Lab 2 - Hosting Nginx

Objectives:

- Introducing Nginx
- Installing Nginx
- Creating and Hosting a Custom HTML Page in Nginx
- Questions

Topic 1: Introducing Nginx

Nginx was designed as an open source web server, but has been adapted for use as a reverse proxy, HTTP cache, and load balancer. It was initially designed to handle large amounts of requests and deliver high performance as a web server, which allows it to operate on minimal resources. It provides compatibility across multiple platforms as well for a variety of users.

Some common use cases of Nginx include the following:

- Reverse proxy with caching
- IPV6
- Load balancing
- FastCGI support with caching
- WebSockets
- Handling of static files, index files, and auto-indexing
- TLS/SSL with SNI

This brings us to Apache, an alternative web hosting service to Nginx. Apache is also open source, compatible with all platforms, and offers many built-in services. Apache also has the unique benefit of allowing the end-user to modify the source code to suit their individual needs.

You might be wondering, why are we deploying Nginx over Apache then?

While Apache is in fact the most widely used web server and has many benefits, Nginx is used more often for high-traffic sites. Sites and services that operate using Nginx include Netflix, Hulu, NASA, CloudFlare, and GitHub.

If you're interested in Apache and would like to explore more, including hosting with Docker, visit their site here: https://httpd.apache.org/

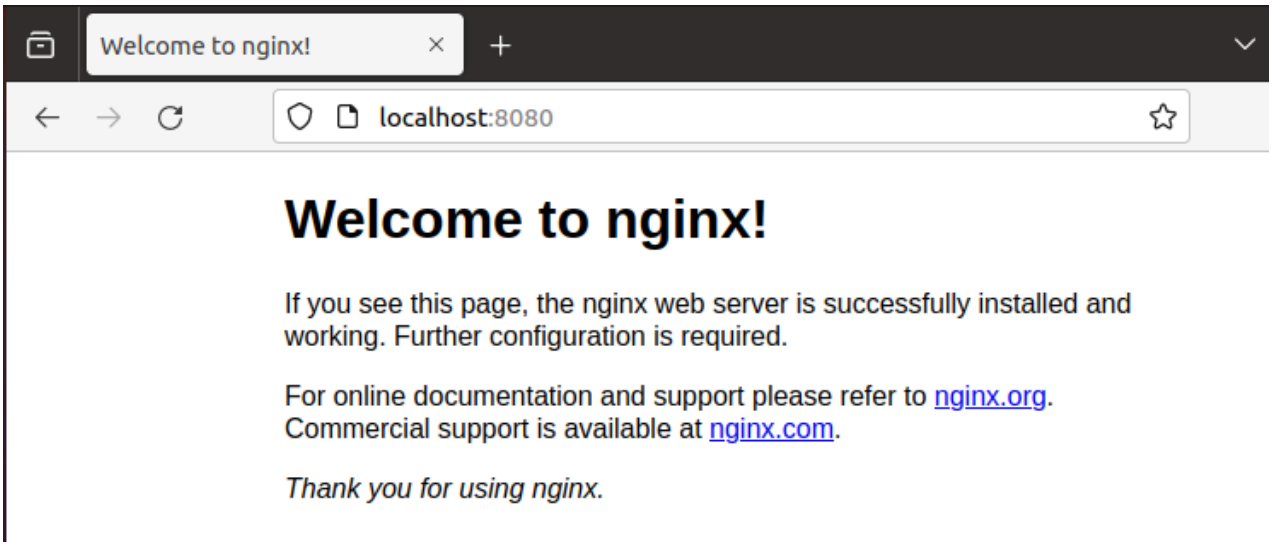
Topic 2: Docker Running on VirtualBox Nginx

---

sudo docker run -it --rm -d -p 8080:80 --name web nginx

```
bgoss@csi5700-lab4:~$ sudo docker run -it --rm -d -p 8080:80 --name web nginx
[sudo] password for bgoss:
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
c57ee5000d61: Pull complete
9b0163235c08: Pull complete
f24a6f652778: Pull complete
9f3589a5fc50: Pull complete
f0bd99a47d4a: Pull complete
398157bc5c51: Pull complete
1ef1c1a36ec2: Pull complete
Digest: sha256:5f44022eab9198d75939d9eaa5341bc077eca16fa51d4ef32d33f1bd4c8cbe7d
Status: Downloaded newer image for nginx:latest
ecbeab8c20a0136d929c19ceb2b6599ddaa2f603b6a17846c49360570c92cae3
```

If you got into any web browser and type http://localhost:8080 then you should see the following.



This is the default web page offered by Nginx.
Another way to check on your docker containers is by running **sudo docker ps**

```
bgoss@csi5700-lab4:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS                                   NAMES
9112e6ae76a4   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.0:8080->80/tcp, :::8080->80/tcp   web
```

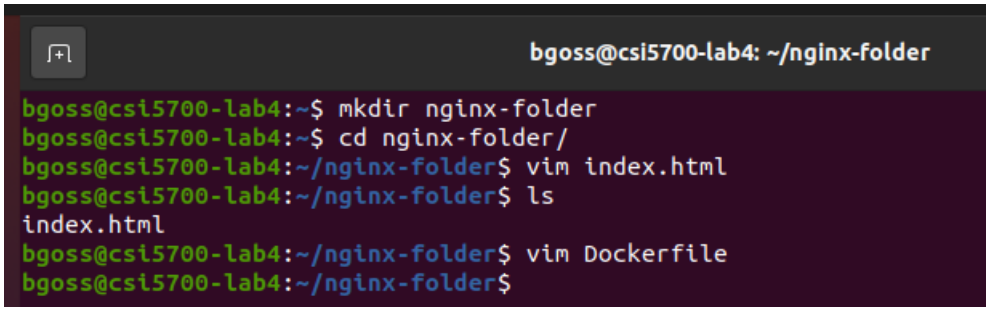
Topic 3: Creating a custom html page to put in Nginx

You'll need to kill your docker container before you move onto this next step. If you don't, you'll have two applications attempting to use the same port.

sudo docker kill web

The easiest way to get a custom html page is by building your own Dockerfile.

