**IPTABLES Presentation Setup Tutorial**

Written by: Bryan Vanek

The following tutorial is intended to set up a lab on your computer for the iptables presentation that will be taking place on Friday, March 4th. This will prepare your computer in such a way that you will be able to see how iptables behaves in several different situations and in a couple different interfaces. The idea is that after the presentation, you will be able to use this lab to configure iptables to certain parameters that I give in order to configure your firewall for a particular situation.

As an extremely brief overview, iptables is a very flexible firewall utility integrated into Linux operating systems. It uses tables provided by the Linux kernel firewall, and utilizes policy chains via a command line interface in order to accept, drop, or reject different types of traffic. We will of course be going into more detail about what iptables is and how it works, but it’s also very important to understand that this is a vital tool for anyone who wants to work as a Linux system administrator-configuring/securing a firewall is one of the first steps that needs to be taken when hardening a server, and this is one tool that allows you to accomplish that.

We will be performing the following procedures in this tutorial to get you set up:

1. Download and install VirtualBox on your computer.
2. Download, install, and configure a Centos 7 minimal server.
3. Install and enable an HTTPD (Apache) web server.
4. Install, configure, and enable Openfire-an RTC (Real Time Collaboration) server.
5. Migrate from firewalld to iptables.

In addition, there are a few other optional procedures to help with further understanding what’s happening behind the scenes and to give you some more practice being a system admin. These options (along with a few other minor optional procedures) are indicated below:

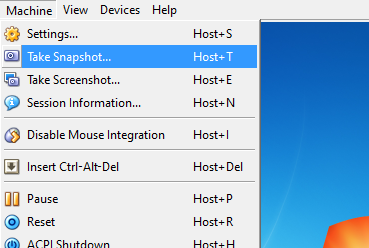
1. Install and configure a PostgreSQL server to run in Openfire.
2. Install, configure, and enable Nagios-an infrastructure monitoring tool.

This tutorial is a little lengthy, but is also intended to be accessible for anyone who is new to system administration. If the tutorial is followed from start to finish, it will likely take around an hour to finish, so be sure to allot enough time to get this done.

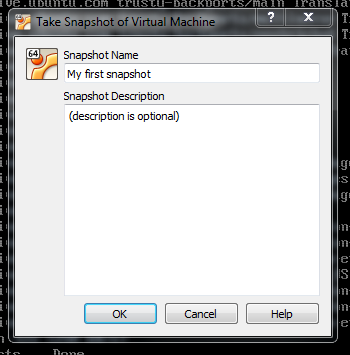
In addition, you’ll also notice that there are a few camera icons with some recommended titles ( http://www.clker.com/cliparts/U/0/C/H/w/V/camera-hi.png - [title name]) at the beginning of certain instructions. Oracle VirtualBox gives you the capability to take a **snapshot** of the current state of your machine, meaning that you can basically travel back/forward in ‘virtual machine’ time. This is a very useful tool which you can use in case a mistake is made, and wherever you see this icon is where I would recommend you take one before starting the next step.

To do this, follow this procedure:

1. Click the ‘Machine’ tab in your virtual environment and select ‘Take Snapshot.’

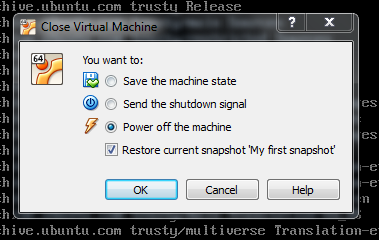


1. Name your snapshot with a title that’s context sensitive (i.e. initial install, post-httpd) and click okay. VirtualBox will then save a snapshot for you.



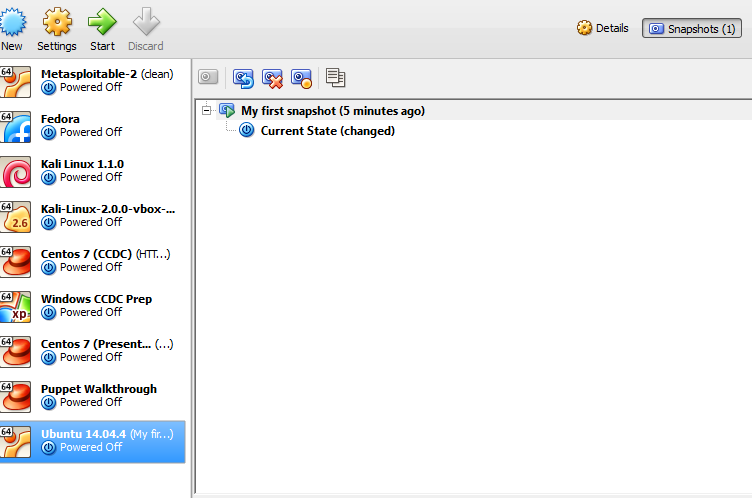
To revert back to that snapshot, you can do one of two things:

1. When powering down your VM, a checkbox will appear giving you the option to restore a snapshot ‘[whatever your title is]’. Check that off, power down, and click okay.



The machine will then revert back to that snapshot whenever you boot back up again.

1. If you’ve already powered down, at the VirtualBox Manager page, click the Snapshots icon in the upper-left hand side of the screen, select your screenshot, and click the ‘Restore Snapshot’ button (  ).

