Purpose

The purpose of this exercise is to create multiple web sites using a single nginx web server.

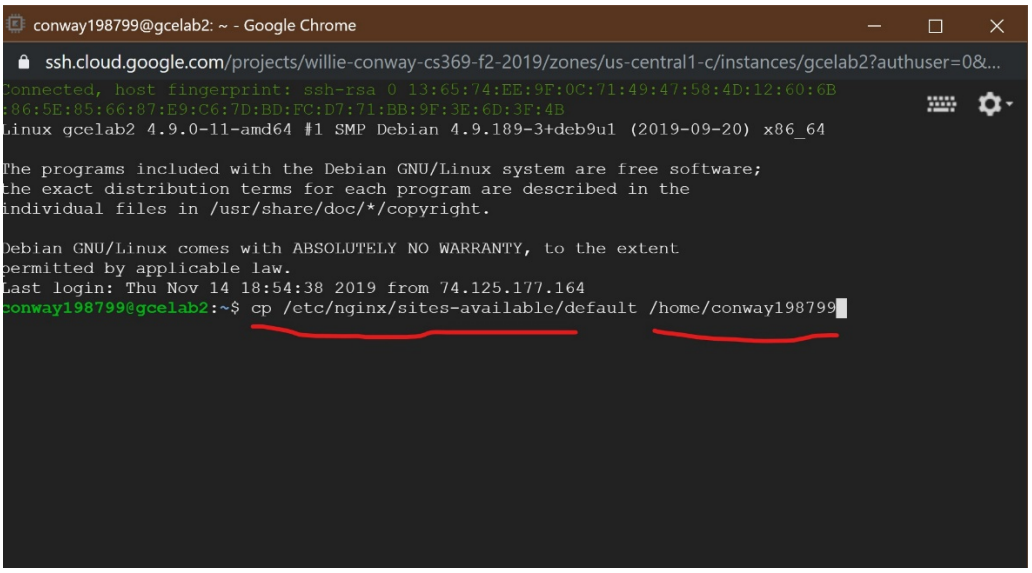
Preparation

You may need to review material about nginx configuration, the Unix tar command, and the concept of symbolic links. Remember, the nginx configuration directory is /etc/nginx and the website directory is /var/www/html. This assignment will be done entirely on your running instance and Cloud Console.

Assignment

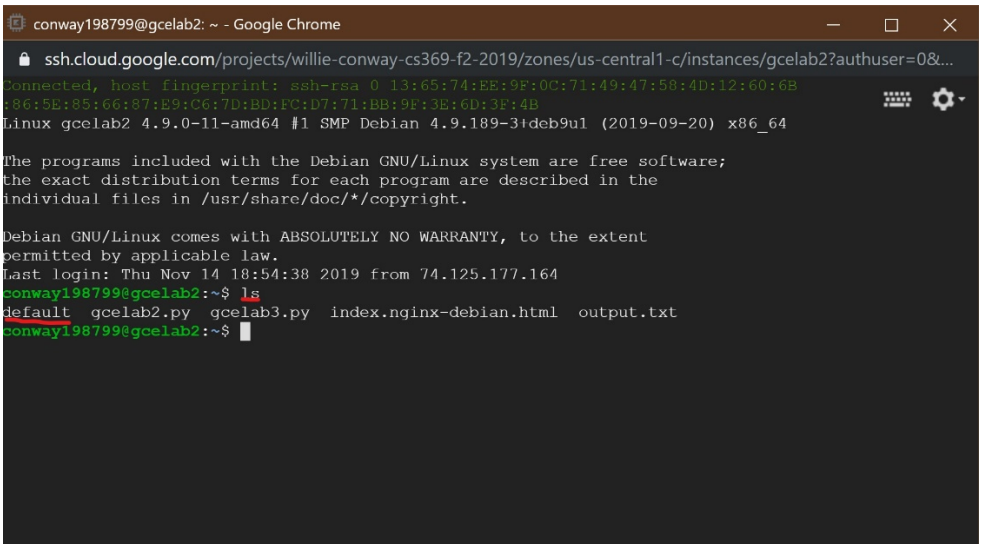
Create a site

Make a copy of the file */etc/nginx/sites-available/default*



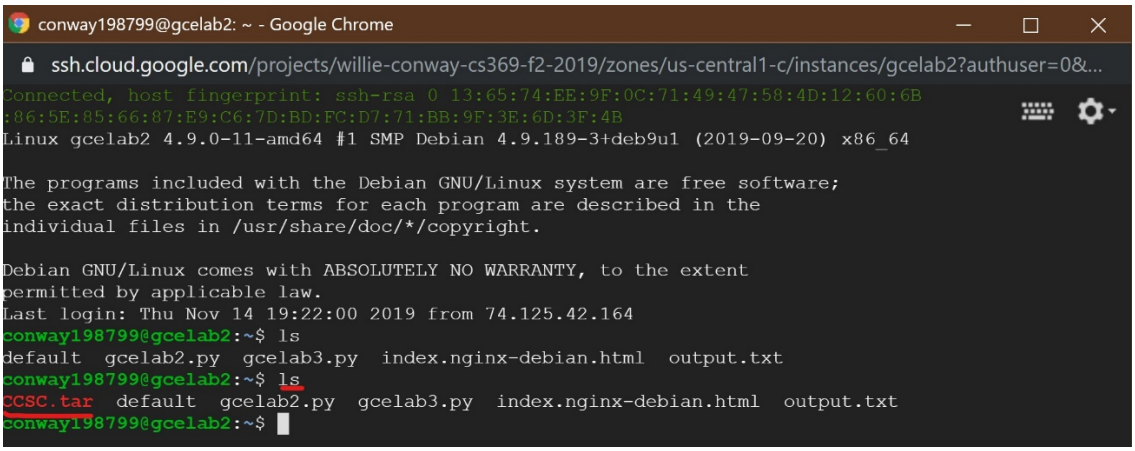
(This screenshot shows that I used the Linux cp command to copy the default file from the sites-available directory using the path /etc/nginx/sites-available to my home directory using the path /home/conway198799. I thought putting the placing the file in my home directory would be the most convenient for me to get to, just in case the configuration file is corrupted during this exercise.)

Put the copy someplace convenient. You'll want to review this in case you ever corrupt the actual configuration file.



