School of Electrical Engineering and Computer Science
INFS3208 – Cloud Computing

Programming Assignment Task II (10 Marks)

Background:

With your Dockerfile in the first assignment, you can develop a simple PHP-based website and swiftly deploy the lightweight website to the customer's machine without worrying about environmental issues (e.g., software and hardware compatibility issues). Now, your manager has decided to adopt micro-services architecture in this project, which is shown in Figure 1:

1. **Micro-service A** (Nginx) is a web server with load balancing to respond to client requests.
2. **Micro-service B** (order.php) accepts all online order requests. To handle massive orders in a short period, MS-B pushes all the order requests onto a message queue and only confirms the products that have been ordered.
3. **Micro-service C** (Redis) is used as a simple message queue (FIFO), storing all the messages sent by Micro-service A in a queue.
4. **Micro-service D** (process.php) will periodically process the orders in the queue in MS-C. For each order that pops out from the message queue in MS-C, MS-D will check the product availability by sending queries to the main product database.
5. **Micro-service E** (MariaDB) is the main product database that stores the information of all the products (stock, price, etc.).
6. **Micro-service F** (phpMyAdmin) is one of the most popular PHP-based MySQL administration tools, which supports a wide range of operations like managing databases, relations, tables, columns, indexes, permissions, users, etc. via a graphical user interface.

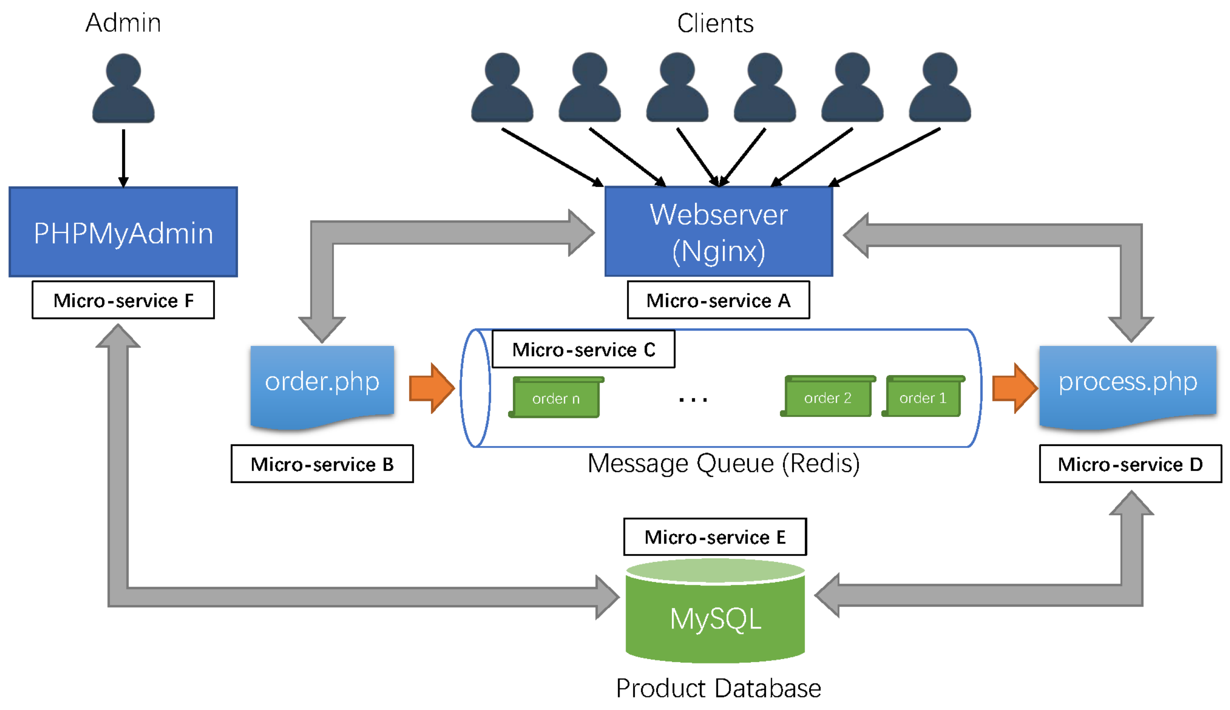


Figure 1: Micro-service architecture.

