Getting Started with Nginx

Introduction

In this series of recipes you will become familair with how to confgure nginx as a web-server as well as a proxy server.

These instrucitons are for use on VMs either locally or remotely.

Recipe 1.0: 📒 Installing Nginx on a Virtual Machine and Testing it 📒

Date: November 26, 2022
Author: Johanna Blumenthal
Category: VM, Nginx, Web - Server Installation
Keywords: Nginx, web-server
Hardware: Desktop/Laptop PC, Mac or Pi
Software: Mac OS, Virtual Box with ubuntu server installed
Preliminary:

1. Installation guide for Ubuntu Server OS: https://help.ubuntu.com/lts/serverguide/index.html and/OR\
2. Installation guide for Virtual box: https://www.virtualbox.org/manual/

Overview

Nginx is a high perfomance web-server developed by Igor Sysoev in the late 90s/early 2000s as a way to allow for a high number of silmultanoues connections to the server. Since its wide-spread release in 2004, Nginx has grown to become the most popular web-server, serving 34% of all websites.

5 steps

1. Open and login to VM running ubuntu 22.1 (you can ssh in if you have that set up already, but make sure you have super user priviledges)

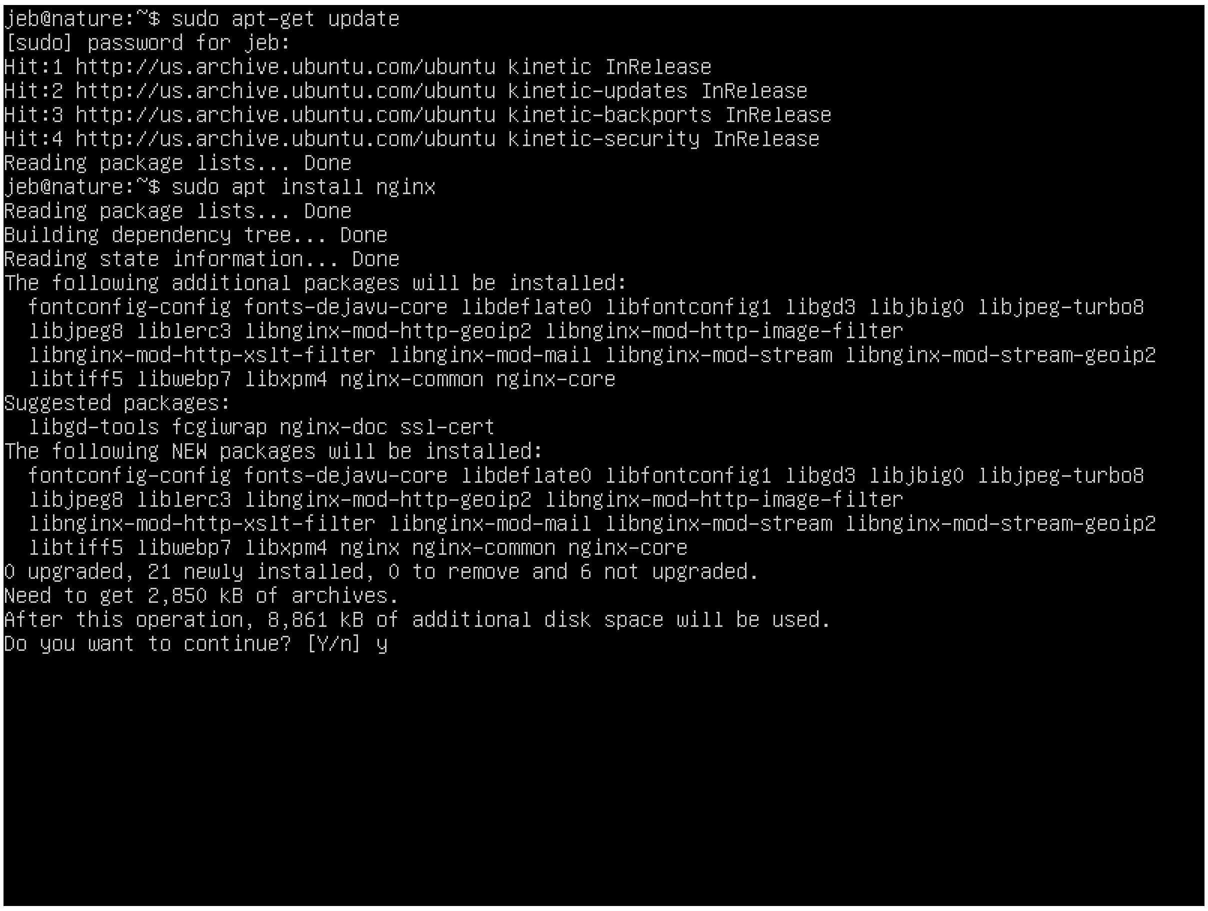
2. Update the system to make sure it is up-to-date and ready to install.

```
sudo apt-get update
```

3. Install Nginx

```
sudo apt install nginx
```

The output should look something like this. Answer y.

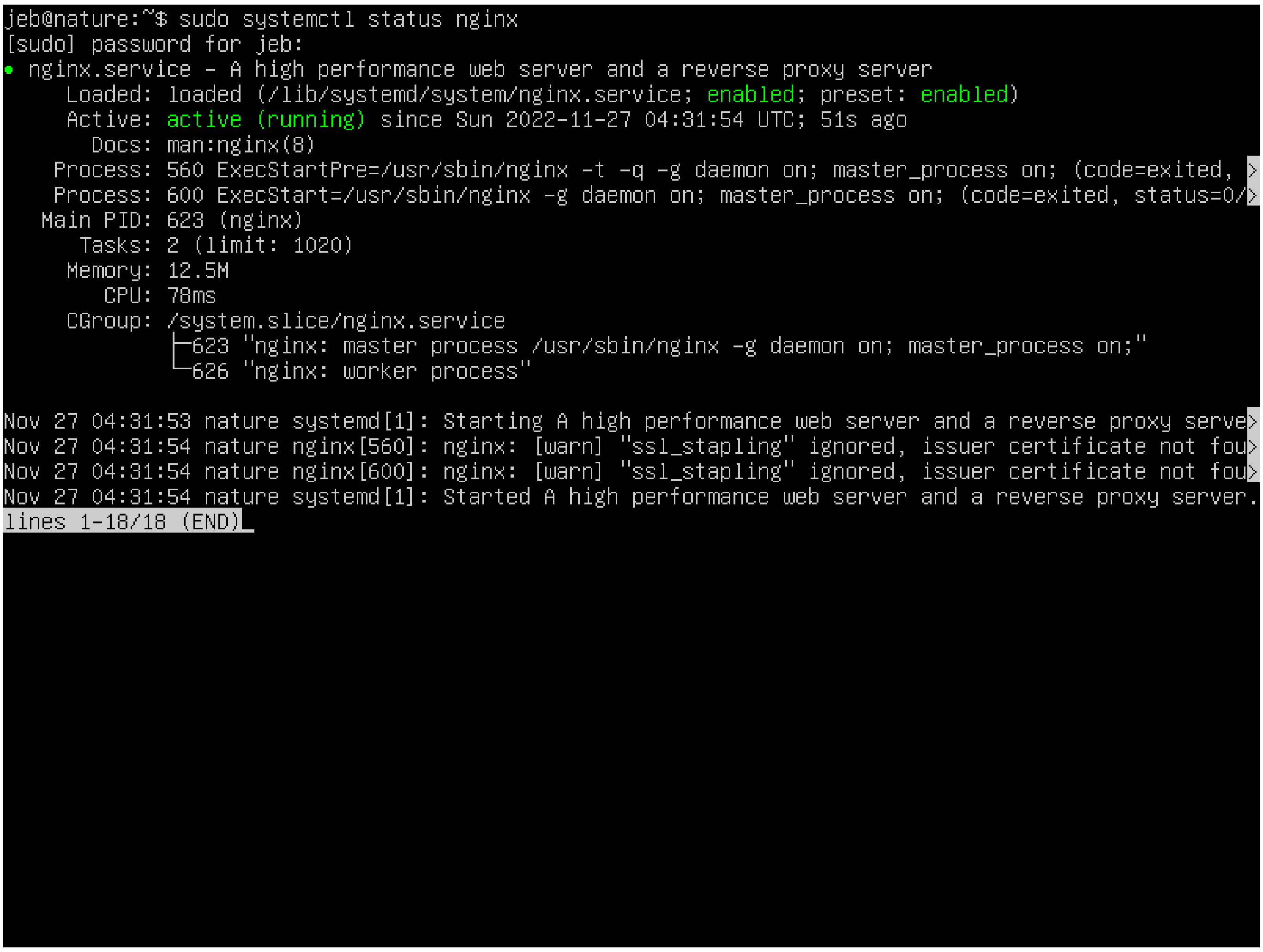


```
jeb@nature:~$ sudo apt-get update
[sudo] password for jeb:
Hit:1 http://us.archive.ubuntu.com/ubuntu kinetic InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu kinetic-updates InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu kinetic-backports InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu kinetic-security InRelease
Reading package lists... Done
jeb@nature:~$ sudo apt install nginx
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  fontconfig-config fonts-dejavu-core libdeflate0 libfontconfig1 libgd3 libjbig0 libjpeg-turbo8
  libjpeg8 liblerc3 libnginx-mod-http-geoip2 libnginx-mod-http-image-filter
  libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream libnginx-mod-stream-geoip2
  libtiff5 libwebp7 libxpm4 nginx-common nginx-core
Suggested packages:
  libgd-tools fcgiwrap nginx-doc ssl-cert
The following NEW packages will be installed:
  fontconfig-config fonts-dejavu-core libdeflate0 libfontconfig1 libgd3 libjbig0 libjpeg-turbo8
  libjpeg8 liblerc3 libnginx-mod-http-geoip2 libnginx-mod-http-image-filter
  libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream libnginx-mod-stream-geoip2
  libtiff5 libwebp7 libxpm4 nginx nginx-common nginx-core
0 upgraded, 21 newly installed, 0 to remove and 6 not upgraded.
Need to get 2,850 kB of archives.
After this operation, 8,861 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

4. To check that the installation occurred check the status of nginx