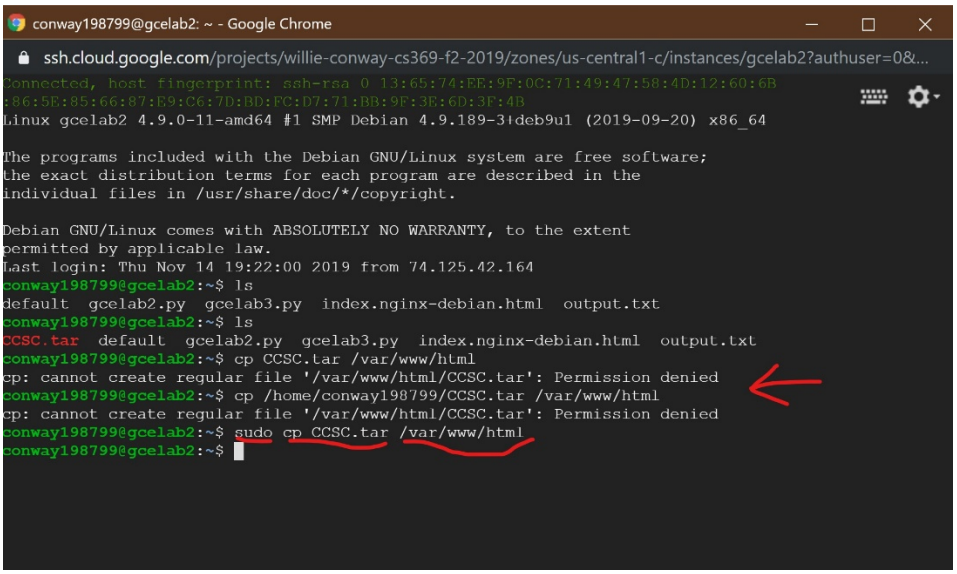
(This screenshot shows that I used the Linux ls command to assure that the default configuration file copied from the sites-available directory. When copying files over from one directory to another, I always double check. The ls command is a basic command in Linux used to List files and directories)

Download the file CCSC.tar from Canvas to your local PC.



(I used the Linux ls command once again to assure that the CCSC.tar file was uploaded fully to my home directory or GCP instance /home/conway198799. Although, there is a confirmation when uploading a file to a instance in pop-up, it doesn't hurt to double check.)

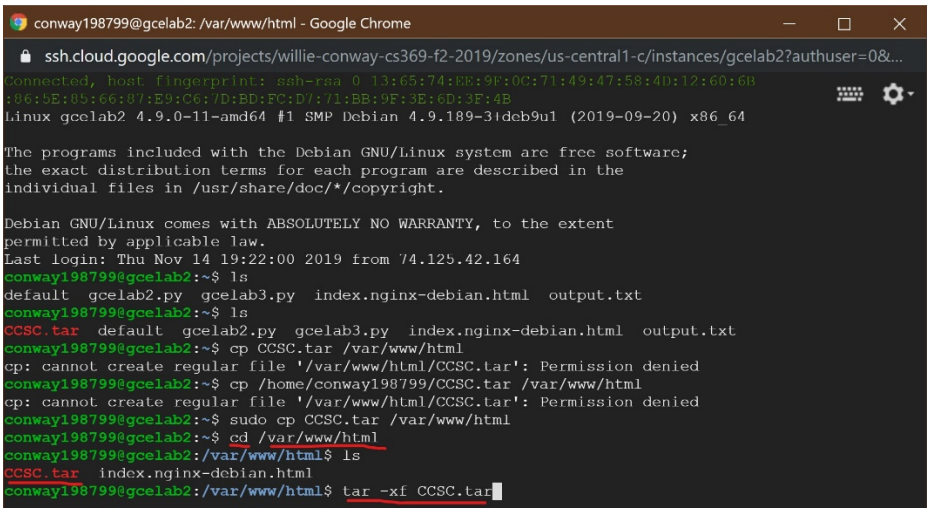
Upload this file to your GCP instance and copy it to the directory /var/www/html.



(This screenshot shows that I had to use the Linux sudo command to bypass permissions and cp command to copy the CCSC.tar file to the /var/www/html directory. On Unix-like operating systems, the sudo command ("superuser do", or "switch user, do") allows a user with proper permissions to execute a command as another user, such as the superuser.)

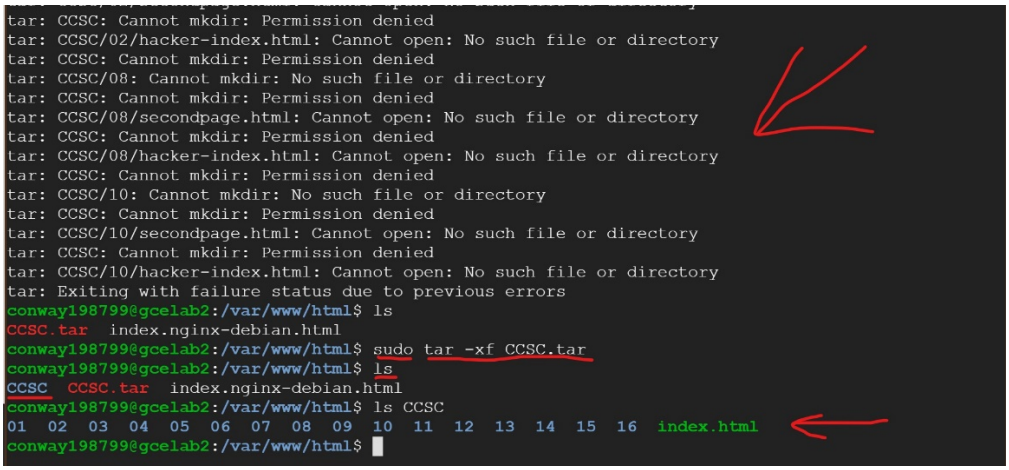
Use the tar command to extract the contents of the archive. Note that tar extracts files into the current directory, so you'll want to cd /var/www/html before you issue the command

tar -xf CCSC.tar



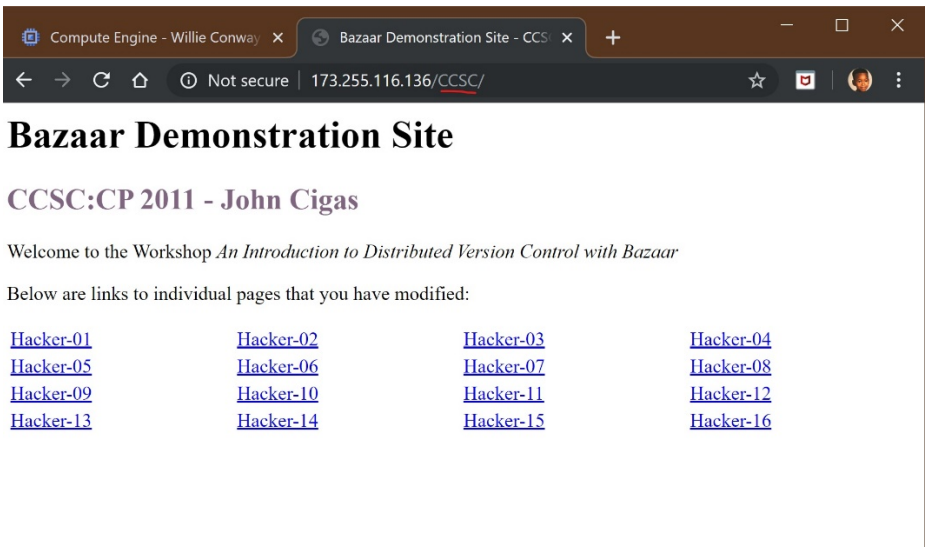
(This screenshot shows that I used the Linux cd command to change over into the /var/www/html directory, then use the Linux ls command to assure the CCSC.tar file copied over into the directory. After I got the positive identification, I then used the Linux tar command to extract the contents of the archive, CCSC.tar file. The x is for the extraction of files and the f is to use the following tar archive for the operation.)

This creates a directory called CCSC, which contains a file index.html along with 16 other directories, 01..16.



