Disabling Multi-Touch Swipe in Chrome on OSX to Prevent Back and Forward Actions

This is a nifty little tidbit that could have saved me many a lost page or form fill in Chrome. If you have the Swipe configured for OSX (as I do), it also works as the back and forward button in your Chrome browser. Swipe to go back sounds like a great idea, right?

Right. This is handy sometimes, but not when you spend 15 minutes filling in a web form and then accidentally swipe left with a second finger on the trackpad and chrome goes back a page losing all your work.

(Spoiler alert: This is a very, very, very anecdotal situation I just experienced)

How to Disable Chrome Swipe to Go Back

Just go out to a Terminal window and type the following:

```
defaults write com.google.Chrome AppleEnableSwipeNavigateWithScrolls -bool FALSE
```

You will have to restart your browser in order to complete the change. That will stop the swipe from invoking the “back” or “forward” action within Google Chrome for the two-finger swipe gesture.

How to Enable Chrome to Swipe Back

As you can imagine, the reverse of the command by changing FALSE to TRUE will bring the swipe actions back for you.

```
defaults write com.google.Chrome AppleEnableSwipeNavigateWithScrolls -bool TRUE
```

Hope that helps you as much as it did for me!

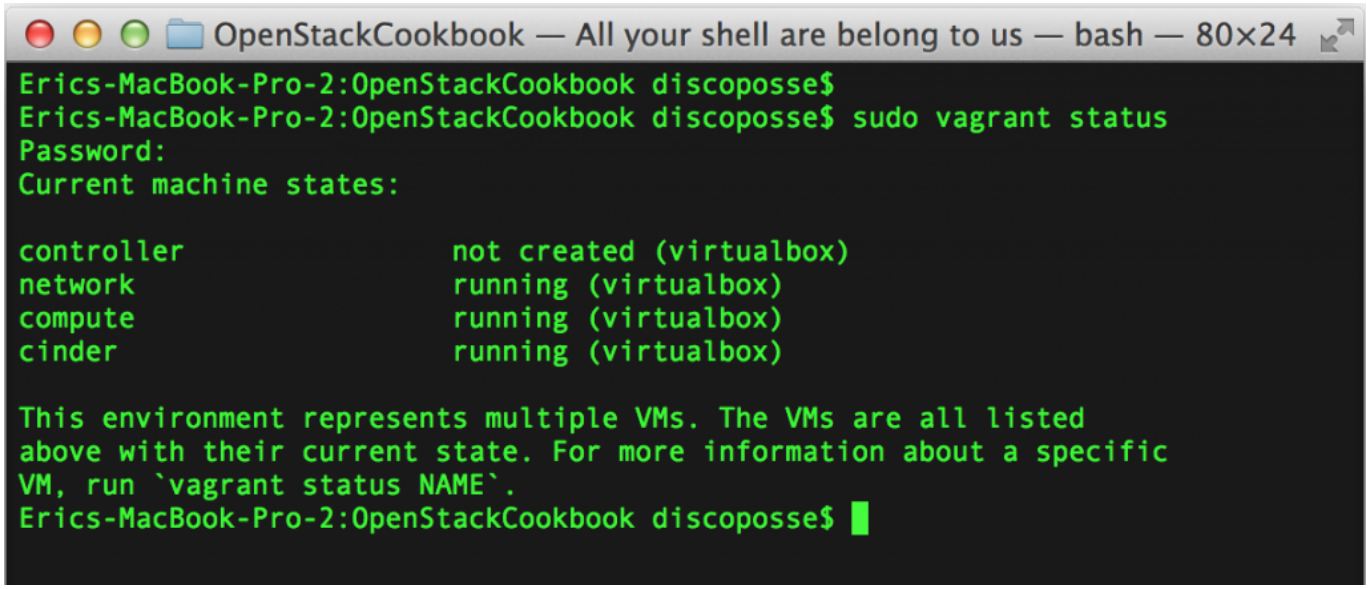
Suspend and Resume Vagrant Boxes in Bulk in OSX and Linux

As a regular user of Vagrant in conjunction with VirtualBox for building and deploying lab environments, I often have situations where I have multiple virtual instances running at a time. The challenge with that is that sometimes my multi-node labs are having to be suspended and resumed to save on resources. It’s not difficult, but it does take multiple commands to complete the task.