```
sudo systemctl status nginx
```

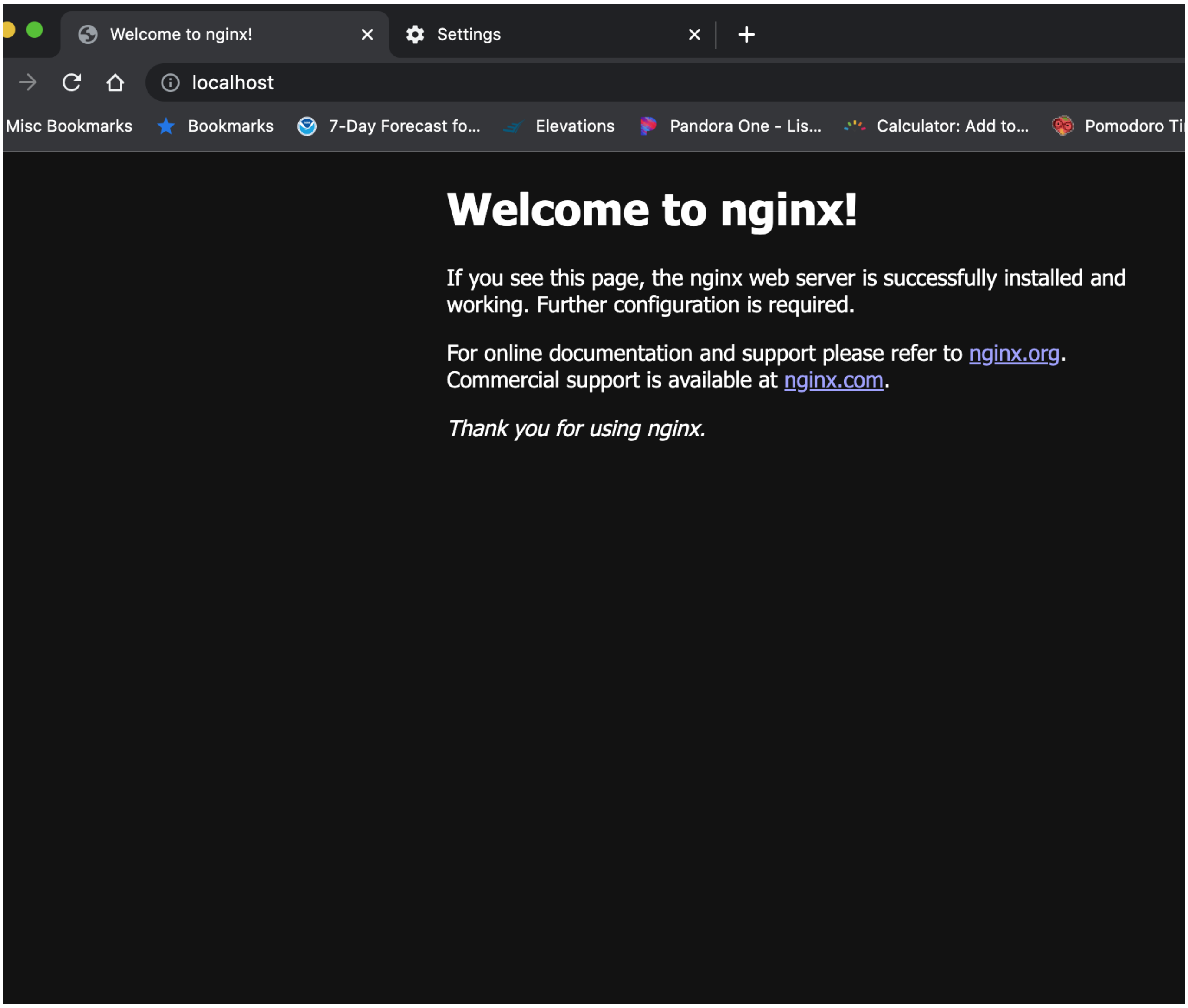
The output should be something like this.



```
jeb@nature:~$ sudo systemctl status nginx
[sudo] password for jeb:
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
     Active: active (running) since Sun 2022-11-27 04:31:54 UTC; 51s ago
       Docs: man:nginx(8)
    Process: 560 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, >
    Process: 600 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/>
   Main PID: 623 (nginx)
      Tasks: 2 (limit: 1020)
     Memory: 12.5M
        CPU: 78ms
     CGroup: /system.slice/nginx.service
             ├─623 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─626 "nginx: worker process"

Nov 27 04:31:53 nature systemd[1]: Starting A high performance web server and a reverse proxy serve>
Nov 27 04:31:54 nature nginx[560]: nginx: [warn] "ssl_stapling" ignored, issuer certificate not fou>
Nov 27 04:31:54 nature nginx[600]: nginx: [warn] "ssl_stapling" ignored, issuer certificate not fou>
Nov 27 04:31:54 nature systemd[1]: Started A high performance web server and a reverse proxy server.
lines 1-18/18 (END)
```

5. Navigate to the browser at http://localhost

you should see the default nginx page



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Troubleshooting

Problem One: If Nginx status is not active and exited because the port is already in use.

1. Check what is running on the port.

```
sudo lsof -i :80
```

Since Nginx by default will try to listen on port 80...this is the port you are looking at. If you need to look at a different port then change 80 to the port you are looking for.

2. you can stop what is running on port 80 by using the pid that outputed and the kill command.

```
sudo kill -9 <pid>
```

Signal Name	Single Value	Effect
SIGHUP	1	Hangup

Signal Name	Single Value	Effect
SIGINT	2	Interrupt from keyboard
SIGKILL	9	Kill signal
SIGTERM	15	Termination signal
SIGSTOP	17, 19, 23	Stop the process

Problem 2: Nothing is being loaded in the browser at http://localhost. This can happen if you are navigating to the browser in your host machine and have not set up to forward the ports from your guest machine (the VM) to the host machine (the machine runniing virtual box).

1. Check if port 80 is being forwarded from virtual machine to host.

Important Commands

Nginx Commands

Check if Nginx is running

```
sudo systemctl status nginx
```

Restart Nginx

```
sudo systemctl restart nginx
```

Reload configurtions for Ngins (this will be important for a later recipe)

```
sudo systemctl reload nginx
```

Definitions:
Web-server : a server that listens and reponds to http or https requests.

Resources:

1. https://www.nginx.com/resources/glossary/nginx/
2. https://w3techs.com/technologies/overview/web_server
3. https://www.linuxfoundation.org/blog/blog/classic-sysadmin-how-to-kill-a-process-from-the-command-line

Recipe 1.1: 📒 Customizing your Nginx site 📒

Date: November 26, 2022
Author: Johanna Blumenthal
Category: VM, Nginx, Web - Server
Keywords: Nginx, web-server
Hardware: Desktop/Laptop PC, Mac or Pi
Software: Mac OS, Virtual Box with Nginx installed
Preliminary:

1. Installation guide for Nginx (see above)
2. A web application you have written or obtained from elsewhere...our example will use the fizz buzz appplicaiton located here: https://github.com/ChayaBasha/CountdownFizz_Buzz. Note this is just a front-end application for ease of this exercise (there is not database for this application).

Overview

Now that we have our nginx server up and running, we probably want our website to be ours rather than the default nginx page. This recipe will give an example of how to change the degault nginx page to whatever you want it to be.

This recipe will also get us more comfortable with the file structures of nginx, which will help for later recipes.

8 steps

1. Locating the frontend files that nginx is serving up.

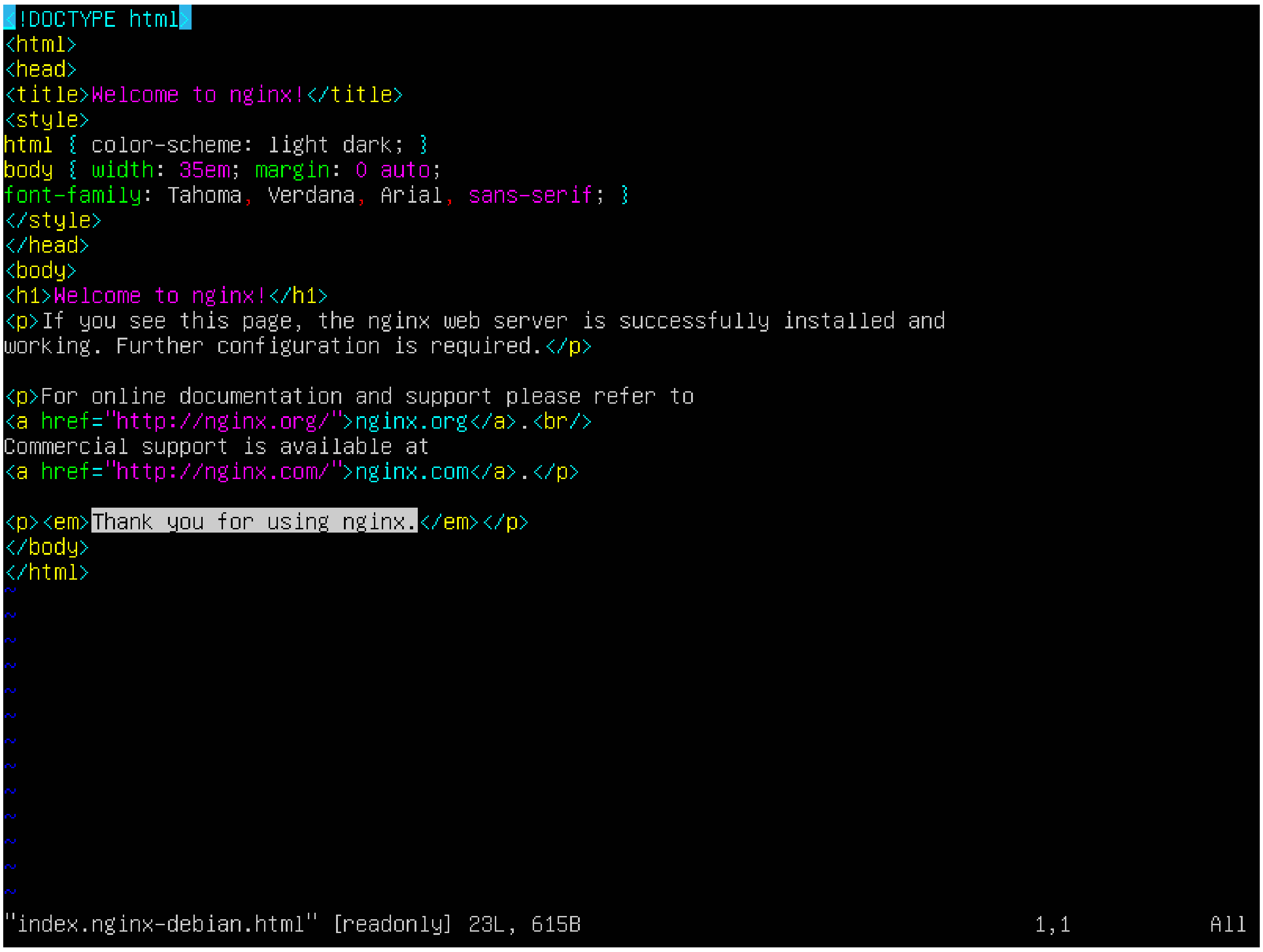
```
cd /var/www/html
```

This is the default directory and folder for the frontend webpage

2. Let's take a look at the index file using vim or whatever text editor you are used to (such as nano)

```
vim index.nginx-debian.html
```

The file should look like this:



```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
~
~
~
~
~
~
~
~
~
~
~
~
~
"index.nginx-debian.html" [readonly] 23L, 615B                1,1           All
```

3. Let's create our new directory from github. We want this to live where the html folder currently is so we need to navigate up to /var/www. Remember git clone will create a directory fort he git files it pulls down so we do not need to manually create the directory.

```
sudo git clone https://github.com/ChayaBasha/CountdownFizz_Buzz.git
```

Check this worked by looking for the directory CountdownFizz_buzz in /var/www/.

4. Now we need to configure the nginx serever so that it loads up the countdown.html file in the CountdownFizz_Buzz idrectory instead of the default in the html directory. First, we are going to inspect the configurations as they are.

