School of Information Technology and Electrical Engineering
INFS3208 – Cloud Computing

Programming Assignment Task I (10 Marks)

Task description:

Your team manager needs a container-based website to swiftly deploy PHP-based webpages of your company using Nginx (webserver) and PHP (server-side script) in one container. As a developer, you would write a dockerfile to containerise these applications in a container and share with the PHP programmers who are working on the PHP website. Your colleagues should be able to upload their PHP websites to the remote directory in the container and immediately test the web without any problems. There are some technical requirements in your dockerfile as follows:

1. You cannot directly pull the existing docker images from Dockerhub.com (e.g. webdevops/php-nginx). You are only allowed to use Ubuntu 18.04 (Long Term Support) (e.g. ubuntu:18.04) as the base image and build the image upon it.
2. (1 mark) In order to contact the creator of the docker images for your colleagues, you need to **leave the maintainer's information** including your name and email with the learned dockerfile instruction. You also need to **define a variable of PHP version** (e.g. PHPVER = 7.4) and the **time zone**. With different values, the dockerfile can install a specified version of PHP.
3. (1 mark) Nginx can deal with static web pages (e.g. html) but it cannot directly process PHP pages. Thus, it needs FastCGI Process Manager (FPM or PHP-FPM) to process PHP pages. You need to **install Nginx, and PHP-FPM** with a specified version to work for PHP websites.
4. (1 mark) To communicate between Nginx and PHP-FPM, the settings of Nginx are key to make Nginx support PHP-FPM. In your dockerfile, you should be able to **copy the modified Nginx setting file to replace the default one** in the container by using a dockerfile instruction.
5. (1 mark) The testing html and php pages are available on Github (`https://github.com/csenw/cca1`), you need to download them, and **copy them to the root folder** (/var/www/html) of the website in the container with the learned instructions. They **must be runnable** after the container is running.
6. (2.5 marks) To be accessed from outside, you need to (1) **expose and publish the port 80 for Nginx and (2) mount the website's root folder of the website as a volume, (3) set the working directory to** (/var/www/html), (4) **start Nginx and PHP-FPM with the entrypoint command and make sure nginx running in the foreground**. In this way, the php files in the website's root folder can be synchronised immediately if there are changes to the php files in the mounting folder on the host. The PHP programmer need no extra efforts to interact with the container.
7. (2 marks) To build the image from the dockerfile and run the container correctly, you need to **provide related docker commands** with adopted options (e.g., *docker build [options]* and *docker run [options]*) in a separate text file. **Tag** the built image as 'a1:<your surname>' and the container as 'cca1_nginx_php'.
8. (1.5 marks) Last but not least, you need to **follow the good practices** of writing a dockerfile making the image as lean as possible.

Preparation:

In this individual coding assignment, you will apply your knowledge of docker commands, dockerfile instructions (in Lecture 4), and the related fields. Firstly, you should understand what the task is and what the technical requirements include. Secondly, you should practise Linux commands to install Nginx, PHP and PHP-FPM on a VM, which are described in the Appendix and Practical/Tutorial sessions. Lastly, you need to write the dockerfile by converting the Linux commands into dockerfile instructions to achieve the task. You can either practise on the GCP's VM or your local machine (Oracle Virtualbox required) if you are unable to access GCP. Please read the example of writing a dockerfile below to have more details.

Assignment Submission:

- You must compress the dockerfile and **a text file** that contains docker commands to properly build the image and run it.
- The name of the compressed file should be named as "FirstName_LastName_StudentNo.zip".
- You must make an online submission to Blackboard before 1:00 PM on Friday, 02/09/2022.
- Only one extension application could be approved due to medical conditions.

Main Steps:

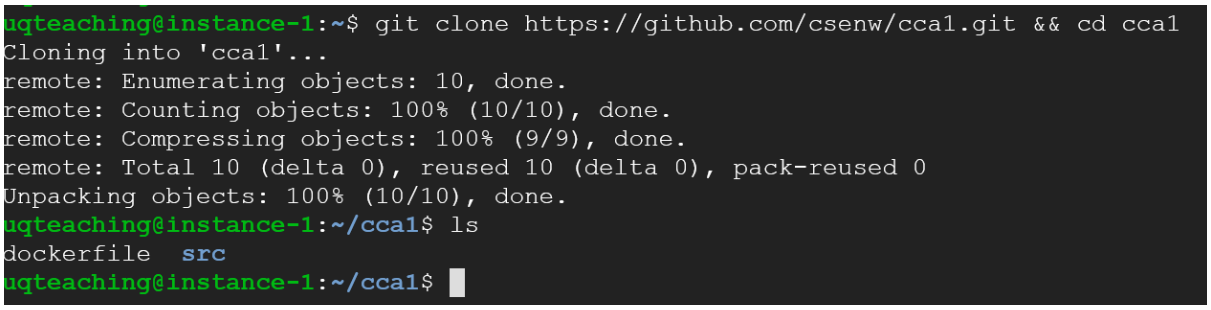
Step 1:

Log in your VM and change to your home directory

Step 2:

```
git clone https://github.com/csenw/cca1.git && cd cca1
```

Run this command to download the required configuration files and testing webpages.

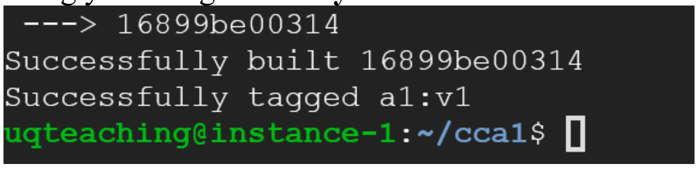


```
uqteaching@instance-1:~$ git clone https://github.com/csenw/cca1.git && cd cca1
Cloning into 'cca1'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 10 (delta 0), reused 10 (delta 0), pack-reused 0
Unpacking objects: 100% (10/10), done.
uqteaching@instance-1:~/cca1$ ls
dockerfile  src
uqteaching@instance-1:~/cca1$
```

In this folder (cca1), please **write your dockerfile**.

Step 3:

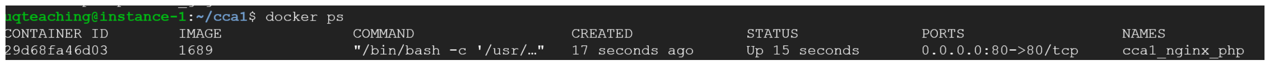
Build your dockerfile and tag your image as **a1:<your surname>.**



```
 ---> 16899be00314
Successfully built 16899be00314
Successfully tagged a1:v1
uqteaching@instance-1:~/cca1$
```

Step 4:

Run the image as a container ‘cca1_nginx_php’:

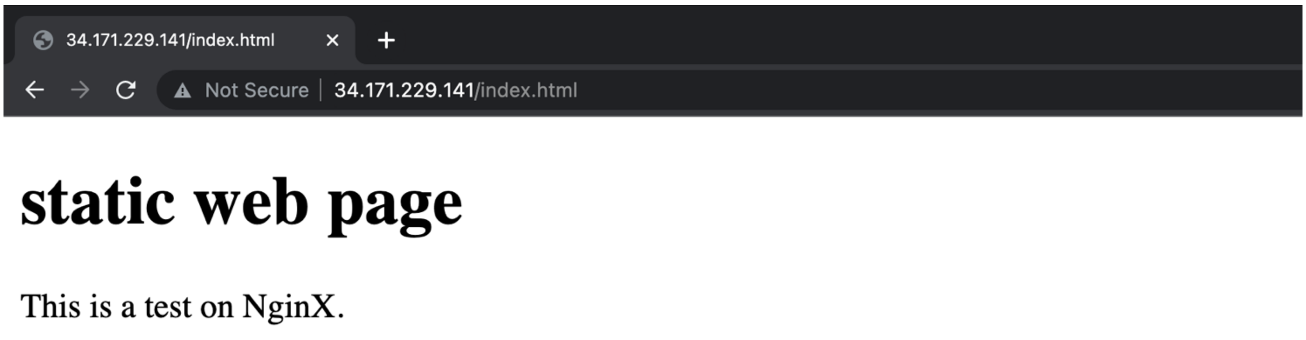


```
uqteaching@instance-1:~/cca1$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                NAMES
29d68fa46d03   1689           "/bin/bash -c '/usr/…"   17 seconds ago   Up 15 seconds   0.0.0.0:80->80/tcp   cca1_nginx_php
```

Step 5:

Test the container on your VM with the preloaded index.html and index.php.