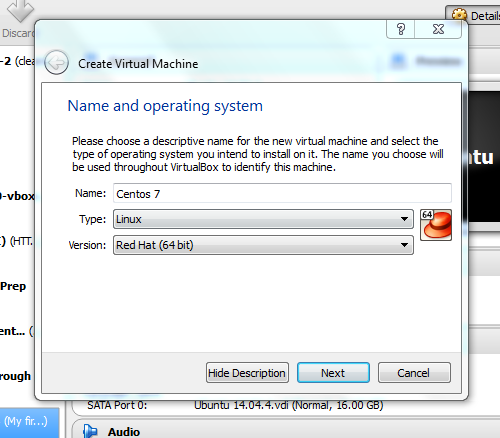


The machine will then revert back to that snapshot when you boot back up.

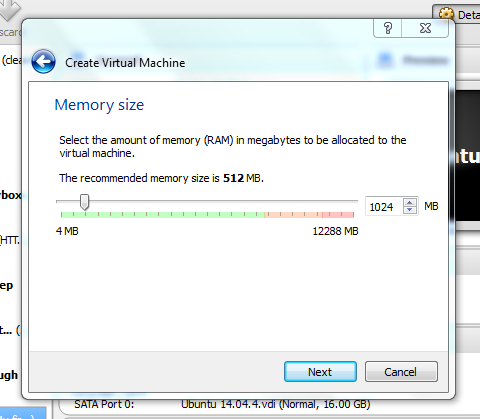
Also, as you’re going through this tutorial, if you have any questions for me, please feel free to email me at [bvanek1@umbc.edu](mailto:bvanek1@umbc.edu) and I’ll try to help out in any way that I can.

**Install Virtualbox and Prepare Centos 7**

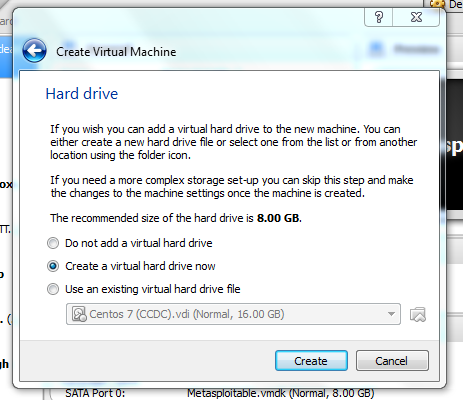
1. Head on over to <https://www.virtualbox.org/wiki/Downloads> and download the most recent version of Oracle VirtualBox. Follow your operating system’s installation wizard; the default settings should be sufficient for our (and your future) needs.
2. Go to <https://www.centos.org/download/> to download Centos 7-a powerful, open-source Linux server environment. We’ll be downloading something called an ISO file-an image of a CD/DVD which is used in the installation of virtual operating systems. We want to download the **Minimal ISO**, so be sure to click that button and click on any of the mirror links. (This file is over 650 MB, so it may take a little while). Take note of where this file is on your computer for later.
3. After the download is complete, open up VirtualBox and hit the ‘New’ button at the top.
4. Name the operating system Centos 7. Note how VirtualBox automatically changes itself to Linux RedHat.



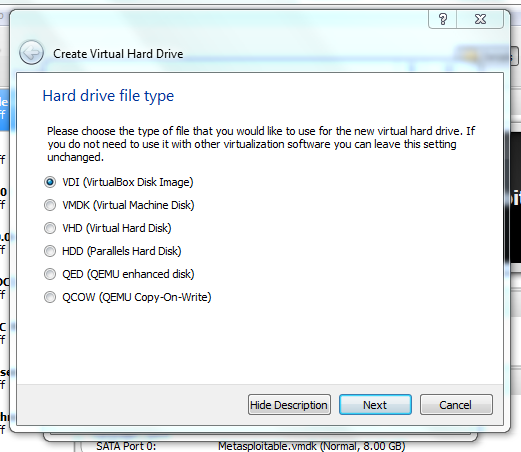
1. Allocate the memory size for Centos. Unless I know I’m going to be needing a lot of memory (for things that are graphic intensive or require a lot of calculations), I usually allocate double the recommended memory size to be safe.



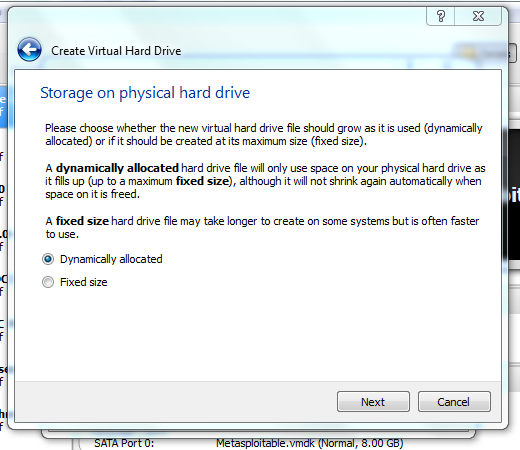
1. Select the option to ‘Create a virtual hard drive now.’