Task Description for docker-compose.yml File:

In this project, you are asked to write a docker-compose.yml file to orchestrate the FIVE containers (MariaDB, Redis, PHP-FPM, Nginx and phpMyAdmin). You should be able to upload your PHP websites to the remote directory in the container and immediately test the web development without any problem. You will be provided with four web pages (`index.html`, `index.php`, `order.php`, and `process.php`), which are used to test whether the deployed containers work properly. The expected results of the testing web pages are shown in Figures 2 - 7.

Objectives:

1. **Define a Network for Communications among Containers (0.5 marks):**
   - Define a network with the **bridge driver** named "mynet" to enable communications among all containers (0.5 marks).
   - The dependencies among the containers should also be defined (refer to Figure 8 in the Appendix).
2. **Deploy the MariaDB Container (0.75 mark):**
   - Use the latest MariaDB image to run the container and name it "mysql" (0.25 marks).
   - Define the environment variables for the database as follows (0.5 marks):
     - `MYSQL_ROOT_PASSWORD` = `MyDBRoot123`
     - `MYSQL_DATABASE` = `cloud_computing`
     - `MYSQL_USER` = `php`
     - `MYSQL_PASSWORD` = `php`
   - Connect the MariaDB container to the predefined network "mynet".
3. **Deploy the Redis Container (0.25 marks):**
   - Use the latest Redis image to run the container and name it "myredis" (0.25 marks).
   - Connect the Redis container to the predefined network "mynet".
4. **Deploy the PHP-FPM Container (1.75 marks):**
   - Build an image based on the Dockerfile in `cca2/src/php/dockerfile`. Please refer to the **Task Description for Dockerfile** below for details.
   - Use the built image to run the container and name it "myphp" (0.25 marks).
   - Expose port 9000 (0.5 marks).
   - Mount the directory of web pages (i.e., `cca2/src`) to the root directory of web pages in the container (i.e., `/var/www/html`) (0.5 marks).
   - As PHP-FPM needs to communicate with MariaDB and Redis, this container ("myphp") should depend on "mysql" and "myredis" (0.5 marks).
   - Connect the PHP-FPM container to the predefined network "mynet".
   - Please note: The official PHP-FPM does not include the extensions (e.g., Redis and MySQL), the test pages `order.php` and `process.php` **would not work without the extensions installed.** Ensure the required extensions (i.e., `redis` and `mysqli`) are installed in the Dockerfile so that `order.php` and `process.php` can work properly.

5. **Deploy the Nginx Container (1.75 marks):**
   - Use the latest Nginx image to run the container and name it "mynginx" (0.25 marks).
   - Map the internal port 80 to the external port 8080 (0.5 marks).
   - Copy the configuration file to the correct location in this container (0.25 marks).
     - Copy `cca2/src/nginx.ini` to `/etc/nginx/conf.d/default.conf`
   - Mount the directory of webpages (i.e., `cca2/src`) to the root directory of webpages in the container (i.e., `/var/www/html`) (0.25 marks).
   - This container ("mynginx") should depend on "myphp" (0.5 marks).
   - Connect the Nginx container to the predefined network "mynet".

6. **Deploy the phpMyAdmin Container (2 marks):**
   - Use the latest phpMyAdmin image to run the container and name it "phpmyadmin" (0.25 marks).
   - Map the internal port 80 to the external port 8082 (0.5 marks).
   - Specify a MySQL host in the "phpmyadmin" container using the hostname of "mysql". Details of the database `cloud_computing` created in the "mysql" container should be visible on the phpMyAdmin page (1.25 mark).
   - Connect the phpMyAdmin container to the predefined network "mynet".

7. **Completeness (2 marks):**
   - To test the completeness of your solution, visit all the testing web pages as shown in Figure 2 - 7 (0.5 marks for each webpage among `index.php`, `order.php`, `process.php` and `phpMyAdmin`).

