Exercise 1 - Creating a Local Lab Environment

Goal:

The goal of this project is to create a local Linux lab environment. Once you have the environment configured you can quickly and easily create and destroy Linux virtual machines on your Windows, Mac, or Linux computer. You will be creating new Linux systems throughout this course using the tools you will install and configure in this exercise.

A Word About Downloads:

Throughout this course I'm going to ask you to download various pieces of software from http://mirror.linuxtrainingacademy.com. This way I can ensure we are all using the same versions and getting the same results. Since other websites are not under my control they can change their addresses and download options at any time without my knowledge. Using a website under my control prevents any problems with acquiring the required software for this course.

With that said, I've also included an "Internet download location" for each piece of software so you know where to download it in the future after you have completed this class. You can use those locations when you are deploying systems in your work or home environments. I cannot guarantee those public Internet links will always be live and accurate, however they should point you in the right direction.

Notes to Windows Users

Do Not Use or Enable Hyper-V for This Class (Windows Users Only)

In this class you are going to use vagrant to manage virtual machines running under VirtualBox. Because VirtualBox is incompatible with Hyper-V, you will need to make sure Hyper-V is disabled. This should only be an issue if you are running a Windows *Server* installation.

Antivirus Software Compatibility (Windows Users Only)

Some Windows Antivirus software, such as Avira, has been known to interfere with the operation of VirtualBox. If you get an error when starting a Virtual Machine, try disabling your antivirus software. Some errors that are an indication of an antivirus conflict include "Error in supR3HardenedWinReSpawm," "VERR_OPEN_FAILED," "E_FAIL (0x80004005)," and "NtCreateFile(\Device\VBoxDrvStub) failed."

Enable Virtualization Support for Your Processor (Windows and Linux Users)

Make sure you have VT-x (for Intel processors) or AMD-v (for AMD processors) enabled in your physical computer's BIOS. This varies from manufacturer to manufacturer, so you might have to look up the documentation for your specific computer. Many computers have these options enabled by default, but some do not. If virtualization support is not enabled you may encounter an error inside your virtual machines like this: "This kernel requires an x86-64 CPU, but only detected an i686 CPU"

Instructions:

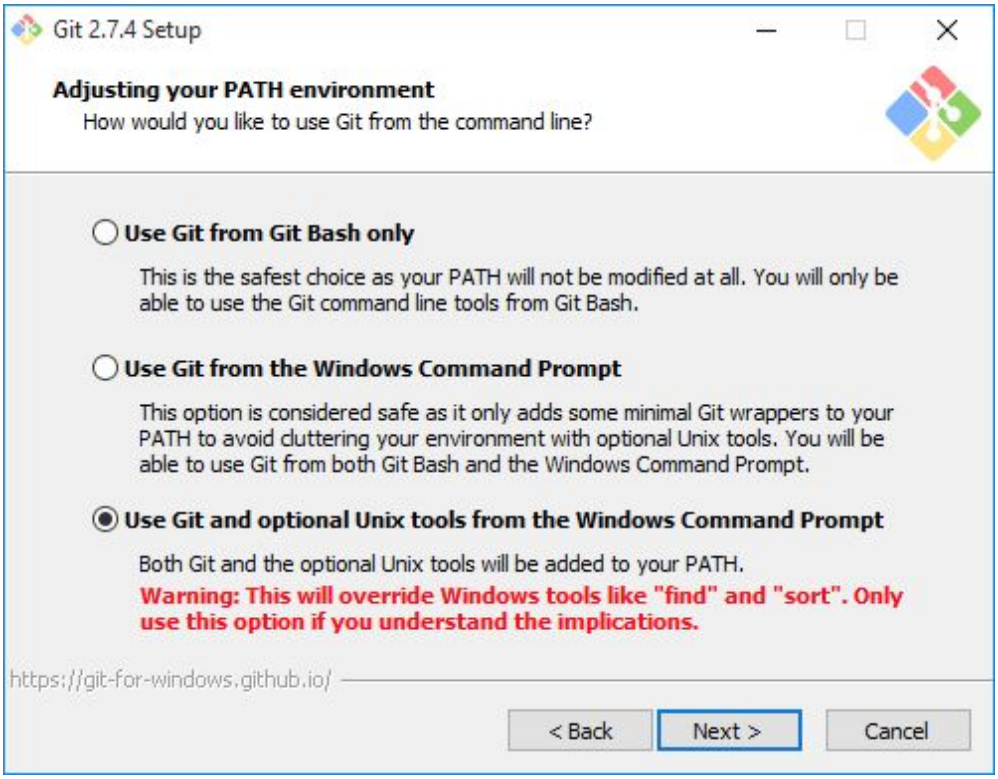
Install an SSH Client (Windows Users Only)

SSH, secure shell, is the network protocol used to connect to Linux systems. By default Windows doesn't include an SSH client. Mac and Linux do, so Mac and Linux users should skip this step.