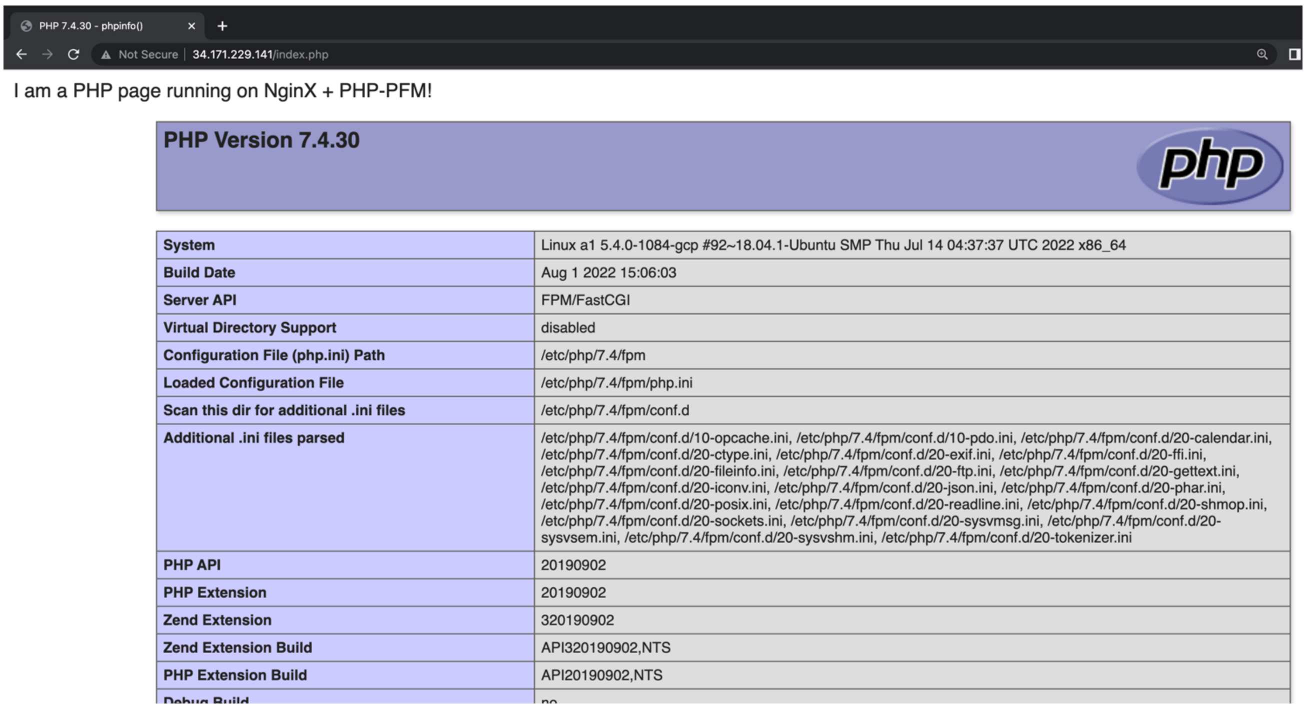
Static webpage test: `http://public_ip/index.html`



static web page

This is a test on NginX.

PHP webpage test: `http://public_ip/index.php`



I am a PHP page running on NginX + PHP-PFM!

PHP Version 7.4.30

System	Linux a1 5.4.0-1084-gcp #92~18.04.1-Ubuntu SMP Thu Jul 14 04:37:37 UTC 2022 x86_64
Build Date	Aug 1 2022 15:06:03
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/fpm
Loaded Configuration File	/etc/php/7.4/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/fpm/conf.d
Additional .ini files parsed	/etc/php/7.4/fpm/conf.d/10-opcache.ini, /etc/php/7.4/fpm/conf.d/10-pdo.ini, /etc/php/7.4/fpm/conf.d/20-calendar.ini, /etc/php/7.4/fpm/conf.d/20-ctype.ini, /etc/php/7.4/fpm/conf.d/20-exif.ini, /etc/php/7.4/fpm/conf.d/20-ffi.ini, /etc/php/7.4/fpm/conf.d/20-fileinfo.ini, /etc/php/7.4/fpm/conf.d/20-ftp.ini, /etc/php/7.4/fpm/conf.d/20-gettext.ini, /etc/php/7.4/fpm/conf.d/20-iconv.ini, /etc/php/7.4/fpm/conf.d/20-json.ini, /etc/php/7.4/fpm/conf.d/20-phar.ini, /etc/php/7.4/fpm/conf.d/20-posix.ini, /etc/php/7.4/fpm/conf.d/20-readline.ini, /etc/php/7.4/fpm/conf.d/20-shmop.ini, /etc/php/7.4/fpm/conf.d/20-sockets.ini, /etc/php/7.4/fpm/conf.d/20-sysvmsg.ini, /etc/php/7.4/fpm/conf.d/20-sysvsem.ini, /etc/php/7.4/fpm/conf.d/20-sysvshm.ini, /etc/php/7.4/fpm/conf.d/20-tokenizer.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902,NTS
PHP Extension Build	API20190902,NTS
Debug Build	no