Task Description for Dockerfile:

To successfully start these services by the docker-compose file, you will need a Dockerfile to properly set up all necessary extensions in the "myphp" container. You should complete the Dockerfile located at `cca2/src/php/dockerfile` to set up a PHP 8.2 environment with specific extensions. The desired environment is a PHP 8.2 with FastCGI Process Manager (FPM) support and should have the capability to communicate efficiently with both Redis and MySQL databases.

Objectives for Dockerfile:

1. **Base Image:**
   - Use an official PHP image with version 8.2 and FPM support (as the provided base image in the Dockerfile).

2. **PHP Extensions (1 mark):**
   - Integrate the igbinary extension for optimized serialization/deserialization. Incorporate the redis extension to facilitate PHP-Redis communication. Include the mysqli extension for MySQL database support in PHP (1 mark).
   - Ensure that the installed extensions are properly configured and loaded every time the "myphp" container starts.

Preparation:

In this individual coding assignment, you will apply your knowledge of docker-compose instructions (in Lecture 5) and the related fields.

- Firstly, you should read the **Background** and **Task Descriptions** to understand the background, motivations, objectives, and technical requirements.
- Secondly, you should practice docker commands and Linux commands to manually orchestrate the containers on your VM instance. Afterwards, you need to convert those deployment steps into a docker-compose file with the correct instructions. Notice that the deployment of Nginx, PHP-FPM, Redis, and MariaDB on a VM is described in **Practical 5**, while the setup of PHP extensions and the installation of phpMyAdmin are NOT included. You need to investigate how to create a Dockerfile to ensure `order.php` and `process.php` can be successfully running and run a phpMyAdmin docker container by yourself.
- Lastly, you need to write a `dockerfile` and a `docker-compose.yml` by converting the docker commands and Linux commands into docker-compose instructions. **All technical requirements need to be fully met to achieve full marks.**

You can practice either on the GCP's VM or your local machine (requiring Oracle VirtualBox or Windows Subsystem for Linux) if you are unable to access GCP.

Assignment Submission:

- You must compress the **docker-compose.yml** file and the **dockerfile**.
- The name of the compressed file should be named "FirstName_LastName_StudentNo.zip".
- You must make an online submission to Blackboard **before 3:00 PM on Friday, 19/09/2025**.
- Only one extension application could be approved due to medical conditions.

Main Steps:

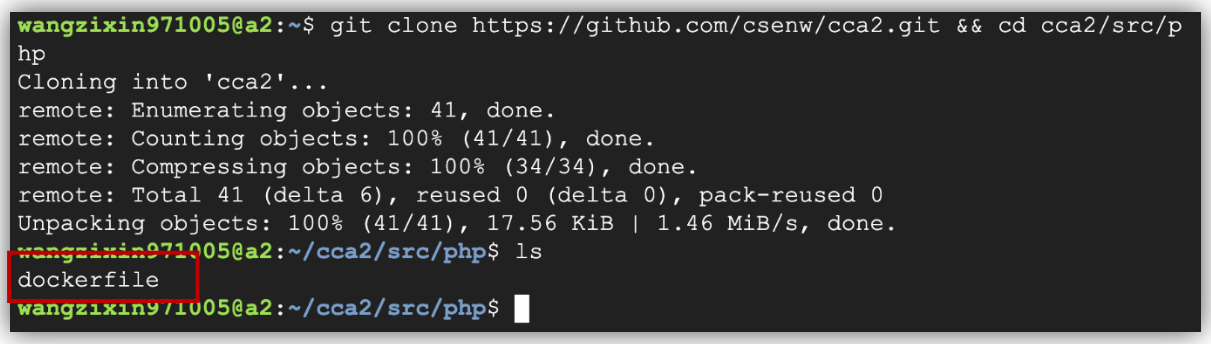
Step 1:

Log in to your VM and change to your home directory

Step 2:

```
git clone https://github.com/csenw/cca2.git && cd cca2/src/php
```

Run this command to download the required configuration file and testing web pages.



