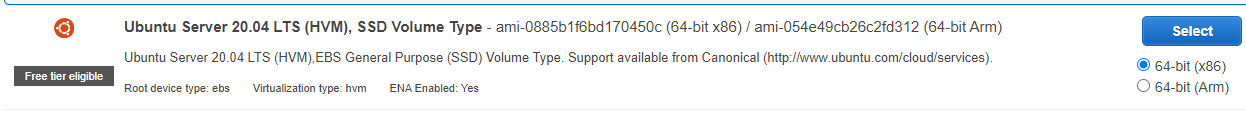
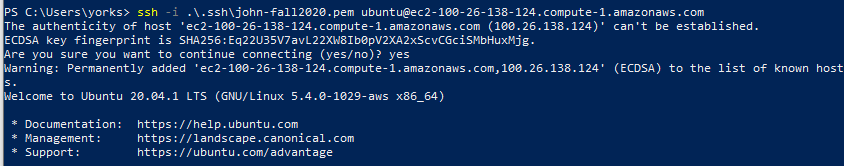
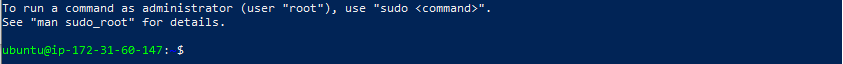
# PHP Lab 1 Web Server Installation

In order to evaluate the security of web sites and web transactions, security engineers must be able to read HTML. They must be able to create web sites in their test environments to evaluate them. One common server installation is called LAMP, for Linux Apache MySQL and PHP. Linux is the operating system, Apache is the web server, and PHP is the web server scripting language. MySQL is a database that attaches to the web server, but we won’t cover that here.

## Create an AWS Instance

Follow the procedure in the Cloud VM Lab to create an instance, EXCEPT choose an Ubuntu server image, since that is what the other students are using for their own VMs.  


Log in to the instance with SSH.

  
<snip>  


## Install Apache web server and PHP

First, we must install a web server on our Ubuntu VM. We'll use the industry standard web server, Apache or httpd, which stands for HTTP Daemon

Install httpd, the Apache web server, and the scripting engine PHP. For more details, see Module 3, Lesson 6, Installations and Conclusions, slides 2 and 3.

sudo apt update #update your app database

sudo apt install apache2 #install apache

sudo apt install php php-cli #install php and the php command line # interface

## httpd service control

You can control the httpd service with these commands.

sudo systemctl start apache2.service #start apache

sudo systemctl stop apache2.service #stop apache

sudo systemctl status apache2.service #check apache status

sudo systemctl restart apache2.service #restart apache

Check to see if your apache2 service is running. If it is you'll need to restart it to get the changes that were made by the php installation. If it is not running, start it.

Note: the older sysV commands, like sudo service apache2 start, will also work.

## Web server configuration

We'll use the default configuration for this lab. You should know that the configuration files are in /etc/apache2/. From the apache2.conf file, you can see that configuration uses multiple files:  
# It is split into several files forming the configuration hierarchy outlined

# below, all located in the /etc/apache2/ directory:

#

# /etc/apache2/

# |-- apache2.conf

# | `-- ports.conf

# |-- mods-enabled

# | |-- \*.load

# | `-- \*.conf

# |-- conf-enabled

# | `-- \*.conf

# `-- sites-enabled

# `-- \*.conf

#

# \* apache2.conf is the main configuration file (this file). It puts the pieces

# together by including all remaining configuration files when starting up the

# web server.

#

# \* ports.conf is always included from the main configuration file. It is

# supposed to determine listening ports for incoming connections which can be

# customized anytime.

#

# \* Configuration files in the mods-enabled/, conf-enabled/ and sites-enabled/

# directories contain particular configuration snippets which manage modules,

# global configuration fragments, or virtual host configurations,

# respectively.

#

# They are activated by symlinking available configuration files from their

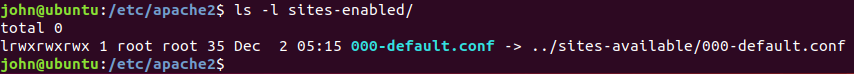
# respective \*-available/ counterparts. These should be managed by using our

# helpers a2enmod/a2dismod, a2ensite/a2dissite and a2enconf/a2disconf. See

# their respective man pages for detailed information.

The configuration files are also described in <https://help.ubuntu.com/lts/serverguide/httpd.html>.

