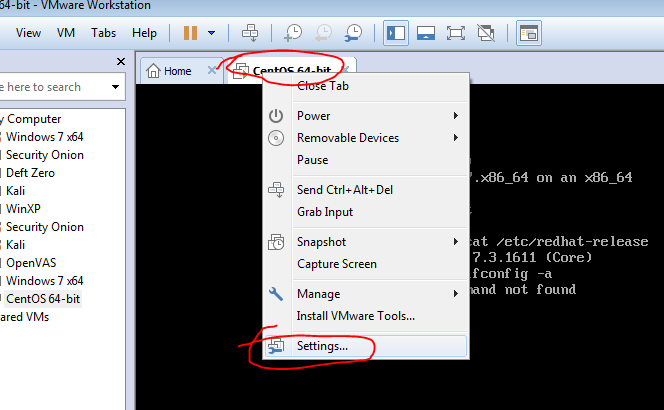
**Class 4**

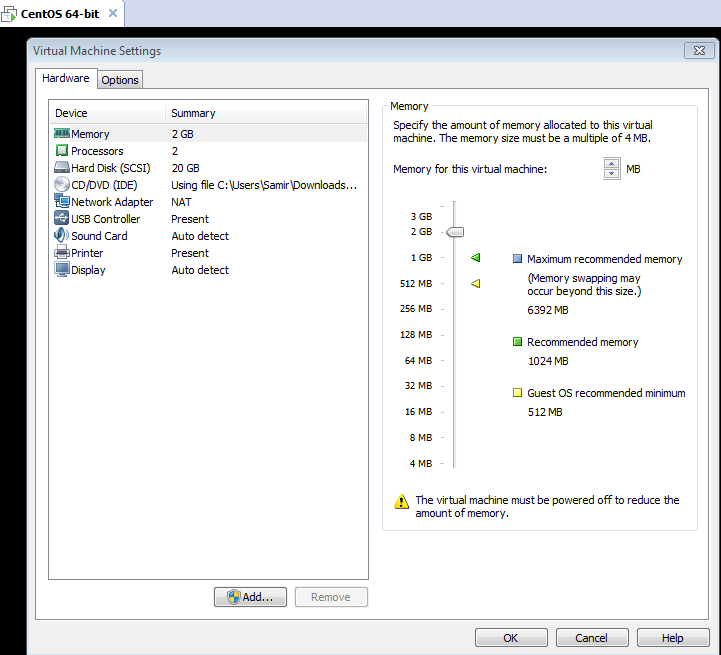
**Networking**

Before looking at Linux commands, let’s first look at how VMware Workstation is setup. Since we are using a type 2 Hypervisor (a kernel that allows us to run virtual machines, in this case it’s VMware Workstation), we have to worry about an “extra layer” that is on top of our Linux operating system.

If you right click on the tab for your virtual machine and go to settings:

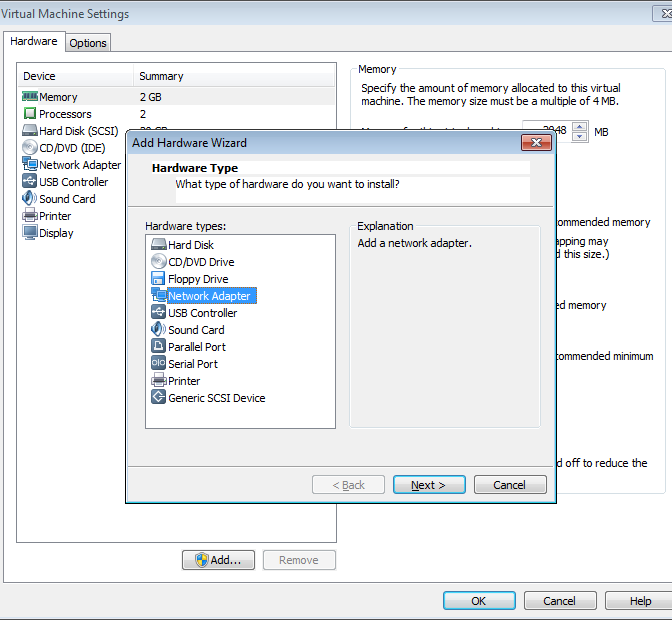


You will get the Virtual Machine settings:

