LINUX (UBUNTU, CENTOS, FEDORA SERVER AND DESKTOP)

SONYA LAO
CCNP P1/2

Purpose

The purpose of this lab was to explore the different technologies available for Linux. Through installing the different flavors of Linux, we learned the basics of a command-line based server and desktop.

Background

Linux is an open-source operating system. This means that anyone can download the operating system and mold it to fit their own needs. For our purposes, we wanted to install SSH and SSL/HTTPS web server on the servers. Secure Socket Shell (SSH) allows remote computers to connect to the server with authentication. This is like managing 2 properties at the same time. For example, if you own two houses, you cannot be at both locations at once. One house may need a new paint coat, and you may need to restock groceries in the other house. You normally cannot be at both locations at once. But with SSH, it is possible to be in both "houses" at once. As long as you have the public and private key password, you can access and manage different servers or routers (the "houses"). Same is true with SSH. As long as private and public authentication keys are configured, the user can enter their username and password to login to another server on the network. The HTTPS web server gives another layer of protection to websites. It requires a secure link in the form of a certificate before information is passed from the web server to the browser. As the server, you are like the gatekeeper. You hold both a secret key and a public certificate. Whenever a guest wants to send a message through you, you use your secret key to protect the message. Whenever a guest wants information, you use the public certificate to decode the information. The Apache HTTPS server operates in a similar manner. It encrypts and decrypts content using a self-signed key and public certificate, respectively.

Lab Summary

First, I installed the 4 different VMs – Ubuntu Server, CentOS Server, Fedora Server, and Fedora Desktop. I also configured the router connected to the host PC with the IP address of 192.168.1.1 on the corresponding interface. Next, I entered the Ubuntu terminal and installed openSSH server and the Apache2 packages. Then, I edited the SSH configuration file using the Nano text editor to add a banner and allow my username access via SSH. Then, I tested the SSH from my host PC through Putty to the Ubuntu Server, and from the Fedora Desktop. Next, I configured the Apache web server for HTTP and HTTPS. First, I enabled the apache module and restarted the server for the changes to take effect. Then, I created a subdirectory for the certificate file to be placed in. Then, I generated the self signed SSL certificate. For more information on the key, visit the Lab Commands section. After created the certificate and restarting the module, I was able to enter the IP address of the Ubuntu server into my web browser and view the Apache Web Server page. To test HTTPS, I added http:// before typing in the address of the server. Instead of directly loading the web server page, I encountered a web page saying that “there was a problem with the web site’s security certificate,” indicating that the certificate was enabled. Setup and configuration of the Fedora and CentOS web servers was relatively similar and straightforward. Since SSH was already installed on both servers, I just set the IP address on each server. This was done using the ifconfig command. After setting the IP address and verifying that I could ping the router from both servers, I tested establishing an SSH connection from my host PC to each server.

Network Commands

PACKAGE AND SERVICE MANAGEMENT

Ubuntu: **`sudo apt install [package name]`**

This command is used to install packages. `sudo` allows the user to issue root commands, given that they have root access. `apt` is an abbreviation for Ubuntu's Advanced Packaging Tool. APT is used to install/remove packages, as well as update the package index and upgrade packages.

Fedora: **`dnf install [package name]`**

On the Fedora Server, DNF is used to install packages. DNF refers to Dandified yum, and allows the user to install, update, and remove packages.

`sudo service [service name] [restart | stop] -` This command allows you to restart or stop a service. Restart is often used after a change has been made to the package and needs to be restarted before changes take effect.

NETWORK COMMANDS

`ip addr –` This command allows you to check the network status of the server on any Linux server.

`ifconfig [interface] [ip address] netmask [subnet mask] -` This command also allows you to check the network status of the server on any Linux server, if you issue `ifconfig` as a standalone command. However, you can also set the IP address of an interface using this command. It must be issued on the root level, so use `sudo` before the command, or have root access.

UBUNTU APACHE WEB SERVER COMMANDS

`sudo a2enmod ssl –` This command enables the Apache2 SSL module. The a2 refers to Apache2, en = enable, mod = module

`sudo service apache2 restart -` This command enables the Apache2 SSL module. The a2 refers to Apache2, en = enable, mod = module

`sudo mkdir /etc/apache2/ssl -` This command creates a subdirectory under the Apache configuration, which is the directory where the certificate will be stored.

`sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt`

This command creates an SSL Certificate and self-signed key pair. The self-signed key is kept secret on the server, while the certificate is public to any user requesting the content. The certificate decrypts the content protected by the SSL key.

Below is a breakdown of each component of the SSL Certificate command:

openssl: This references the OpenSSL tool that is used to create and manage certificates and keys

req: This is a subcommand for X.509 certificate signing request (CSR) management. SSL follows X.509, which is a public key infrastructure standard.

-x509: This option specifies that we want to make a self-signed certificate file instead of generating a certificate request.

-nodes: This option tells OpenSSL that we do not wish to secure our key file with a passphrase, because if we put a password, we would have to authenticate every time Apache2 restarted.

-days 365: This specifies that the certificate we are creating will be valid for 365 days.

-newkey rsa:2048: This option simultaneously creates the certificate request and private key. The rsa:2048 tells OpenSSL to generate an RSA key that is 2048 bits long.

-keyout: This parameter names the output file for the private key file that is being created.

-out: This option names the output file for the certificate that we are generating.

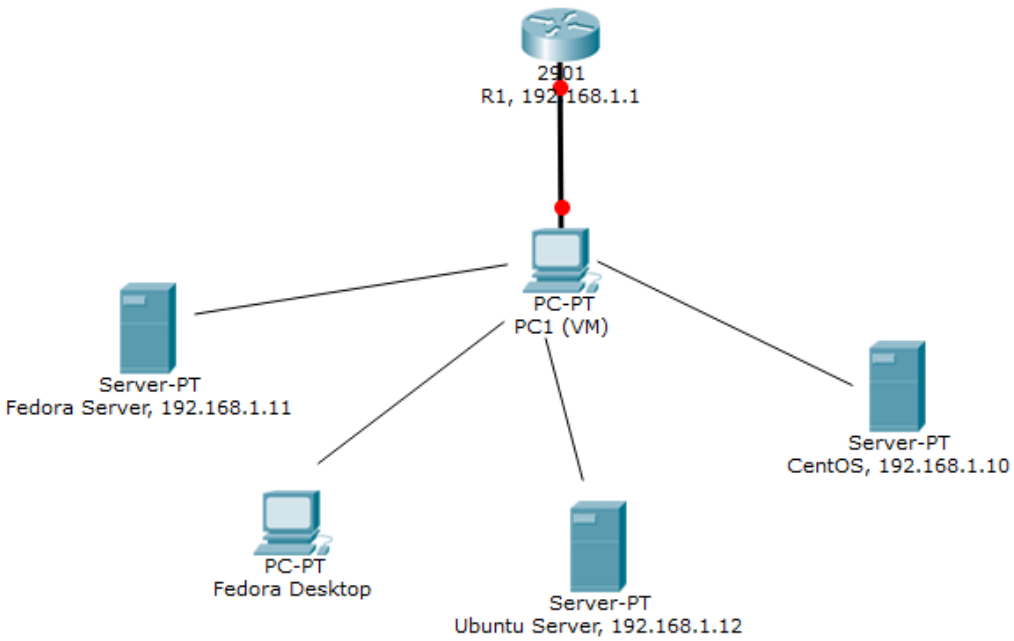
sudo nano /etc/apache2/sites-available/default-ssl.conf - This command uses the nano text editor to access the default-ssl.conf file, to modify the Apache configuration