The /etc/apache2/sites-enabled directory contains entries for the sites that are actually running on the server. Often they are symbolic links that point to sites that can be used, which are listed in /etc/apache2/sites-available.



When you are reading the configuration files remember that the ‘#’ character denotes a comment. Comments are used to document the settings. The file will often show you the default value of a setting by starting the line with a ‘#’ (comment). Non-default settings are shown without a comment.

You'll need the answers to the following questions so you can create a site.

a) What port does the server listen on? (Look in /etc/apache2/ports.conf)

b) What is the path to the document root? This is where you'll put your html files for the web site content. (Look in /etc/apache2/sites-enabled/000.default.conf. Since it is a symbolic link, you’ll really be looking at /etc/apache2/sites-available/000-default.conf.)

c) What is the path of the server root, where the configuration, error, and log files are kept? (Look in /etc/apache2/apache2.conf.)

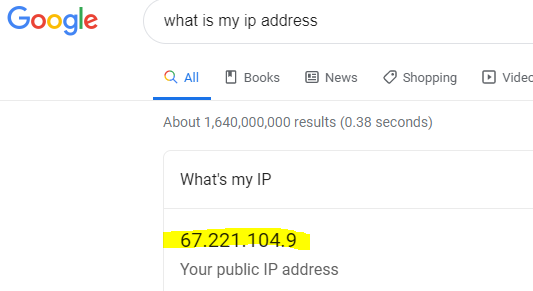
d) The directory index determines what html page the site returns as the default, when the user just asks for the root or a directory. What is the name of the page? (It may be in the apache2.conf and 000-default.conf files as the DirectoryIndex directive. If DirectoryIndex is not included in either file you can find the default value in /etc/apache2/mods-available/dir.conf.)

## Allow TCP port 80 through the AWS firewall

Right now, the web site on your new instance is not visible to the outside world. We need to allow inbound connections to the instance on TCP port 80. A note of caution: We will be purposefully making insecure configurations to demonstrate vulnerabilities, and it is not good to make them available to the world. A simple precaution we can take is to only allow connections from our computer’s IP address.

### What is my IP Address?

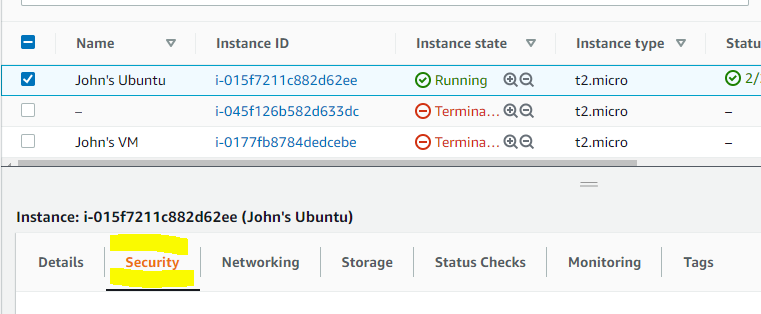
Remember that our computers have private IP addresses which are not routable on the Internet. For review, private address ranges are 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16. Our router changes our private IP address to a public IP address using Network Address Translation (NAT). We need to know our public address, since that is what AWS will see when we browse to our site. A simple way to discover our public address is to Google “what is my ip address?”



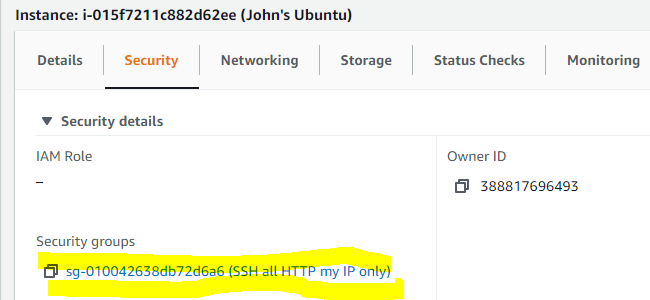
Do not use the IP address from the image. Run the Google query yourself, since your address will be different from the one shown above.