There Has to be a Better Way!

Just like the infomercials always say: “There has to be a better way!”, and in this case there absolutely is. Let’s use an example where I have my OpenStack 4-node lab running in Virtual box. Because I’m security conscious, I also use sudo for all of my processes (and you should too), so let’s just have a look at the process I go through to view, suspend and resume my 4-node lab with relative ease.

First, we want to list our vagrant boxes on the system: `sudo vagrant status`



We can see that we have a list of all of the nodes which shows them by name and with the status showing as **running**. In fact, I even have one machine that I haven’t deployed that is shown as **not created**, which means that we will have even more confusion when trying to work with bulk commands potentially.

In order to work with our running machines only, we can simply pipe the output to a grep command to filter the results:

```
vagrant status | grep running
```

Now we have a fairly good bit of data to work with, but the problem is that our `vagrant suspend` command needs to use the instance name to control the VM. This means that we will need to pull one of the columns of data to pipe it to vagrant to issue the suspend command. So, let’s do that!

Getting AWK-ward: Finding Text in the Stream

Because our last command produced a table of data which is able to be read and manipulated, we will use the `awk` command to print the first part of the data stream which is coming in.

```
sudo vagrant status | grep running | awk '{print $1}'
```

With this new set of data, we have exactly what we need to pass the output into a `vagrant suspend` command in order to suspend our virtual machines. In order to do this, we just add

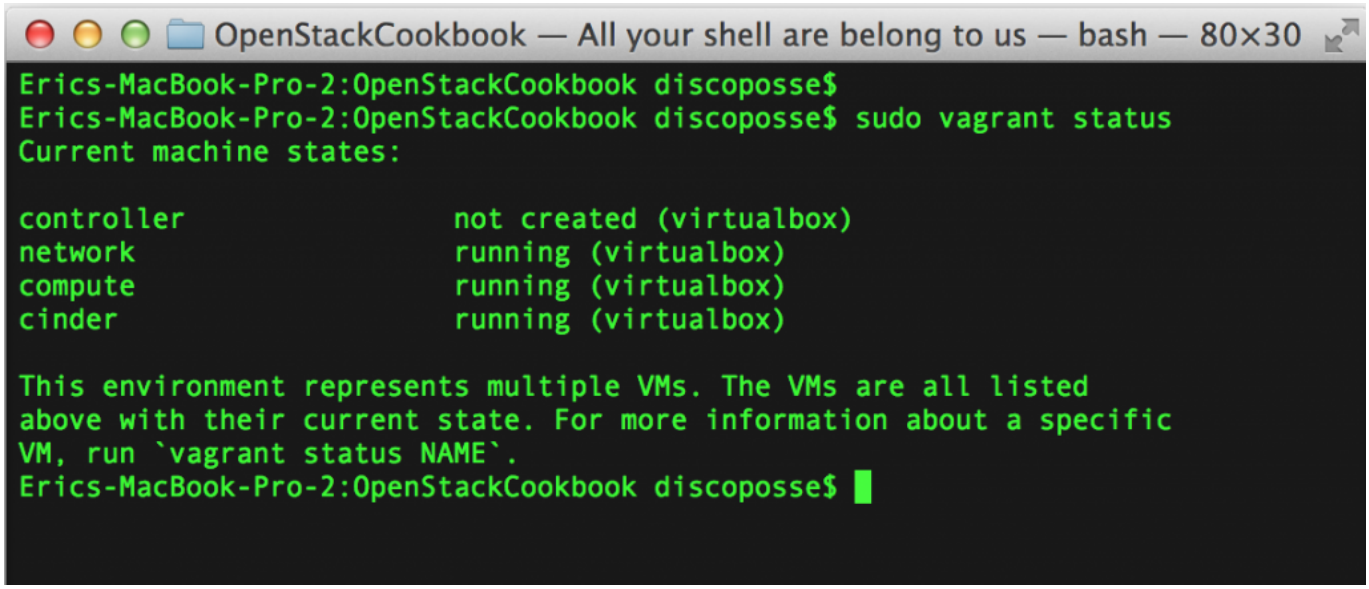
another pipe command into the one-liner:

```
sudo vagrant status | grep running | awk '{print $1}' | sudo vagrant suspend
```

Now to resume our machines, we simple revers the process. By checking the status and finding machines which are in the saved state, we can issue a resume command using this relatively simple one-liner:

```
sudo vagrant status | grep saved | awk '{print $1}' | sudo vagrant resume
```

Now you can run the vagrant status command and see that your virtual machines are up and running again:



This is a handy tip, especially when you're on battery and need to quiesce your lab machines to save processing power. Even when they seem idle, they are using additional resources. Beyond that, it's a great way to start working with the awk command and learn to start parsing data with it. As you will see with my OpenStack install scripts, awk is a powerful tool for dynamically parsing and piping data.