```
cd /etc/nginx/sites-available
```

You will see that there is one file in there now named default

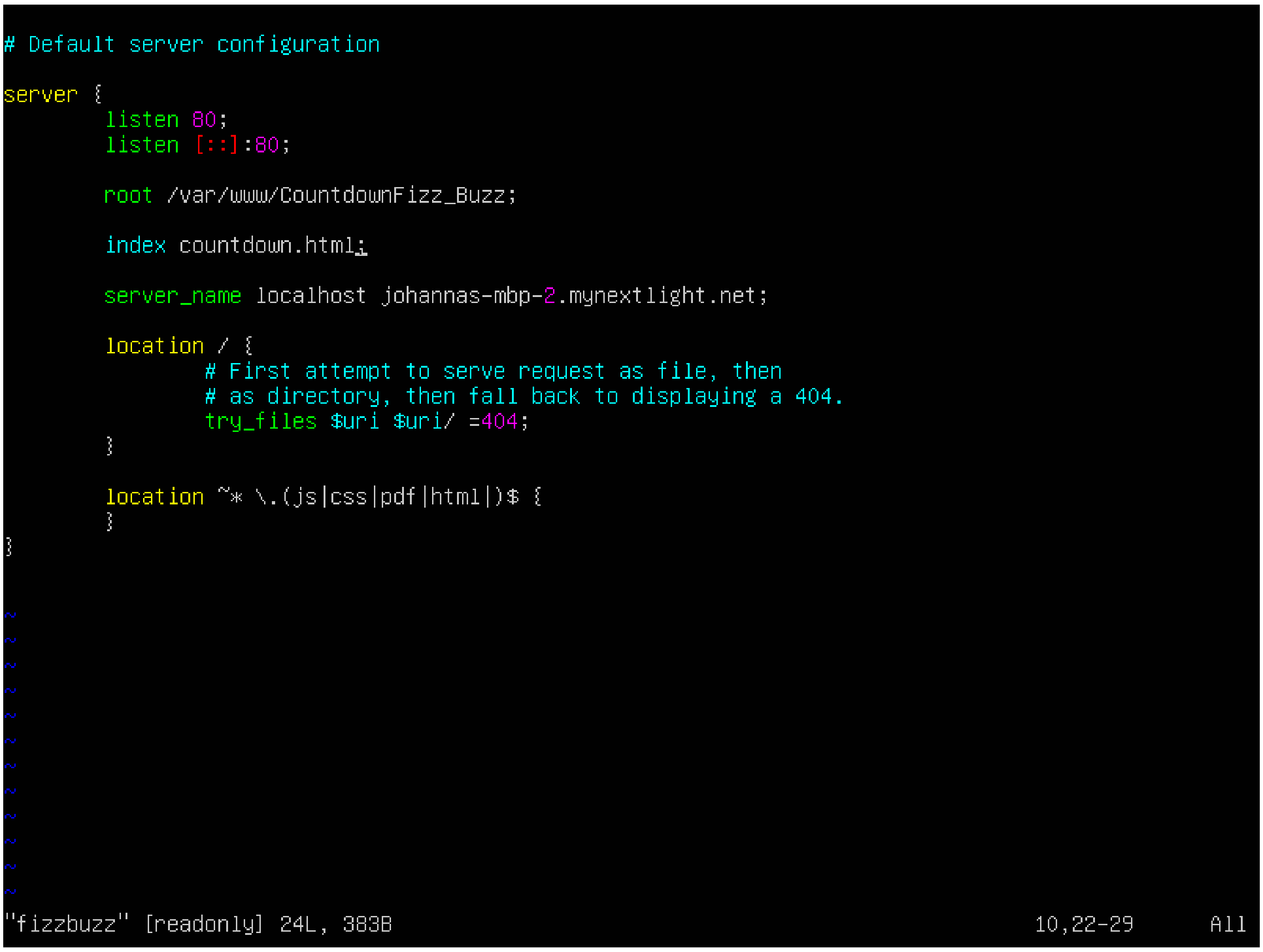
```
vim default
```

In here are a lot of instrucitons about how to use these configurations.

5. We don't want to change the default file in case we need it again. Some instrucitons have you copy it before editing. We are going to avoid that by just creating a new file in the same directory.

```
sudo vim fizzBuzz
```

This will create a new file called fizzBuzz. Edit the file so that it looks as follows:



```
# Default server configuration

server {
        listen 80;
        listen [::]:80;

        root /var/www/CountdownFizz_Buzz;

        index countdown.html;

        server_name localhost johannas-mbp-2.mynextlight.net;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }

        location ~* \.(js|css|pdf|html|)$ {
        }
}

~
~
~
~
~
~
~
~
~
~
~
~
"fizzbuzz" [readonly] 24L, 383B                                      10,22-29       All
```

notice we have changed the root from the original root var/www/html to var/www/CountdownFizz_Buzz. This will tell nginx to look for the index file there instead of in html where it was before.

The location ~* .(js|css|pdf|html)$ {

} tells the nginx server to allow these types of files.

6. Now we need to get our new configuration file enabled. Navigate to /etc/nginx/sites-enabled and you will see that only the default file is enabled. We can change that by following:

```
ln -s /etc/nginx/sites-available/fizzbuzz /etc/nginx/sites-enabled
```

look in the directory /etc/nginx/sites-enabled and you should now see default and fizzbuzz.

Because it can interere with the directions for nginx, we want to disable the default. We can do this by simply removing it from the sites-enabled directory.

```
sudo rm default
```

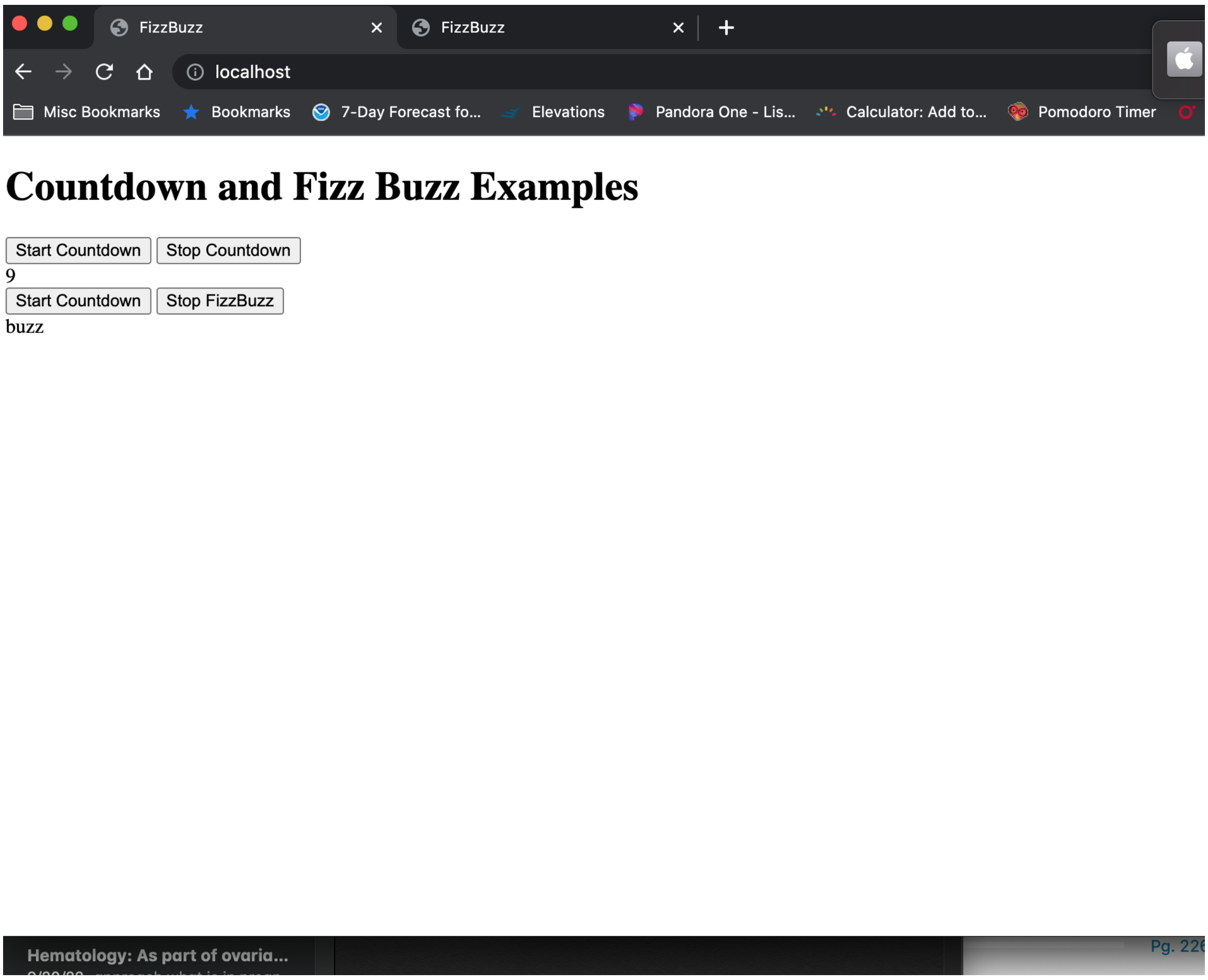
Don't worry this still exists in our sites-available directory if we want to bring it back.

7. We need to reload nginx in order for the changes to take hold

```
sudo systemctl reload nginx
```

8. Navigate to the browser at http://localhost

You should see our new site



Countdown and Fizz Buzz Examples

9

buzz

Troubleshooting

Problem one: When navigating to the browser you get a 403 forbidden error.

1. check the permissions of the directory where the files are located. They need to be read/write/executable for the js files to work.

```
chmod 755 <dir name>
```

2. make sure that the file you want to load is listed in the index of the configuration file located in /etc/ginx/sites-available

In our example, there is no index.html so you need to make sure countdown.html is listed.

3. Make sure that the file type is allowed in the etc/nginx/sites-available file

Important Commands

enabling and disabling the sites

To enable

```
ln -s /etc/nginx/sites-available/<name of site> /etc/nginx/sites-enabled
```

To disable

```
sudo rm <file name>
```

Resources:

1. https://www.linode.com/docs/guides/how-to-enable-disable-website/
2. https://linuxhint.com/fix-nginx-403-forbidden/

Recipe 1.2: 📒 Enabling https and redirect from http to https 📒

Date: November 26, 2022
Author: Johanna Blumenthal
Category: VM, Nginx, Web - Server
Keywords: Nginx, web-server
Hardware: Desktop/Laptop PC, Mac or Pi
Software: Mac OS, Virtual Box with Nginx installed
Preliminary:

1. Installation guide for Nginx (see above)
2. A non-default site configuration in /etc/nginx/sites-available (see above)
3. A crt certificate and a private key located in the /etc/ssl folder

Overview

HTTPS has become the standard. Most browsers will automatically append a domain with https at the beginning and will label content as insecure that does not have https enabled. Google's search engine favors https sites over http sites in its page ranking.

This recipe shows you how to change the nginx configurations to use https. The recipe assumes you have either a self-signed certificate or a purchased CA ready for use.

8 steps

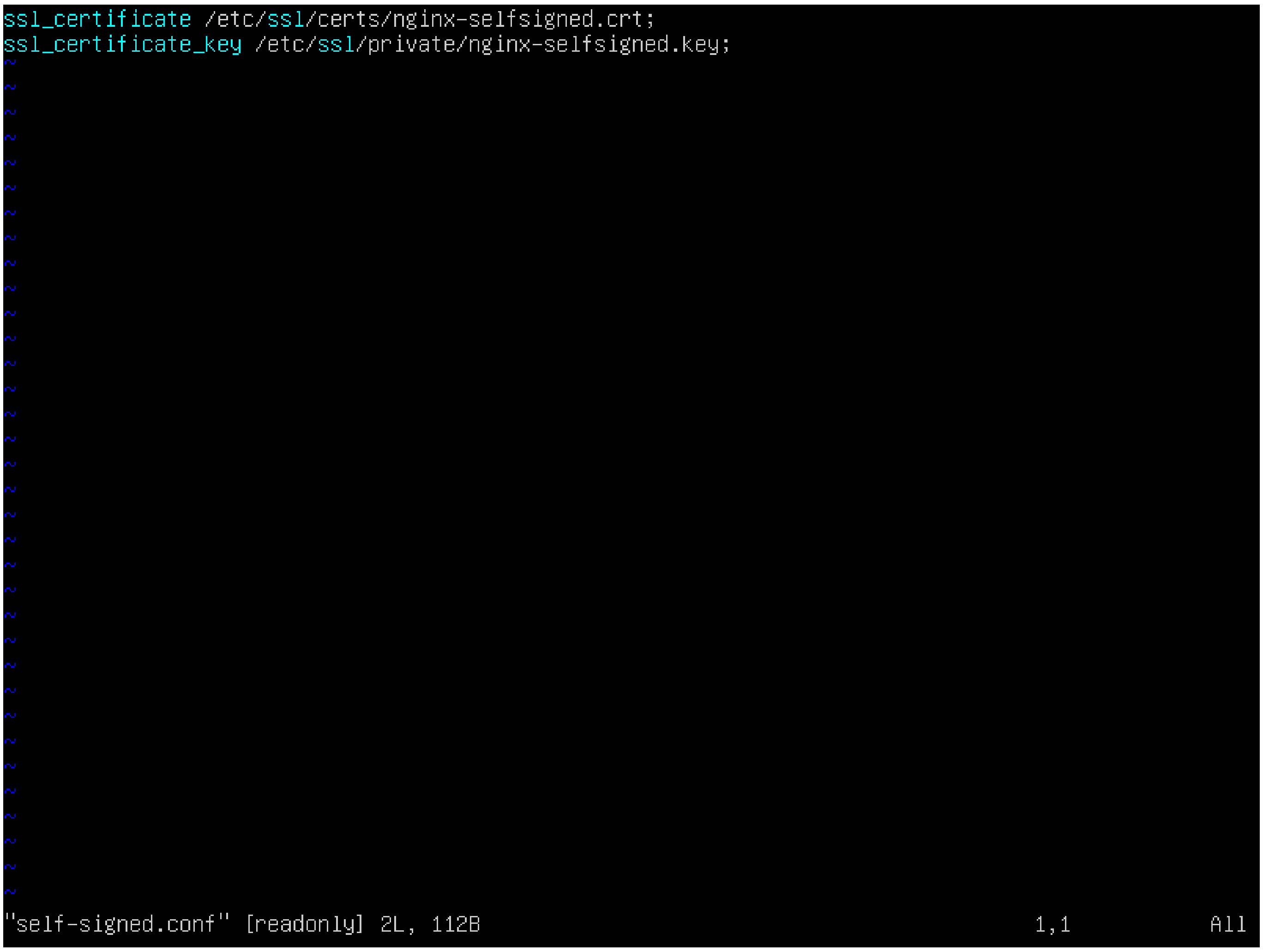
1. Double check the location and name of your key and certificate files as we will be using that to create some snippets for nginx to import into our configuration files.

