates a user “foo” with a password “bar”.

Not only does the shell script create the username with a password but it also associates the new username with the SSH key(s) of the “core” user. This will allow you to log into the Guest with this username using the SSH Client since the insecure\_private\_key is already configured. Try that out.

It is strongly encouraged that you read through the user\_up.sh shell script as a working example for creating some of your own shell scripts. You will also want to check out the chmod command with the +x option. chmod +x <myshell.sh> would add execute attributes to the file <myshell.sh> given.

#### Test the Guest Shared Folder (Pull)

It is expected that you have the “rsyncuser” and “foo” user set up as described above. The “user\_up.sh” script exists in the “shared” folder and you have reloaded core-01.

On the host at your coreos-vagrant command shell type or copy and paste the following command.

rsyncvmshare core-01 rsyncuser y

Now you have a password (Password1!) for the rsyncuser and you can complete the command when prompted for that password. The “y” switch will give you the before and after versions of the sharecore-01 directory. You may not find this very interesting if the core-01 directory on the Guest is empty.

Log into the core-01 Guest as “foo”.

cd core-01

touch afile

At the coreos-vagrant command shell type

rsyncvmshare core-01 rsyncuser y

You should see “afile” in the second listing of the sharecore-01 directory.

### Review 3:

Got a picture of the Host and Guest folder structure

Set up the folders on the Host and Guest to support this structure

Modified the Vagrantfile to configure the folder shares

Tested the push to the new Guest shared folders using vagrant reload

Learned about rsyncvmshare.bat file that executes commands to rsync from Guest share to Host share

Solved our issue of needing a username and password for rsyncvmshare.bat using user\_up.sh shell script

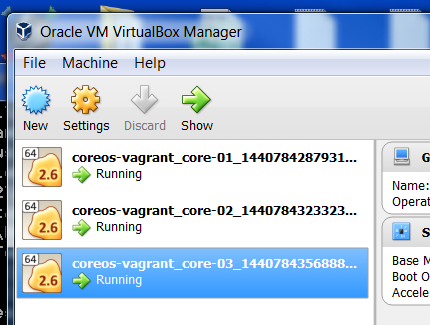
Tested the pull of files from the Guest share to Host share using rsyncvmshare.bat and user we set up

Optionally we read through the rsyncvmshare.bat file and user\_up.sh file to find out more of how they work. Unknowingly we started to learn UNIX commands (cd, touch, chmod).

We also picked up that you need to invoke a UNIX shell script by prefixing a “./” to the full name of the file.

This may not be out of reach right? Again you can play but it is strongly suggested you finish the next section to “Recreate the Cluster”.

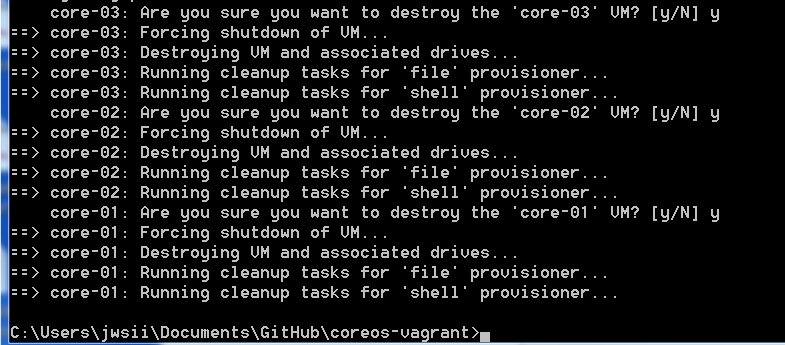
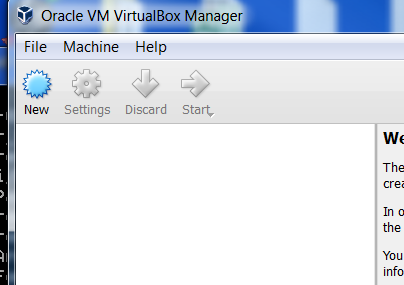
## Recreate the Cluster

Now that we have our Vagrant, config.rb, user-data and help.rb modified to work under Windows 7, make sure we can create a fresh version of the environment and test that fresh version.

Log out of all your open SSH sessions on the Guests. This will not prevent things from happening but is just good practice. At the coreos-vagrant command shell issue the follow command.

vagrant destroy

This will prompt you to confirm you want to destroy each Guest machine. In this case yes we do.