In order to get an SSH client on Windows you can install Git. Git is used for version control, but we're interested in the SSH client that it ships with. Download git here:
http://mirror.linuxtrainingacademy.com/git/windows/

Internet download location: https://git-scm.com/download/win

Start the installer. Be sure to select "Use Git and optional Unix tools from The Windows Command Prompt" when presented with the option. Otherwise, use the defaults. If you're asked for an administrator user and password, be sure to enter it.



Install Vagrant Dependencies (Windows Users Only)

Vagrant requires the "Microsoft Visual C++ 2010 SP1 Redistributable Package (x86)" to be installed if you are running Windows. Download it from https://www.microsoft.com/en-us/download/details.aspx?id=8328. Start the installer and accept the defaults to complete the installation.

Install VirtualBox (All Users)

Download VirtualBox from http://mirror.linuxtrainingacademy.com/virtualbox/. If you are using Windows, download the file from the windows subdirectory. If you are using a Mac, download the file from the mac subdirectory. If you are using RedHat or Centos, download the file from the centos directory.

Internet download location: https://www.virtualbox.org/wiki/Downloads.

Install the software on your local machine, accepting all the defaults. If you're asked for an administrator user and password, be sure to enter it. Reboot your system if requested by the installer.

Install Vagrant (All Users)

Download Vagrant from http://mirror.linuxtrainingacademy.com/vagrant/. If you are using Windows, download the file from the windows subdirectory. If you are using a Mac, download the file from the mac subdirectory. If you are using RedHat or CentOS, download the file from the centos directory.

Internet download location: https://www.vagrantup.com/downloads.html

Install the software on your local machine, accepting all the defaults. If you're asked for an administrator user and password, be sure to enter it. Reboot your system if requested by the installer.

Create a Working Folder (All Users)

Create a folder to keep your course work in. First, start a command line session on your local machine.

For Windows users, start the Command Prompt. (Click the Start button. In the Search box, type "Command Prompt", and then, in the list of results, double-click Command Prompt.)

For Mac users, start the Terminal application which is located in the /Applications/Utilities folder.

For Linux users, start your favorite terminal emulator. Examples include GNOME Terminal, Konsole, and xterm.

```
mkdir shellclass
```

When the Command Prompt (Windows) or the Terminal (Mac/Linux) is launched you will be placed in your home directory. For example, if I'm logged into a Windows system as "jason" my home directory could be "C:\Users\jason". (Note: this might vary depending on the version of Windows you are using.) If I'm logged into a Mac system as "jason" my home directory will be "/Users/jason". If I'm logged into a Linux system as "jason" my home directory will be "/home/jason".

Change into the Working Folder

Now let's move into the folder we just created.

```
cd shellclass
```

Add a Box to Vagrant

A "box" in Vagrant speak is an operating system image. The "`vagrant box add`" command will download and store that box on your local system. You only need to download a box once as this image will be cloned when you create a new virtual machine with Vagrant using the box's name.

I created a box specifically for this class and uploaded it to the public Vagrant box catalog. Run the following command on your local machine to download it.

```
vagrant box add jasonc/centos7
```

The format of the command when downloading a public box is "vagrant box add USER/BOX". There are several public boxes available to download. You can search for boxes here, but be sure to use the "jasonc/centos7" box for this class. https://atlas.hashicorp.com/boxes/search

Create a Vagrant Project Folder

Vagrant uses the concept of projects. A Vagrant project must consist of a folder and a Vagrant configuration file, called a Vagrantfile. Start out by creating a "testbox01" folder.

```
mkdir testbox01
```

Create Your First Vagrant Project

To create the Vagrant configuration file (Vagrantfile), run the "`vagrant init <BOX_NAME>`" command. Be sure to be in Vagrant project directory you just created. Also, use the "jasonc/centos7" box you downloaded earlier.

```
cd testbox01
vagrant init jasonc/centos7
```

Create Your First Virtual Machine

The first time you run the "`vagrant up`" command Vagrant will import (clone) the vagrant box into VirtualBox and start it. If Vagrant detects that the virtual machine already exists in VirtualBox it will simply start it. By default, when the virtual machine is started, it is started in headless mode meaning there is no UI for the machine visible on your local host machine.