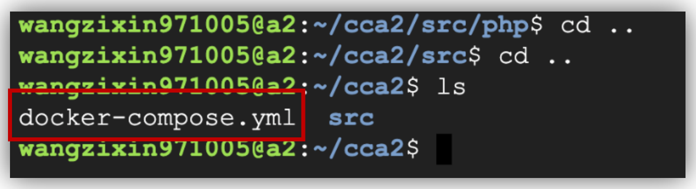
```
wangzixin971005@a2:~$ git clone https://github.com/csenw/cca2.git && cd cca2/src/p
hp
Cloning into 'cca2'...
remote: Enumerating objects: 41, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 41 (delta 6), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (41/41), 17.56 KiB | 1.46 MiB/s, done.
wangzixin971005@a2:~/cca2/src/php$ ls
dockerfile
wangzixin971005@a2:~/cca2/src/php$
```

In the folder `cca2/src/php/`, please **write the dockerfile to ensure the order.php and process.php can be shown without any error (1 mark).**

Step 3:

```
cd ../.. && ls
```

In the folder `cca2/`, please **write your docker-compose.yml (7 marks).**

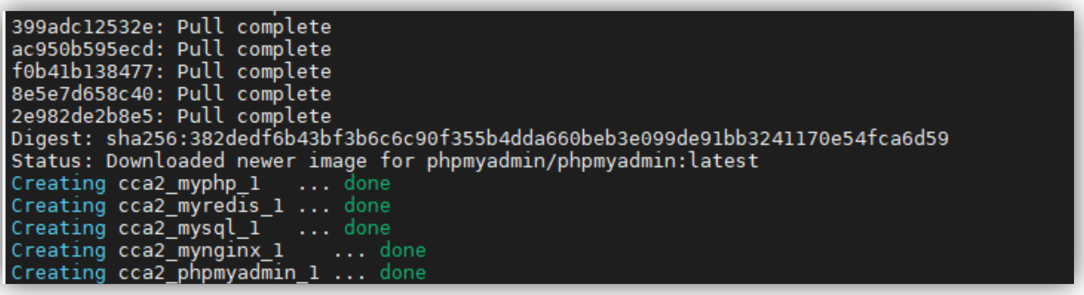


```
wangzixin971005@a2:~/cca2/src/php$ cd ..
wangzixin971005@a2:~/cca2/src$ cd ..
wangzixin971005@a2:~/cca2$ ls
docker-compose.yml  src
wangzixin971005@a2:~/cca2$
```

Step 4:

```
docker-compose up -d
```

Run all the containers:

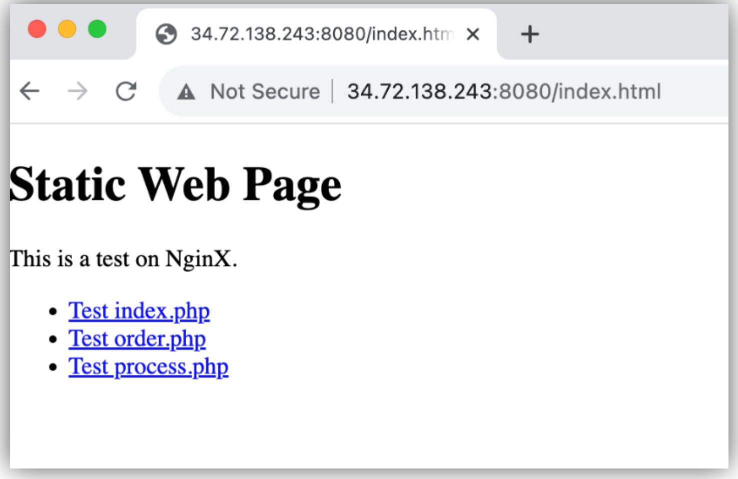


```
399adc12532e: Pull complete
ac950b595ecd: Pull complete
f0b41b138477: Pull complete
8e5e7d658c40: Pull complete
2e982de2b8e5: Pull complete
Digest: sha256:382dedf6b43bf3b6c6c90f355b4dda660beb3e099de91bb3241170e54fca6d59
Status: Downloaded newer image for phpmyadmin/phpmyadmin:latest
Creating cca2_myphp_1    ... done
Creating cca2_myredis_1 ... done
Creating cca2_mysql_1    ... done
Creating cca2_mynginx_1     ... done
Creating cca2_phpmyadmin_1 ... done
```

Step 5:

Test the completeness of your solution. Check each of the pages (2 marks).