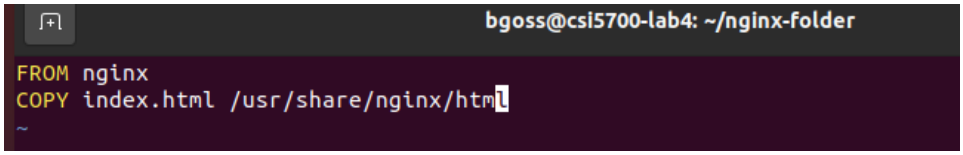
- Create a folder
- Make an index.html file and have it include the following text:
  - Oakland University Bridge 2 Cyber
  - This is <your_name>'s Unit 3 Lab 2 submission.
- Create a file called **Dockerfile**



- Inside that file type the following:

  FROM nginx

  COPY index.html /usr/share/nginx/html



  - This pulls the latest docker version of Nginx and copies your index. html file into the default html file location for the Nginx container.
- So at this point you should have a folder with two items inside: the Dockerfile and the index.html file, and your current working directory should be that new folder
- Build your Docker image from that Dockerfile:

  sudo docker build -t <new_image_name> .
  - The "**.**" is important here & as you can see below, camelcase is not allowed.

```
bgoss@csi5700-lab4:~/nginx-folder$ sudo docker build -t myNginx .
[+] Building 0.0s (0/0)                                                 docker:default
ERROR: invalid tag "myNginx": repository name must be lowercase
bgoss@csi5700-lab4:~/nginx-folder$ sudo docker build -t my-nginx .
[+] Building 0.8s (7/7) FINISHED                                        docker:default
 => [internal] load build definition from Dockerfile                             0.0s
 => => transferring dockerfile: 86B                                              0.0s
 => [internal] load metadata for docker.io/library/nginx:latest                  0.6s
 => [internal] load .dockerignore                                                0.0s
 => => transferring context: 2B                                                  0.0s
 => [internal] load build context                                                0.0s
 => => transferring context: 140B                                                0.0s
 => CACHED [1/2] FROM docker.io/library/nginx:latest@sha256:84c52dfd55c467e12ef85cad6a252  0.0s
 => [2/2] COPY index.html /usr/share/nginx/html                                  0.0s
 => exporting to image                                                           0.0s
 => => exporting layers                                                          0.0s
 => => writing image sha256:05a6c2ef1ae71d90105bf509b202c63d98db7cd6bdcc69f11b0041e881a31  0.0s
 => => naming to docker.io/library/my-nginx                                      0.0s
```

- You can check your new creation by:
  sudo docker image ls

```
bgoss@csi5700-lab4:~/nginx-folder$ sudo docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED              SIZE
my-nginx     latest    05a6c2ef1ae7   About a minute ago   187MB
try6         latest    6be776ed6954   4 days ago           187MB
```

- Now you can run the image:
  sudo docker run -d --name <name-container> -p 8080:80 <new_image_name>

```
bgoss@csi5700-lab4:~/nginx-folder$ sudo docker run -d --name myNgnix -p 8080:80 my-nginx
f5f226f81c0820893ea54bf108c37748b8757b11c44040fa9a53268d1a0b7664
```

- To check if your container is now running, you can run:
  sudo docker ps

```
bgoss@csi5700-lab4:~/nginx-folder$ sudo docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS
                          NAMES
f5f226f81c08   my-nginx   "/docker-entrypoint.…"   52 seconds ago   Up 51 seconds   0.0.0.0:8080
->80/tcp, :::8080->80/tcp   myNgnix
```

- Lastly, check your http://localhost:8080 to make sure that the nginx is up and running your custom index.html.

**Tip: Sometimes your web browser might cache localhost. Make sure to empty your cache if you keep getting the default Nginx page instead of the new html file you created.

Topic 4: Questions

1. What version of nginx are you pulling by using **FROM ngnix**?
2. When you run **docker run -d --name <name-container> -p 8080:80 <new_image_name>,** what does the -d stand for?
3. Why is the "**.**" so important in the command **sudo docker build -t <new_image_name> . ?**
4. What is the difference between running **sudo docker ps** and running the command **ls**?
5. What is happening when you run **COPY index.html /usr/share/nginx/html**?
6. What do you find when you run **sudo docker exec <name-container> ls**?
7. What is your use case for nginx in this scenario?
8. Can you edit files within the docker container? Why or why not?
9. Kill your container again. Do you get the same output as when you ran **sudo docker exec <name-containter> ls**? Why or why not?
10. Re-run all of these commands but in a different directory. Do you get different results? Why or why not?
11. Is this a stateful or stateless docker container?

****Include a screenshot of your localhost:8080 with the customized index.html page for credit.**