Let's bring up your first virtual machine running Linux with Vagrant.

```
vagrant up
```

Troubleshooting

On some systems you may see the following message, though it is rare.

```
Timed out while waiting for the machine to boot. This means that
Vagrant was unable to communicate with the guest machine within
the configured ("config.vm.boot_timeout" value) time period.

If you look above, you should be able to see the error(s) that
Vagrant had when attempting to connect to the machine. These errors
are usually good hints as to what may be wrong.

If you're using a custom box, make sure that networking is properly
working and you're able to connect to the machine. It is a common
problem that networking isn't setup properly in these boxes.
Verify that authentication configurations are also setup properly,
as well.

If the box appears to be booting properly, you may want to increase
the timeout ("config.vm.boot_timeout") value.
```

If you do see that message it most likely means the virtual machine started with its networking interface disabled. To force the network interface to be enabled we'll need to update the Vagrantfile. The Vagrantfile controls the behavior and settings of the virtual machine. Open it with your favorite text editor. You may need to use the File Explorer (Windows) or the Finder (Mac) to navigate to the folder and then open it with your desired editor.

By the way, Atom is a nice text editor that works on Mac, Windows, and even Linux. You can download it for free at Atom.io.

Add the following line somewhere after `"Vagrant.configure(2) do |config|"` and before `"end"`. A good place could be right after the `'config.vm.box = "jasonc/centos7"'` line:

```
config.vm.provider "virtualbox" do |vb|
  vb.customize ['modifyvm', :id, '--cableconnected1', 'on']
end
```

Reboot the virtual machine with Vagrant.

```
vagrant reload
```

Confirm That It's Running

Start the VirtualBox application. On Windows double click on the "Oracle VM VirtualBox" icon on your desktop. For Mac users, start the /Applications/VirtualBox application.

Confirm that you see a virtual machine running. It will start with the name of your Vagrant project folder.

You can also use the "vagrant status" command to check the status of the virtual machine. Confirm that it shows the virtual machine is in a running state.

```
vagrant status
```

Connect to the Virtual Machine

SSH, secure shell, is the network protocol used to connect to Linux systems. Vagrant provides a nice shortcut to ssh into the virtual machine.

```
vagrant ssh
```

Now you are connected to the Linux virtual machine as the vagrant user. This default vagrant account is used to connect to the Linux system. For your convenience, the Vagrant application takes care of the details that allow you to connect to the box over SSH without a password. For reference, the password for the vagrant account is "vagrant". The password for the root account is also "vagrant". The vagrant user has full sudo (administrative) privileges that allow you to further configure the system. You will learn more about accounts, privileges, and sudo later in this course.

After running "`vagrant ssh`" you should be presented with a prompt that looks similar to this:

You'll be working a lot at the Linux command line in this course. For now, let's log out of the Linux system by running the "`exit`" command.

```
exit
```

Stop the Virtual Machine

The "vagrant halt" command shuts down the virtual machine. When you run this command you will not lose any work you've performed on the virtual machine. The virtual machine will still exist in VirtualBox, it will simply be stopped.

```
vagrant halt
```

Open the VirtualBox application and verify that the virtual machine is still define, yet stopped.

Start the Virtual Machine Again

Remember, to start the virtual machine run "vagrant up". This time when you run the command it will not need to import the box image into VirtualBox as the virtual machine already exists.

Switch back to the command line on your local machine and run the following command.

```
vagrant up
```

Open the VirtualBox application and verify that the virtual machine is now running.

Change the Virtual Machine's Name

The Vagrantfile controls the behavior and settings of the virtual machine. Open it with your favorite text editor. You may need to use the File Explorer (Windows) or the Finder (Mac) to navigate to the folder and then open it with your desired editor.

By the way, Atom is a nice text editor that works on Mac, Windows, and even Linux. You can download it for free at Atom.io.

Add the following line somewhere after `"Vagrant.configure(2) do |config|"` and before `"end"`. A good place could be right after the `'config.vm.box = "jasonc/centos7"'` line:

```
config.vm.hostname = "testbox01"
```

Be sure to save your changes.

At this point you could run "vagrant halt" followed by "vagrant up" to activate this change. However, Vagrant provides a shortcut: "vagrant reload" which restarts the virtual machine, loads the new Vagrantfile configuration, and starts the virtual machine again. Give it a go:

```
vagrant reload
```

Connect to the virtual machine to confirm that it's hostname has changed.

```
vagrant ssh
```

You should see a prompt similar to this one containing the hostname that you configured:

Disconnect from the virtual machine:

```
$ exit
```

Assign the Virtual Machine an IP Address

During this course you are going to create virtual machines that will be able to communicate with each other as well as your local workstation. Let's give this virtual machine the IP address of "10.9.8.7". To do that, insert the following line of configuration into the Vagrantfile. Remember, it

needs to be inserted somewhere after "`Vagrant.configure(2) do |config|`" and before "`end`".

```
config.vm.network "private_network", ip: "10.9.8.7"
```

Save your changes.