```
cd etc/ssl
```

2. create a snippets file for the location of the key and certificate files in /etc/nginx/snippets/

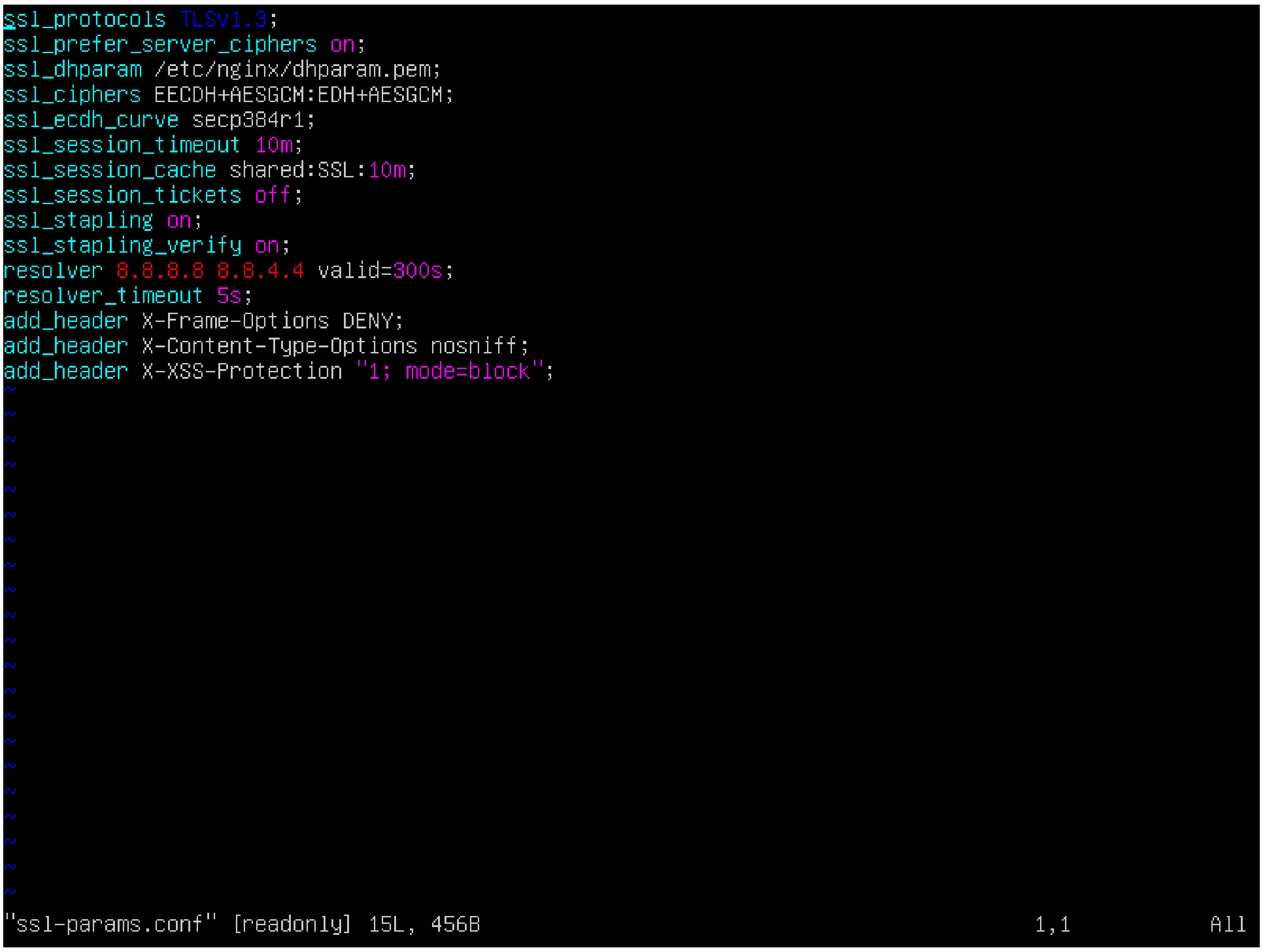
```
sudo vim /etc/nginx/snippets/<name of your file>
```

3. Edit your file as follows. Make sure to change the path and name to your .key and .crt files



```
ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
"self-signed.conf" [readonly] 2L, 112B                                1,1           All
```

4. Create another snippets file called ssl-params.conf and add the following to it

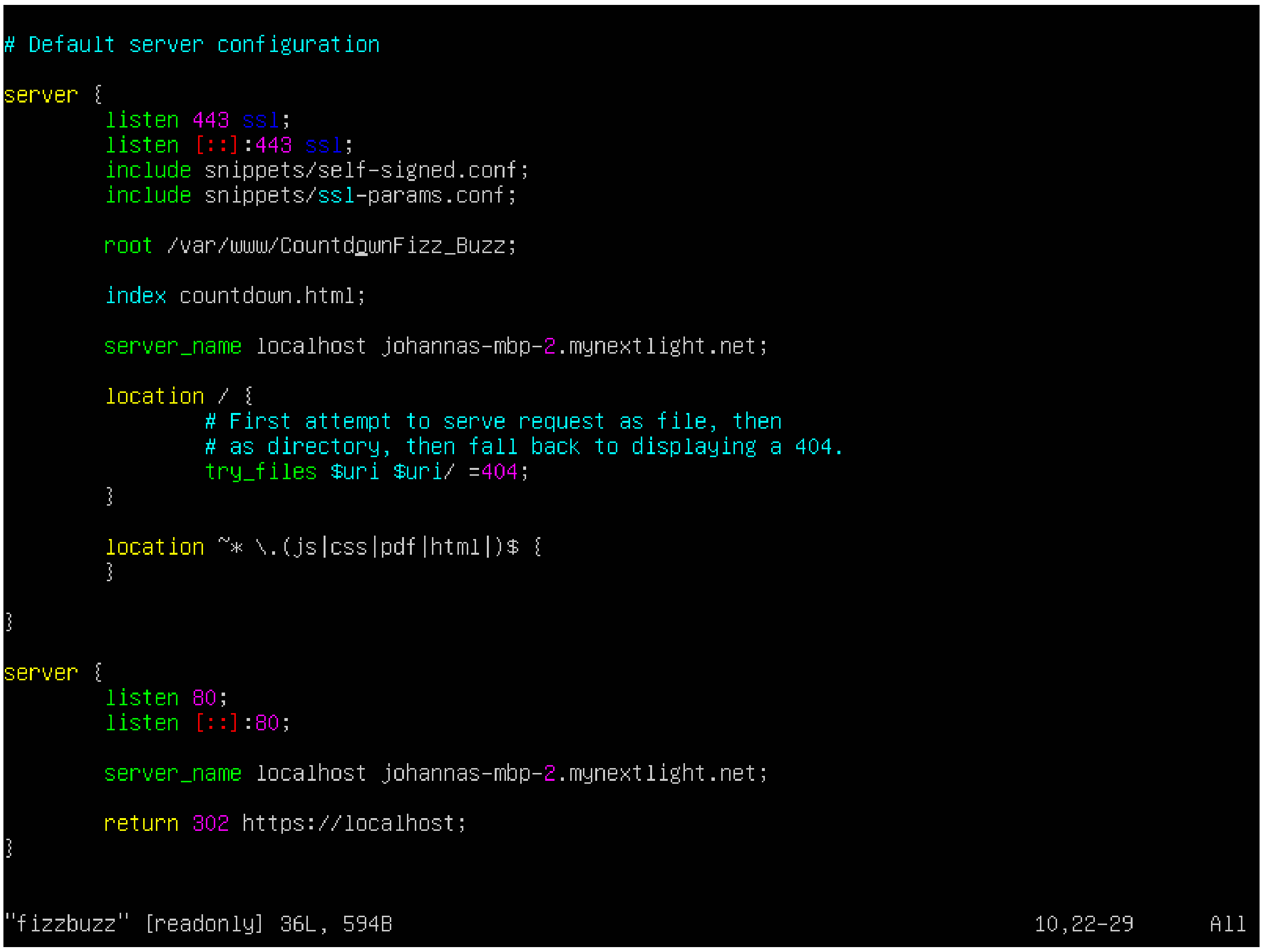


```
ssl_protocols TLSv1.3;
ssl_prefer_server_ciphers on;
ssl_dhparam /etc/nginx/dhparam.pem;
ssl_ciphers EECDH+AESGCM:EDH+AESGCM;
ssl_ecdh_curve secp384r1;
ssl_session_timeout 10m;
ssl_session_cache shared:SSL:10m;
ssl_session_tickets off;
ssl_stapling on;
ssl_stapling_verify on;
resolver 8.8.8.8 8.8.4.4 valid=300s;
resolver_timeout 5s;
add_header X-Frame-Options DENY;
add_header X-Content-Type-Options nosniff;
add_header X-XSS-Protection "1; mode=block";
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"ssl-params.conf" [readonly] 15L, 456B                          1,1           All
```

5. We need to make changes to our fizzbuzz configuration file, but first we want to make a backup of our original file in case something goes wrong and we want to go back.

```
cd /etc/nginx/sites-available
sudo cp fizzbuzz bu.fizzbbuzz
```

6. Now we will edit the original file to add ssl port listening and a redirect from port 80 to 443. The file should look like this when you are done.



```
# Default server configuration

server {
        listen 443 ssl;
        listen [::]:443 ssl;
        include snippets/self-signed.conf;
        include snippets/ssl-params.conf;

        root /var/www/CountdownFizz_Buzz;

        index countdown.html;

        server_name localhost johannas-mbp-2.mynextlight.net;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }

        location ~* \.(js|css|pdf|html|)$ {
        }

}

server {
        listen 80;
        listen [::]:80;

        server_name localhost johannas-mbp-2.mynextlight.net;

        return 302 https://localhost;
}

"fizzbuzz" [readonly] 36L, 594B                               10,22-29      All
```

7. We need to reload Nginx, but before we do let's test if the new configurations are correct.