Visit the static web page: `http://external_ip:8080/index.html` (ensure the Nginx is running)



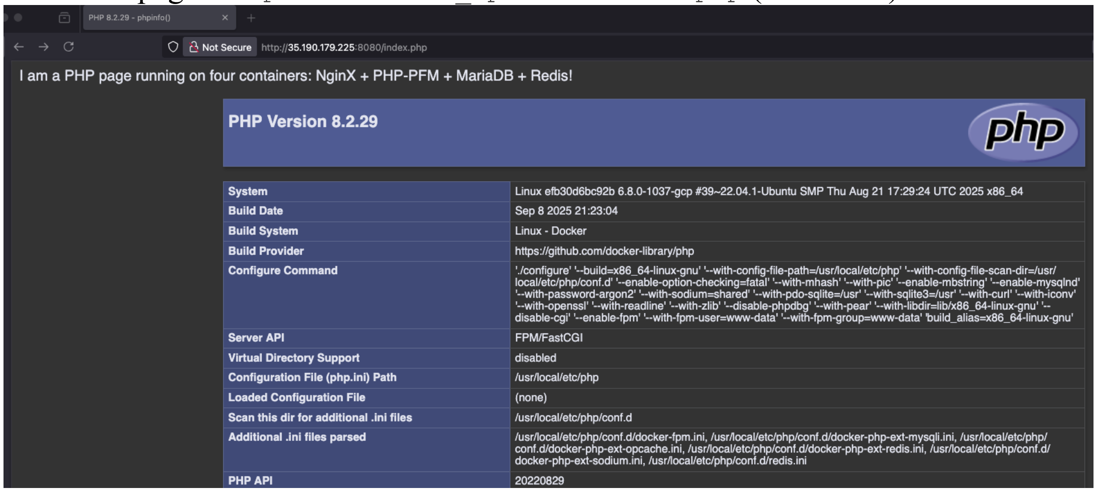
Static Web Page

This is a test on NginX.

- Test index.php
- Test order.php
- Test process.php

Figure 2: A Nginx test page.

Test the PHP web page: `http://external_ip:8080/index.php` (0.5 marks)



PHP 8.2.29 - phpinfo()

Not Secure http://35.190.179.225:8080/index.php

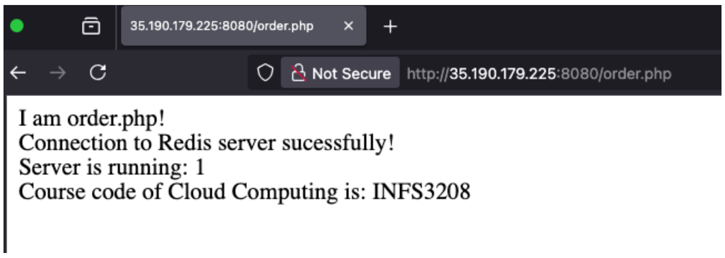
I am a PHP page running on four containers: NginX + PHP-PFM + MariaDB + Redis!

PHP Version 8.2.29

System	Linux efb30d6bc92b 6.8.0-1037-gcp #39~22.04.1-Ubuntu SMP Thu Aug 21 17:29:24 UTC 2025 x86_64
Build Date	Sep 8 2025 21:23:04
Build System	Linux - Docker
Build Provider	https://github.com/docker-library/php
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--with-pic' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-iconv' '--with-openssl' '--with-readline' '--with-zlib' '--disable-phpdbg' '--with-pear' '--with-libdir=lib/x86_64-linux-gnu' '--disable-cgi' '--enable-fpm' '--with-fpm-user=www-data' '--with-fpm-group=www-data' 'build_alias=x86_64-linux-gnu'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-fpm.ini, /usr/local/etc/php/conf.d/docker-php-ext-mysqli.ini, /usr/local/etc/php/conf.d/docker-php-ext-opcache.ini, /usr/local/etc/php/conf.d/docker-php-ext-redis.ini, /usr/local/etc/php/conf.d/docker-php-ext-sodium.ini, /usr/local/etc/php/conf.d/redis.ini
PHP API	20220829

Figure 3: A PHP-FPM test page.