Below are the changes made to the default-ssl conf file:

```
<VirtualHost _default_:443>
                ServerAdmin your_email@example.com
                ServerName server_domain_or_IP
                ServerAlias domain_name
                DocumentRoot /var/www/html
                ErrorLog ${APACHE_LOG_DIR}/error.log
                CustomLog ${APACHE_LOG_DIR}/access.log combined

                SSLEngine on
                SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt
                SSLCertificateKeyFile /etc/ssl/certs/apache-selfsigned.key
                <FilesMatch "\.(cgi|shtml|phtml|php)$">
                                SSLOptions +StdEnvVars

        </VirtualHost>
```

sudo a2ensite default-ssl.conf - This command enables the SSL Virtual Host at the `default-ssl.conf` location

Network Diagram



Configurations

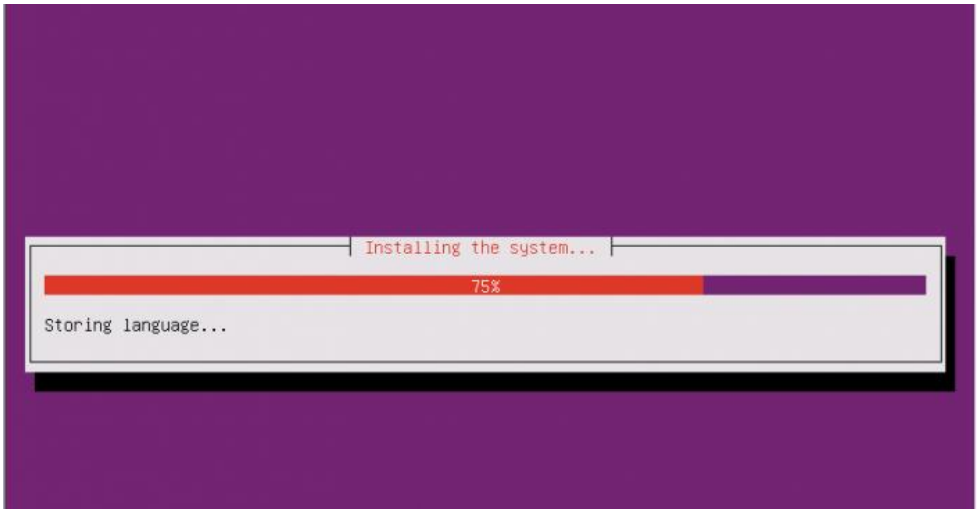
R1 show run:

```
en
conf t
int g0/0
ip address 192.168.1.1 255.255.255.0
no shutdown
exit
```

Ubuntu Server Install:

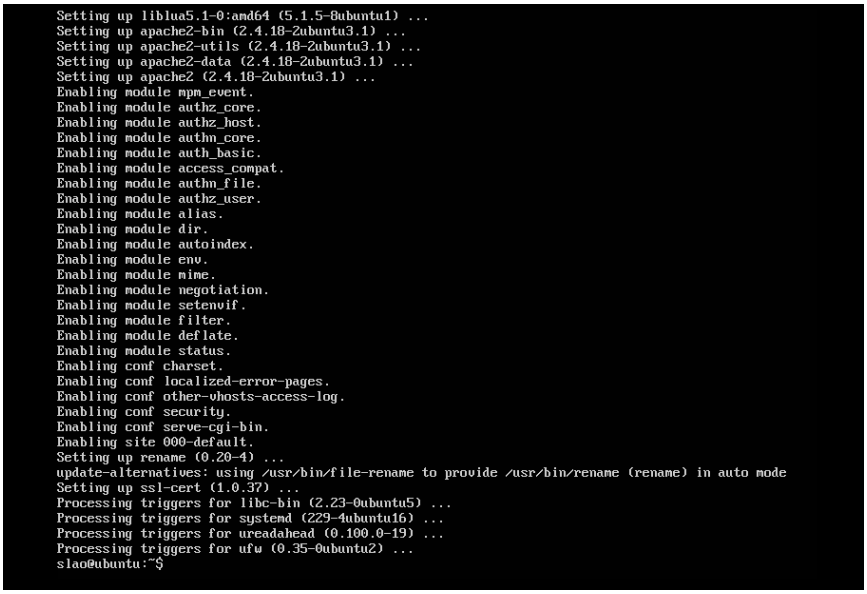
Create a new Virtual Machine and use your Ubuntu Server Disc Image file.

Follow through the guided installation, setting the IP address of the server and the default gateway. Create the administrator user and password.

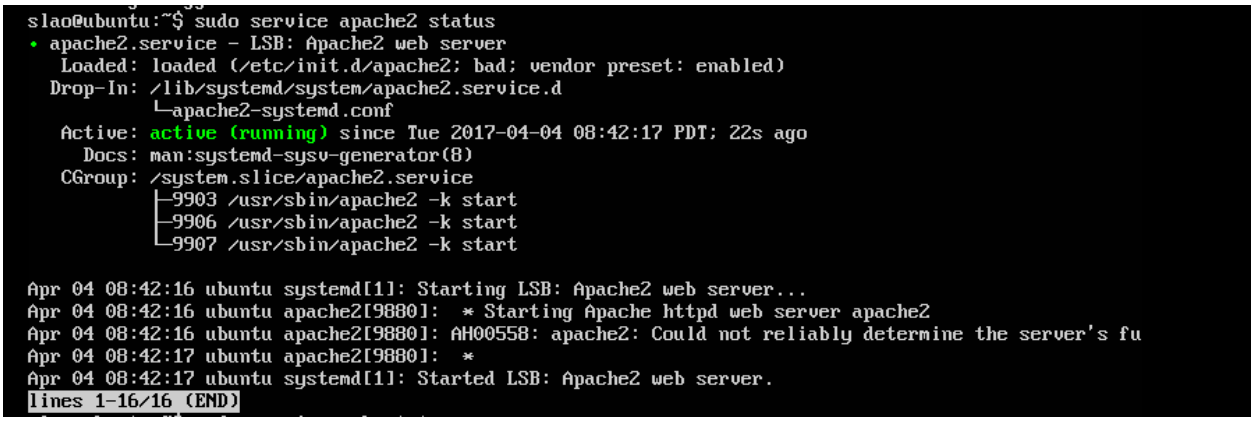


Ubuntu Apache/HTTPS Server Configuration:

First, I issued the `sudo apt-get install apache2` command. The host PC must have internet connectivity for the install to begin.