(This screenshot shows that when I tried to use the tar command to extract the CCSC.tar file, all permissions were denied. I had to use Linux sudo command again to bypass permissions when using the tar command tar -xf CCSC.tar. To assure that the files were extracted over, I used the Linux ls command to list the files in the /var/www/html directory. I then was able to notice the CCSC directory, which was created from extraction of the CCSC.tar file. I performed the Linux ls file from the last time to see which files and directories were placed in the CCSC directory. This is when I notice there were a total of 16 directories and one index.html file.)

Test your web site by appending /CCSC to the IP address in a browser window. You should see a new web page about a Bazaar workshop. Click around on the links. They should all work.

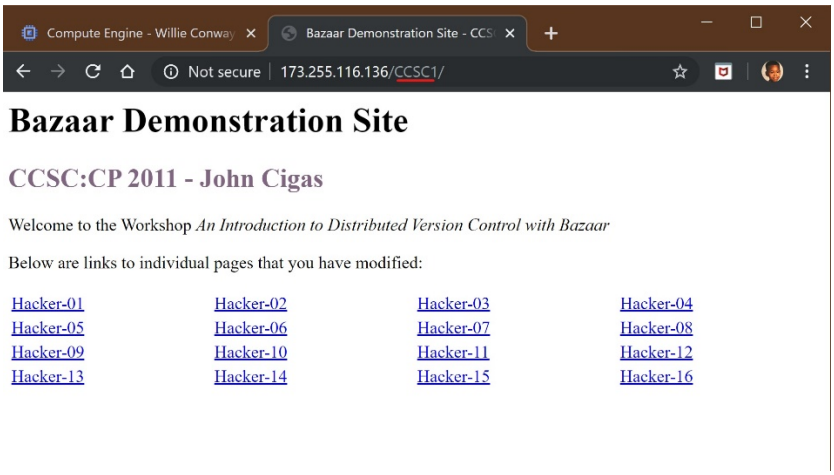


(Testing the website by appending the /CCSC to my IP address for my instance, in a browser window. This displays the Bazaar workshop with 16 links to individual pages.)

Change the name of the directory from CCSC to CCSC1. Check that you can still access the site using IP#/CCSC1 in a web browser.

```
conway198799@gcelab2:/var/www/html$ ls CCSC
01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  index.html
conway198799@gcelab2:/var/www/html$ mv CCSC CCSC1
mv: cannot move 'CCSC' to 'CCSC1': Permission denied
conway198799@gcelab2:/var/www/html$ sudo mv CCSC CCSC1
conway198799@gcelab2:/var/www/html$ ls
CCSC1  CCSC.tar  index.nginx-debian.html
conway198799@gcelab2:/var/www/html$
```

(Using Linux sudo command to bypass permissions and Linux mv command to rename or change the directory CCSC to CCSC1.)

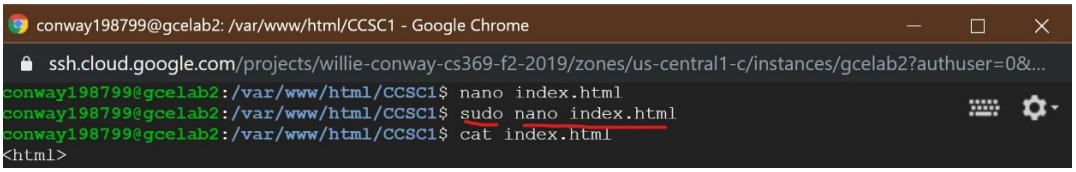


(Checking to see if I can still access the site using IP#/CCSC1 in a web browser.)

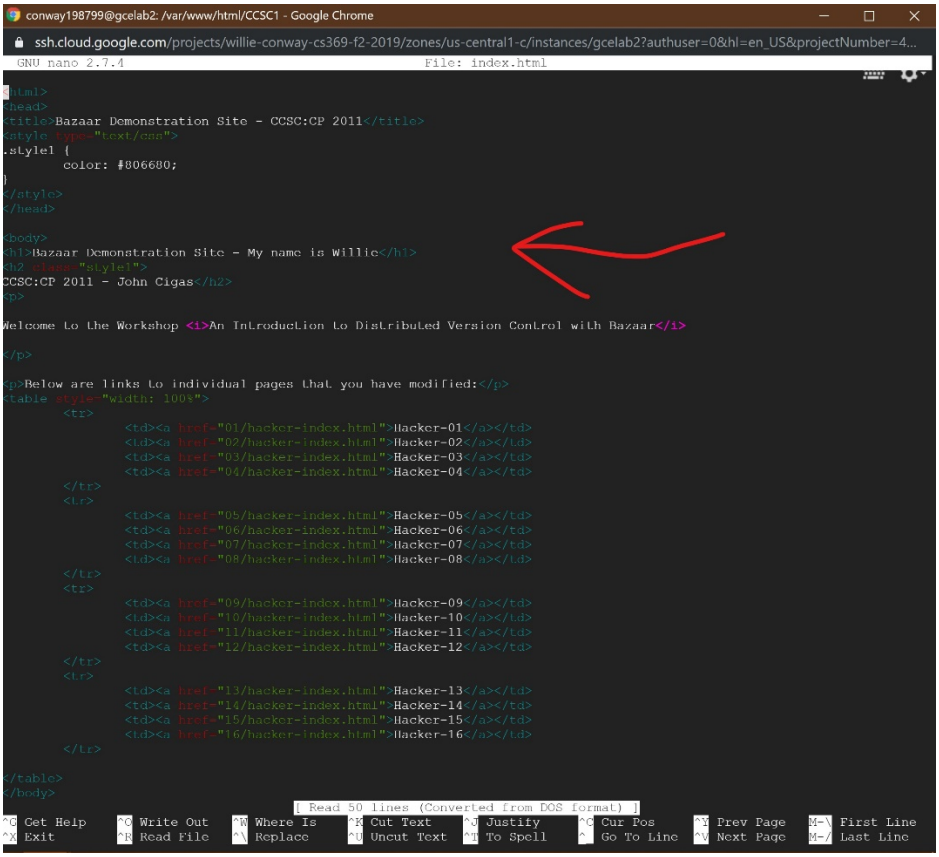
Put your name prominently on the opening web page, similar to what you did in Unit 2.

```
conway198799@gcelab2:/var/www/html$ cd /var/www/html
conway198799@gcelab2:/var/www/html$ ls
CCSC1  CCSC.tar  index.nginx-debian.html
conway198799@gcelab2:/var/www/html$ cd /var/www/html/CCSC1
conway198799@gcelab2:/var/www/html/CCSC1$ ls
01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  index.html
conway198799@gcelab2:/var/www/html/CCSC1$
```

(This screenshot shows that I used the Linux cd command to change over into the /var/www/html/CCSC1 directory to edit the index.html file.)