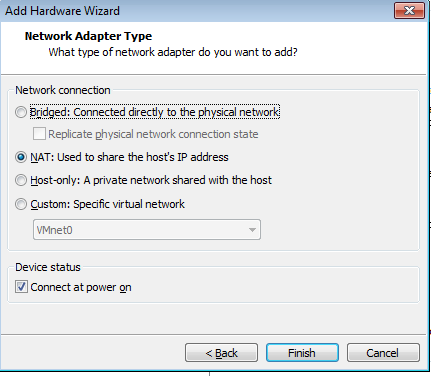


Notice that by default we have a Network Adapter in NAT mode. Note: all of this stuff is virtual – that means you can add as many adapters and hard disks as your system will allow – that’s part of why using virtual machines is useful for learning and testing.

If you click on the Add button (bottom left of the picture above) you will see the following:



After click on “Next” you will see the following:



**Bridged mode** – this would add a network adapter that “piggy-backs” off your computer’s network adapter. What it does is that it shares that same adapter and it tries to communicate to your local network’s DHCP server to get an IP address.

**NAT** – this would mean that VMware Workstation creates a Network Address Translation within VMware. What’s really happening is VMware has a virtual adapter installed on your local computer (with its own IP address) and then that IP is “translated” to different IP inside of VMware so that you have a unique private IP inside of your VMs.

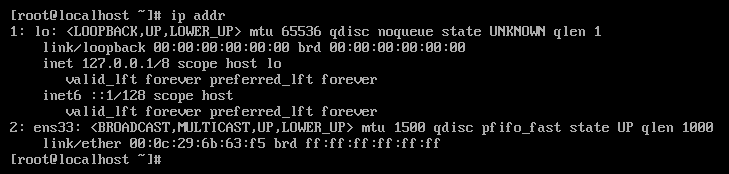
**Custom** – This allows you to add a virtual adapter that is on a custom virtual network. You could have different virtual machines on their own virtual network (VMnet 2 for example) so that they are “segregated” from the rest. You’d have to assign your own custom IP addresses to make this work.

**Host-only** – This makes use of VMware’s built-in DHCP server option. You can set your own scope (range) of IP addresses and VMware will automatically assign IPs.

Don’t worry if this is confusing to you, it’s not a big deal right now, but it’s important for your learning as you will see these networking concepts as you progress in IT.

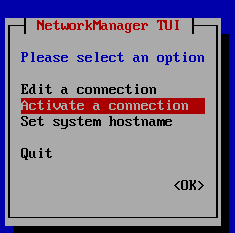
Let’s run the first command for Centos 7, **ip addr** - this command will show the current network interfaces (adapters) that your operating system sees. The example below shows lo (loopback) – every operating system has a loopback address – it uses this address to “ping itself.” If your operating system is unable to ping the loopback address, then chances are you have a bad network adapter or drivers.

You also see the “ens33” network adapter – this is the default NAT adapter that we have added when we first installed our Virtual Machine.

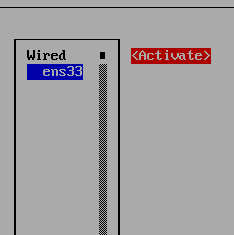


We have to activate the network adapter and to do so, we use the **nmtui** command (this is new in Centos 7).

The command brings up the following screen:



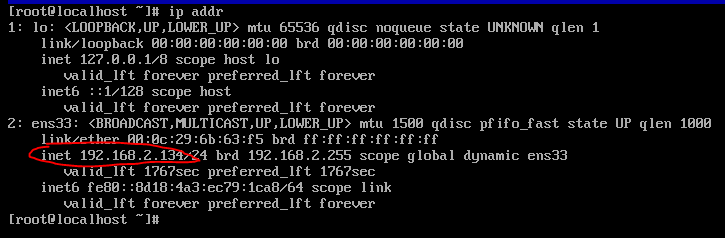
Use your arrows to highlight “Activate a connection” and use the TAB key on your keyboard to go to “OK” and push “ENTER.”