Next, issue the `sudo service apache2 status` to check that Apache is active



Next, type the command `sudo a2enmod ssl` to enable the SSL Module, then issue `sudo service apache2 restart` to restart the web server, and allow the change to take effect.

Then, create a subdirectory within Apache to place the certificate files in using the `sudo mkdir /etc/apache2/ssl` command.

Next, create the key and certificate and place them in the newly created subdirectory, as shown below. For information on each part of the command, view the Commands section of this lab.

```
cisco@ubuntu:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/ap
ache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt
Generating a 2048 bit RSA private key
.....................................................................+++
.....................................................................+++
writing new private key to '/etc/ssl/private/apache-selfsigned.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Washington
Locality Name (eg, city) []:Bellevue
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Cisco
Organizational Unit Name (eg, section) []:Cisco
Common Name (e.g. server FQDN or YOUR name) []:1.1.1.3
```

You will also need to enter some basic information about the key, including the Country Name, State/Province, Name, Organization Name, etc. The most important is the Common Name – this is where you place the IP address of the server. You can also place the domain name there if you have one.

Next, change the default-ssl.conf file that contains the default SSL configuration. Use the command `sudo nano /etc/apache2/site-available/default-ssl.conf` to edit the file using the Nano editor.

To set the server to use the virtual host, you need to configure the ServerAdmin, ServerName, ServerAlias, and DocumentRoot (see highlighted areas).

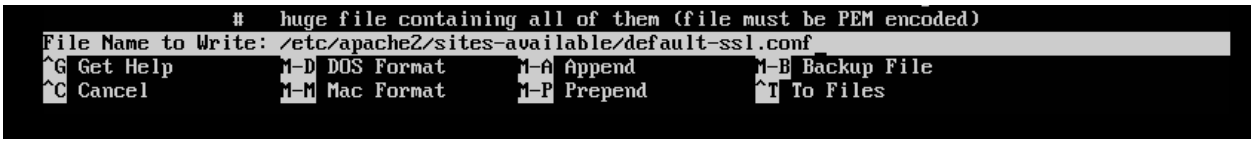
```
 GNU nano 2.5.3        File: /etc/apache2/sites-available/default-ssl.conf

<IfModule mod_ssl.c>
        <VirtualHost _default_:443>
                ServerAdmin sonya@cisco.com
                ServerName cisco.com
                ServerAlias www.cisco.com

                DocumentRoot /var/www/html

                # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
                # error, crit, alert, emerg.
                # It is also possible to configure the loglevel for particular
                # modules, e.g.
                #LogLevel info ssl:warn

                ErrorLog ${APACHE_LOG_DIR}/error.log
                CustomLog ${APACHE_LOG_DIR}/access.log combined
```

```
#   A self-signed (snakeoil) certificate can be created by installing
#   the ssl-cert package. See
#   /usr/share/doc/apache2/README.Debian.gz for more info.
#   If both key and certificate are stored in the same file, only the
#   SSLCertificateFile directive is needed.
SSLCertificateFile      /etc/apache2/ssl/apache.crt
SSLCertificateKeyFile /etc/apache2/ssl/apache.key
```

The only other modification to the file is to set the SSLCertificateFile and SSLCertificateKeyFile to the correct file locations.

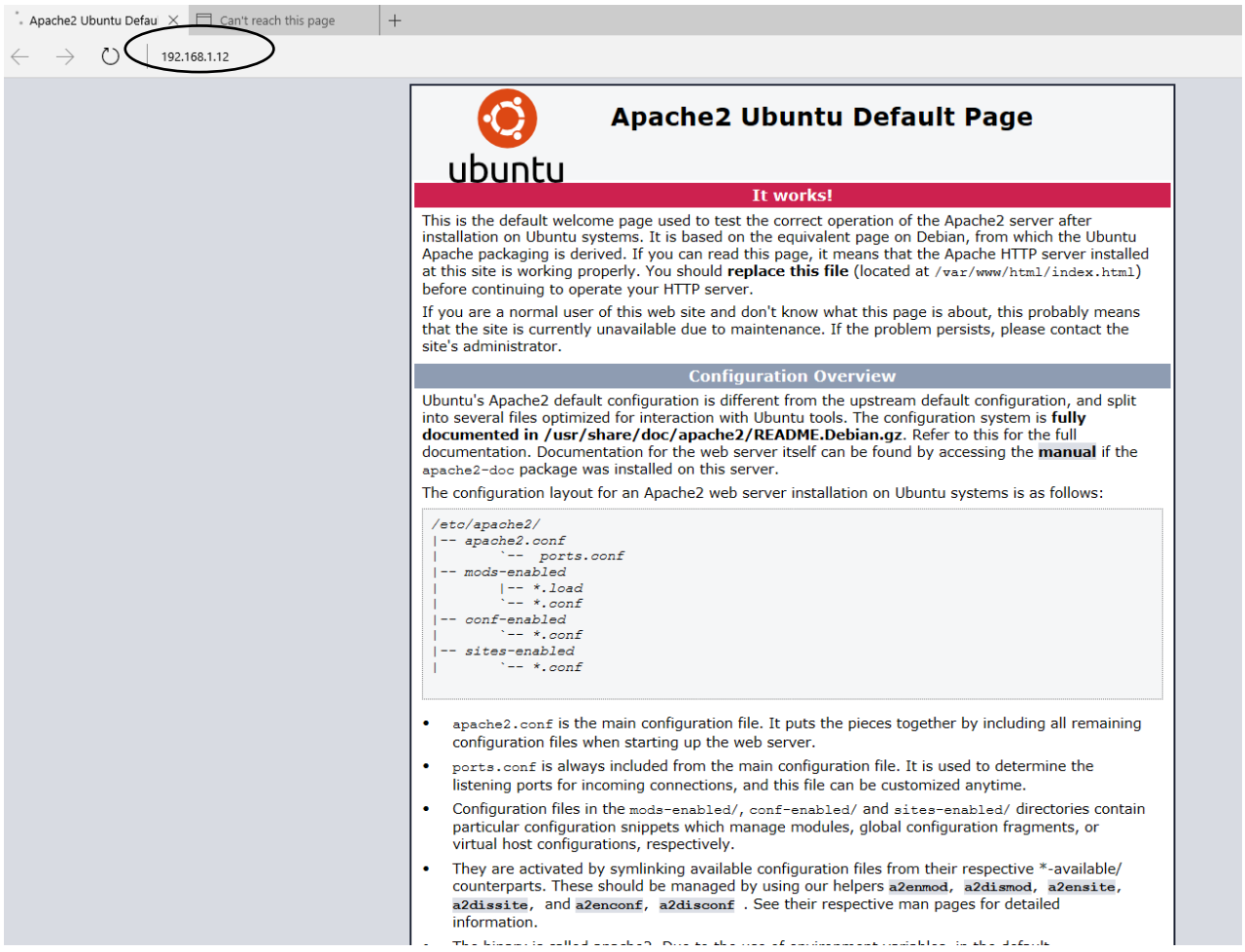
To save the file, use the Ctrl-O keyboard shortcut

Click Enter after confirming that the text editor is writing the right file.