Check the VirtualBox Manager and you will see that all of the Guest machines are now gone.

Check to make sure the “rsyncvmshare.bat” exists in the coreos-vagrant directory along with your “path.bat” file. The “shared” folder should have at least a copy of user\_up.sh shell script.

Remember we have to modify the user-data with a new cluster token from <https://discovery.etcd.io/new?size=3> so the new cluster will be registered. Do that now.

Should be ready to launch the new clusters right? No we have one new item to resolve.

### Removing the Old Host SSH Signatures

Before you can log into the new Guest machines we need to clean up the SSH signatures that were registered on our Windows 7 machine. After a “vagrant destroy” for a Guest machine you need to remove the signature from a file used by Cygwin in the .SSH directory. Assuming you are doing this for all of your vagrant hosts the commands below will work.

cd c:\cygwin64\home\<username>\.ssh

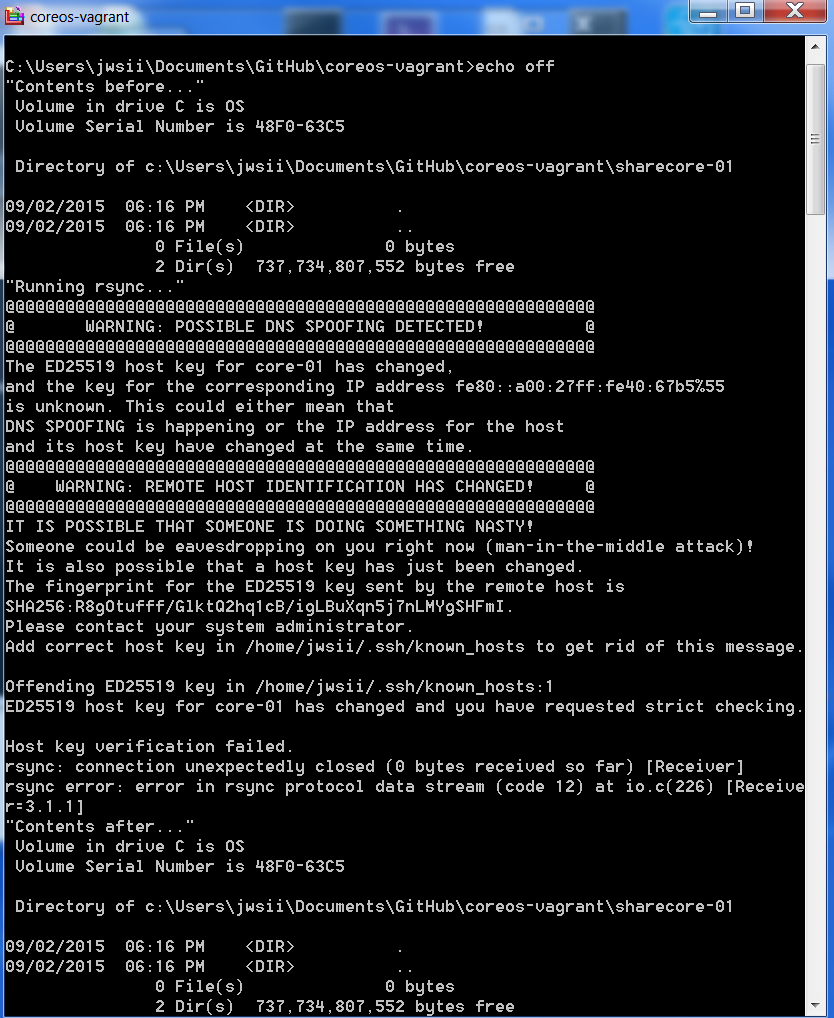
del known\_hosts

If you need to get rid of selected hosts edit the file.

cd c:\cygwin64\home\<username>\.ssh

del known\_hosts

Good luck with this editing of the file since it is not yet tested.

If you should forget to do this you will get the following error(s):

Warning POSSIBLE DNS SPOOFING DETECTED!

Warning REMOTE HOST IDENTIFICATION HAS CHANGED

### Bring up a Fresh Cluster

We have destroyed the Guest machines, retrieved a new cluster token and updated the user-data file, removed the SSH signatures from the known\_hosts file and now we are finally ready to bring up the fresh cluster. Easy as…

vagrant up

You can watch the progress of the cluster forming in the VirtualBox Manager. The Guest machines run through a cycle of appearing Powered off, to Starting to Running.