Here you push TAB again to go over “Activate” and push ENTER.

After it’s activated, you can hit TAB to go to “Back” and use your arrows to go to “Quit” to get out.

Run the ip addr command again and you will see that we have an IP address now (192.168.2.134):

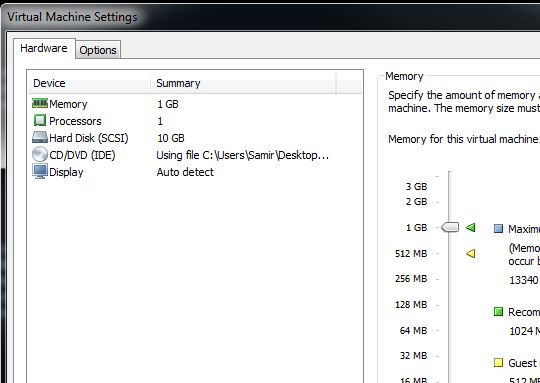


For versions prior to Centos 7, you would use “ifconfig” – this is still very relevant. Here are previous notes (I separated this section with long horizontal bars, you will resume the Centos 7 stuff when I start mentioning about sshd).

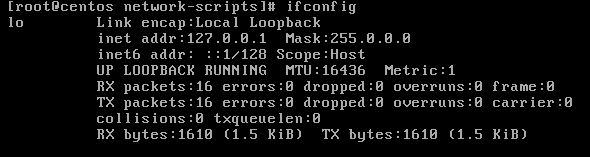
------------------------------------------------------------------------------------------------------------------------

**ifconfig –a** this command will list all of the network interfaces attached to the server as well as the IPs set, MAC address, etc.

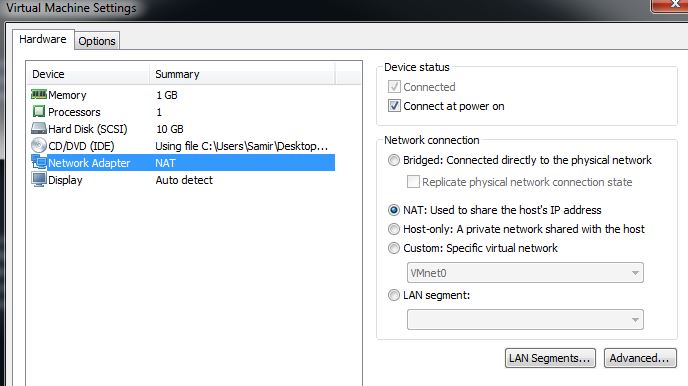
My server right now has no network adapters set in VMware workstation:



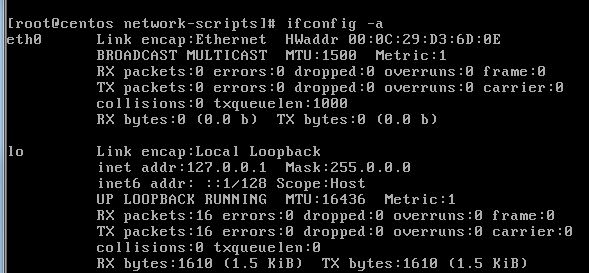
Thus if I run ifconfig –a , all I see is the lo interface (stands for loopback or the 127.0.0.1 private IP, read more about loopback here if you would like: <http://en.wikipedia.org/wiki/Loopback>)



Now I added a Network Adapter and set it to “NAT: used to share the host’s IP address” (Bridge mode should also work fine at HCC).