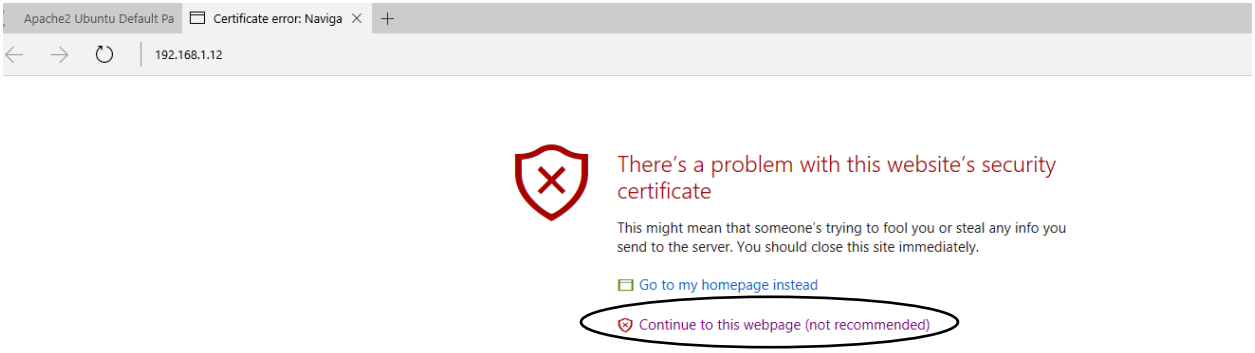
Then, enable the SSL virtual host by typing `sudo a2ensite default-ssl.conf`

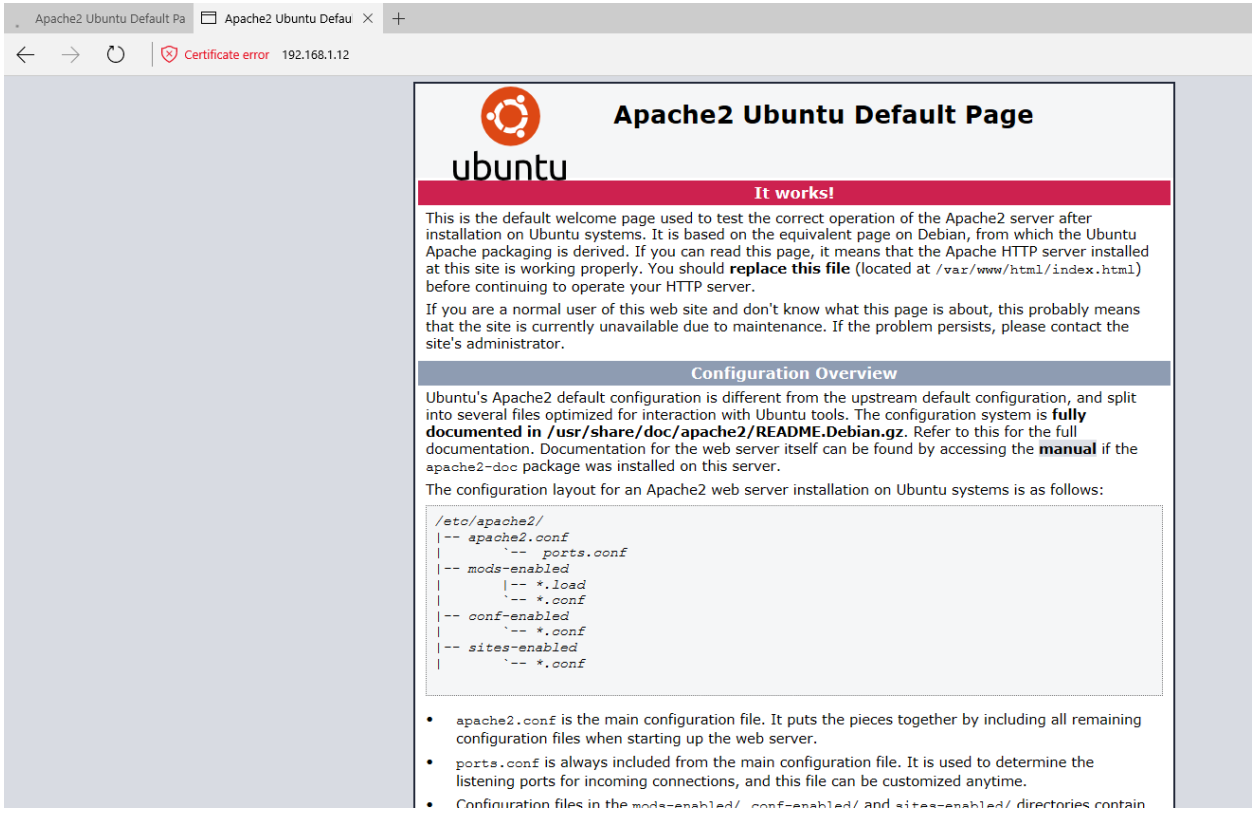
Restart Apache to load the new virtual host file (`sudo service apache2 restart`)



To test HTTP, enter the IP address of the server into the browser. The Apache2 Ubuntu Default Page should appear. To test HTTPS, enter [https://[IP](https://[IP) address of the Server]

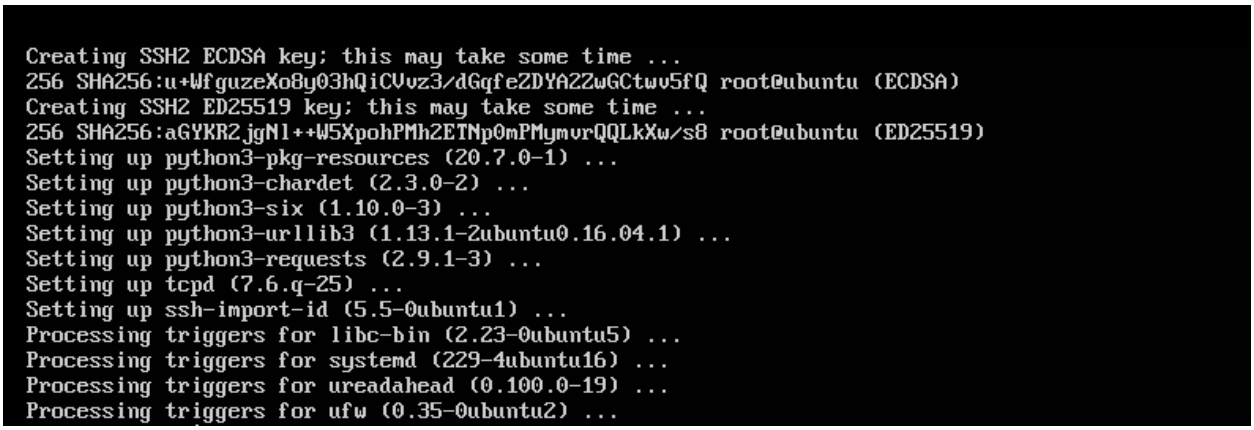
There should be a security certificate message. Click “Continue to this webpage”, and the Apache2 Ubuntu Default Page will appear.