Test the connection between PHP-FPM and Redis: `http://external_ip:8080/order.php` (0.5 marks)



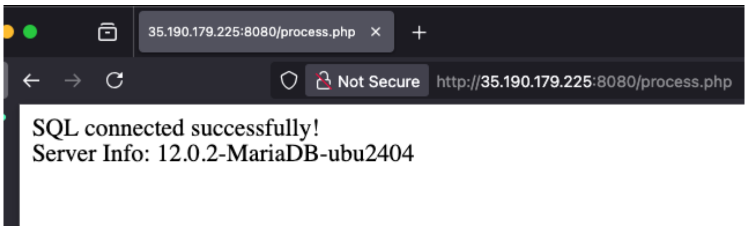
35.190.179.225:8080/order.php

Not Secure http://35.190.179.225:8080/order.php

```
I am order.php!
Connection to Redis server sucessfully!
Server is running: 1
Course code of Cloud Computing is: INFS3208
```

Figure 4: A Redis test page.

Test the connection between PHP-FPM and MariaDB: `http://external_ip:8080/process.php` (0.5 marks)



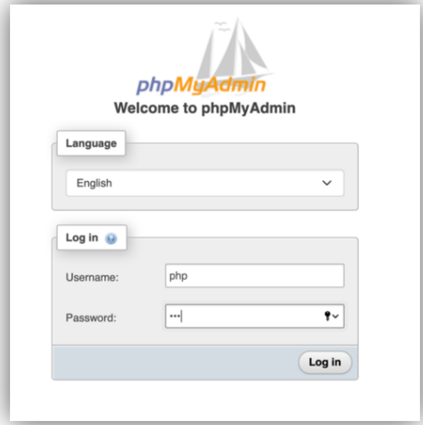
35.190.179.225:8080/process.php

Not Secure http://35.190.179.225:8080/process.php

```
SQL connected successfully!
Server Info: 12.0.2-MariaDB-ubu2404
```

Figure 5: MariaDB test page.

Test phpMyAdmin: `http://external_ip:8082` (0.5 marks)



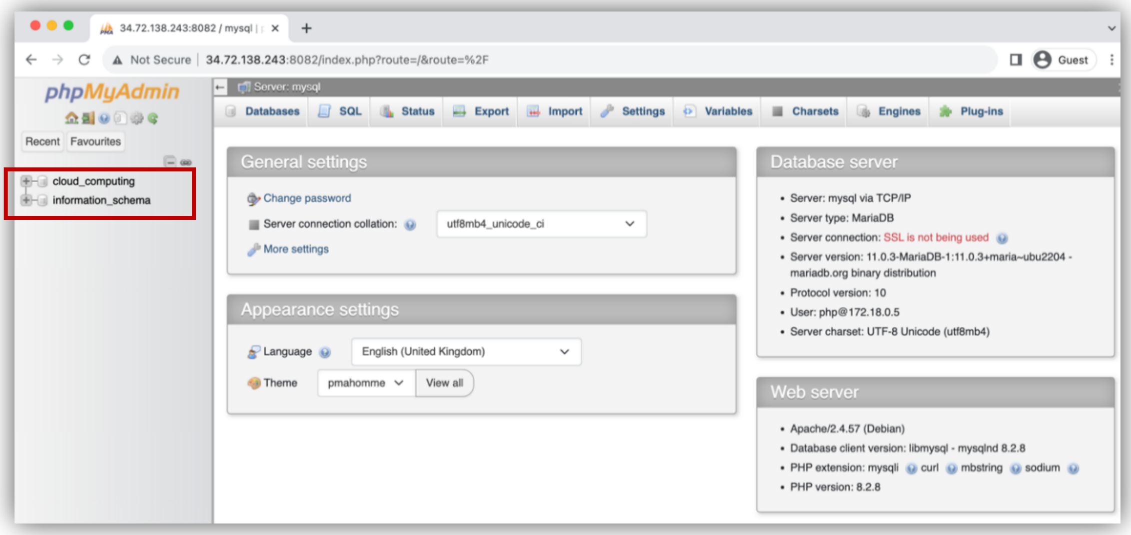
Language: English

Username: php

Password: •••

Figure 6: A phpMyAdmin login page.