```
sudo nginx -t
```

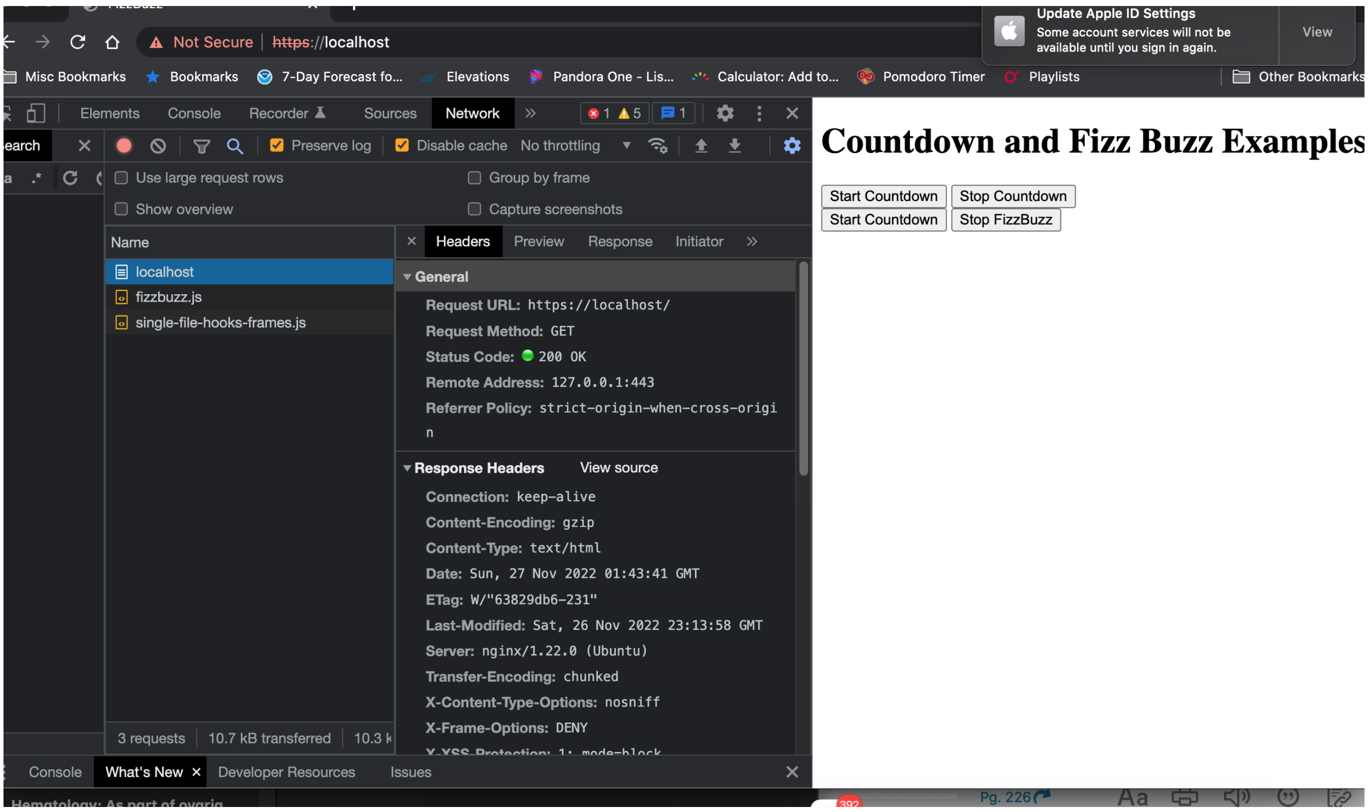
If this gives us an error it will tell us what line of the configurations are the problem.

8. Reload nginx to get the configuration to work

```
sudo systemctl reload nginx
```

9. Test the https in the browser https://localhost

Note that chrome will still say it is insecure if you are using a self-signed certificate.



Not Secure | https://localhost

Update Apple ID Settings
Some account services will not be available until you sign in again.

Elements Console Recorder Sources Network

Preserve log Disable cache No throttling

Use large request rows Group by frame
Show overview Capture screenshots

Name
localhost
fizzbuzz.js
single-file-hooks-frames.js

Headers Preview Response Initiator

▾ General

```
Request URL: https://localhost/
Request Method: GET
Status Code: 200 OK
Remote Address: 127.0.0.1:443
Referrer Policy: strict-origin-when-cross-origin
```

▾ Response Headers View source

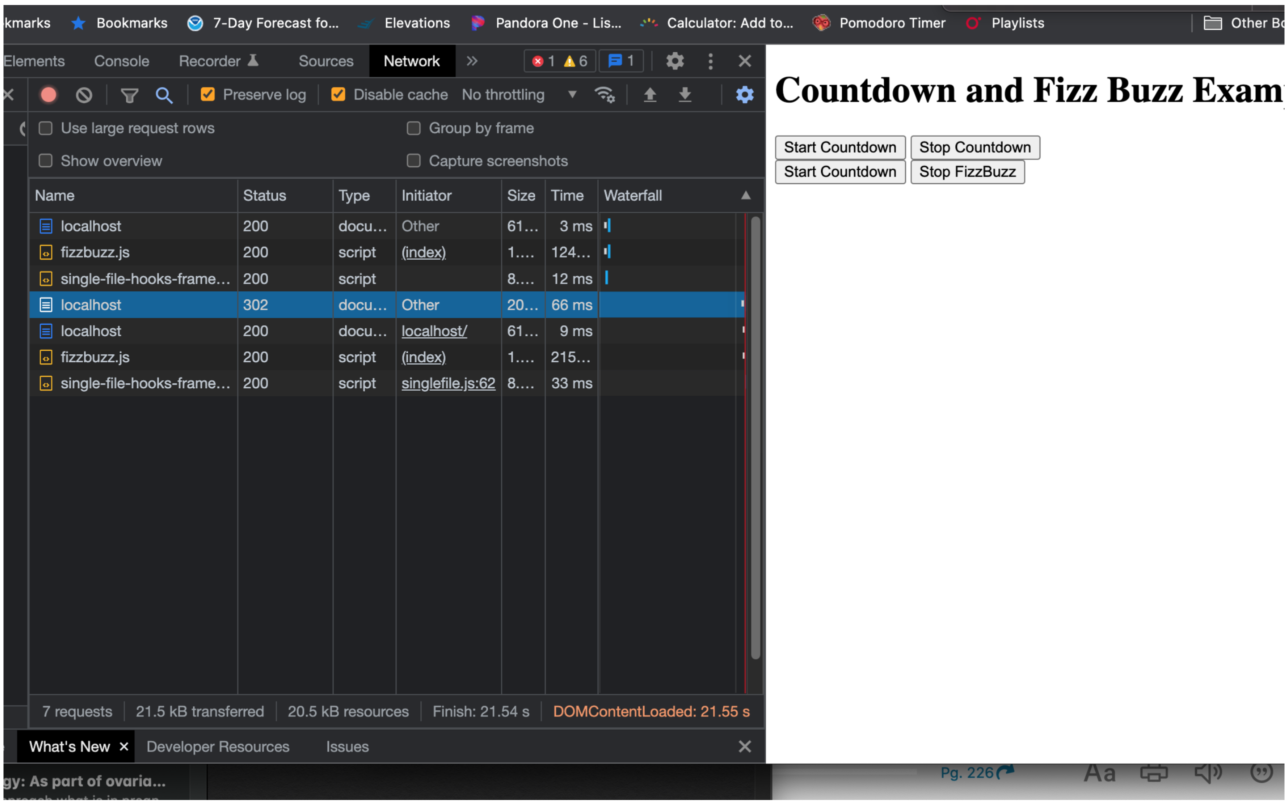
```
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html
Date: Sun, 27 Nov 2022 01:43:41 GMT
ETag: W/"63829db6-231"
Last-Modified: Sat, 26 Nov 2022 23:13:58 GMT
Server: nginx/1.22.0 (Ubuntu)
Transfer-Encoding: chunked
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
```

3 requests | 10.7 kB transferred | 10.3 k

Countdown and Fizz Buzz Examples

Start Countdown Stop Countdown
Start Countdown Stop FizzBuzz

8. Test the redirect by going to http://localhost. You should be able to see the redirect when you inspect the network.



Elements Console Recorder Sources Network

Preserve log Disable cache No throttling

Use large request rows Group by frame
Show overview Capture screenshots

Name	Status	Type	Initiator	Size	Time
localhost	200	docu…	Other	61…	3 ms
fizzbuzz.js	200	script	(index)	1.…	124…
single-file-hooks-frame…	200	script		8.…	12 ms
localhost	302	docu…	Other	20…	66 ms
localhost	200	docu…	localhost/	61…	9 ms
fizzbuzz.js	200	script	(index)	1.…	215…
single-file-hooks-frame…	200	script	singlefile.js:62	8.…	33 ms

7 requests | 21.5 kB transferred | 20.5 kB resources | Finish: 21.54 s | DOMContentLoaded: 21.55 s

Countdown and Fizz Buzz Exam

Start Countdown Stop Countdown
Start Countdown Stop FizzBuzz

Troubleshooting

Problem One: Cannot connect to server.

1. Did you remember to forward the 443 port from guest machine to host machine?
2. If you have a firewall, did you remember to allow 443?

Problem Two: No matter what the browser won't load because of the "insecure" certificate.

1. try culring to see if the configuration is working

```
curl -k https://localhost -v
```

-k allows it to accept a self-signed certificate This should show you the html source if it is working

2. try redirect through curl as well

```
curl -k http://localhost -v
```

The -v should show you the redirect information

Important Commands

Check the configuration of nginx