Appendix

Nginx (pronounced "engine X"), stylized as NGINX or Nginx or NginX, is a web server that can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache. The software was created by Igor Sysoev and publicly released in 2004. Nginx is free and open-source software, released under the terms of the 2-clause BSD license. A large fraction of web servers uses NGINX, often as a load balancer. Nginx can be deployed to serve dynamic HTTP content on the network using FastCGI, SCGI handlers for scripts, WSGI application servers, and it can serve as a software load balancer. Note that Nginx cannot directly deal with PHP but can serve PHP applications through the FastCGI protocol. Nginx employs PHP-FPM (FastCGI Process Manager) that is running in the background as a daemon and listening for CGI requests. Nginx uses an asynchronous event-driven approach, rather than threads, to handle requests. Nginx's modular event-driven architecture can provide more predictable performance under high loads. Nginx default configuration file is nginx.conf

PHP [2] is a general-purpose scripting language that is especially suited to web development. It was created by Danish-Canadian programmer Rasmus Lerdorf in 1994; the PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the recursive initialism PHP: Hypertext Preprocessor. PHP code is usually processed on a web server by a PHP interpreter implemented as a module, a daemon or as a Common Gateway Interface (CGI) executable. On a web server, the result of the interpreted and executed PHP code – which may be any type of data, such as generated HTML or binary image data – would form the whole or part of an HTTP response. Various web template systems, web content management systems, and web frameworks exist which can be employed to orchestrate or facilitate the generation of that response. Additionally, PHP can be used for many programming tasks outside of the web context, such as standalone graphical applications and robotic drone control. Arbitrary PHP code can also be interpreted and executed via the command-line interface (CLI).

PHP-FPM (FastCGI Process Manager) [2] is an alternative FastCGI implementation for PHP, bundled with the official PHP distribution since version 5.3.3. When compared to the older FastCGI implementation, it contains some additional features, mostly useful for heavily loaded web servers. Figure 1 shows how PHP-FPM helps Nginx process PHP pages.

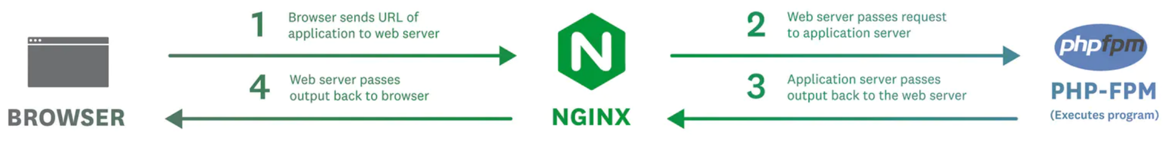

BROWSER
1 Browser sends URL of application to web server
4 Web server passes output back to browser
NGINX
2 Web server passes request to application server
3 Application server passes output back to the web server
phpfpm
PHP-FPM
(Executes program)

Figure 1.

How PHP and Nginx work together (Image credit: DataDog)

[1] Nginx, https://en.wikipedia.org/wiki/Nginx
[2] PHP, https://en.wikipedia.org/wiki/PHP

An example of installation steps on a Linux VM with the following configuration:

- OS: Ubuntu 18.04,
- Network: using dynamical IP and Port 80
- PHP-FPM version: 7.4

A1 How to install Nginx and PHP7.4-FPM

The installation steps of Nginx, PHP, and PHP-FPM **on a VM** will be described in this section. You can practise installing the entire framework on your VM. Afterwards, you need to **convert** installation steps on your VM into dockerfile instructions.