Hope you find this helpful!

Making a Sublime Command Link for OSX

I've just switched back to using a Mac platform and started using Sublime as my text editor of choice. In the past I've used TextMate, but Sublime has met all of my needs easily including single file edits, project folder edits, context display for languages, and good quick response.

One thing I love to do is to work in the shell (Terminal) and work among the folder structure for

things line Vagrant projects. In order to launch my text editor from the command line, there is a simple trick to creating a shortcut to Sublime which is done like so:

```
sudo ln -s "/Applications/Sublime Text.app/Contents/SharedSupport/bin/subl" /usr/local/bin/sublime
```

You'll be prompted for your password unless you've already type it into your shell, but the result should be successful as shown above with no errors displayed. Next you can browse to a folder where you have a file you want to edit and you can use the simple command **`sublime FILENAME`** to launch the editor with your file already loaded:

This will launch a Sublime window loaded with your **Vagrantfile** in this case:

Loading a folder as a project is also just as easy. In this case, we have a folder with 3 files in it for my sample:

By launching Sublime with the command line **`sublime .`** you will load the folder which displays all of the files in the editor for simple multi-file edits:

It's just that easy! Happy editing!

Deploying the VMware vCenter Virtual Appliance on OSX

For those who want to leverage VMware vCenter for your lab, but you are running on Mac OSX with VMware Fusion, there are a couple of quick steps to get you working. The secondary title to this article could really be "How to deploy an OVF template on VMware Fusion", but the specific goal I'm after is my vCenter appliance today.

I am assuming that you are already comfortable with VMware Fusion, and with administering your lab environment so this is really more of a quick walkthrough just to deploy the virtual appliance using command line tools.

The virtual appliance is available as an OVF file. I've used it for my production deployment, but for this case I want to use the same OVF package to deploy into my nifty little lab that I pack inside my

Macbook. Here is the challenge: there is no native way to import or deploy the OVF template using VMware Fusion.

Luckily we have the OVFTool available which lets us use the same files we have by converting the OVF file into a VMX to be used by VMware Fusion. The OVFTool details and documentation are found here: http://www.vmware.com/technical-resources/virtualization-topics/virtual-appliances/ovf

First, you need to download the VMware vCenter virtual appliance files from the VMware download site. There are 3 files which will include the System disk, the Data disk and the OVF file which is what you would normally use to deploy into your ESX host using the Virtual Infrastructure Client.

The vCenter Server Appliance download can be found here: https://my.vmware.com/group/vmware/details?downloadGroup=VC50&productId=229

Secondly, you need to download and install the OVFtool package for OSX. The file for OSX is a tar.gz file which contains a PKG file. You will find that tar file here: http://communities.vmware.com/community/vmtn/server/vsphere/automationtools/ovf

Once you unzip the file into a folder and run the PKG file to install the product it will place the contents in the **/Applications/VMware OVF Tool/** folder.

Open a Terminal window. Change directory to the location of the OVF tool (note that it will show as **cd /Applications/VMware OVF Tool/** because of the spaces in the path).

Next you run the ovftool using the the command line of ***ovftool source destination*** which in this case I've stored the source file under my Downloads folder in a vCenter subfolder.

The tool will work its magic for a while and when it is done it will have crafted a ready to use VMX file and VMDK file set which you can now open with your VMware Fusion application. Because we have converted to a new VM file set, the files are duplicated so in this case you have to be sure you have space for another 5.5 GB or so. You will be notified of the successful completion:

Simply go to your VMware Fusion app now and on the File menu select Open

Point to your location that you have stored your new VMX and VMDK files. For my example I have simply left them in place. You will most likely want to move them to a folder with your other virtual machines.

Now we see the machine in our Library view and you may now start up the server. It is just that easy!