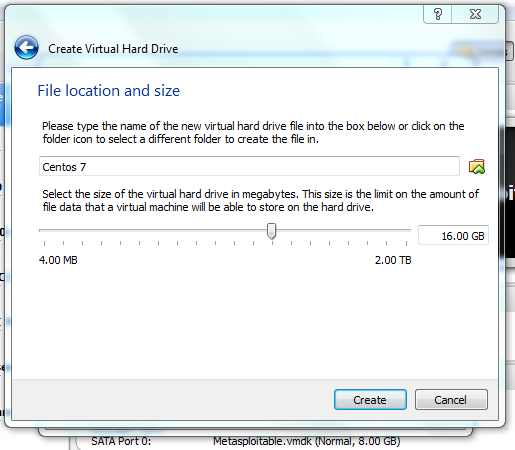
1. For the window that pops up, select VDI (VirtualBox Disk Image).



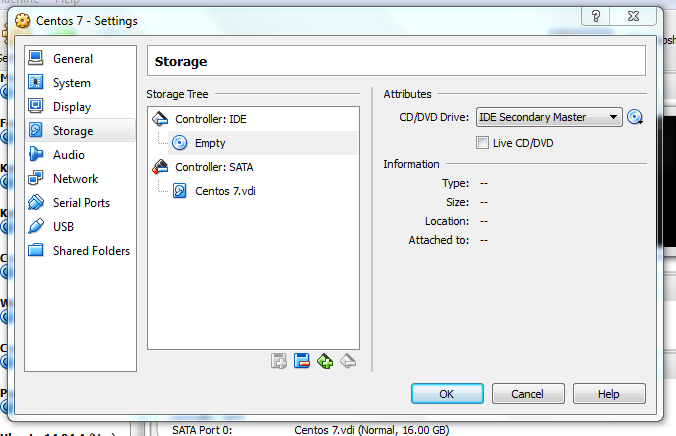
1. Set the hard drive storage to be dynamically allocated.



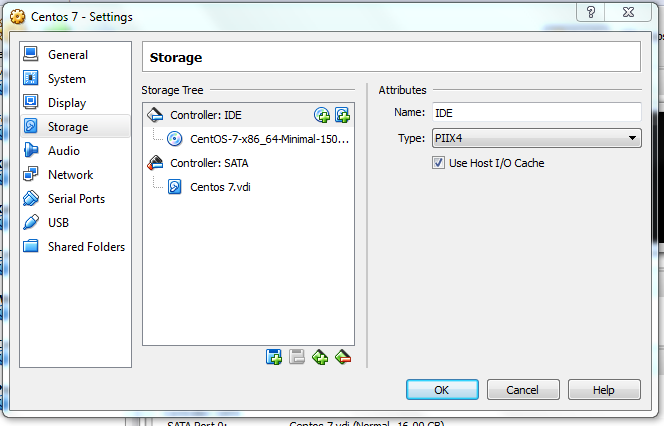
1. Keep the name Centos 7 and set the file size. The recommended file size is 8.00 GB, but again, as a rule of thumb, I always like to go with double the recommended size.



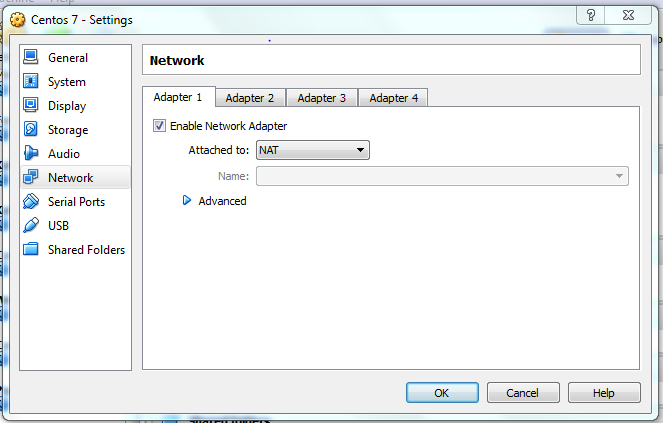
1. Now that we’ve created our VM, we want to add our installation disk. With the VM highlighted, click the settings button and go to **Storage**.



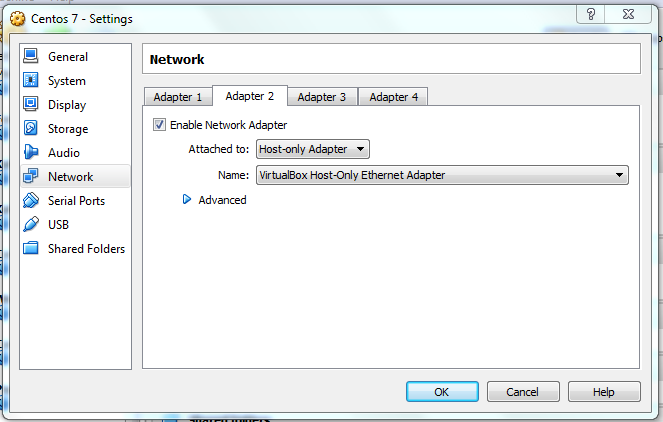
1. Right click the empty disk underneath the Controller: IDE section and click ‘Remove Attachment’. Upon removal, click the ‘Add CD/DVD’ button (  ) and select the ‘**Choose Disk**’ option. Locate and select the downloaded Centos ISO file. When it’s all said and done, it should look something like this:



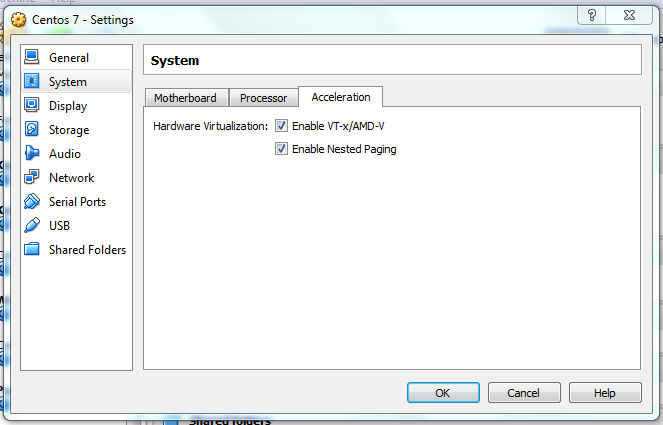
1. For the purposes of this lab, we want to configure a couple different network adapters. While still in the settings menu, click the **Network** option. Ensure that the first adapter is attached to NAT.



1. Click the **Adapter 2** tab, check off the ‘Enable Network Adapter’ tab, and attach it to a **Host-Only Adapter**.



1. Finally, check and see if Virtualization is enabled by going to **System** and clicking the **Acceleration** tab. If the two boxes are checked off, then you’re good to go.



* + *If these options are not checked off, or if you get an error message upon trying to start the VM, you will have to enable virtualization on your computer.*To do this, you will have to access your hardware bios and enable it. This involves figuring out what your BIOS access key is from your computer manufacturer’s website, shutting down, and hitting that key during bootup. (Mine is the ESC key; other common options include, F2, F10, F12, and DEL). Once you’re in, find the option that allows you to enable virtualization and enable it.

**Install and Configure Centos 7**

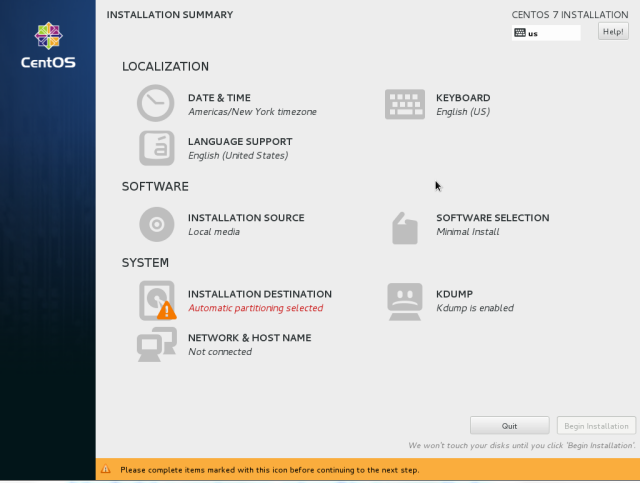
Start the Centos 7 VM. For those who have never worked in a VM before, note that when you click inside the window that your mouse will disappear-this is because you are now inside of a virtual environment! Also note that when we’ve finished installing Centos, you will not have a mouse at all-everything will be done from the command line inside of the VM. It’s a good idea to start getting away from relying on a GUI interface all the time, because as a system admin, you won’t always have access to one.

A couple things to note about working in VirtualBox:

* + You can switch between your physical machine and the VM by hitting the **Host** key, which be default is set to **Right Ctrl**.
  + To enter and exit full-screen mode, hit **Host-F**.
  + Another option is scaled mode, which can act like full screen, but can also act like a filled window with no scroll bars. The disadvantage of this is that the menu options at the top go away, so keep that in mind. To enter and exit scaled mode, hit **Host-C**.

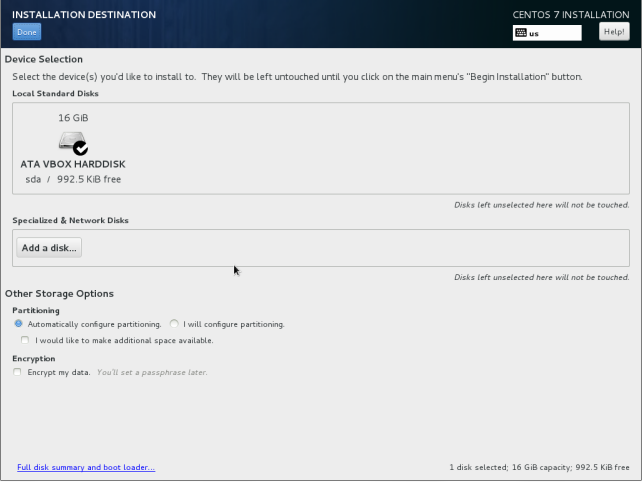
We can now proceed with installing Centos.