Step 1: Install build dependencies and libraries.

```
sudo apt-get update
```

Run this command periodically to make sure your source list (/etc/apt/sources.list) is up-to-date. This is the equivalent of "Reload" in Synaptic or "Fetch updates" in Adept.

```
sudo apt-get install software-properties-common
```

Run this command to install build dependencies.

Step 2: Install Nginx, and PHP-FPM

```
sudo add-apt-repository ppa:ondrej/php && sudo apt-get update
```

Run this command to add the repository of php (https://launchpad.net/~ondrej/+archive/ubuntu/php), where can find the latest version of PHP, and update the package list.

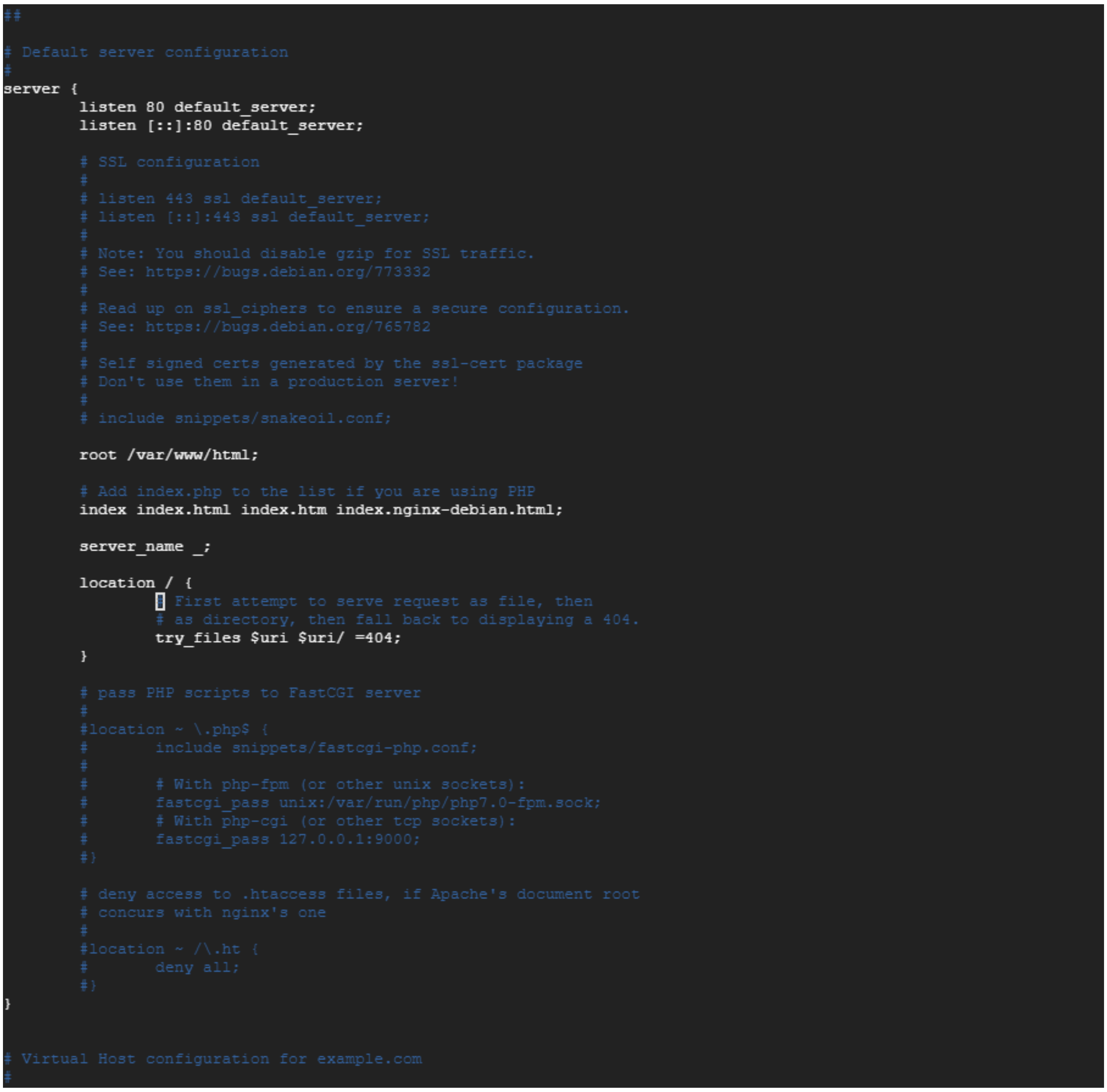
```
sudo apt-get install nginx php7.4-fpm
```

Run this command to install Nginx and PHP-FPM 7.4.