Centos 6.5: When running **ifconfig –a** again, the new interface, called **eth0** shows up:

****

At this point, we still do not have Internet access because we do not have an IP address setup on eth0 (and a number of other things)

These two articles are good to describe how to add IPs and what ifconfig is about:

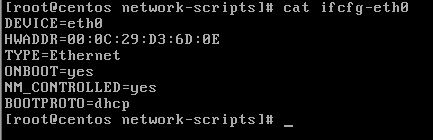
<http://www.informit.com/articles/article.aspx?p=20883&seqNum=2>

<http://www.tecmint.com/ifconfig-command-examples/>

**Steps for Centos 6.5/6.7**

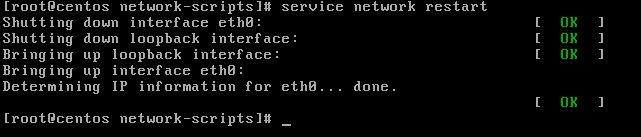
In my case, I am going to make a configuration called **ifcfg-eth0** and it will be saved under **/etc/sysconfig/network-scripts** (this is where the interface configuration files are stored)

This is what it looks like:

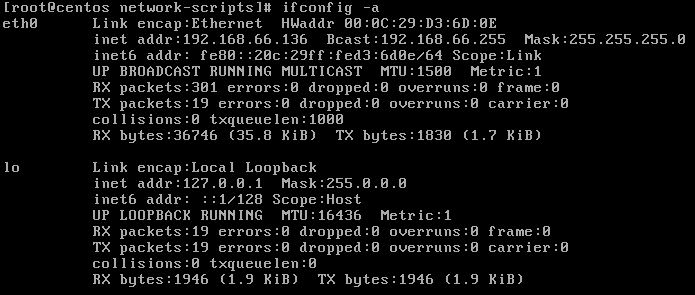


Note that the **HWADDR** is the MAC address (I got that with the original ifconfig –a) and that the bootproto is set to DHCP (DHCP basically means that if there is a DHCP server on the network, which most companies have including HCC, an IP address will be set automatically. Otherwise a specific IP could be set here). Also note that whenever you add a new network adapter in Vmware Workstation, you can go to its properties -> Advanced and find out the MAC address so you can match the interface script properly.

Now I use the **service network restart** command – this will restart the network service so that the server will re-initiate its interfaces and bring them online if they have not (there are other ways to bring interfaces such as doing **ifup eth0** rather than restarting everything. **ifdown eth0** would bring the interface down.)



Now when checking ifconfig –a, an IP address has been automatically assigned (because of DHCP):



After you change the IP, run the following commands:

Service network restart

Ifdown interfacename (where interfacename is the name of the interface you get from doing ifconfig –a, so for example: en1523523)

Ifup interacename

If /etc/sysconfig/network-scripts you should now see something like this (If you didn’t give it a name):

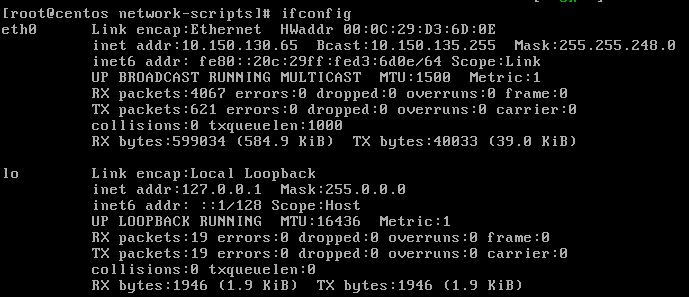
Ifcfg-Wired\_connection\_1

I can also ping google.com now. Part of the reason why I am able to ping google.com is because the **/etc/resolv.conf** was also created (it was created automatically because of the DHCP setting in the interface configuration) – the **/etc/resolv.conf** configuration file stores resolver IPs. Resolvers provide DNS service (think of it as the yellow book for the Internet for domain names) so that when I try to ping google.com, my system is able to translate that name into an IP address and communicate. A popular set of DNS resolvers comes from Google itself and they are: **8.8.8.8 and 8.8.4.4**

Notice how the current IP is set to 192.168.66.136 – that is a private IP (there are three private IP ranges including 10.0.0.0 – 10.255.255.255, 172.16.0.0 -172.31.255.255, 192.168.0.0 – 192.168.255.255 which are used for internal/private IP purposes, they cannot be used for public Internet access by design).

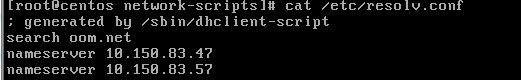
The reason why it is a private IP is because I picked NAT and NAT stands for “Network address translation” – this is basically what companies use to “share” a public IP address with multiple users in a building.