```
nginx -t
```

use curl to check website

```
curl -k https://domain
```

Resources:

1. https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-nginx-in-ubuntu-20-04-1

Recipe 2.1: 📒 Configuring Nginx as a Proxy Server on the same machine and Between Machines 📒

Date: November 26, 2022
Author: Johanna Blumenthal
Category: VM, Nginx, Web - Server
Keywords: Nginx, web-server
Hardware: Desktop/Laptop PC, Mac or Pi
Software: Mac OS, Virtual Box with Nginx installed, another web server that is configurable by you
Preliminary:

1. Installation guide for Nginx (see above)
2. Configuring as a web-server with https (see above)

3. Another VM running another web-server (for this recipe we assume a lamp stack running an apache server on another VM)

Overview

Now that we know how to use Nginx as a web-server, we are ready to take advantage of another feature Nginx offers, the ability to use it as a proxy seerver.

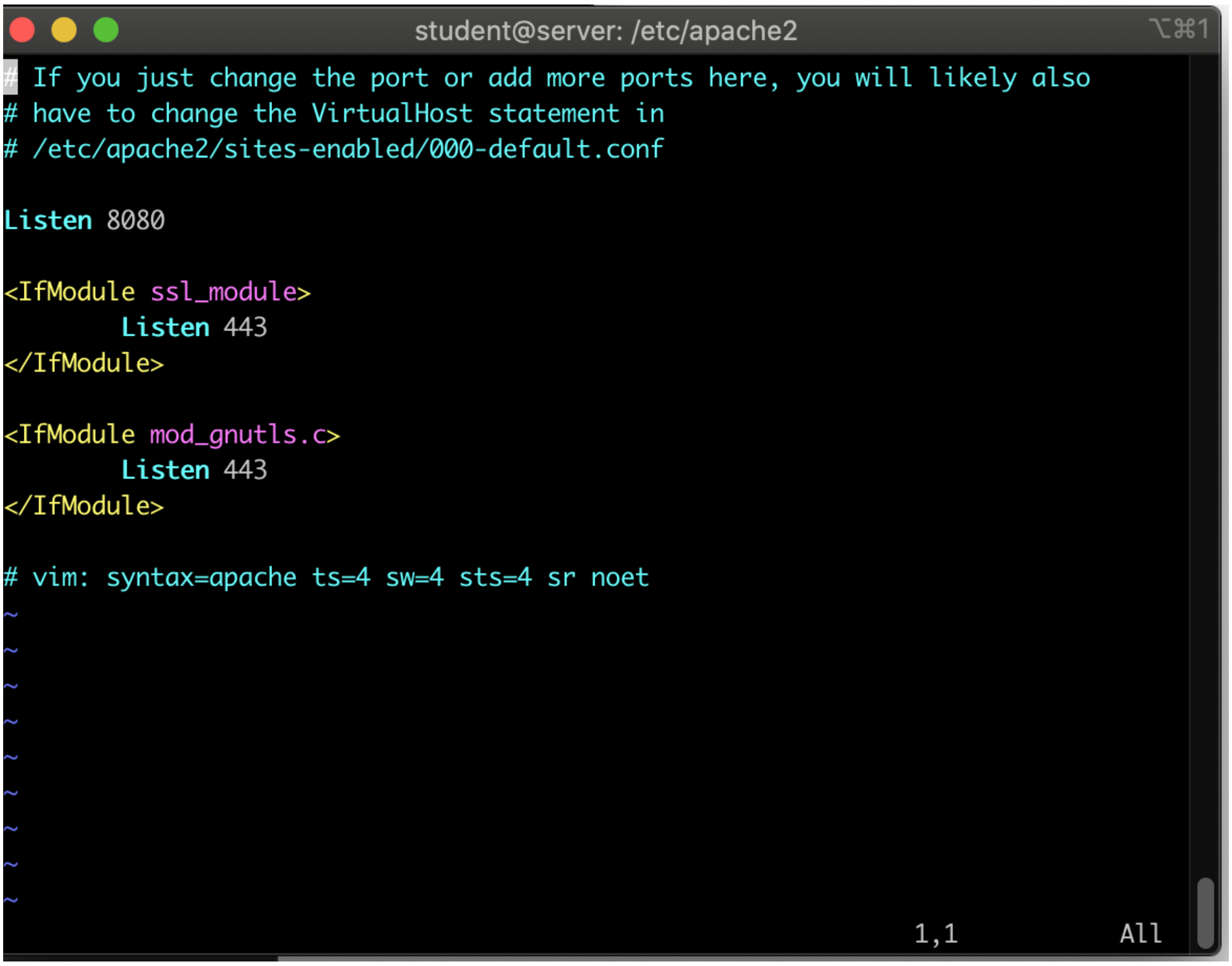
If you have used the internet, you have probably used a proxy server whether you knew it or not. That is the beauty of a proxy server. A proxy server is a server that takes in a request, forwards it along to another server that handles the request and then sends the response back all without the user knowing what happened behind the scenes.

This recipe describes how to configure nginx to serve as a proxy server for another web-sever.

9 steps

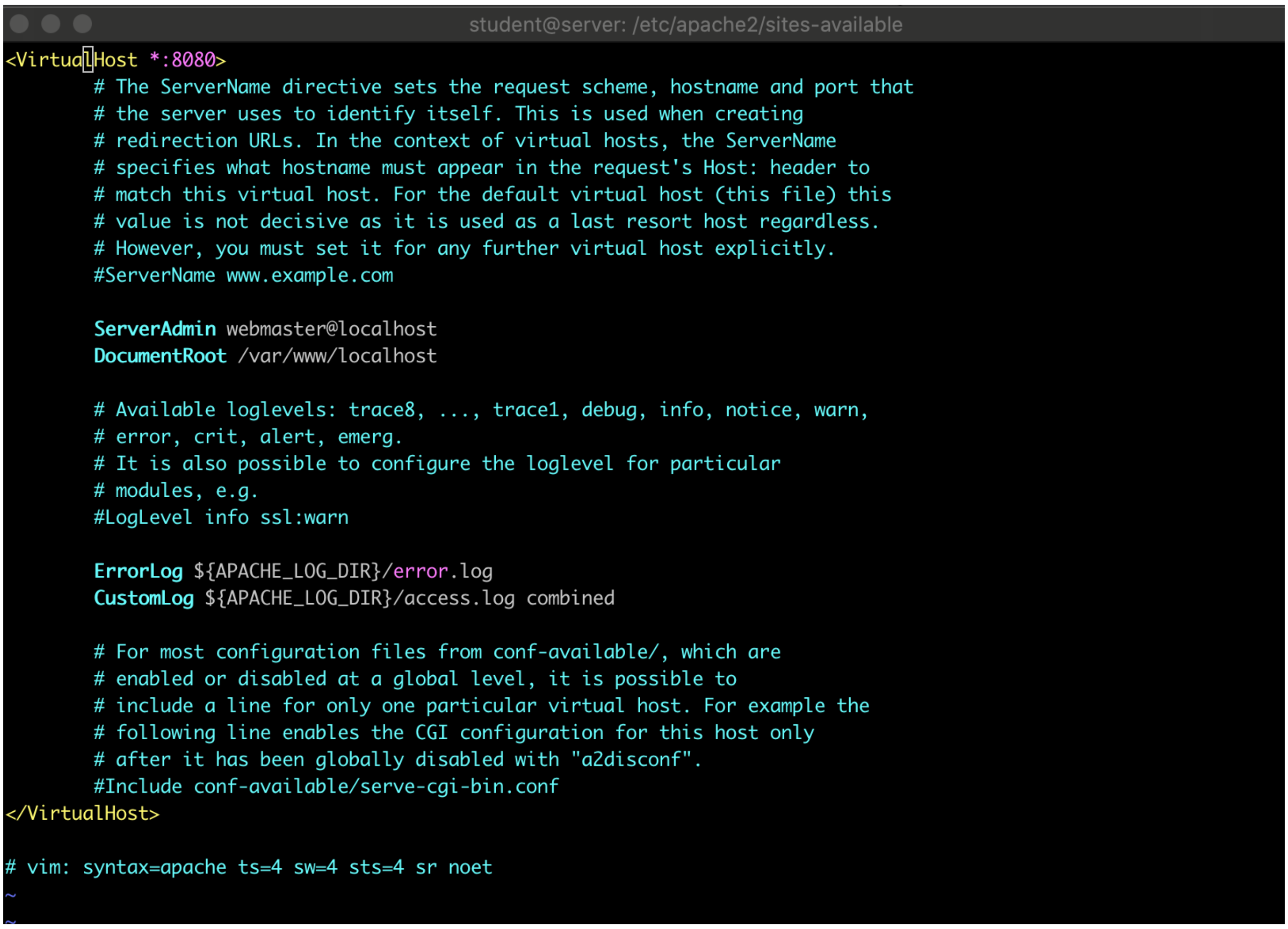
1. Start the web server (for this example I am using apache). Make sure the web server that is not the Nginx server is working by going to the browser. If you never changed the port, then this would be on port 80.

2. Now navigate to /etc/apache2 Open the ports.conf file and edit the port to listen on to something else... I used 8080 for one example and 8000 for another (see below).

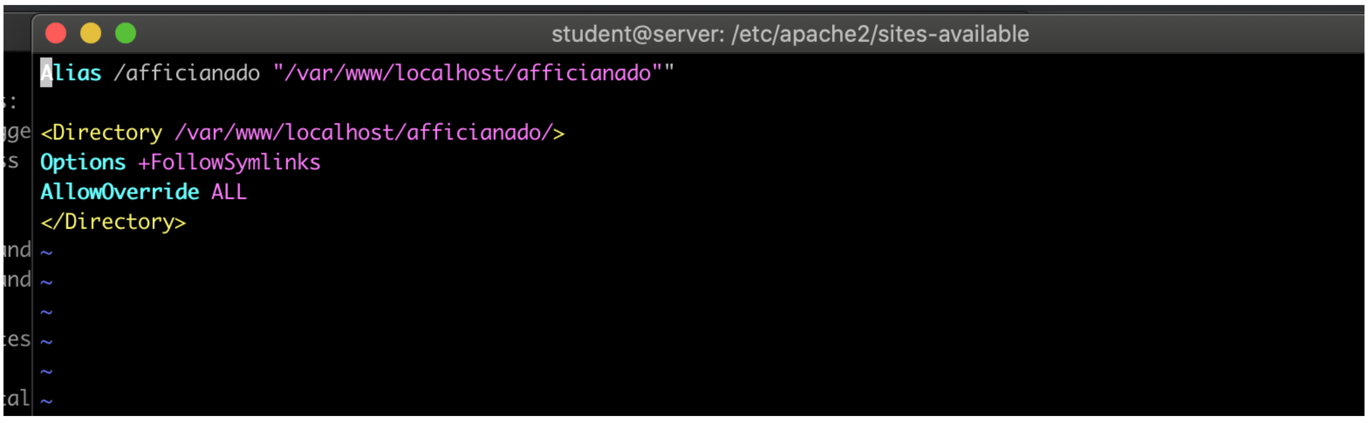


```
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 8080

<IfModule ssl_module>
        Listen 443
</IfModule>

<IfModule mod_gnutls.c>
        Listen 443
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
~
~
~
~
~
~
~
~
~
                                                              1,1           All
```

3. Navigate to the virtualhost configuration file /etc/apache2/sites-available. If you used an alias for your other sites, make sure you are editing the main configuration file. For this example my sites-available file has a 000-deault.conf file and an afficianado.conf file. In the 000-default.conf file we are changing the VirtualHost port to 8080. Your other file should not need to change.



```
<VirtualHost *:8080>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/localhost

        # 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

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
~
~
```



```
Alias /afficianado "/var/www/localhost/afficianado""

<Directory /var/www/localhost/afficianado/>
Options +FollowSymlinks
AllowOverride ALL
</Directory>
~
~
~
~
~
~
```

4. restart apache

```
sudo systemctl restart apache2
```

5. Open your machine with Nginx running and navigate to /etc/nginx/sites-available. Create a backup if you wish. Add another location block. This is where your apache server proxy is going to go.

Inside the main server block, after the

```
location / {
    try_files $uri $uri/ =404;
}
```

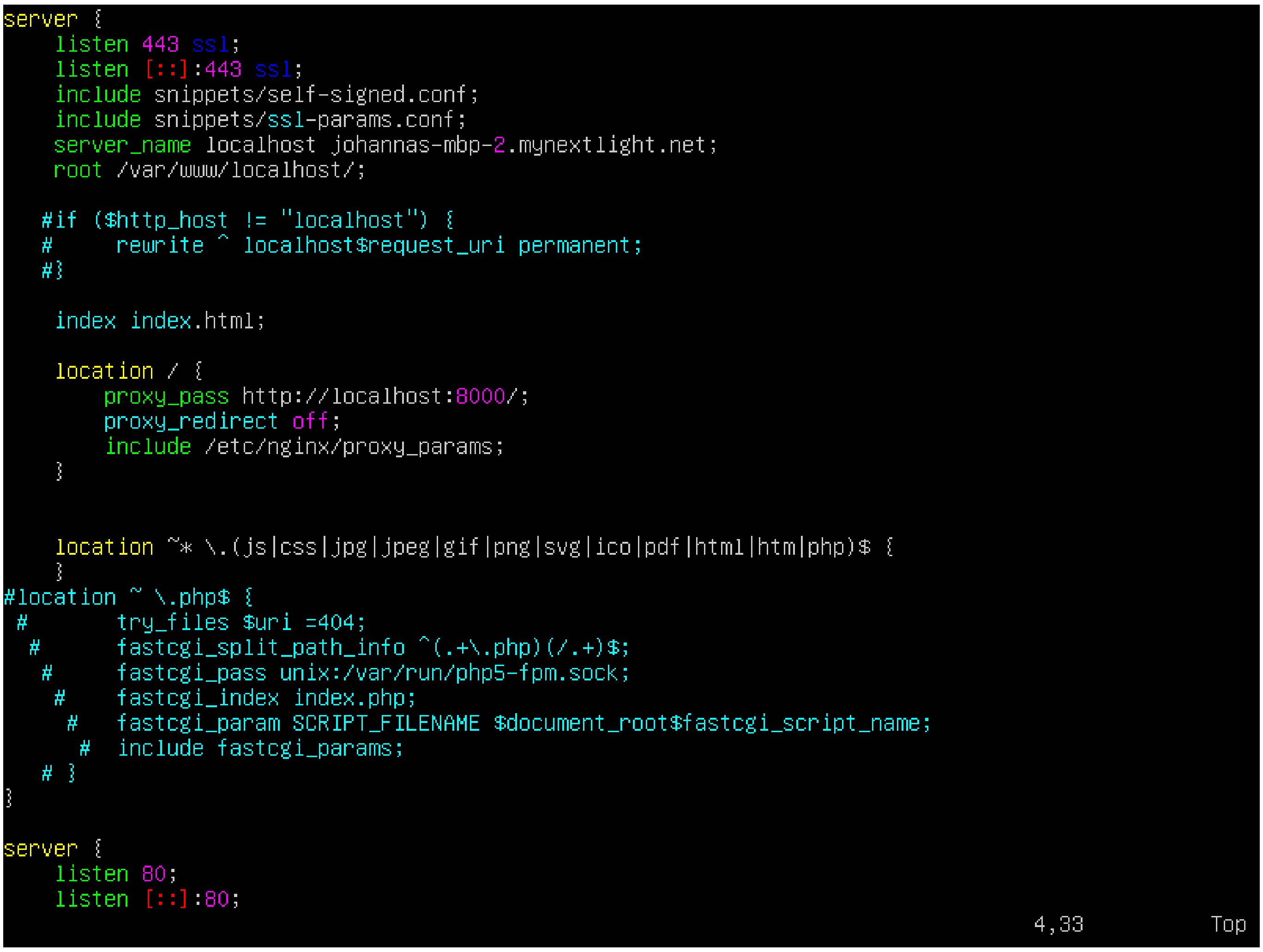
Add

```
    location /<whatever you want the subdomain to be {
        root <the root for the other web-server>;
        proxy_pass http://<whatever the domain is you are proxying to>;
        include /etc/nginx/proxy_params;
        proxy_redirect <domain proxying to> <domain proxynig from>;
    }
```

For example for the LAMP in Cloud VM I am using

```
    location /afficianado {
        root /var/www/localhost
        proxy_pass http://192.168.76.71:8080/afficianado;
        include /etc/nginx/proxy_params;
        proxy_redirect http://192.168.76.71:8080
http://192.168.76.82/afficianado;
    }
```

For the localhost example (apache and nginx are on the same machine) the complete configuration file looks as follows (note the server redirect should look as described in the previous recipe...here the screenshot is a little cut off). You should also note that the var/www/localhost file has a directory afficinaado that has all of the LAMP stack php files, including a configuration to mysql, the two files from our countdown front end application described in recipe 2.1.