1. Once you get to the boot screen, hit enter and wait for the initial setup to finish loading. Select English as your language, and you’ll be at a screen like this:

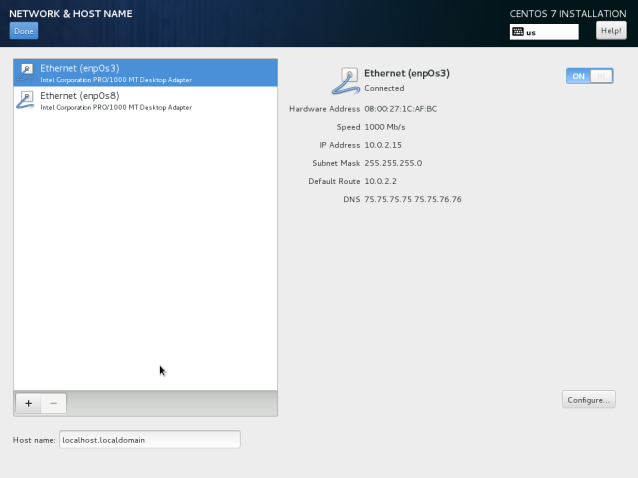


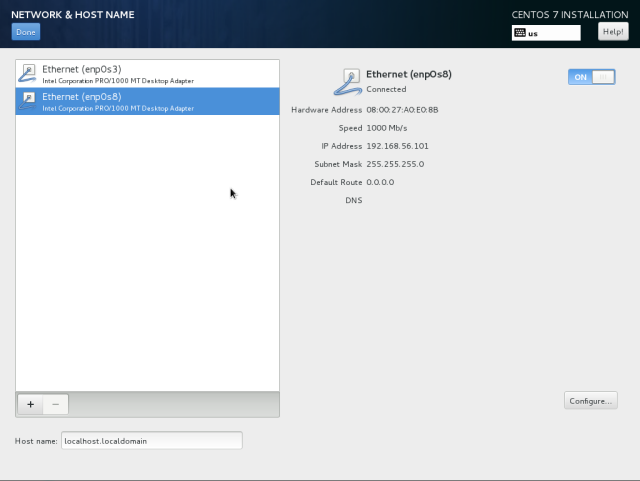
Be sure that everything in the localization and software sections match what’s listed above; the only thing that’s really crucial is the Software Selection section. Ensure that this is set to **Minimal Install**.

1. Click the **Installation Destination** section. Ensure that the 16 GiB ATA hard drive is selected, and that automatic partitioning is selected. Click Done.

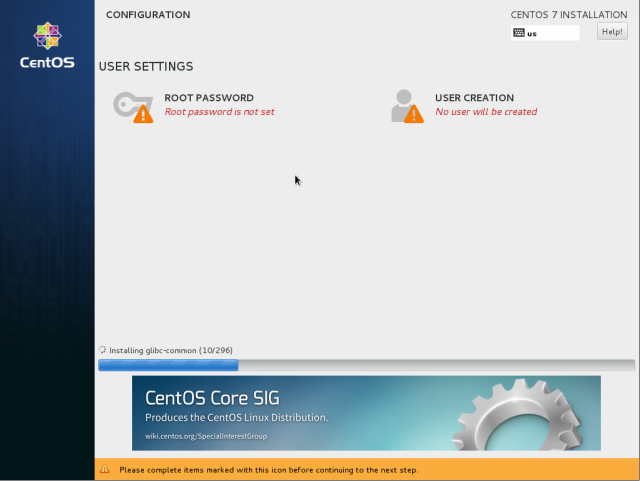


1. Click the **Network & Host Name** section. Ensure that BOTH of the network adapters are turned on. Click done.





1. Once this is done, go ahead and click the ‘Begin Installation’ button. While the install is happening, go on ahead and set your root password, and create an admin user (the message prompts are pretty straightforward here-just be sure that the ‘Make this user an admin’ box is checked).



1. Once this has completed, click the reboot button and log in as root. You are now in the Centos 7 environment! (Note: to scroll up and down, press **Shift-pg up/pg dn**).
2. ( http://www.clker.com/cliparts/U/0/C/H/w/V/camera-hi.png - Mint Installation) One thing that’s very important to do as system administrator is to ensure that your system is constantly up to date. Different versions of Linux provide different ways of performing these updates, and in the case of Red Hat Linux (the version of Linux backing Centos), the way to do that is with the **yum** command. Type in the following command and hit enter (typing **y** and hitting enter whenever it prompts):

yum update

* + *Before proceeding onward from this point, ensure that you know how to perform some basic Linux commands (traversal, making, editing, and removing directories, etc) and can work with the vi text editor*. If you need help with command line stuff, I would recommend Codecademy’s Command Line tutorial (<https://www.codecademy.com/learn/learn-the-command-line>) which gives you some hands on experience with this. If you need some help with learning how to work with vi, there’s a lovely little tool called **vimtutor** that you can get by running the following command:

yum install vim-enhanced

Once the installation is complete, type in:

vimtutor

You will then be launched into a text-based tutorial that will let you learn about using vi. Go through until you’re comfortable with using it.

1. We need to do some network configuration now. First, let’s get some tools to help us with this:

Yum install net-tools

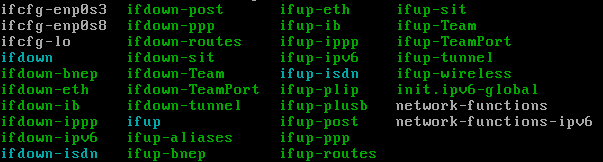
This will give us some important commands such as **ifconfig**.