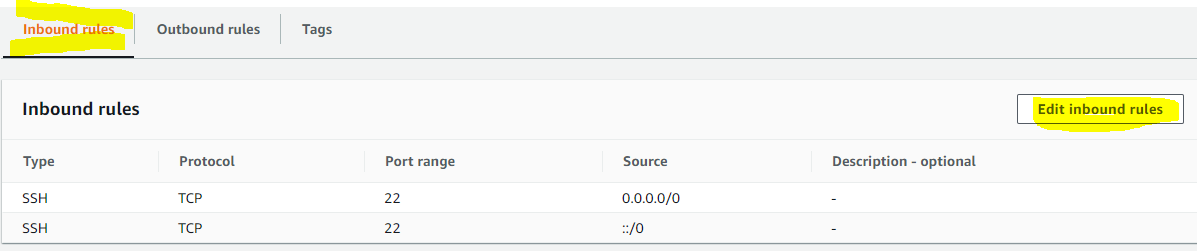
### Allow your address through the AWS firewall for TCP port 80

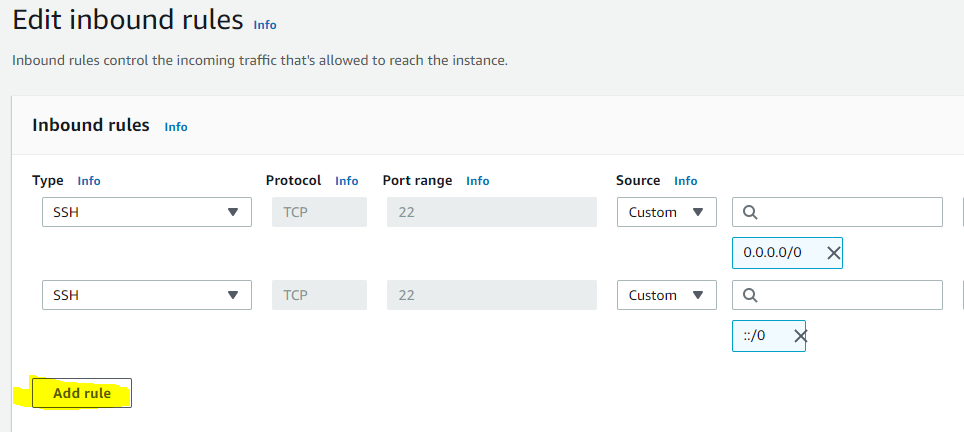
Select your instance, and then select the Security tab.

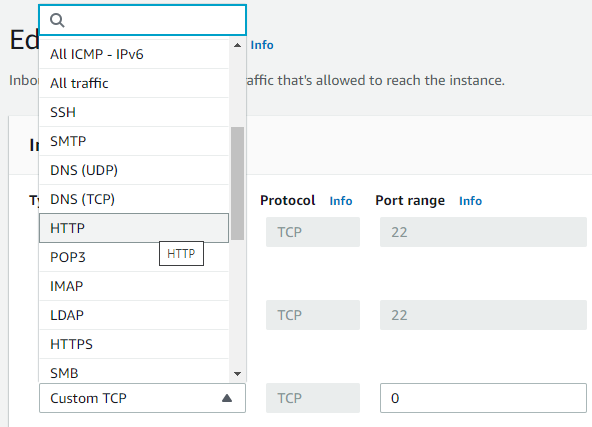


Then select your Security Group, which contains your firewall settings.

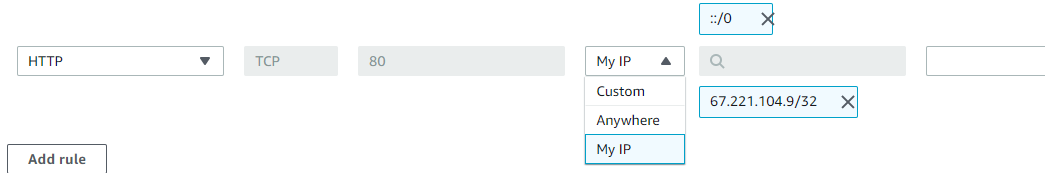


Select Inbound rules, and select Edit inbound rules.  