```
server {
    listen 443 ssl;
    listen [::]:443 ssl;
    include snippets/self-signed.conf;
    include snippets/ssl-params.conf;
    server_name localhost johannas-mbp-2.mynextlight.net;
    root /var/www/localhost/;

   #if ($http_host != "localhost") {
   #     rewrite ^ localhost$request_uri permanent;
   #}

    index index.html;

    location / {
        proxy_pass http://localhost:8000/;
        proxy_redirect off;
        include /etc/nginx/proxy_params;
    }


    location ~* \.(js|css|jpg|jpeg|gif|png|svg|ico|pdf|html|htm|php)$ {
    }
#location ~ \.php$ {
 #       try_files $uri =404;
  #      fastcgi_split_path_info ^(.+\.php)(/.+)$;
   #     fastcgi_pass unix:/var/run/php5-fpm.sock;
    #    fastcgi_index index.php;
     #   fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
      #  include fastcgi_params;
   # }
}

server {
    listen 80;
    listen [::]:80;
                                                              4,33          Top
```

Make sure to add any file types you may be using into the location block

```
location `* \.(js|css|html|jpeg|jpg|gif|pdf|htm|php)$ {

}
```

6. Create the proxy_params file referenced in the location block for the proxy.

```
sudo vim /etc/nginx/proxy_params
```

Add

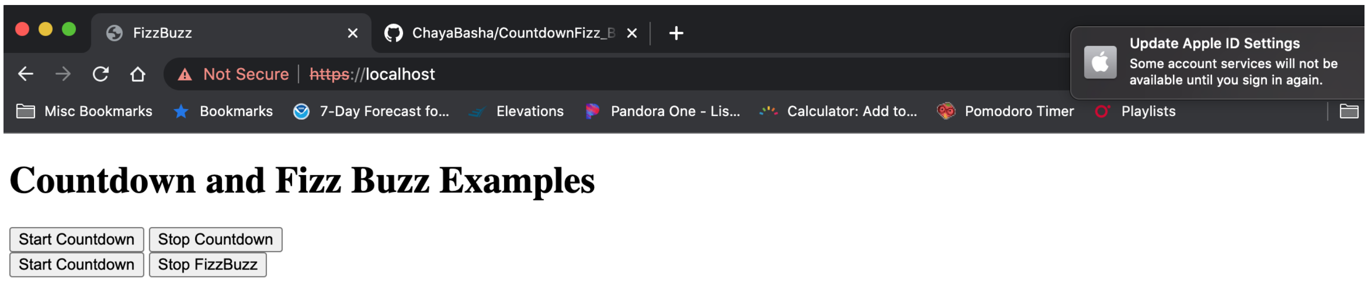
```
proxy_set_header Host $http_host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded_Proto $scheme;
proxy_set_header X-Forwarded-Port $server_port;
```

7. Reload Nginx configurations

```
sudo systemctl reload nginx
```

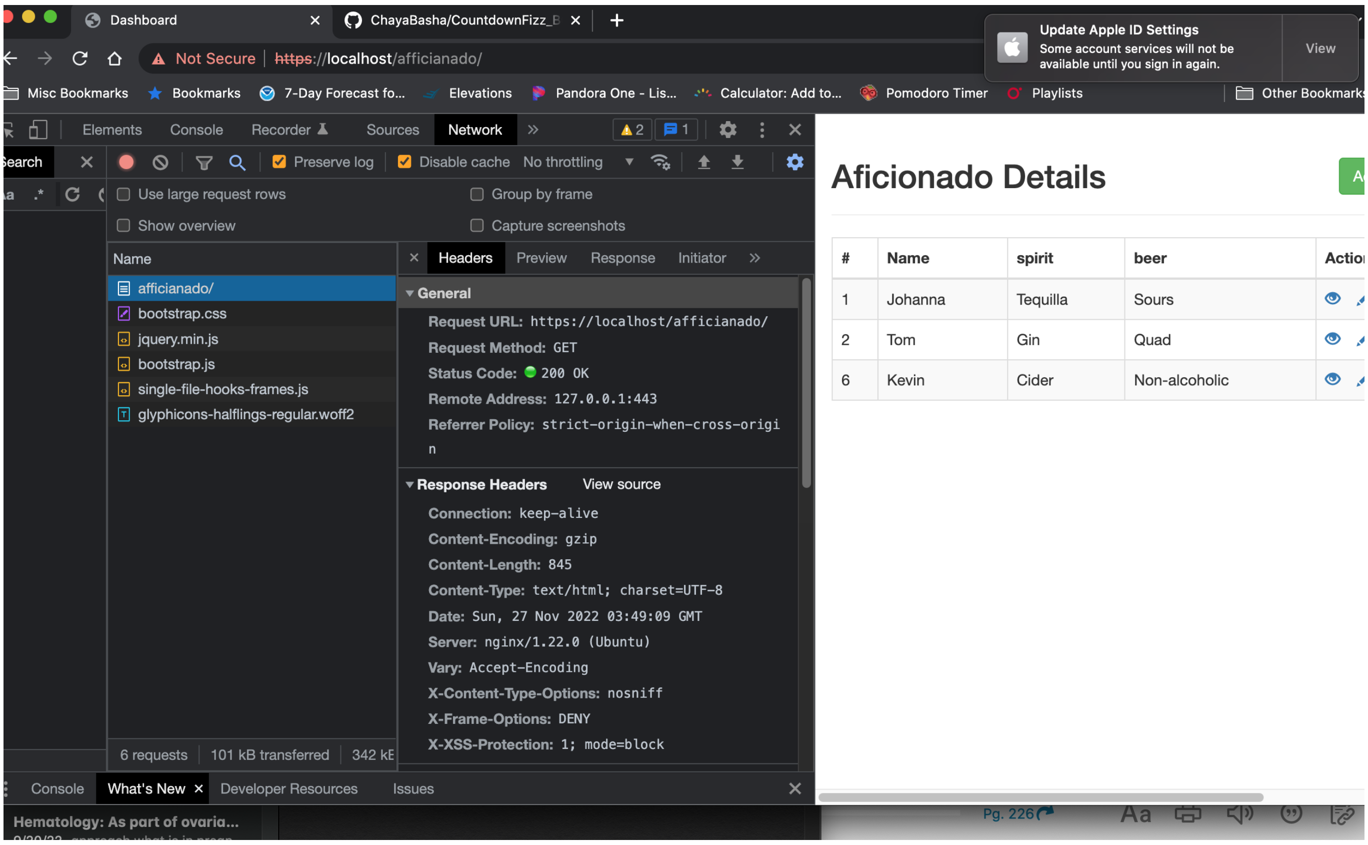
8. check the browser https://localhost

This should load the nginx server (since that is what the default root domain is set to). We are expecting this to look the same as before we did the proxy configurations.



Countdown and Fizz Buzz Examples

9. check the proxy in the browser https://localhost/afficianado/ Altohugh this is runing on port 443 and apache is running on 8080, it should look like it is coming from the same place to the client. Note the server is Nginx/1.22.0



Name
afficianado/
bootstrap.css
jquery.min.js
bootstrap.js
single-file-hooks-frames.js
glyphicons-halflings-regular.woff2

```
General
Request URL: https://localhost/afficianado/
Request Method: GET
Status Code: 200 OK
Remote Address: 127.0.0.1:443
Referrer Policy: strict-origin-when-cross-origin

Response Headers        View source
Connection: keep-alive
Content-Encoding: gzip
Content-Length: 845
Content-Type: text/html; charset=UTF-8
Date: Sun, 27 Nov 2022 03:49:09 GMT
Server: nginx/1.22.0 (Ubuntu)
Vary: Accept-Encoding
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
```

Aficionado Details

#	Name	spirit	beer	Actio
1	Johanna	Tequilla	Sours	
2	Tom	Gin	Quad	
6	Kevin	Cider	Non-alcoholic	

Troubleshooting

Similar Troubleshooting for the previous recipes. Check port and firewalls. Check the types of files allowed. Check the roots for both servers and the index files.

Resources:

1. https://www.linode.com/docs/guides/use-nginx-as-a-front-end-proxy-and-software-load-balancer/

Recipe 2.1: 📒 Configuring Nginx as a Proxy Server in A Docker Container Network 📒

Date: December 11, 2022
Author: Johanna Blumenthal
Category: Docker Containers, Nginx, Web-Proxy-Server
Keywords: Nginx, Docker-Compose, Docker Network
Hardware: Desktop/Laptop PC, Mac or Pi
Software: Mac OS, Virtual Box donfigured as Ubuntu Server with Docker and Docker compose installed.

Preliminary:

1. Docker and Docker Compose installed on Ubuntu Server (an excellent guide exists from Digital Ocean at https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-compose-on-ubuntu-22-04)
2. At least two different application servers with their apps runniing (in this guide we are using the RESTfulCRUD app available at https://github.com/RegisUniversity/RESTfulCRUD/tree/master) and the coundown app available at https://github.com/ChayaBasha/CountdownFizz_Buzz.)

Overview

Now that we have figured out how to configure Nginx as a proxy server. We want to take advantage of those features in another popular environment, a containerized environment.

Docker is a popular software platform for creating containers (isolated sandbox environments for development and/or deployment of applications).

This recipe assumes that you have some familiarity with docker imgages and docker compose files. If you are not, please follow the getting-started guide for Docker on their website (https://docs.docker.com/get-started/) before porceeding.

This Recipe will show you how to configure a nginx docker container so that it can serve as a wen-proxy for other Docker containers.

13 steps

1. Select the docker contianer images you will need for your network. For this recipe we will create three containers:

- The Nginx Container (configured both as a web-server to serve CountDown application and as a web proxy)
- A php apache container (this will serve up our aficianado application)
- A mySql database container configured to work with the aficianado data

2. Getting Your File System Set up

It is really important for docker to work correctly that you are clear about where your docker-compose.yml file is and where your application files are. We will be specifying this as we continue the recipe. For now, create a folder for your applicaiton. For our example we will call ours multistack.

In the multistack directory put the aficianado application files. The example assumes this is in a directory called aficianado and that the aficianado directory contains the files located at https://github.com/RegisUniversity/RESTfulCRUD/tree/master.

Also in the mulistack directory, create a directory called app. In the app directory place the fizzBuzz applicaiton files in a directory labeled fizzBuzz. In the fizzBuzz directory place the files from https://github.com/ChayaBasha/CountdownFizz_Buzz.

In the app directory, I am creating an index.html file for the entry into my environment. This will be served by the Nginx container and containes links to the other two applications. This is simply for ease of navigating and testing that our web-server and proxy server work.

3. Create a docker-compose.yml file in /multistack. We will be filling this in as we go.

4. We are going to create the aficianado container first. Edit the docker-compose.yml file as follows:

```
version: '3.7'
services:
  php-apache-environment:
    container-name: php-apache2
    build:
      context: ./aficianado
    volumes:
      - ./aficianado:/var/www/html/
    ports:
      - "8000:80"
```

Right now our docker-compose file has one container in it. The container is called php-apache2. The application files for aficianado live in our host computer at ./aficianado. We are specifying for the container to put those files in the /var/www/html directory of the container when it is created.

You could go ahead and run

```
docker compsoe up
```

and see the container load in localhost:8000, but it won' work properly because it doesn't have its mysql database container set up. Let's set that up next

5. configure and create the mysql database container. Go back in to edit your docker-compose.yml file and add the database container

```
version: '3.7'
services:
  php-apache-environment:
    container-name: php-apache2
    build:
      dockerfile: Dockerfile
      context: ./aficianado
    depends_on:
      - db2
    volumes:
      - ./aficianado:/var/www/html/
    ports:
      - "8000:80"

  db2:
    container_name: db2
    image: mysql
    volumes:
      - aficianado-mysql-data-2:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: aficianado
      MYSQL_USER: JB
      MYSQL_PASSWORD: puppePress
    ports:
      - "9906:3306"
volumes:
  aficianado-mysql-data-2:
```

As you will see our services now include two contaniers, the php-apache2 container and the db2 container. The environment variables of the db2 container specify the root password and database to be created for the mysql container that will be create. The volumes specify a directory on our host machine so that our data is retained when destroying the container itself.

Notice we addded another build command in the docker-compose file. It references a dockerfile. We need to create that in order to make sure that the database and application can be configured correctly together.

6. Creat the Dockerfile mentioned. Navigate to the multistack/aficianado and create a Dockerfile

```
FROM php:8.0-apache
RUN docker-php-ext-install mysqli &&  docker-php-ext-enable mysqli
RUN apt-get update && apt-get upgrade -y
```

This adds some additional commands when building the container

7. Create the containers

```
docker compose up
```

8. Navigate to http://localhost:8000

You should see the afficaiado app, but there will be some errors with the mysql.

Let's fix those.

9. Edit the mysql container.

The way aficainado was written, we need to add the People table to the afciainado database manually. Let's go ahead and do that.

To get into the container we need is id

```
docker ps
```

```
docker exec -it <id for mysql container> mysql -u root -p
```

This should get us into the mysql console of the container. Remember we se the password as 'secret'

in the mysql console you should see the aficianado database using

```
SHOW DATABASES;
```

lets add the people table

```
use aficianado;
```

```
CREATE TABLE people (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    spirit VARCHAR(255) NOT NULL,
    beer VARCHAR*255) NOT NUll
);
```

We need to add permissions for our user we specified in the docker-compose fle.

```
CREATE USER 'JB'@'localhost' IDENTIFIED BY 'puppePress';

GRANT ALL ON aficianado.* TO 'JB'@'localhost';
```

Now our afificando app should work at localhost:8000.

10. Create our Nginx Container

We need to create the container by adding it to the docker-compose file

```
version: '3.7'
services:
 web:
   image: nginx:alpine
   ports:
     - "80:80"
   volumes:
     - ./app:/usr/share/nginx/html
```

Run the container and you should see the application at http://localhost

But this is not configured as a proxy yet...it is just a web-server right now.

11. We need to configure the nginx container as a proxy. First we need to write some configuration files.

Create a file called multistack.conf in your multistakc directory. This is where we are going to put our configurations that we are used to making for a proxy (see recipe 2.0 above)

```
server {
    listen 80;
    listen [::]:80 default_server;

    server_name localhost 127.0.0.1;
    root /usr/share/nginx/html

    index index.html index.php;

    location / {
        try_files $uri $uri/ = 404;

    }

    location /aficianado {
        root www /var/www/localhost;
        proxy_pass http://php-apache2/;
        proxy_set_header HOST $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded_Proto $scheme;
        proxy_set_header X-Forwarded_Port $server_port;
        proxy_redirect http://php-apache2/ http://localhost/aficainado;
    }
}
```

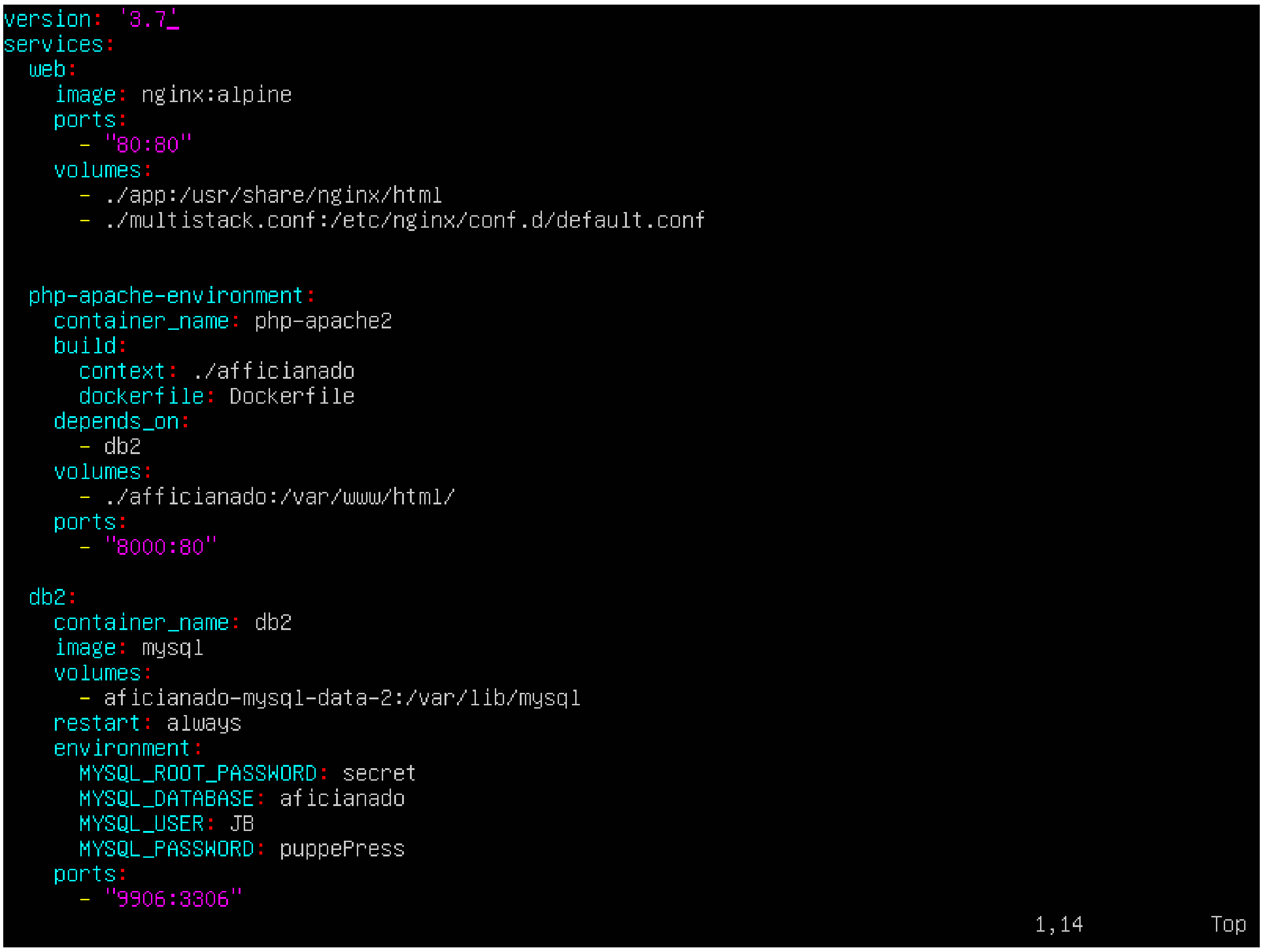
You will note some differences here form recpe 2.0 above. The proxy_params are passed in under the location instead of in its own file. This was just a choice of ease in passing in the new .conf files to the Nginx container.

You will also note that http://php-pache2/ is used as the ip adress/url for the proxy_pass. This takes advatnages of the native network setup docker does for use when we create multiple services in the same docker-compose file. They can be aliased by name of the container.

12. We need to add our custom configuration to the nginx container when it is built. Change the docker-compose.yml file

```
version: '3.7'
services:
 web:
   image: nginx:alpine
   ports:
     - "80:80"
   volumes:
     - ./app:/usr/share/nginx/html
     - ./multistack.conf:/etc/nginx/conf.d/default.conf
```

Your full docker-compose.yml file should look like this:



```
version: '3.7'
services:
  web:
    image: nginx:alpine
    ports:
      - "80:80"
    volumes:
      - ./app:/usr/share/nginx/html
      - ./multistack.conf:/etc/nginx/conf.d/default.conf


  php-apache-environment:
    container_name: php-apache2
    build:
      context: ./afficianado
      dockerfile: Dockerfile
    depends_on:
      - db2
    volumes:
      - ./afficianado:/var/www/html/
    ports:
      - "8000:80"

  db2:
    container_name: db2
    image: mysql
    volumes:
      - aficianado-mysql-data-2:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: aficianado
      MYSQL_USER: JB
      MYSQL_PASSWORD: puppePress
    ports:
      - "9906:3306"
                                                                  1,14          Top
```

13. navigate to http://localhost (you shuld see the index.html page you created in munltistack/app/index.html)

either click the link from the index.html page or navigate to http://localhost/aficianado and you should see the aficianado application and you should be able to create read update delete

either click the linkn from the index.html page or navigate to http://localhost/fizzBuzz/countdown.html and it should show the countdown application and you should be able to interact with it.

Troubleshooting

1. Problem: 502 Gateway Error

This is the error you will get if the nginx container can't find the correct IP address for the other container. If you watch the docekr logs you will see that the proxy request url is not being created properly. This can happen if you try to use the port number as in localhost:8000 instead of the network alias name.

2. Problem: Nginx container is exiting before being created.

This can happen if you are trying to create the container without knowledge of how the container's file system currently exists in the image. AKA you assume the image has files that are exactly the same as the Nginx systems when we loaded it on ubuntu server.

To inspect what the iamge looks like you can use the docker exec command

```
docker exec -it <container id> /bin/sh
```

This will get you into the shell of the nginx container to look around at how the image is configured so you can change just what you need to for the proxy. In this case we just changes the default.conf to multistack.conf.

3. Problelm: Docker comtainers exiting when trying to build due to insufficeint space

If you have been running and debugging for a bit, your mysql volume might get clogged with log data. You can prune it through various commands. Be careful with these that you are only deleted what you want to.

```
docker prune -a
```

You can prune volumes to but be sure that is what you want to remove.

4. Proxy only working for the first page and not the additional links in the proxy path.

This looks something like http://localhost/aficianado working, but then when you click add on aficainado it does this in the url http://localhost/create.php and gives a 403 or 404 error. Notice it is not correct. It should got to http://localhos/aficianado/create.php.

You can see the error in the url and more specifically in the docker logs where it will show what directory it is looking for the create.php file to be in.

This can happen if PHP forms are using variables that the Nginx server is interpreting wrong. It will end up looking like a full path instead of a relative path. You can edit the code to make sure it is a relative path.

Resources

1. https://docs.docker.com/get-started/
2. https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04
3. https://docs.docker.com/network/
4. https://www.section.io/engineering-education/dockerized-php-apache-and-mysql-container-development-environment/

Summary

These recipes give a good introduciton to using Nginx as a proxy. The knowledge learned here can be used to set up a proxy to domains that are on different networks.