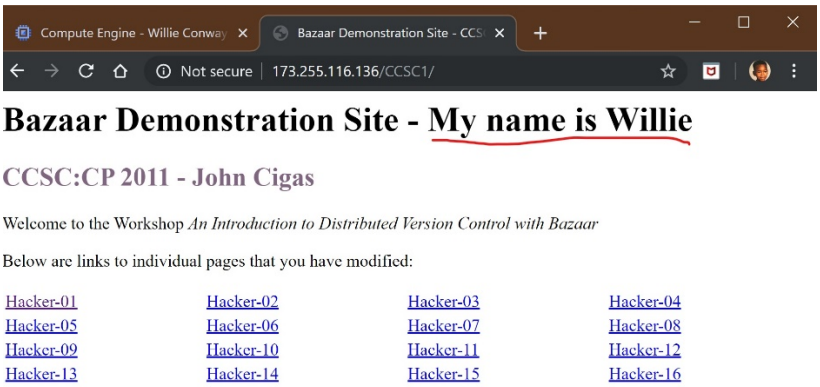


(To edit the file, I used the Linux sudo command to bypass permissions and the command nano before the index.html file to grant permissions to edit the html file in the nano editor.)



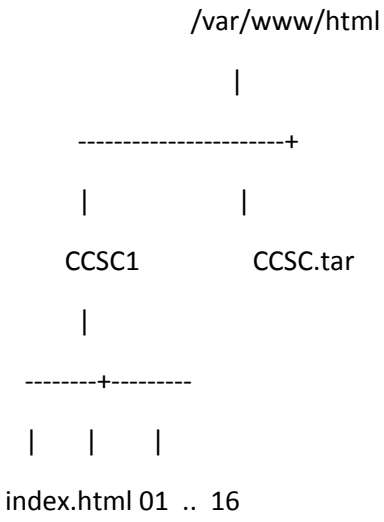
(Once in the editor I applied the same scenario as followed in Unit 2 assignment in editing a html file. I added my name to the opening title of the webpage.)

Take a screen shot showing your browser window with the URL, along with your name on the displayed page.



(Checking to see if I can still access the site using IP#/CCSC1 in a web browser. This time, with my name displayed on the opening webpage of the Bazaar workshop. The title reads "Bazaar Demonstration Site - My name is Willie.")

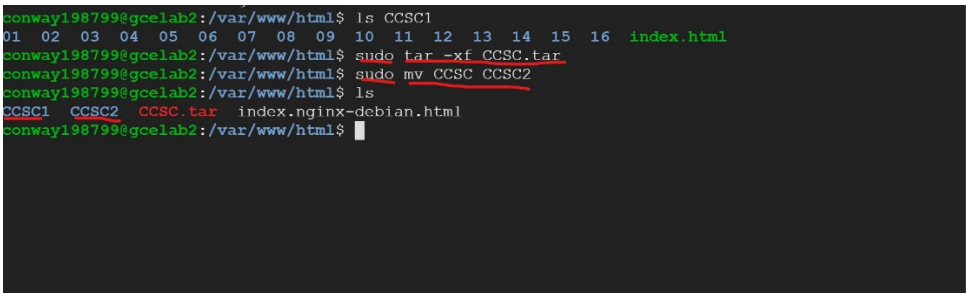
When you have finished this step, the directory tree should look like this.



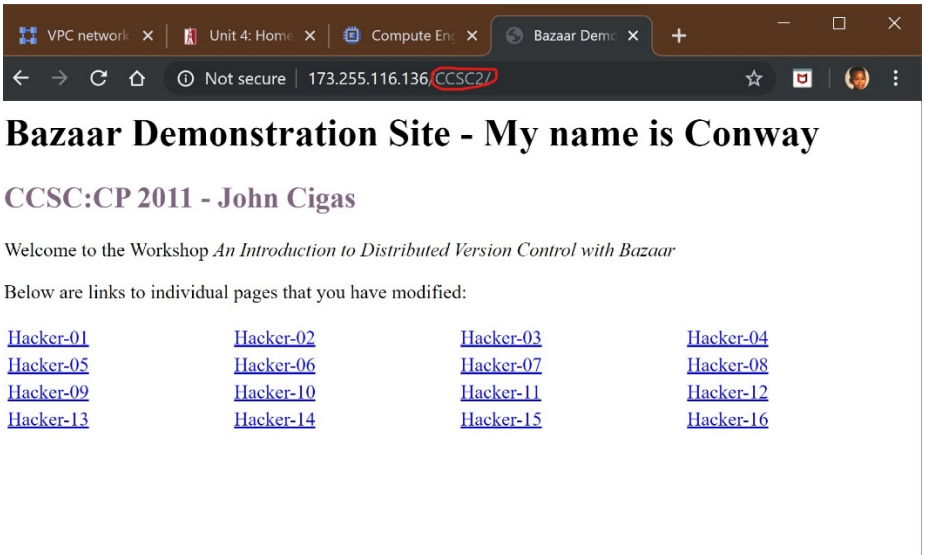
Create another site

This means creating another directory tree, another configuration file, another symbolic link, and finally, adding a firewall rule to enable access to your new sites.

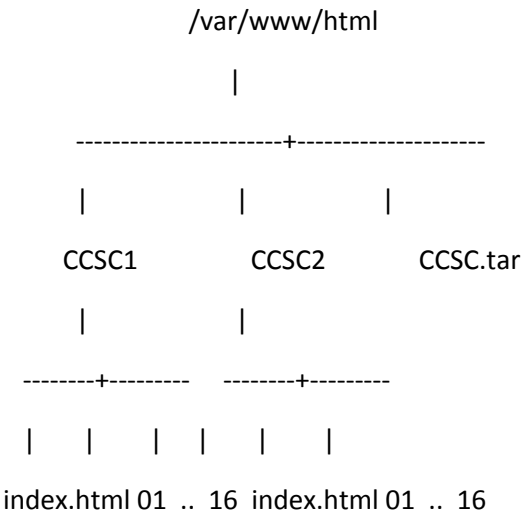
Extract the files in CCSC.tar again, then change the name of the directory to CCSC2. Youl'll have two directories each with a copy of the same files. You'll find it helpful to change the text on the in each index.html to indicate which site is which.



(To create another site, I had to apply the same fundamentals that I applied in the first half of the exercise. Since I was already in the /var/www/html directory, I went ahead and extracted the CCSC.tar file again, using Linux sudo and tar command. Then, I proceeded to rename the extracted CCSC directory to CCSC2, using the Linux sudo and mv command. After, I did a Linux ls command check to assure that I had two CCSC directories.)