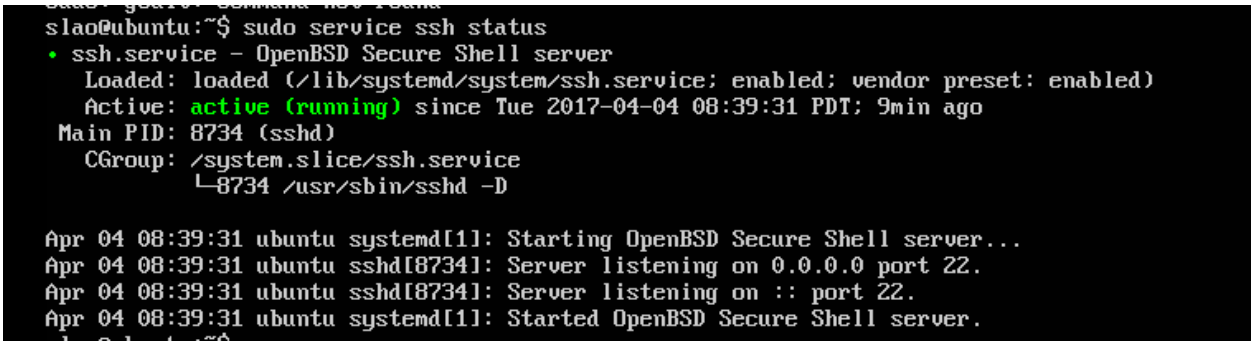


Ubuntu SSH Server Installation:

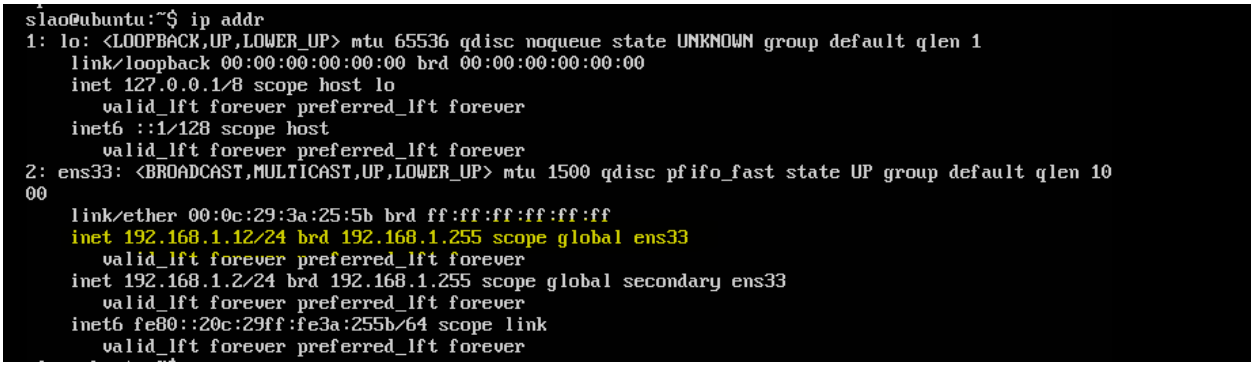
First, issue the `sudo apt-get install openssh-server` command.

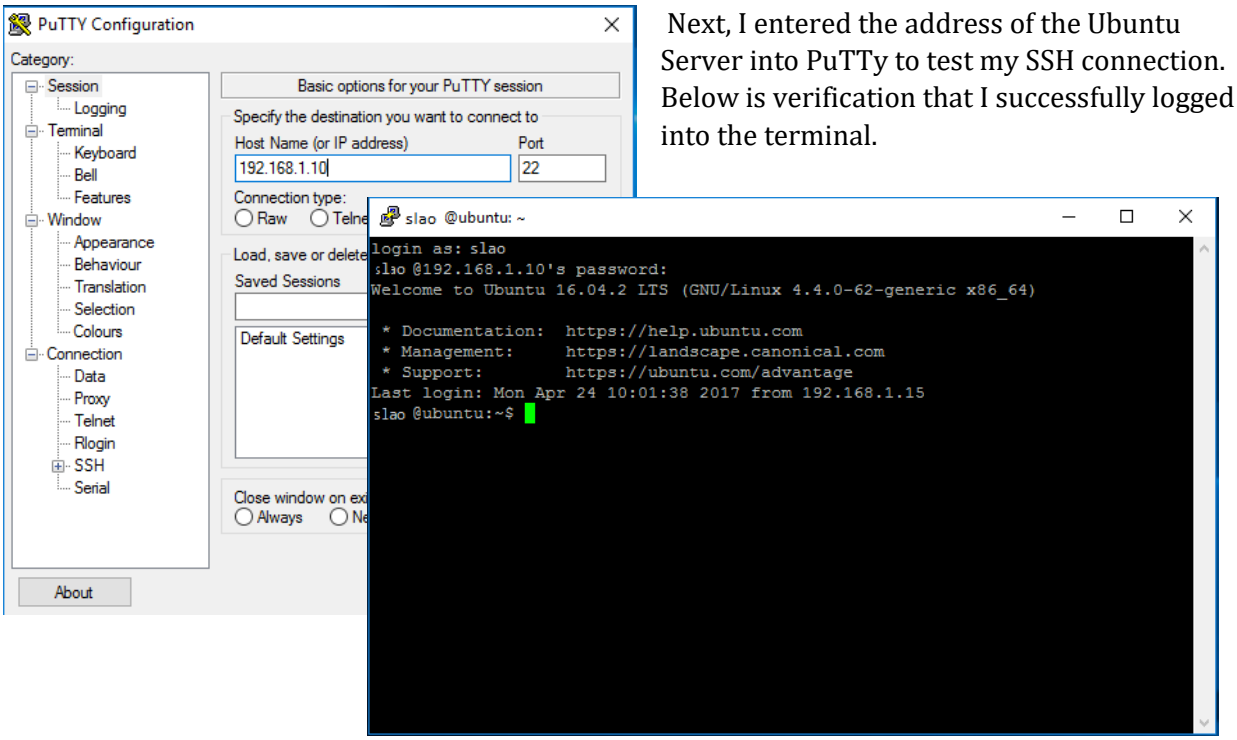


Issue the `sudo service ssh status` to check the OpenSSH install status



Use the `ifconfig [interface] [ip address] netmask [subnet]` command to set the IP address of the server. To check the network status of the server, use the `ip addr` command.