Reload the virtual machine and let Vagrant assign the IP address to it.

```
vagrant reload
```

Test the connection by pinging the virtual machine.

Window users, run the following command:

```
ping 10.9.8.7
```

Mac and Linux users, run the following command:

```
ping -c 3 10.9.8.7
```

The ping command is one simple way to test network connectivity. If you see replies, then you can safely assume the IP address is reachable and the host is up. If you see "timeout" messages then the system is not answering your ping requests. In the "real world" this doesn't necessarily mean the system is "down." It means it is not answering your ping requests which could be for a variety of reasons. However, for our purposes here if you get "timeout" messages, then you can assume this system is down or something is wrong. The first thing to try is to simply reboot the VM by running:

```
vagrant reload
```

If the ping command fails again, double check the contents of the Vagrantfile paying special attention to the config.vm.network line. Make any necessary changes, restart the virtual machine and try again.

The final step is to reboot the host operating system, I.E. your physical computer. This troubleshooting step primarily applies to Windows users.

Destroy the Virtual Machine

If you're are done with the virtual machine or you want to start over with a fresh copy of the virtual machine, run "vagrant destroy".

```
vagrant destroy
```

It will prompt you to confirm that you want to delete the virtual machine. Answer "y". If the virtual machine is running it will stop it and then delete it. If it's already stopped, it will simply delete it.

Final Vagrantfile for testbox01

Here are the contents of the `shellclass/testbox01/Vagrantfile` file with all the comments removed.

```
Vagrant.configure(2) do |config|
  config.vm.box = "jasonc/centos7"
  config.vm.hostname = "testbox01"
  config.vm.network "private_network", ip: "10.9.8.7"
end
```

Create Another Vagrant Project with Multiple Virtual Machines

Let's create another Vagrant Project and two virtual machines. First, let's return to our "`shellclass`" directory. The "`cd ..`" command changes to the parent directory which is represented by "`..`".

```
cd ..
```

Next, let's create the Vagrant project folder and change into that folder.

```
mkdir multitest
cd multitest
```

Initialize the Vagrant project. This step creates the Vagrantfile.

```
vagrant init jasonc/centos7
```

Add two virtual machine definitions. The first host definition will define the test1 VM with a hostname of "test1" and an IP address of 10.9.8.5. The second host definition will define the test2 VM with a hostname of "test2" and an IP address of 10.9.8.6. Here are the contents of the `shellclass/multitest/Vagrantfile` file with all the comments removed.

```
Vagrant.configure("2") do |config|
  config.vm.box = "jasonc/centos7"

  config.vm.define "test1" do |test1|
    test1.vm.hostname = "test1"
    test1.vm.network "private_network", ip: "10.9.8.5"
  end

  config.vm.define "test2" do |test2|
    test2.vm.hostname = "test2"
    test2.vm.network "private_network", ip: "10.9.8.6"
  end
end
```

Start the virtual machines. (Remember, that if you do not specify a VM name all the defined VMs will be started.)

```
vagrant up
```

Connect to the test1 virtual machine to confirm that it's working and then exit it.

```
vagrant ssh test1
$ exit
```

Connect to the test2 virtual machine to confirm that it's working. While you are logged into the test2 VM, ping the test1 VM to prove that the two virtual machines can communicate with each other over the network.

```
vagrant ssh test2
ping -c 3 10.9.8.5
exit
```

When you brought up the virtual machines you may have noticed a message similar to this one:

```
==> test2: Mounting shared folders...
    test2: /vagrant => /Users/jason/shellclass/multitest
```

You can access the files in the vagrant project directory that resides on your local machine inside the virtual machine. The vagrant project directory is mounted, or shared, via the /vagrant directory. The only file in our local directory is the Vagrantfile. You can look at the file from within the vm. Run the following commands while you're still logged into the test2 VM:

```
ls /vagrant
cat /vagrant/Vagrantfile
```

Exit out of the test2 VM.

```
exit
```

Stop the Virtual Machines

In upcoming projects you'll be working more with Vagrant, virtual machines, IP addresses and more. Feel free to explore the Linux system if you'd like. (Connect by running "`vagrant ssh [VM_NAME}`" within the project folder.) When you're ready to stop or take a break, halt the virtual machine. Remember, you can always pick up where you left off as long as you don't destroy the virtual machine.

```
vagrant halt
```