Now log into the Guest machines and create a user so you can rsyncvmshare the Guest Shares. Below is a suggested exercise that will prove the value of the work you have done so far. As always you are free to jump right in and explore.

### Exercise for the Fresh Cluster

In this exercise we will log into each of the three (3) machines in our cluster, create a user “rsyncuser” with a password of “doit” and create some interesting files in the Guest share folder. Once we have finished with each Guest machine we will pull the files from the Guest machines using rsyncvmshare.bat, do some modifications on each of the sharecore-0x folders and then perform some rsync updates to the machines to see what transpires.

#### Setting up the Guest Machines for Our Exercise

Log into core-01 using the SSH client and the “core” username. Set up our user for use.

./share/user\_up.sh rsyncuser doit

This will create the required rsyncuser on core-01. Now create an interesting file in the core-01 directory. We do this using the following two commands

cd core-01

touch core01stuff1

ls

This will give us a file “core01stuff1” in that directory. Wondering where “afile” came from? Remember we created it in an exercise above. If you do not have it you may be jumping around. That’s OK. Move on.

Log into core-02 using the SSH client and the “core” username. Set up our user for use.

./share/user\_up.sh rsyncuser doit

This will create the required rsyncuser on core-02. Now create interesting files in the core-02 directory. We do this using the following two commands

cd core-02

touch core02stuff{1..2}

ls

This will give us two files core02stuff1 and core02stuff2. This may be handy for you if you need to create many test files in a directory. Move on.

Log into core-03 using the SSH client and the “core” username. Set up our user for use.

./share/user\_up.sh rsyncuser doit

This will create the required rsyncuser on core-03. Now create interesting files in the core-03 directory. We do this using the following two commands

cd core-03

touch core03stuff1{1..3}

ls

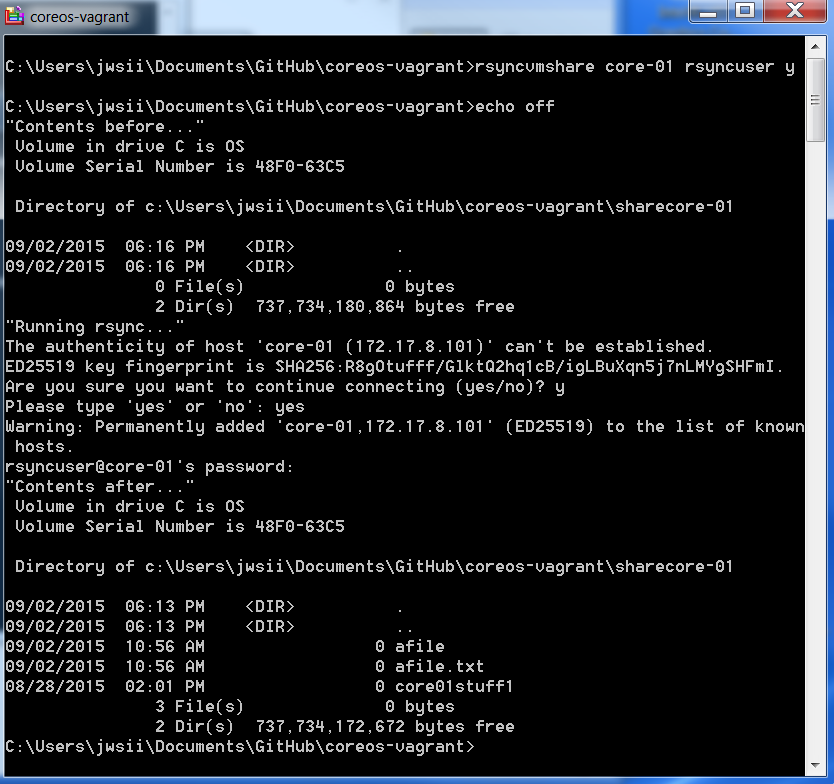
This will give us three files core03stuff1, core03stuff2 and core03stuff3.

We now have a username “rsyncuser” on each Guest machine with the same password. Each Guest share folder has a unique set of files. We are ready to go to the Host and move some files around.