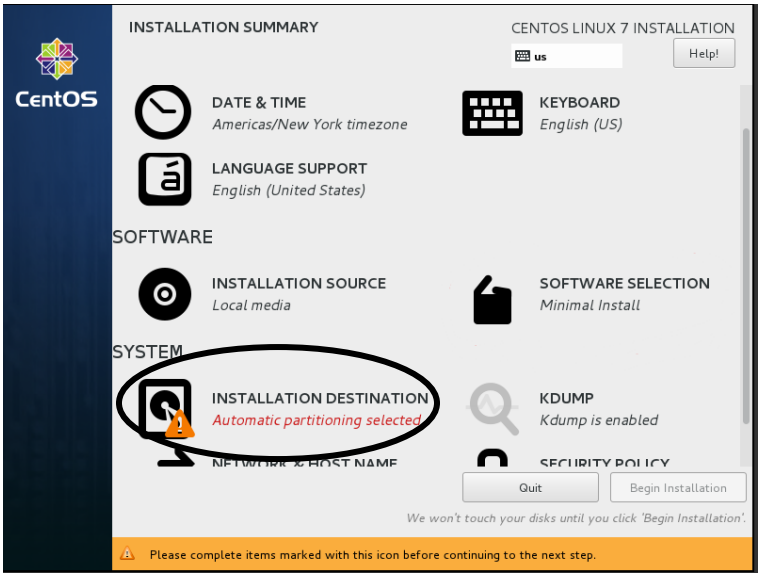


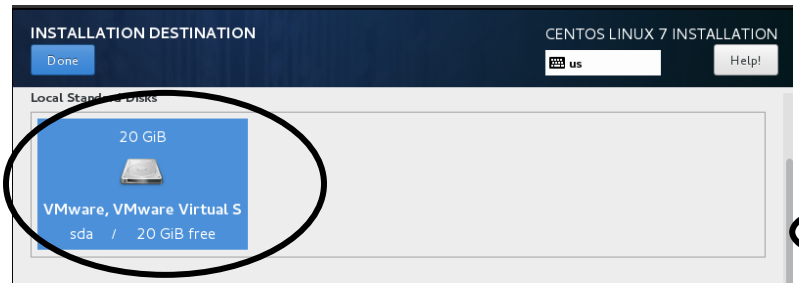


Next, I entered the address of the Ubuntu Server into PuTTy to test my SSH connection. Below is verification that I successfully logged into the terminal.

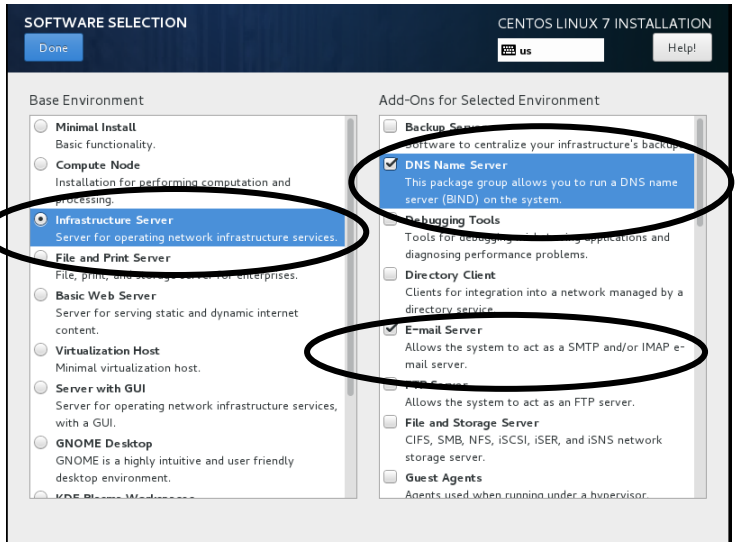
CentOS Installation:

First, load the correct ISO image into VMWare and boot the Virtual Machine. Next, select the Language (I chose English), and select Continue. Then the Installation Summary page is shown. Click on “Installation Destination” under System.



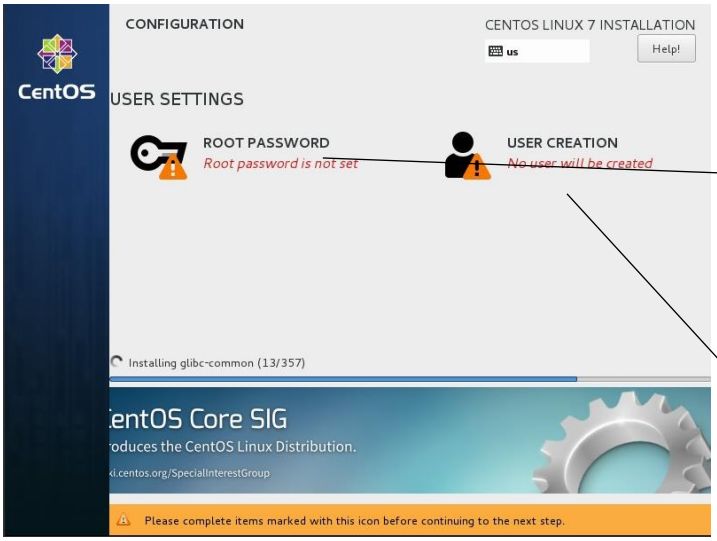


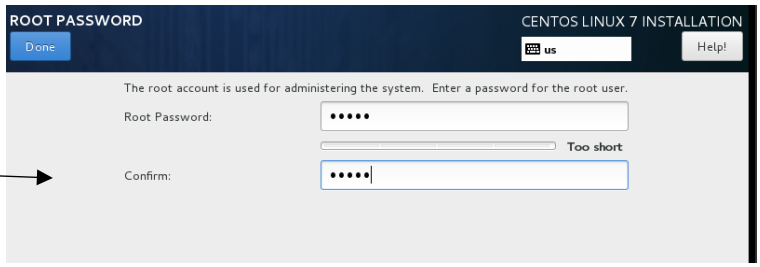
Select the 20GB VMWare disk (it should already be selected) and click “Done”.

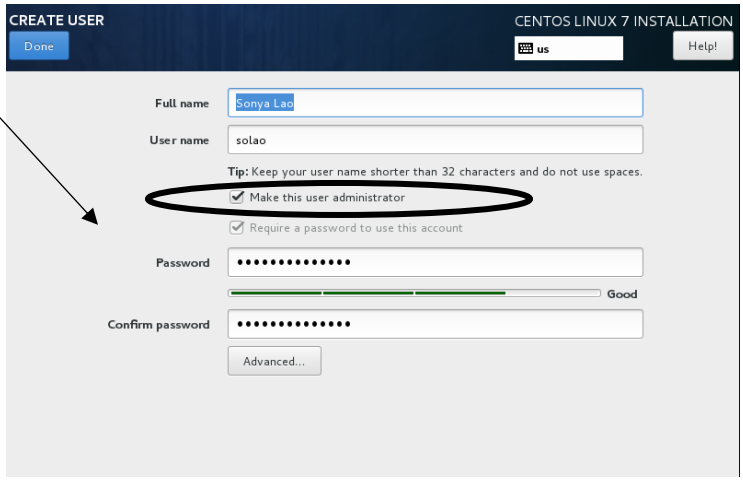


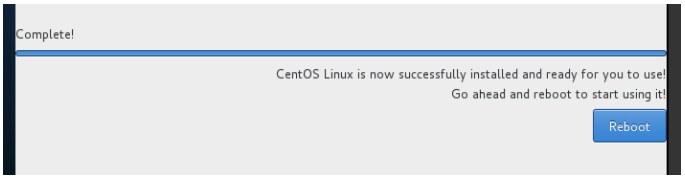
Then, return to the main Installation Summary page and click on “Software Selection”. Choose “Infrastructure Server” as the Base Environment, and check “DNS Name Server” and “E-mail Server” as add-ons for the Selected Environment. Then click “Done”, and Begin Installation.