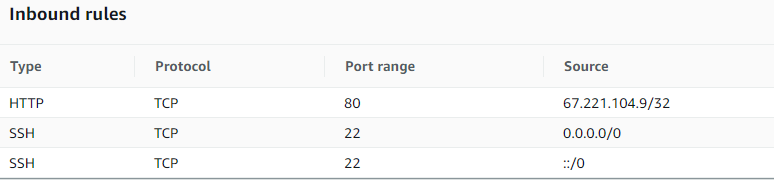
Select Add rule.  


In the left column, select HTTP.  


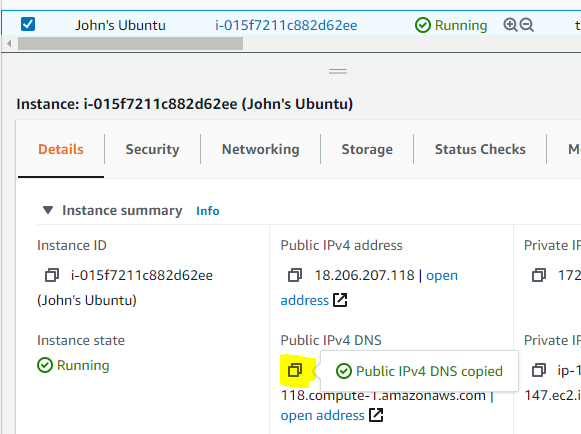
The page will fill in TCP and port 80 for you.  

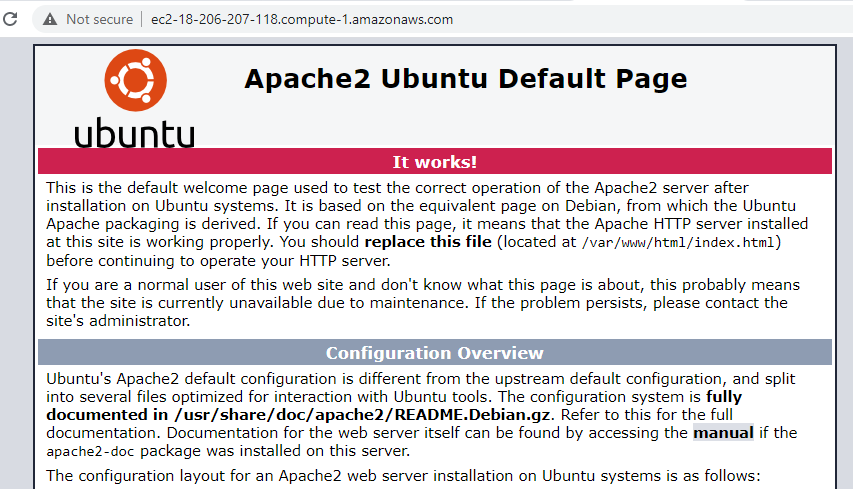

In the Source column, select My IP.  


The IP address that the site selects should be the same address you saw when you Googled “what is my IP address?”

Save the rule. When it is done, you should see something like this. Do not worry if your rules do not have the IPv6 entry for SSH.  


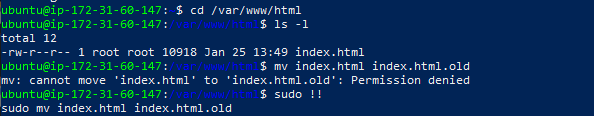
## Testing

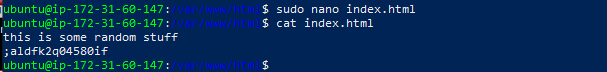
If your apache2 is running and your firewall is open, you should be able to browse to site and see the default page that Apache displays for unconfigured web sites. Copy the DNS address from your instance.  


Paste that address into your browser.  


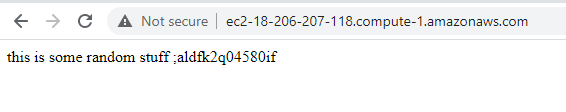
WooHoo!! It works!

You should have a file called index.html in /var/www/html. This is the default page you just browsed to. Rename it and create a new index.html file. (The file is owned by root with rw-r--r-- privileges, so you’ll have to use sudo.) Put some random stuff in your new index.html file. You don't have to use html language yet--the server will repeat any text you put in this file.





Open a browser and browse to your instance again. If you see your random stuff, the web server is working.



# Hand in

Turn in a screenshot of the browser of your host, browsing to your random stuff page on your AWS Ubuntu Apache server to demonstrate that your server and firewall are properly configured.