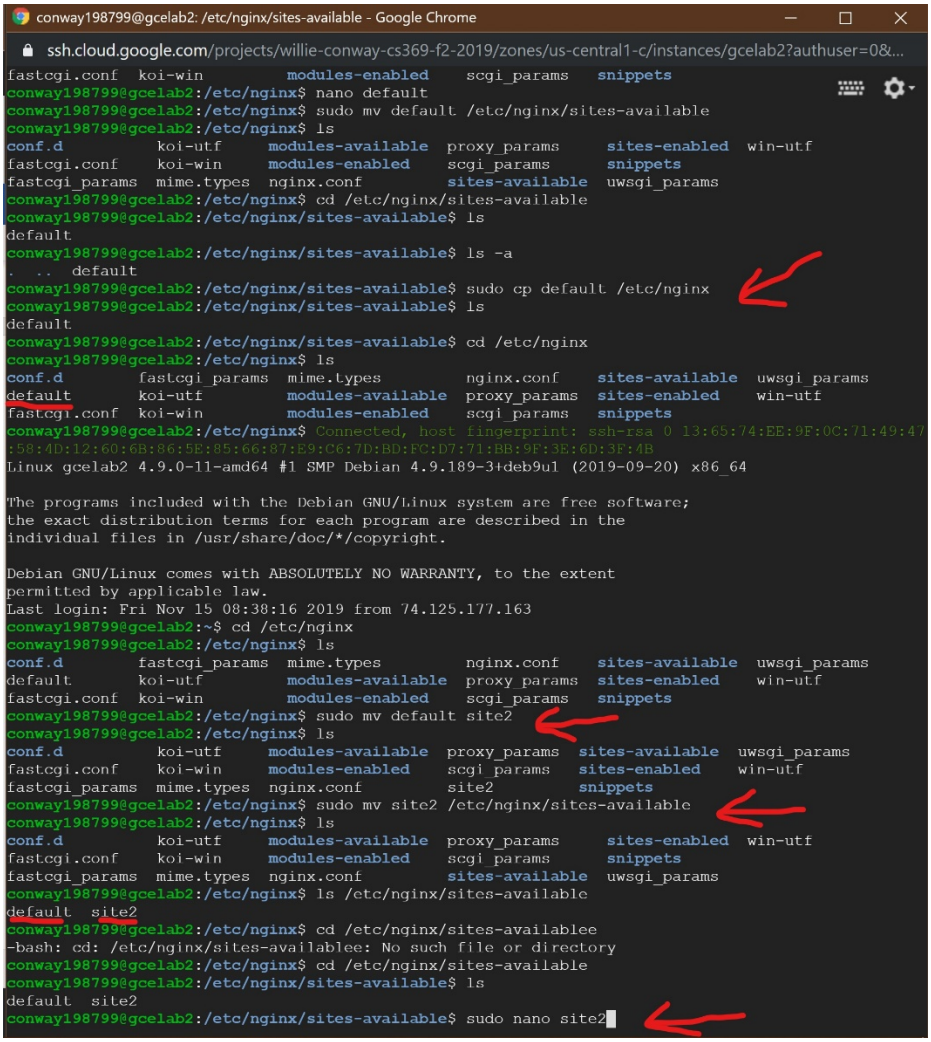


(Checking to see if I can still access the site using IP#/CCSC2 in a web browser. This time, with my name displayed on the opening webpage of the Bazaar workshop. The title reads "Bazaar Demonstration Site - My name is Conway." I edited the CCSC2 file, with my last name to distribute the difference between the two sites.)



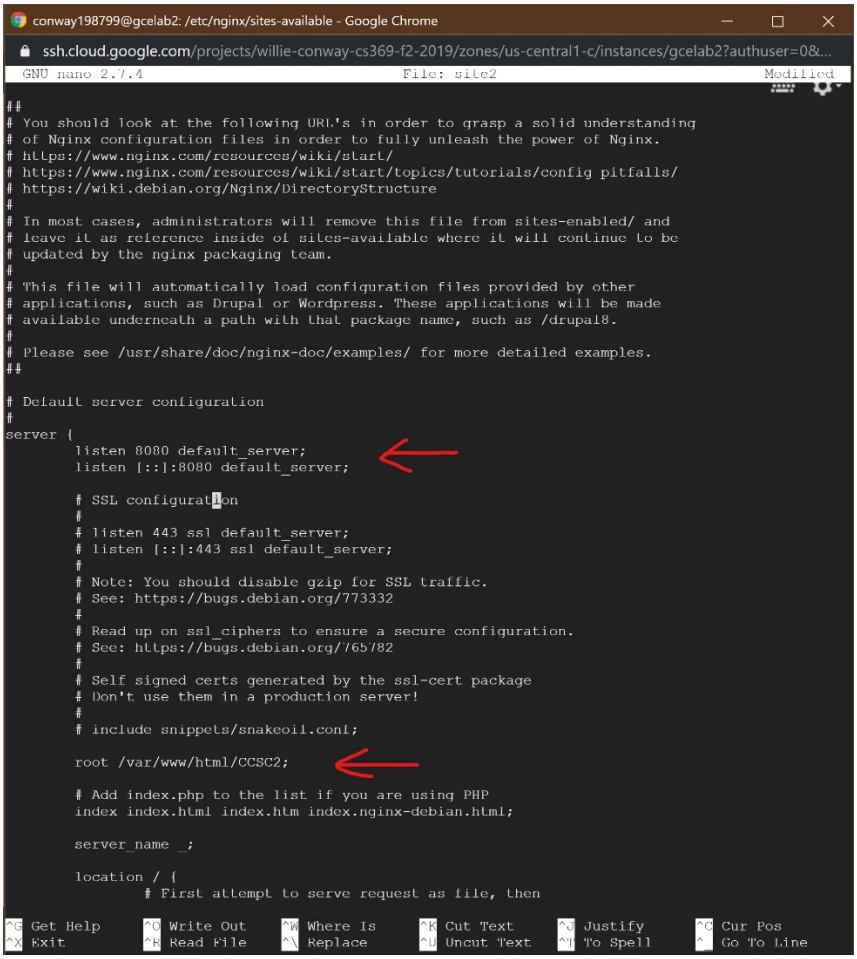
This time though, you're going to access your web site though a different mechanism. Normally you would use a different domain name to access the new site, but instead we're just going to use a different port number.

Make a copy of the default configuration file in the sites-available directory. In the copy you'll need to edit the root element to point to the CCSC2 directory you just created, and you'll need to edit the listen elements to use port 8080.



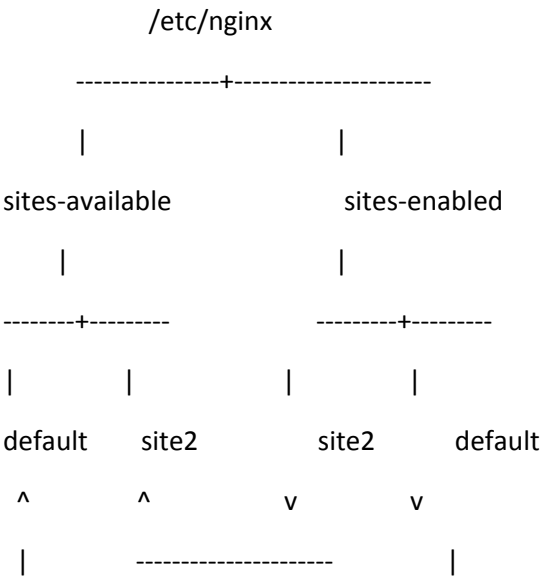
(In this screenshot it looks like I took a longer route in copying the default configuration file then a shorter one. That was due to overthinking the situation. I used the Linux sudo and cp command to bypass permissions and copy the default configuration file from the /etc/nginx/sites-available directory to the /etc/nginx directory. During this point I re-read the instructions and reviewed the diagram and that's when I notice that I needed to rename the default configuration file to distribute it from the regular file, since I was going to be editing the contents. So, I used Linux sudo and mv command to bypass permissions and rename, my copy default configuration file from default to site2 (as shown in the directory tree diagram).)