1. A lot of network configuration information can be found within the **sysconfig** directory in linux. This, in turn, is found within the **etc** (pronounced ‘et-see’) directory, which can be thought of as the nerve center of any Linux system. It contains a lot of (if not all of) the configuration files for your server application files.

We’re concerned with altering our basic network data for now. Change to the following directory:

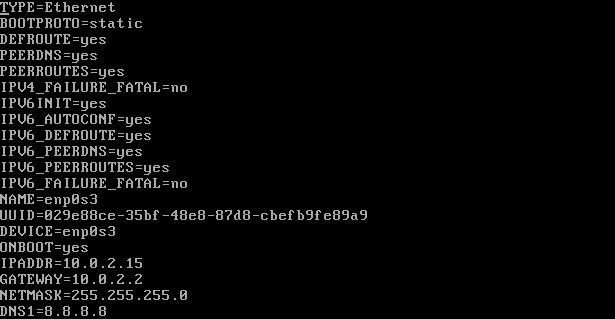
cd /etc/sysconfig/network-scripts

When you look in this directory, you’ll see something like this:



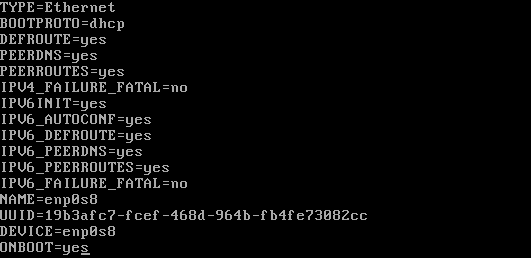
Note the two **ifcfg-enp0s[x]** files in the upper left. These are the files where we want to perform a few tweaks to.

1. Open the first file (ifcfg-enp0s3) in vi. We’ll be configuring a static ip address here. Edit your file to look like this (the UUID value is inconsequential):



The changes include switching the **BOOTPROTO** variable to *static*, the **ONBOOT** variable to *yes*, and the addition of the four items at the bottom.

1. We also need to perform a quick tweak to the ifcfg-enp0s8 file (again, the UUID value is inconsequential):



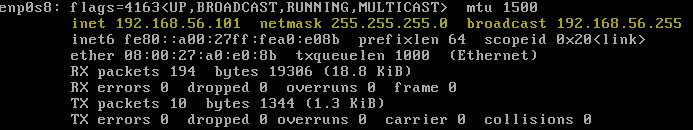
Note the only change here is switching the **ONBOOT** value to *yes*.

1. Once the files have been tweaked and saved, type in the following command:

systemctl restart network

**systemctl** is a very important command within the context of what we’re going to be setting up; it helps with starting and stopping services. In this case, we restarted our network with it. If everything was performed correctly, there should be no error messages.

1. To confirm that everything was configured correctly, simply type in **ifconfig**. You’re looking for something like the highlighted portion in enp0s8 (it won’t actually be highlighted):

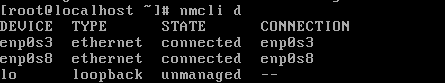


Note: your IP address will likely be different in enp0s8!! This is using DHCP to get an IP for you to work with, so just be aware of that. Also, **take note of the dhcp address**; we’ll be using it later on to access httpd, Openfire, and Nagios.

1. Power off your machine and power it back on, then run **ifconfig** once more to ensure everything’s good to go. You can also type in this command:

nmcli d

**nmcli** stands for **N**etwork **M**anager **C**ommand **L**ine **T**ool, which is used for controlling things in **NetworkManager**. We won’t be going into this for this lab, but you should basically see the following devices (that’s what the d stands for) and statuses listed:



**Install and enable HTTPD**

This is where things are going to start to get a little involved. The remainder of this tutorial is really going to be executing commands and working a little with an external web browser, so if you’re new to this, it’s very important that you TAKE YOUR TIME with these next few parts. And again, don’t be afraid to use your snapshots!

Apache web servers are another critical component for an individual to understand if they want to enter the field of system administration. It allows the system admin and other people who have access to the server to use services through a web browser. It also comes with a bunch of useful modules that can be immediately applied to a lot of other downloadable services and packages.

We’re merely going to be installing Apache here and doing a few minor configuration tweaks, but I would recommend that you read more into what all HTTPD has to offer (and, more importantly, how to secure it properly).

1. (http://www.clker.com/cliparts/U/0/C/H/w/V/camera-hi.png - Post Network Config) Install httpd:

yum install httpd

1. We need to tweak our current firewall settings to allow httpd to run on a browser. These commands are associated with another firewall program called **firewalld**, which we won’t be going into. So you don’t really need to understand these commands for the time being.