While the installation is running, set the root password and create a user. Make yourself an administrator as well.









Reboot the VM once the installation is complete.

```
CentOS Linux 7 (Core)
Kernel 3.10.0-514.el7.x86_64 on an x86_64

localhost login: solao
Password:
[solao@localhost ~]$ su root
Password:
[root@localhost solao]#
```

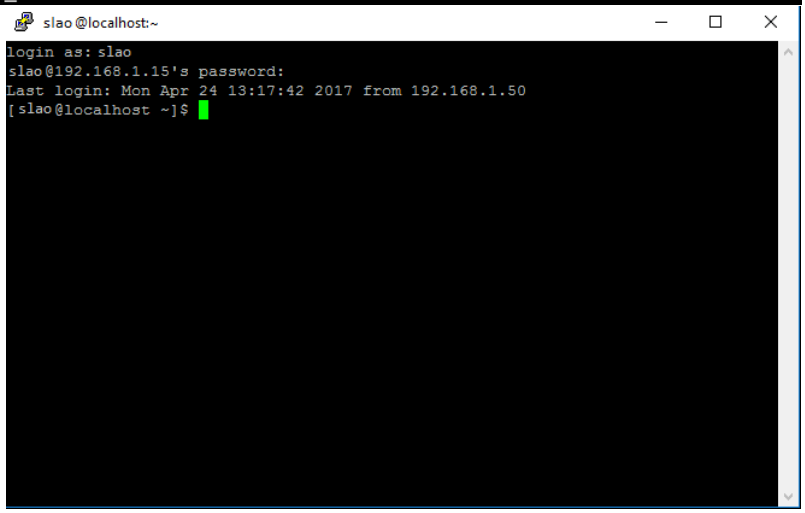
CENTOS: `service sshd status`

```
[root@localhost solao]# service sshd status
Redirecting to /bin/systemctl status  sshd.service
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2017-04-18 11:48:25 EDT; 1 weeks 2 days ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 1409 ExecStart=/usr/sbin/sshd $OPTIONS (code=exited, status=0/SUCCESS)
 Main PID: 1417 (sshd)
   CGroup: /system.slice/sshd.service
           └─1417 /usr/sbin/sshd

Apr 24 12:18:12 localhost.localdomain sshd[40035]: Failed password for invalid user slao from 192.168.1.70 port 49977 ssh2
Apr 24 12:18:19 localhost.localdomain sshd[40035]: pam_unix(sshd:auth): check pass; user unknown
Apr 24 12:18:21 localhost.localdomain sshd[40035]: Failed password for invalid user slao from 192.168.1.70 port 49977 ssh2
Apr 24 12:18:36 localhost.localdomain sshd[40035]: Connection closed by 192.168.1.70 [preauth]
Apr 24 12:19:00 localhost.localdomain sshd[40038]: Accepted password for solao from 192.168.1.70 port 49979 ssh2
Apr 24 12:20:16 localhost.localdomain sshd[40092]: Accepted password for solao from 192.168.1.3 port 49981 ssh2
Apr 24 13:18:50 localhost.localdomain unix_chkpwd[41003]: password check failed for user (solao)
Apr 24 13:18:50 localhost.localdomain sshd[41001]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.1....er=solao
Apr 24 13:18:52 localhost.localdomain sshd[41001]: Failed password for solao from 192.168.1.30 port 50164 ssh2
Apr 24 13:18:56 localhost.localdomain sshd[41001]: Connection closed by 192.168.1.30 [preauth]
Hint: Some lines were ellipsized, use -l to show in full.
[root@localhost solao]# _
```

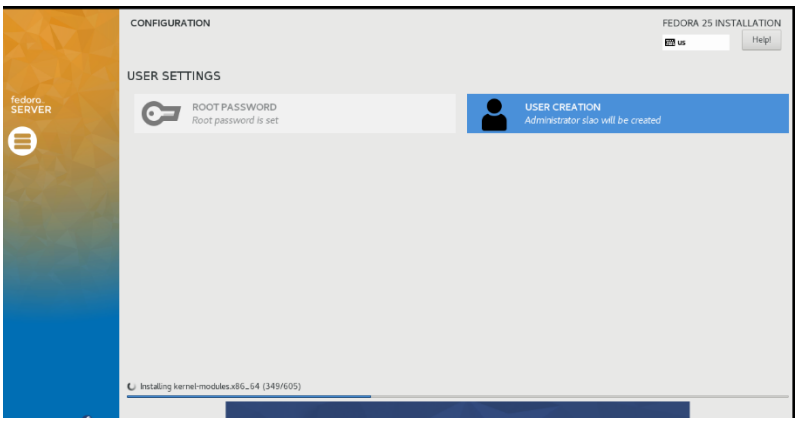
CENTOS: `ip addr`

```
[root@localhost solao]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:36:09:8f brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.10/24 brd 192.168.1.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::edfc:cd05:6cfd:bd9c/64 scope link
       valid_lft forever preferred_lft forever
```

Since CentOS already contained the SSH package, after I set the IP address of the CentOS server to 192.168.1.15, I was able to SSH into the server from my local PC.



The Fedora Server Installation was practically the same as the CentOS install. Follow the same steps to install the Fedora Server as shown above with CentOS. Create a root password and user, making yourself the admin.