The created database “cloud_computing” can be viewed in the sidebar after logging in.



cloud_computing
information_schema

General settings

Server connection collation: utf8mb4_unicode_ci

Appearance settings

Language: English (United Kingdom)

Theme: pmahomme

Database server

- Server: mysql via TCP/IP
- Server type: MariaDB
- Server connection: SSL is not being used
- Server version: 11.0.3-MariaDB-1:11.0.3+maria~ubu2204 - mariadb.org binary distribution
- Protocol version: 10
- User: php@172.18.0.5
- Server charset: UTF-8 Unicode (utf8mb4)

Web server

- Apache/2.4.57 (Debian)
- Database client version: libmysql - mysqlnd 8.2.8
- PHP extension: mysqli curl mbstring sodium
- PHP version: 8.2.8

Figure 7: A phpMyAdmin test page.

Appendix

MariaDB [1] is a community-developed, commercially supported fork of the MySQL relational database management system (RDBMS), intended to remain free and open-source software under the GNU General Public License. Development is led by some of the original developers of MySQL, who forked it due to concerns over its acquisition by Oracle Corporation in 2009. MariaDB intended to maintain high compatibility with MySQL, ensuring a drop-in replacement capability with library binary parity and exact matching with MySQL APIs and commands. However, new features diverge more. It includes new storage engines like Aria, ColumnStore, and MyRocks. You can regard MariaDB as an alternative to MySQL and feel free to interchangeably use these free relational databases in your project.

Redis (Remote Dictionary Server) [2] is an in-memory data structure project implementing a distributed, in-memory key-value database with optional durability. Redis supports different kinds of abstract data structures, such as strings, lists, maps, sets, sorted sets, HyperLogLogs, bitmaps, streams, and spatial indexes. The project is mainly developed by Salvatore Sanfilippo and as of 2019 is sponsored by Redis Labs. It is open-source software released under a BSD 3-clause license.

phpMyAdmin [3] is a free and open-source administration tool for MySQL and MariaDB. As a portable web application written primarily in PHP, it has become one of the most popular MySQL administration tools, especially for web hosting services.

For information about Nginx, PHP, and PHP-PFM, please refer to Appendix in Assignment I.

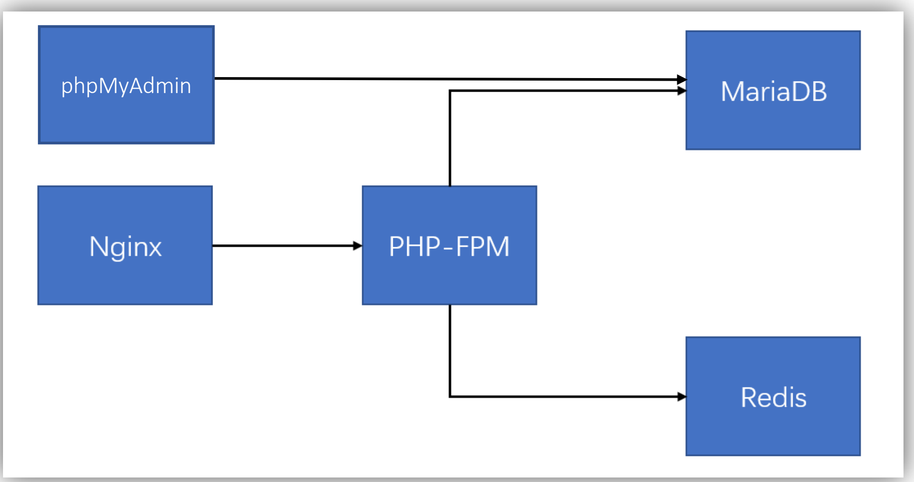


Figure 8. The dependencies among five containers: Nginx + PHP + MariaDB + phpMyAdmin + Redis.

Reference:
[1] MariaDB, https://en.wikipedia.org/wiki/MariaDB
[2] Redis, https://en.wikipedia.org/wiki/PHP
[3] phpMyAdmin, https://en.wikipedia.org/wiki/PhpMyAdmin