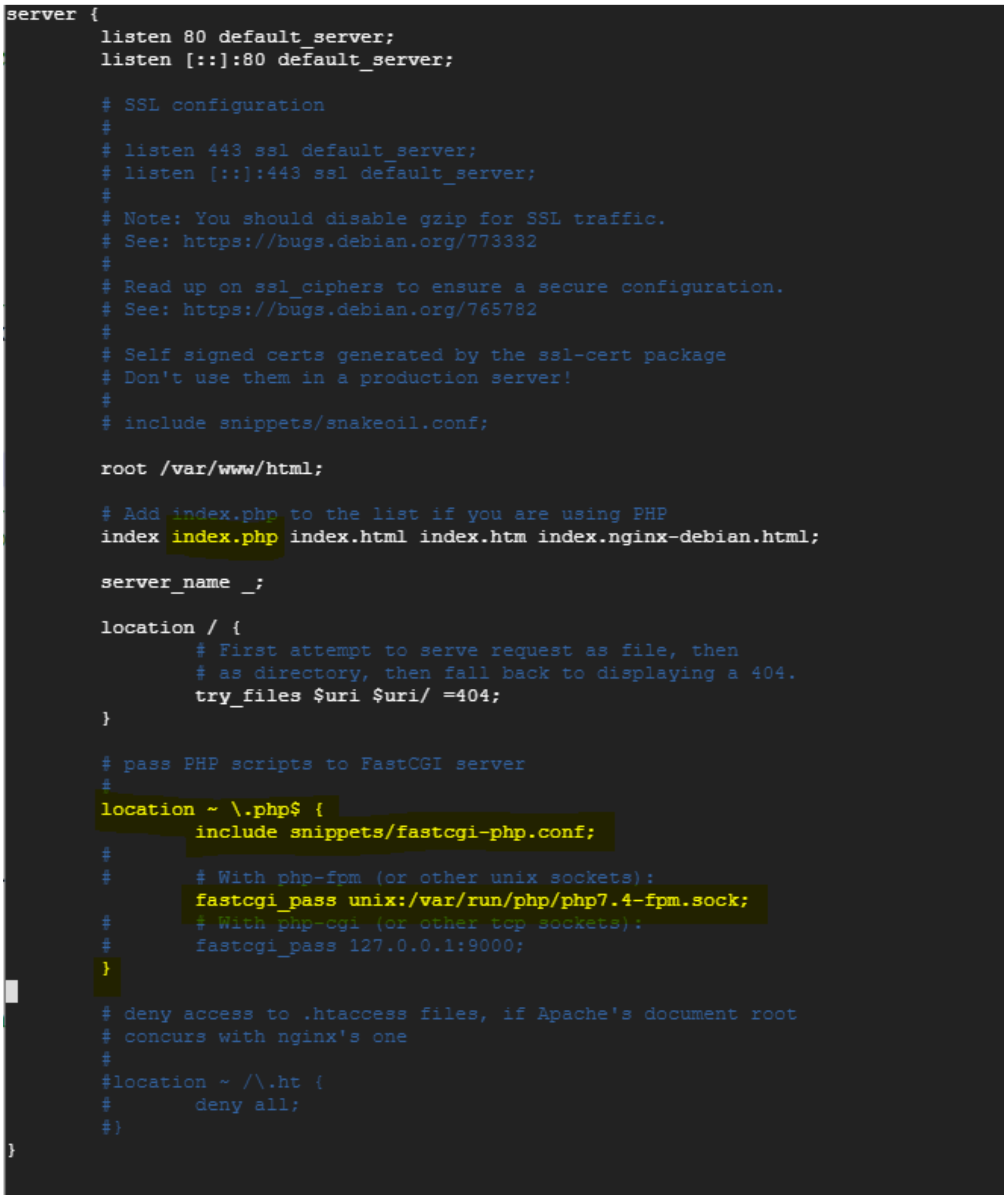
Step 3: Setup Nginx

With **SUDO**, open `/etc/nginx/sites-available/default`, and make the following modifications to enable php-fpm:

We have provided a sample configuration file in https://github.com/csenw/cca1/tree/master/src.



```
##

# Default server configuration
#
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configuration.
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        server_name _;

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

        # pass PHP scripts to FastCGI server
        #
        #location ~ \.php$ {
        #       include snippets/fastcgi-php.conf;
        #
        #       # With php-fpm (or other unix sockets):
        #       fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
        #}

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        #       deny all;
        #}
}


# Virtual Host configuration for example.com
#
```



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

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configuration.
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.php index.html index.htm index.nginx-debian.html;

        server_name _;

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

        # pass PHP scripts to FastCGI server
        #
        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
        #
        #       # With php-fpm (or other unix sockets):
                fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
        }

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        #       deny all;
        #}
}
```

Note that the highlighted contents are the configuration you need to modify.

```
        root /var/www/html;


        # Add index.php to the list if you are using PHP
        index index.php index.html index.htm index.nginx-debian.html;
        server_name _;

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

        # pass PHP scripts to FastCGI server
        #
        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
        #
        #       # With php-fpm (or other unix sockets):
                fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        }

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        #       deny all;
        #}
```

Step 5: Run PHP-FPM and Nginx

```
sudo service php7.4-fpm start
```

Run this command to start `php7.4-fpm`. Note that `php7.4-fpm` will not automatically run as a system service after installation. To apply the new nginx configuration, you can restart nginx service with:

```
sudo service nginx restart
```

Step 6: Test static and dynamic webpages

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

Run this command to change directory to the root of the website.

With **SUDO**, add the following html code to a html file (index.html) with an editor (e.g. vim)

```
<!DOCTYPE html>

<html>

        <body>

                <h1>static web page</h1>

                <p>This is a test on NginX.</p>

        </body>

</html>
```

Add the following php code to a php file (index.php) with an editor (e.g., vim)

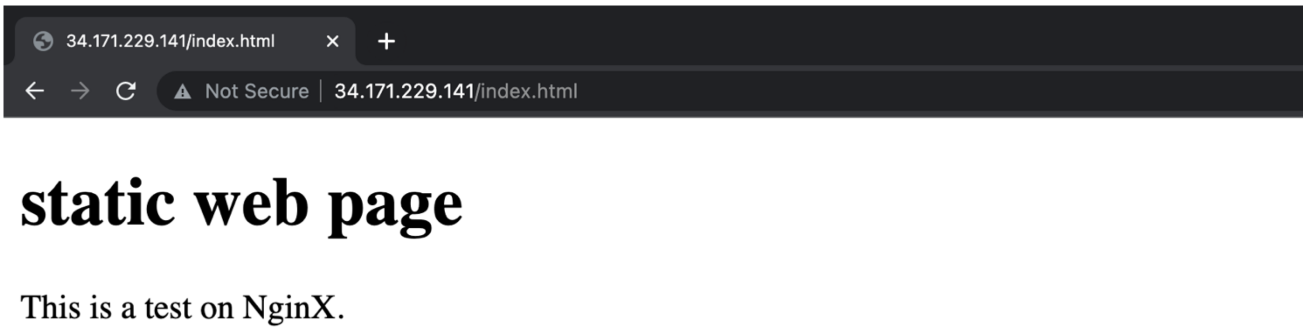
```
<?php

echo "I am a PHP page running on NginX + PHP-PFM!";