FEDORA SERVER: `service sshd status`

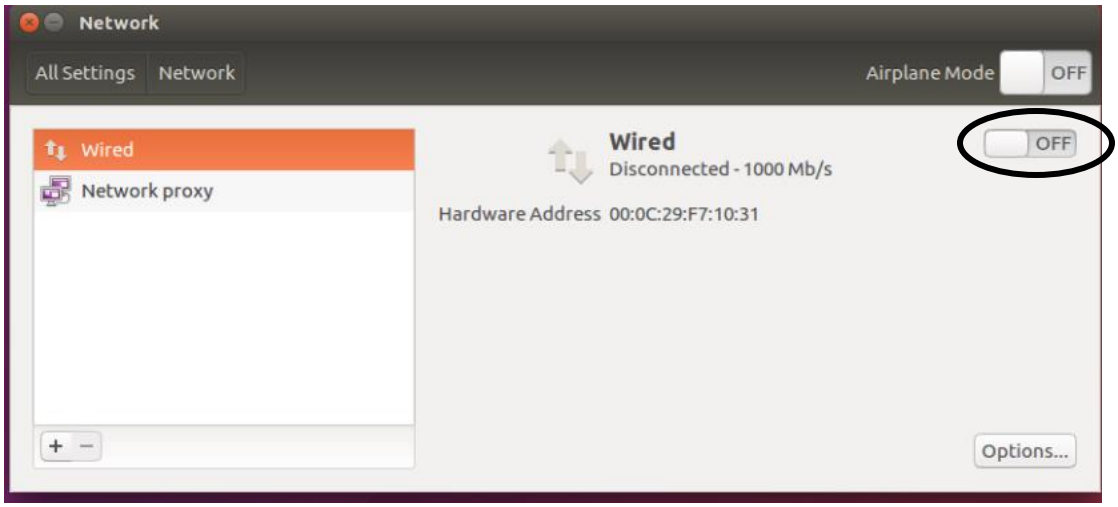
```
[root@localhost slao]# service sshd status
Redirecting to /bin/systemctl status  sshd.service
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2017-04-24 08:48:37 PDT; 1h 33min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 662 ExecStart=/usr/sbin/sshd $OPTIONS (code=exited, status=0/SUCCESS)
 Main PID: 685 (sshd)
    Tasks: 1 (limit: 19660)
   CGroup: /system.slice/sshd.service
           └─685 /usr/sbin/sshd

Apr 24 08:48:37 localhost.localdomain systemd[1]: Starting OpenSSH server daemon...
Apr 24 08:48:37 localhost.localdomain sshd[685]: Server listening on 0.0.0.0 port 22.
Apr 24 08:48:37 localhost.localdomain sshd[685]: Server listening on :: port 22.
Apr 24 08:48:37 localhost.localdomain systemd[1]: Started OpenSSH server daemon.
Apr 24 09:22:44 localhost.localdomain sshd[1133]: Accepted password for slao from 192.168.1.30 port
Apr 24 10:19:04 localhost.localdomain sshd[1259]: Accepted password for slao from 192.168.1.30 port
lines 1-17/17 (END)
```

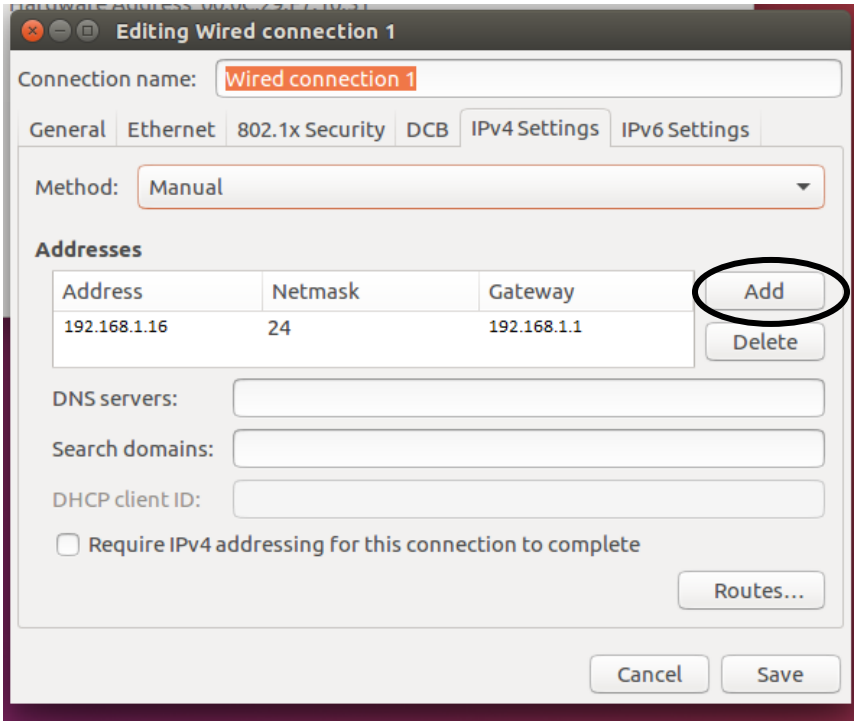
FEDORA SERVER: `ifconfig`

```
[root@localhost slao]# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.11  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::7f1:92f3:f4ef:a050  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:21:5c:10  txqueuelen 1000  (Ethernet)
        RX packets 60415  bytes 79639558 (75.9 MiB)
        RX errors 0  dropped 13  overruns 0  frame 0
        TX packets 28780  bytes 1970966 (1.8 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1  (Local Loopback)
        RX packets 247  bytes 22555 (22.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 247  bytes 22555 (22.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

For the Ubuntu Desktop, it was a very straightforward installation. However, the network is not set by default.

Enter the Settings > Network, then check if the Wired connection is On. If it is off, turn it on.





Click “Add” and type in the IP address that you want for the desktop, and set the address of the gateway.

Problems

One problem that I encountered involved openSSH on the Ubuntu Server. My initial install and set up of SSH was very smooth and I was able to enter the terminal from Putty on my host PC without any issues. However, when I attempted to access the Ubuntu Server via SSH a few days later, I was unable to do so. After running the sudo service openssh-server status command, I learned that the SSH service was no longer active. So, I tried issuing the sudo service openssh-server restart command to restart the service and re-activate it. However, nothing changed. Then I tried resetting my IP address on the server, thinking it was a networking issue, but I was still unable to connect via SSH. Next I tried to install openSSH again. This returned a multitude of errors. Then, I looked online to see if other people had similar problems. One suggestion was to completely uninstall openSSH and reinstall it. So I tried to uninstall openSSH using the service ssh stop command, but was unable to stop the service or uninstall it. Then, I tried the apt-get –purge remove openssh-server command, which also failed. Then I tried the sudo apt-get remove ssh command, which failed, as well as the sudo apt-get remove –auto-remove ssh command, which also failed. However, I kept trying to issue any of the above uninstall commands, and finally, it was successfully uninstalled. After re-installing OpenSSH Server, everything ran smoothly again.

Another problem I faced was setting up a DNS server in the CentOS server. We were originally planning on creating an email server in CentOS, but I was unable to install anything on the CentOS server. I tried to fix the problem by changing the IP address of the server, and testing its connection to the router, but both changes proved normal. For some reason, every yum install command returned a long list of "fail to reach the XYZ repository" messages. As a result, I was unable to install the necessary packages to setup a mail server. Thankfully, SSH was already preinstalled and set up, so I did not need to install SSH on the CentOS server.

Conclusion

Through the Linux lab, I learned how to make a directory, issue commands from the root, edit files using a text editor on the command line, install packages, and test connectivity. Knowing how to work in a Linux shell environment is a very useful skill and I'm thankful for the opportunity to learn a new skill. I hope to apply my server building skills and virtual machine use knowledge to practical situations in the future.