(Then, I used Linux sudo and mv command to move the site2 configuration file to the /etc/nginx/sites-available directory. Next, I performed a Linux ls command check for the /etc/nginx/sites-available directory, and that's when I notice the default configuration file and the site2 configuration file were in the same directory. So, this means all I have to do now, is use the Linux sudo command and the nano editor to edit the site2 configuration file.)



(Editing the root element to point to the CCSC2 directory and the listen elements to point to port 8080.)

Create a symbolic link in sites-enabled to point back to the new configuration file. Your nginx directory should now look like this



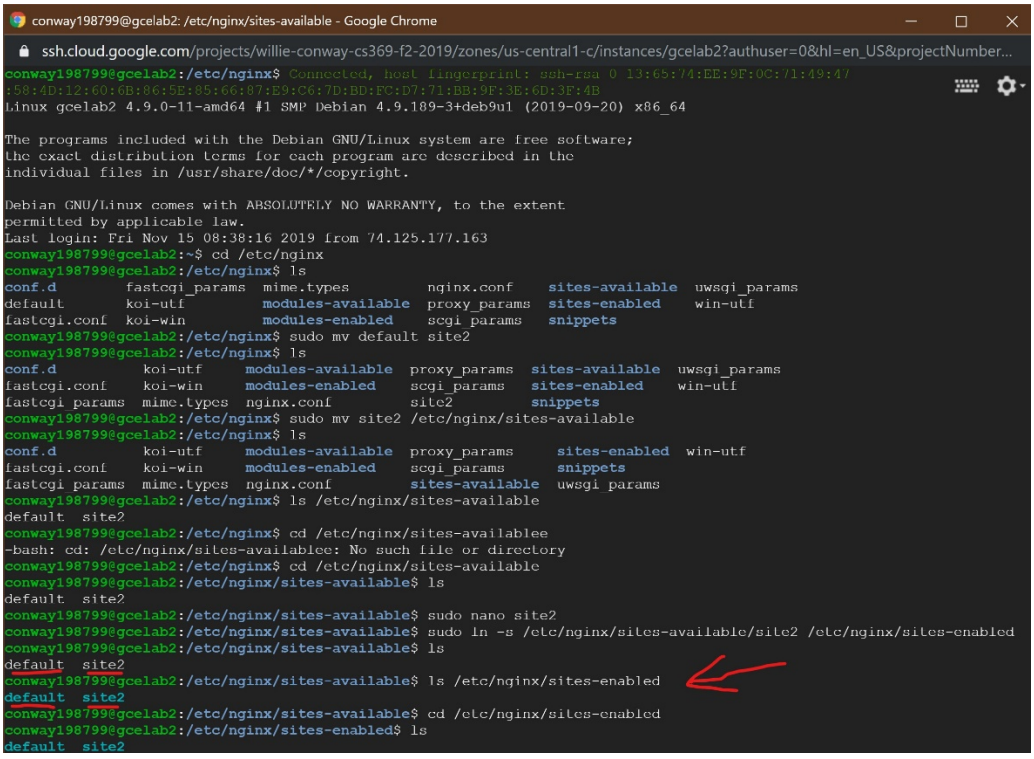
|----------|

```
conway198799@gcelab2:/etc/nginx$ Connected, host fingerprint: ssh-rsa 0 13:65:74:EE:9F:0C:71:49:47
:58:4D:12:60:6B:86:5E:85:66:87:E9:C6:7D:BD:FC:D7:71:BB:9F:3E:6D:3F:4B
Linux gcelab2 4.9.0-11-amd64 #1 SMP Debian 4.9.189-3+deb9u1 (2019-09-20) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Nov 15 08:38:16 2019 from 74.125.177.163
conway198799@gcelab2:~$ cd /etc/nginx
conway198799@gcelab2:/etc/nginx$ ls
conf.d        fastcgi_params  mime.types         nginx.conf    sites-available  uwsgi_params
default       koi-utf         modules-available  proxy_params  sites-enabled    win-utf
fastcgi.conf  koi-win         modules-enabled    scgi_params   snippets
conway198799@gcelab2:/etc/nginx$ sudo mv default site2
conway198799@gcelab2:/etc/nginx$ ls
conf.d          koi-utf     modules-available  proxy_params  sites-available  uwsgi_params
fastcgi.conf    koi-win     modules-enabled    scgi_params   sites-enabled    win-ulf
fastcgi_params  mime.types  nginx.conf         site2         snippets
conway198799@gcelab2:/etc/nginx$ sudo mv site2 /etc/nginx/sites-available
conway198799@gcelab2:/etc/nginx$ ls
conf.d          koi-ulf     modules-available  proxy_params     sites-enabled  win-ulf
fastcgi.conf    koi-win     modules-enabled    scgi_params      snippets
fastcgi_params  mime.types  nginx.conf         sites-available  uwsgi_params
conway198799@gcelab2:/etc/nginx$ ls /etc/nginx/sites-available
default  site2
conway198799@gcelab2:/etc/nginx$ cd /etc/nginx/sites-availablee
-bash: cd: /etc/nginx/sites-availablee: No such file or directory
conway198799@gcelab2:/etc/nginx$ cd /etc/nginx/sites-available
conway198799@gcelab2:/etc/nginx/sites-available$ ls
default  site2
conway198799@gcelab2:/etc/nginx/sites-available$ sudo nano site2
conway198799@gcelab2:/etc/nginx/sites-available$ sudo ln -s /etc/nginx/sites-available/site2 /etc/nginx/sites-enabled
```

(While in the /etc/nginx/sites-available directory, I used the Linux sudo and symbolic/soft link command to create a link in the /etc/nginx/sites-enabled directory to point back to the new configuration file.)



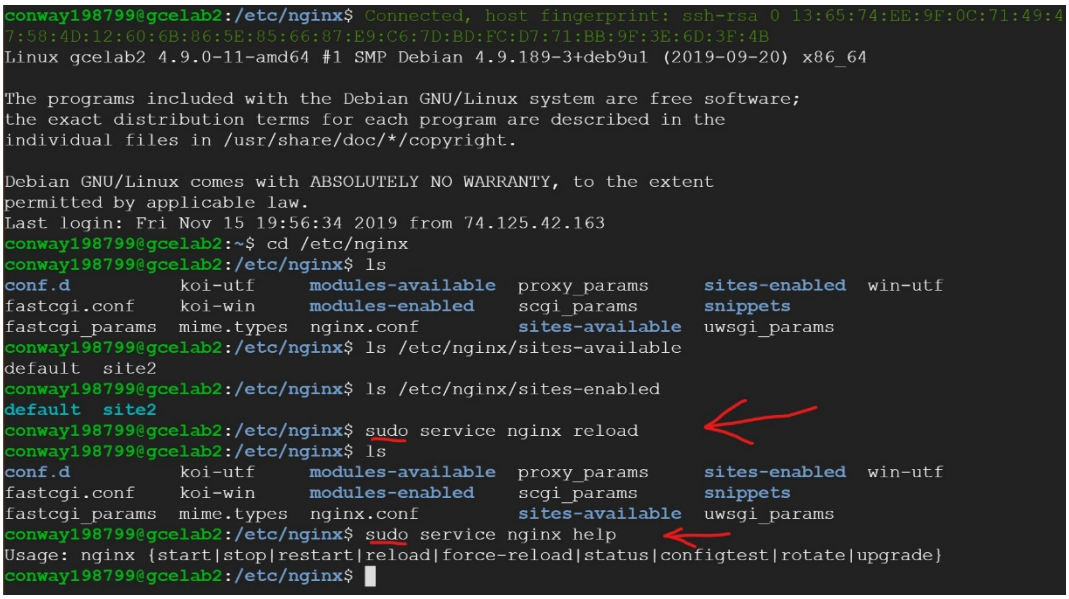
(Doing a Linux ls command check for the /etc/nginx/sites-available and /etc/nginx/sites-enabled directory to assure the symbolic link was a success.)