firewall-cmd --permanent --add-port=80/tcp

firewall-cmd --permanent --add-port=443/tcp

firewall-cmd --reload

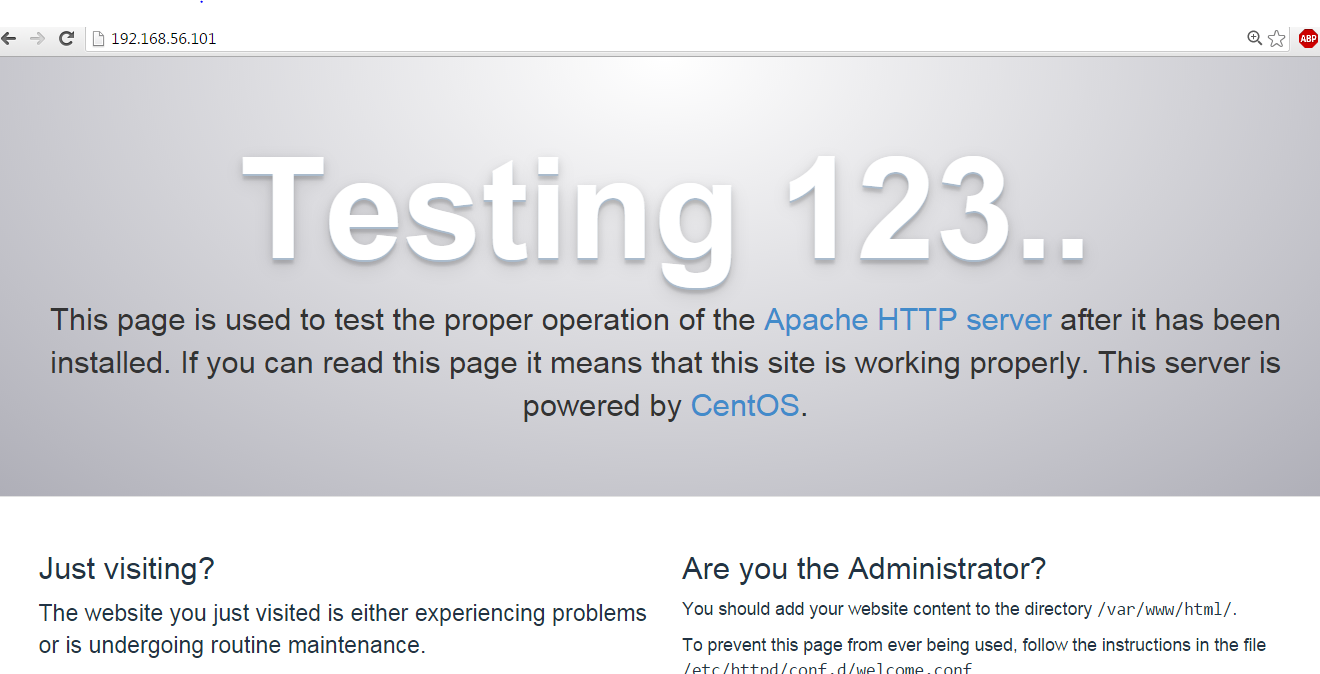
1. We now want to start httpd and have it turn on each time we turn Centos on/off. To do that, we can use systemctl:

systemctl start httpd

systemctl enable httpd

The **enable/disable** actions are used in conjunction with systemctl to have services turn on (and stay on) on start up.

1. To ensure that you installed everything correctly, power off/on and open up a web browser in your physical machine. Type in the IP address of your local host (the dhcp address you took note of). If you get to the following page, you’re good to go.



**Install, Configure, and Enable Openfire**

Openfire is known as an RTC (Real Time Collaboration) server, which is really just a fancy way to say ‘chat server.’ What’s interesting about it is the fact that it runs on an open standard called XMPP (or Jabber), which is used by a lot of other companies for things from infrastructure to security. So despite it’s simple exterior appearance, there’s actually a lot going on behind the scenes.

One of your tasks will be getting Openfire to run behind a proxy in the iptables lab. But before we do that, we need to get it set up. Again, for those who are new, take your time, and use snapshots!

1. (http://www.clker.com/cliparts/U/0/C/H/w/V/camera-hi.png - Post HTTPD Config) We need to install a tool that allows us to grab content from the web. A utility called **wget** allows us to access items over http and https.

yum install wget

1. At the time of writing this tutorial, the most recent version of Openfire is 4.0.1. Use wget to grab it, then install it:

wget <http://download.igniterealtime.org/openfire/openfire-4.0.1-1.i386.rpm>

yum install openfire-4.0.1-1.i386.rpm

1. There’s also a C library that we need to install:

yum install glibc.i686

1. We then need to start and enable openfire. We can use systemctl for this, but we have another option that we can use as well called **chkconfog** for enabling purposes:

chkconfig openfire on

Just another way to show that within Linux, there’s usually always more than one way to do things. Go ahead and enable openfire:

systemctl start openfire

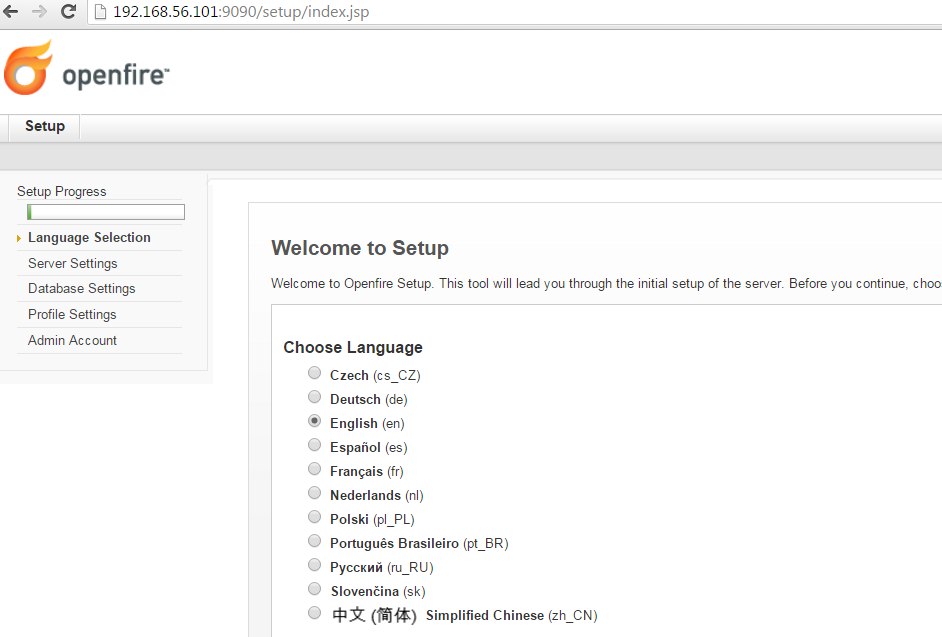
1. We also need to update our current firewall again in order to allow openfire to operate:

firewall-cmd --permanent --zone=public --add-port=9090/tcp

firewall-cmd --permanent --zone=public --add-port=9091/tcp

firewall-cmd --reload

1. At this point, you should be able to directly access the openfire setup though port 9090. Open a web browser and type in the following: [localhost]**:9090**, where localhost is your dhcp IP address. It should look like this:



1. In addition to what we just configured, we also want to set up the proxy itself. This involves manually altering the apache configuration file to work with an interface called ProxyPass. Go ahead and open the httpd.config file:

vi /etc/httpd/conf/httpd.conf

1. Scroll to the bottom and insert the following text to enable the proxy:

<VirtualHost \*:80>

ProxyPass /openfire <http://localhost:9090>

ProxyPassReverse /openfire http://localhost:9090

</VirtualHost>

1. There’s one more thing we need to do: if we try to access Openfire through the proxy right now, we’ll receive a 503 error. This is due to another utility on Linux called SELinux (SE standing for Security Enhanced), an enterprise security enhancement that was merged into the Linux kernel in the early 2000s. While it’s very useful in helping separate good and bad network traffic, it can be a bit troublesome to deal with when getting a few things set up as a system admin.

Fortunately, the solution this time around is rather simple: what we just wrote in our configuration file is a proxy script, and we need to tell SELinux that it’s okay if that proxy script connects to the network. We can do that like this:

setsebool -P httpd\_can\_network\_connect on

1. After we’ve done that, we merely need to reset httpd:

systemctl restart httpd

You should now be able to connect to the Openfire login page via the following url (local host being your dhcp address):

<http://[localhost]/openfire/>

**Migrate from firewalld to iptables**

There’s one more mandatory step that we need to take at this point, and that’s to switch from Centos’ currently activated firewall utility (**firewalld**) to the iptables utility. Firewalld can be thought of as a more recent version of iptables; both of them use netfilter to access packets (or rather, both do packet filtering), and in fact, firewalld uses the iptables command for various operations. However, while it has more flexibility than iptables, it has a whole new set of commands to learn, and is more geared towards desktop distributions. Iptables is also still very heavily used in the industry, making it another necessity for system admins to understand.

1. (http://www.clker.com/cliparts/U/0/C/H/w/V/camera-hi.png - Post Openfire Config) Install the iptables-services package:

yum install iptables-services

1. Test your connection by using a shell script (with the sh command):

sh -c ‘iptables-restore -t < /etc/sysconfig/iptables’

1. Stop Firewalld and start iptables simultaneously:

systemctl stop firewalld && systemctl start iptables

1. Verify that firewalld is no longer running:

firewall-cmd --state

1. Flush out the current iptables rules (we’ll be going over this in the lab):

iptables -F

1. Save your cleared table over to iptables with the following script:

iptables-save > /etc/sysconfig/iptables

1. Proceed to disable and mask firewalld to prevent it from interfering with future configurations (masking prevents the service from being both automatically and manually started):

systemctl disable firewalld

systemctl mask firewalld

1. Enable and start iptables:

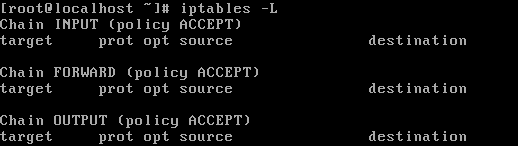
systemctl enable iptables

systemctl start iptables

1. If you’ve done everything, power off/on type in the following command:

Iptables -L

If you see this:



Then you’ve done everything correctly!

1. Last thing to do is to make one final snapshot (http://www.clker.com/cliparts/U/0/C/H/w/V/camera-hi.png - Post iptables config) and ensure that all of your links (httpd, openfire with and without proxy) still work.

That’s it for the mandatory items; you’re now ready for the iptables lab!

*(Optional configurations still to come; will probably be done by Tuesday)*

As stated above, if you have any questions for me, please feel free to email me at [bvanek1@umbc.edu](mailto:bvanek1@umbc.edu) and I’ll try to help out in any way that I can.