phpinfo(); ?>
```

Test the static webpage and php webpage with the address:

- `http://public_ip/index.html`
- `http://public_ip/index.php`

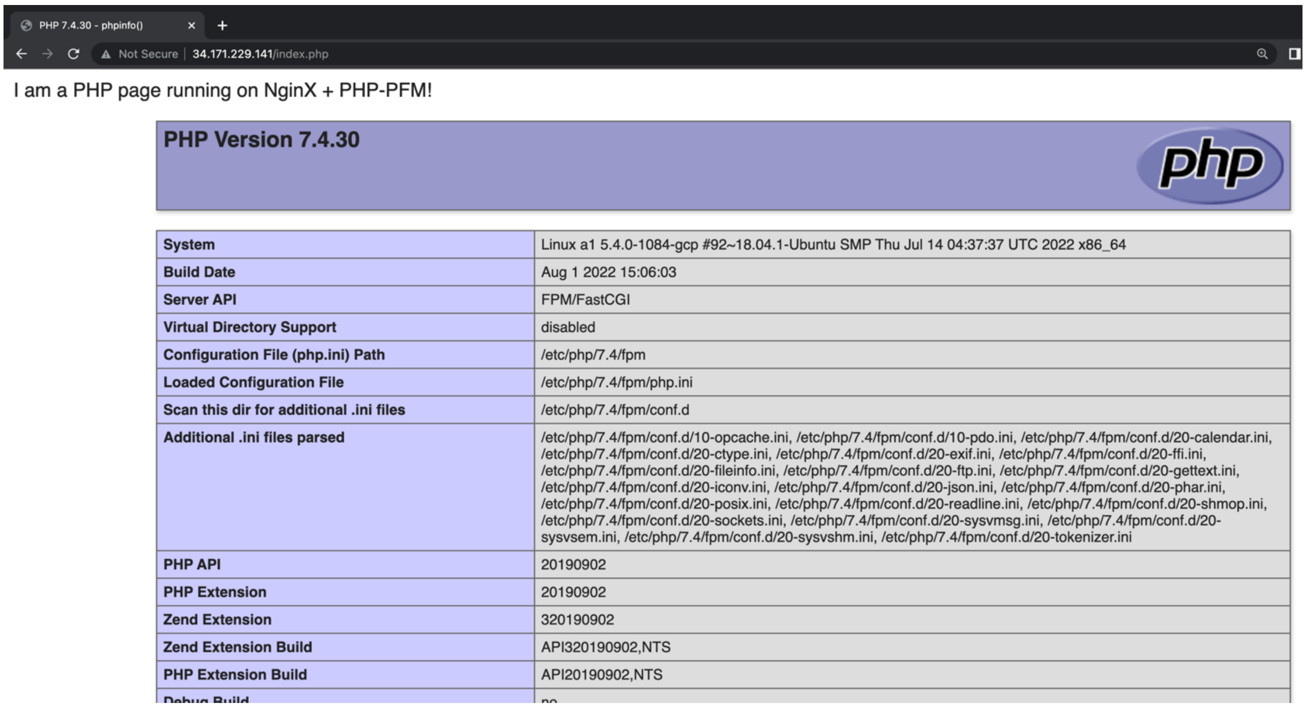
The public_ip is your VM's IP on GCP.

Well done, you have completed this installation and can run any html/php pages on your VM.



static web page

This is a test on NginX.



I am a PHP page running on NginX + PHP-PFM!

PHP Version 7.4.30

System	Linux a1 5.4.0-1084-gcp #92~18.04.1-Ubuntu SMP Thu Jul 14 04:37:37 UTC 2022 x86_64
Build Date	Aug 1 2022 15:06:03
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/fpm
Loaded Configuration File	/etc/php/7.4/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/fpm/conf.d
Additional .ini files parsed	/etc/php/7.4/fpm/conf.d/10-opcache.ini, /etc/php/7.4/fpm/conf.d/10-pdo.ini, /etc/php/7.4/fpm/conf.d/20-calendar.ini, /etc/php/7.4/fpm/conf.d/20-ctype.ini, /etc/php/7.4/fpm/conf.d/20-exif.ini, /etc/php/7.4/fpm/conf.d/20-ffi.ini, /etc/php/7.4/fpm/conf.d/20-fileinfo.ini, /etc/php/7.4/fpm/conf.d/20-ftp.ini, /etc/php/7.4/fpm/conf.d/20-gettext.ini, /etc/php/7.4/fpm/conf.d/20-iconv.ini, /etc/php/7.4/fpm/conf.d/20-json.ini, /etc/php/7.4/fpm/conf.d/20-phar.ini, /etc/php/7.4/fpm/conf.d/20-posix.ini, /etc/php/7.4/fpm/conf.d/20-readline.ini, /etc/php/7.4/fpm/conf.d/20-shmop.ini, /etc/php/7.4/fpm/conf.d/20-sockets.ini, /etc/php/7.4/fpm/conf.d/20-sysvmsg.ini, /etc/php/7.4/fpm/conf.d/20-sysvsem.ini, /etc/php/7.4/fpm/conf.d/20-sysvshm.ini, /etc/php/7.4/fpm/conf.d/20-tokenizer.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902,NTS
PHP Extension Build	API20190902,NTS
Debug Build	no