Adding files to the directory tree doesn't affect the nginx server, but changing a configuration file does. You'll need to tell nginx to read its configuration files again. Use

sudo service nginx reload

You can find all the commands available by using

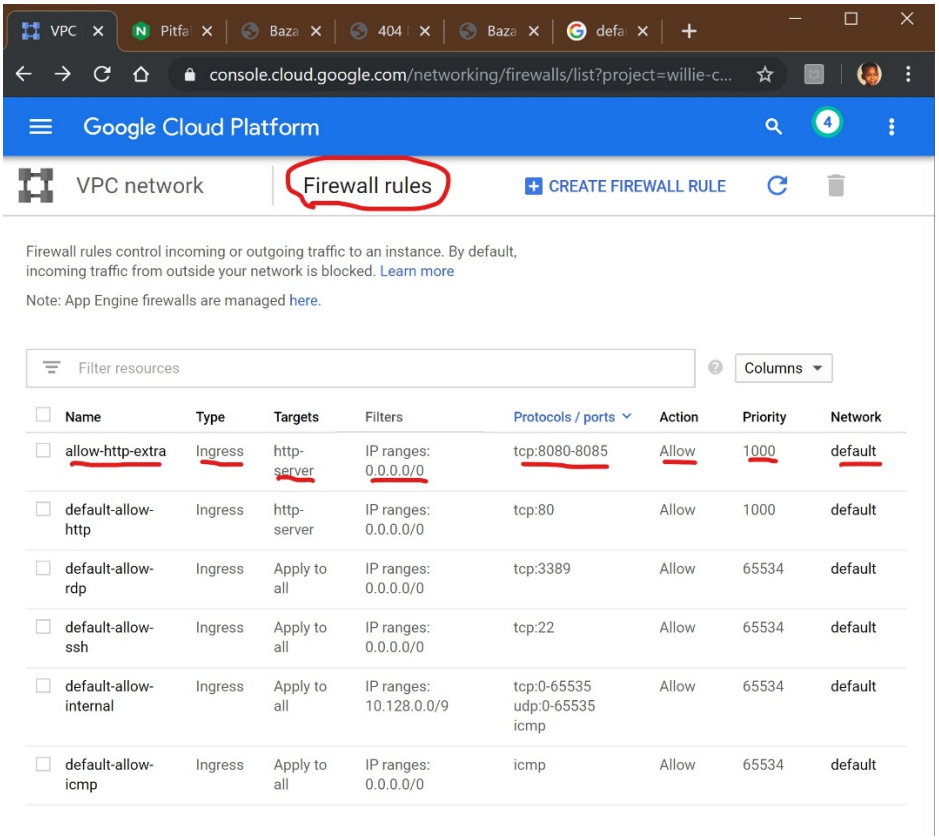
sudo service nginx help



(Using the Linux sudo command to bypass permisions and use the service nginx reload command to tell nginx to read its configuration files again. Then using sudo service nginx help to find all commands available.)

Firewall rules

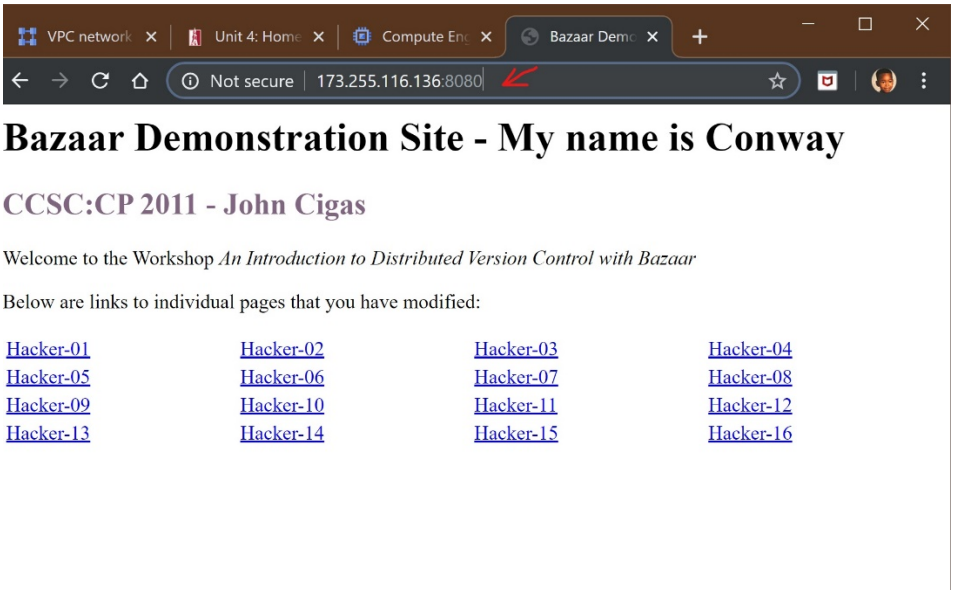
To create a firewall rule, you'll have to use the GCP console. See the attached image for where this is and what the settings should be. You want to enable a small range of ports, like 8080-8085, since you will be hosting several different web sites. Make sure that the Target is http-server. Note that the default rule is already there.



(Creating a firewall rule to enable ports tcp:8080-8085. Also, listing all the following filter resources that was listed in the screenshot for this part of the exercise.)

Once the firewall is opened, you should be able to get to your new website with a URL like http://1.2.3.4:8080, using your IP address and site port number. You'll know it's the correct site, because you have already modified the index.html file to show this is second site.