Now if I change my adapter to “Bridged” and check “replicate physical network connection state” I get:



Now I get an IP 10.150.130.65. My local laptop’s IP is 10.150.30.39.

My /etc/resolv.conf has also changed to:

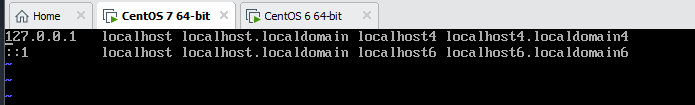


These are the DNS resolvers that we use at work (not HCC)

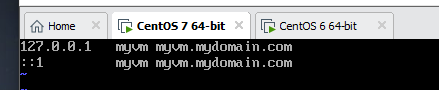
Centos 7 \*resume here\*

**Hostname** – To update your hostname (a name for your machine that uses a domain name structure), use this command as an example: hostname myvm.mydomain.com

Then you will need to edit the /etc/hosts file with the new hostname. Change this:



To this:



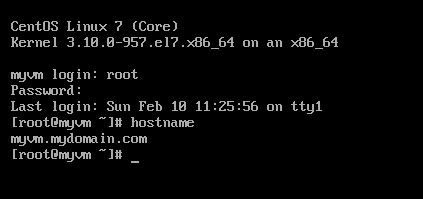
Also change /etc/hostname from this:



To this:



Reboot your VM and your hostname should be set:

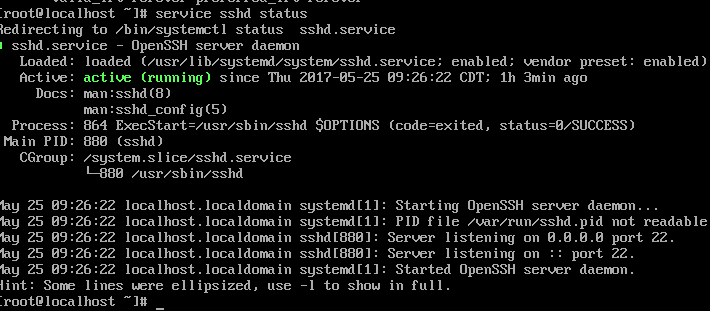


**SSH (Secure Shell)**

With our minimal install, the **sshd** service was already installed – this allows our server to SSH (connect to another server in a secure manner) or to be accessed via SSH.

You can use the following command to check that sshd is running:

**service sshd status**



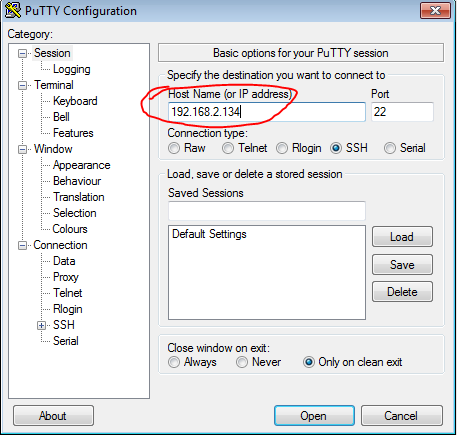
Note: If it’s not running, then type: service sshd start

Now I can use a shell program such as **Putty** to SSH into my server. PuTTY is a popular client program for SSH. It gives us freedom from not having to use the VMware Workstation console to type commands. It’s useful for copy pasting things and running multiple windows. You run PuTTY on a Windows machine, and tell it to connect to (for example) a Unix machine. PuTTY opens a window. Then, anything you type into that window is sent straight to the Unix machine, and everything the Unix machine sends back is displayed in the window. So you can work on the Unix machine as if you were sitting at its console, while actually sitting somewhere else.