### Exercise for Fresh Cluster Shared Folders

At the coreos-vagrant command shell on the host type the following command

rsyncvmshare core-01 rsyncuser y

If you get an error check “Removing the Old Host SSH Signatures” section above.

Notice that we have to respond yes to the query to continue connecting. It is here that the SSH signatures are registered in the known\_hosts file. Above we cleaned out the sharecore-01 directory before executing the rsync command. The files from the host are moved.

Now do the same thing for core-02 and core-03.

rsyncvmshare core-02 rsyncuser y

rsyncvmshare core-03 rsyncuser y

Now we should have the files from the Guest machines share folders in our corresponding sharecore-0x folders. Use rsync to move files from the Host to the Guests.

vagrant rsync

This command spins through each of our Guests and moves files from the Host shared folders to the Guest. To do a single Guest call it out in the command.

vagrant rsync core-01

Now delete some files on the Host. Cut and paste the four (4) commands below in coreos-vagrant command shell.

del sharecore-02\core02stuff2

del sharecore-03\core03stuff2

dir sharecore-02

dir sharecore-03

What is left is sharecore-02\core02stuff1, sharecore-03\core03stuff1 and sharecore-03\core03stuff3. Now perform the following commands in coreos-vagrant command shell.

rsyncvmshare core-02 rsyncuser y

rsyncvmshare core-03 rsyncuser y

We notice that the files sharecore-02\core02stuff2 and sharecore-03\core03stuff2 are now back! They were still on the Guest machine share folders.

We need to delete files in both the Guest and the Host if we want to get rid of them. There are cases where a delete on the Host will be final. Say if it is done prior to a “vagrant destroy” and we have not successfully used rsyncvmshare on that Guest.

As they used to say, “That’s all there is there ain’t no more!”. Yet we will update as your cards and letters keep coming and we gain more insight into this environment.

### Review 4:

We ensured that all our configuration files were updated and in place in coreos-vagrant and shared

Destroyed our Guest machines with vagrant destroy

Removed the Old Host SSH Signatures deleting known\_hosts

Brought up a new Cluster with vagrant up

Optionally ran the Exercise for the Fresh Cluster

Created a username on each Guest and some interesting files

Used the rsyncvmshare.bat to move files from the Guest shares to the Host sharecore-0x folders

Used vagrant rsync to move the files from all the Host shared folders to all the Guest shared folders

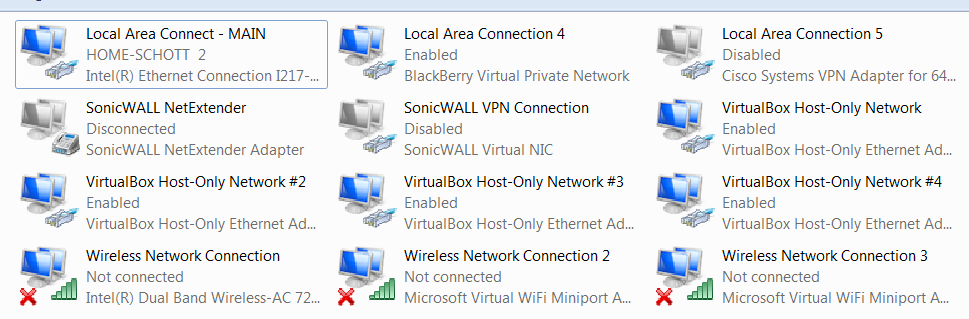
Tried vagrant rsync core-01 to move the Host shared folders associated with core-01 to that Guest

Deleted some files on the host for the core-02 and core-03 machines

Saw that they reappeared when we used rsyncvmshare for those machines

Have everything we need configured to Search and Destroy using CoreOS cluster on our Windows 7 machine!

## Known Issues

One incident where the Windows “lost” the adapter configuration. It is supposed but not proven that my coreos-vagrant environment hijacked the adapter. I did notice that there are several adapters now configured and dedicated to the VirtualBox environment.

A mere reboot of the laptop didn’t fix it at first. But quitting for the night and booting it up off the docking station had me wondering if it “…was ever there at all”. The music never stopped.

The systems cleared up and have not appeared in the following days.

# Research Links Used

Attempt to document the work of others used in creating this tutorial. 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>

<https://www.digitalocean.com/community/tutorials/how-to-use-rsync-to-sync-local-and-remote-directories-on-a-vps>