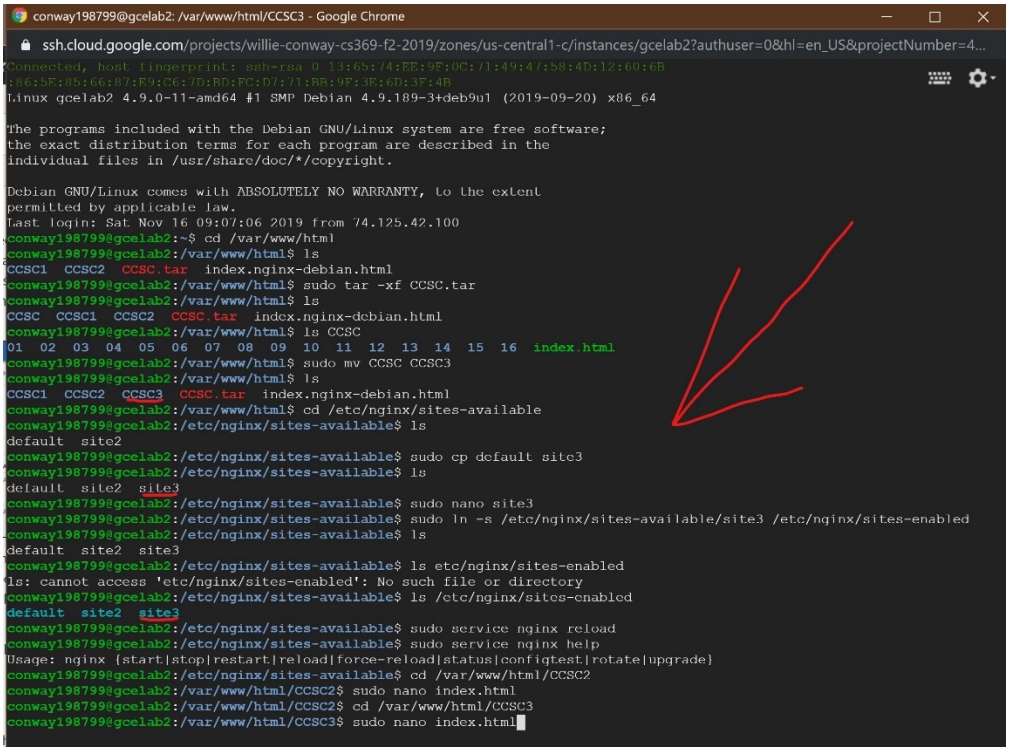
Take a screen shot showing your browser window with the URL, along with the modified front page.



(Screenshot of modified front page, IP address and site(CCSC2) port number (8080).)

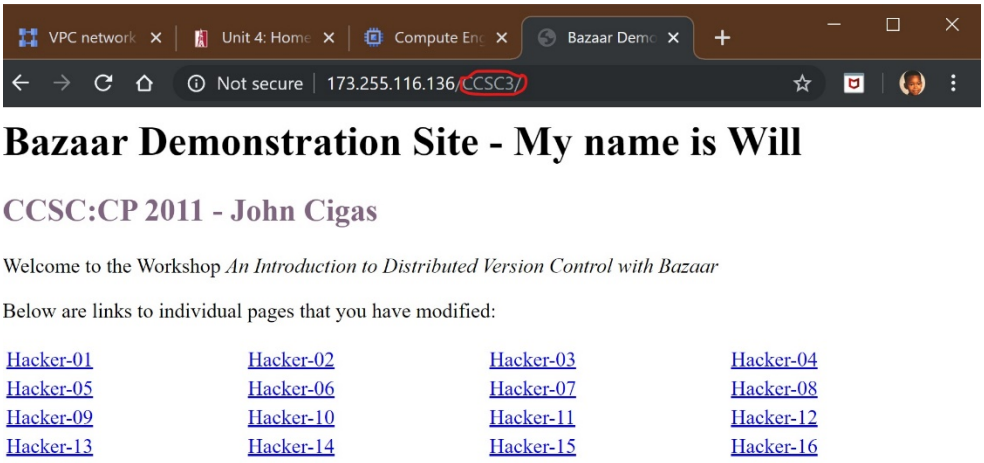
Create a third site

Now repeat this all one more time, just so you get to practice the steps. The good news is that you don't have to create another firewall rule, as long as you use a port number in the right range.

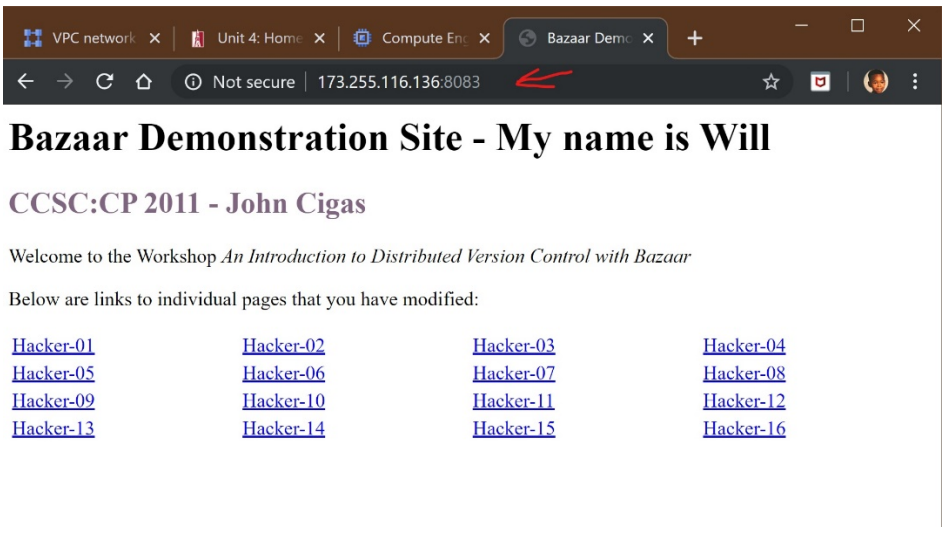


(This screenshot just shows that I applied the same fundamentals in creating a third site, as I did when I created a second site. I just named the directory CCSC3, the configuration file site3, edit the site3 configuration file and index.html file to distribute the site from the other sites.)

Take a screen shot showing your browser window with the URL, along with the modified front page.



(Checking to see if I can still access the site using IP#/CCSC3 in a web browser. I edited the index.html file, to change the opening webpage title to read "Bazaar Demonstration Site – My name is Will," to distribute the site from the previous sites.)



(Screenshot of modified front page, IP address and site(CCSC3) port number (8083). Since I already created a firewall rule using tcp:8080-8085, I didn't have to make another firewall rule. I just decided to choose a port number that I knew would fit between the range, which was 8083.)

When you are finished, you will be able to access 3 web sites, each using a different port number on your server.

Reflection

Add your screenshots to a Word document, along with a brief summary of the steps you needed to take to get the web sites to work. This will be a reminder for how to repeat this assignment at a later time. Also include the answers to the following questions:

In a sentence or two, what did you learn? *In this assignment I learned how to create multiple websites using the nginx web server. It was my first time utilizing the tar command to extract files into a directory from a tar file. From last week's exercise I learned how to upload files from my local computer to my GCP instance. In this exercise I got to experience more with my IP address with utilizing different tcp ports to locate different websites. Other than the first week's tutorial, this was my first time creating a symbolic link from one directory to another, then testing it out on a site. The last time I worked with using tcp ports, it was in my computer networking class, so doing this exercise kind of open up more thoughts that I have about ports.*

In a sentence or two, what did you like about this project? *I like the fact that this project taught me something unique about ports. This was my first time understanding more about VPC networks and how they work. I never had any experience making a firewall rule until now. I thought that was cool, understanding the different resources, and how they all contribute to connection with your server. Also, I got to learn to different Linux commands for nginx.*

In a sentence or two, what did you find confusing or would like to see done differently regarding this project?

I didn't have much confusion with this assignment other then knowing that the diagram is a visual understanding of a directory tree. I was lost at first until I realize that the files are alphabetized when displayed with the Linux ls command. I never edited a configuration file before, so this is something I can put in my notes, that I understand how to do.