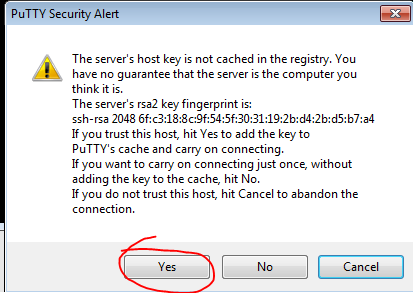
Go to this website: <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

Download: putty.exe for Windows on Intel x86

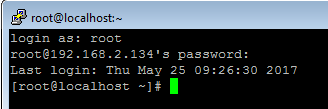
Now open Putty and enter your IP address for the “hostname” – mine was 192.168.2.134 as you saw earlier, and the port is 22 by default (make sure the connection type is SSH and that is also default):



Click “Yes” when the security alert sign comes up (this is asking you to accept the newly created RSA private key to be stored on your computer for the session):



You will then be prompted for a user (root) and password (password you set) and then you will have access:



Putty is great because now you can highlight text and then just right click to paste it and you can also modify the font/colors (click on the top left putty icon and go to change settings, appearance/colors).

Note: We can edit the /etc/ssh/sshd\_config to change the Port number from 22 to something else for example – that’s considered safer to hide your server but the only drawback of that is if you want to SSH to your server from a foreign network, you may not be able to communicate via the new port that you set (in which case you’d ideally VPN to your own network and SSH to your server). A good command that saved us to see if we had a syntax error in our config was: **sshd -t**

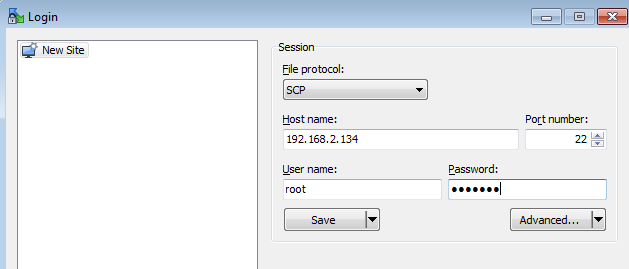
There is also SuperPutty that a lot of techs like, something you can explore later on -> <https://github.com/jimradford/superputty> or https://code.google.com/p/superputty/ (cool thing about SuperPutty is some of the customizable things you can do and the ability to have multiple tabs for your sessions).

The terminal program from Macintosh operating systems could also be used to SSH into a server.

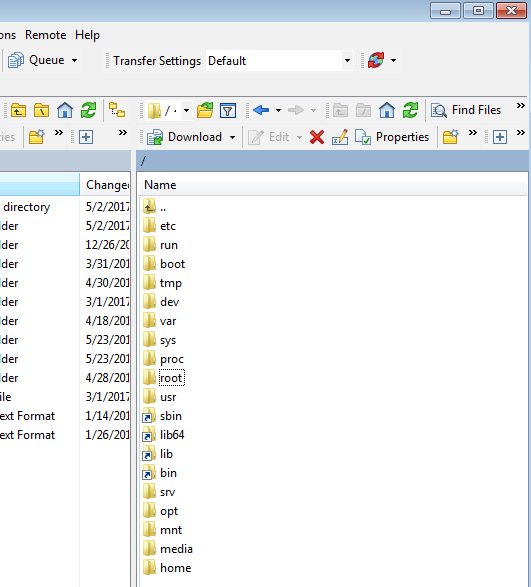
**WinSCP** –is a program that can serve as a client for different protocols such as FTP and SCP. We use it for SCP (secure copy) which allows us to login to our server (port 22) and view all of our files and upload/download easily while being in a secure session.

The website is: <http://winscp.net/eng/index.php>

This is optional, but recommended you try:



After connecting, I am able to see all of my files and I can drag / drop them to my local computer and vice-versa.



Note: I had to hit the “..” up arrow once to move up a directory after first connecting because it put me in the /root folder (the home user for the root user) first.

**yum –** yum is an interactive package manager – it can be used to install tools such as “man” (manual) “postfix” (email) “httpd” (apache web server) etc. It can also be used to perform system updates.

Full description on yum: <http://linux.die.net/man/8/yum>

Details on adding or changing repositories if needed:

<http://www.cyberciti.biz/tips/rhel5-fedora-core-add-new-yum-repository.html>

To check for a list of repos: <http://www.cyberciti.biz/faq/centos-fedora-redhat-yum-repolist-command-tolist-package-repositories/>