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 environment issues (e.g., software and hardware compatibility issues). Now, your manager decides 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 popped 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.

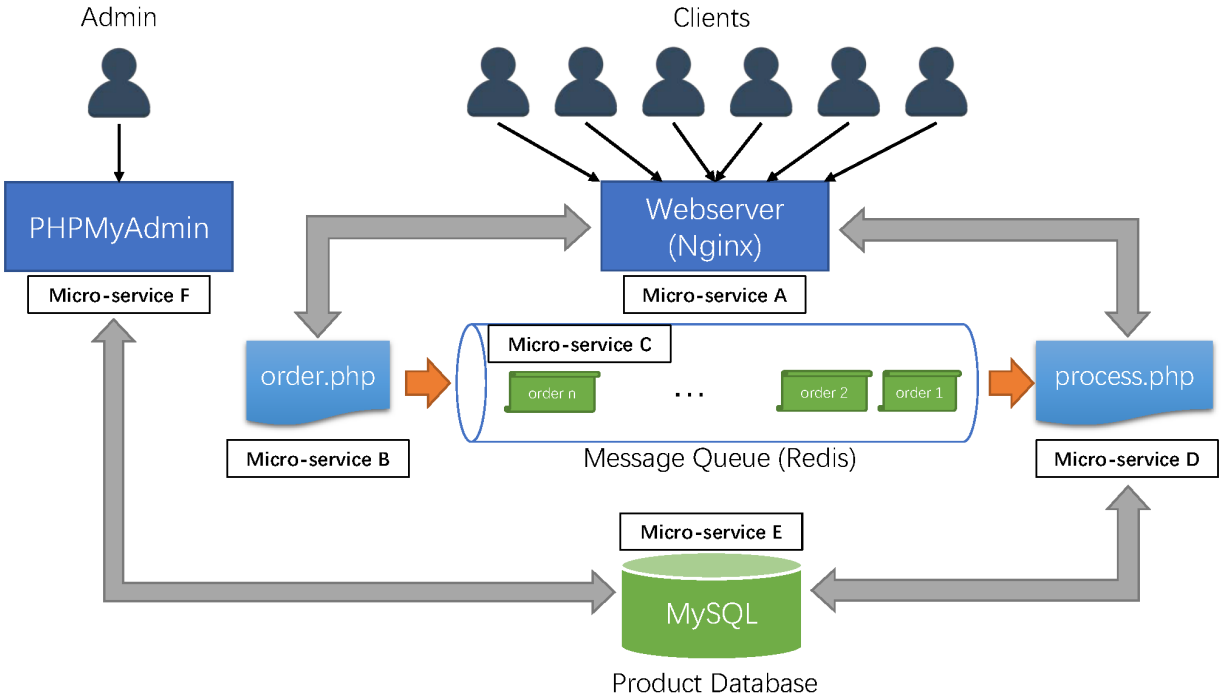


Figure 1: Micro-service architecture.

Task Description for docker-compose.yml:

In this project, you are asked to write a docker-compose.yml file to orchestrate FIVE containers (Nginx, PHP-FPM, phpMyAdmin, Redis, and MariaDB). You should be able to upload your PHP websites to the remote directory in the container and immediately test the web development without any problems. There are some technical requirements for your docker-compose file as follows:

1. First, you need to use `git` to download the configuration files and test web pages. In your docker-compose file, you should copy the given configuration files for Nginx and PHP-FPM to the corresponding places inside the docker containers by using docker-compose instructions.
2. Apart from the configuration files, you will be given four web pages (`index.html`, `index.php`, `order.php`, and `process.php`). They are used to test whether the deployed containers work properly. The expected results of the test web pages are shown in Figures 2-7.
3. For each container, you can choose to either pull an existing official image from the Docker Hub or create a customised image.
4. In your docker-compose file, you need to define a **network** with the **bridge** driver named as "mynet" to enable communications among all containers. The **dependencies** among the containers should also be defined (refer to Figure 8 in the Appendix).
5. For the **MariaDB** container:
    a. Name the container as "mysql".
    b. Define the environment variables for the database as follows:
        i. `MYSQL_ROOT_PASSWORD` = `MyDBRoot123`
        ii. `MYSQL_DATABASE` = `cloud_computing`
        iii. `MYSQL_USER` = `php`
        iv. `MYSQL_PASSWORD` = `php`
    c. Connect the container to the predefined network "mynet".
6. For the **Redis** container:
    a. Name the container as "myredis".
    b. Connect the Redis container to the predefined network "mynet".
7. For the **PHP-FPM** container:
    a. This service **should be built from `cca2/src/php/dockerfile`.**
    b. Name the container as "myphp" and expose port `9000`.
    c. Mount the directory of the web pages (i.e., `cca2/src`) to the root directory of the website in the container (i.e., `/var/www/html`)
    d. As PHP-FPM needs to communicate with MariaDB and Redis, this container ("myphp") needs to depend on "mysql" and "myredis".
    e. As 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 (i.e., `cca2/src/php/dockerfile`) so that `order.php` and `process.php` can work properly. See **Task Description for Dockerfile** below for the instructions.
    f. Connect the PHP-FPM container to the predefined network "mynet".
8. For the Nginx container,
    a. Name the container as "mynginx" and map the external port `8080` to internal port `80`.
    b. Copy the prepared configuration files to the correct locations in this container.
        i. Copy `nginx.ini` to `/etc/nginx/conf.d/default.conf`
    c. Mount the directory of the web pages (i.e., `cca2/src`) to the root directory of the website in the container (i.e., `/var/www/html`).
    d. This container ("mynginx") needs to depend on "myphp".
    e. Connect the Nginx container to the predefined network "mynet".

9. For the phpMyAdmin container,
    a. Name the container as “phpMyAdmin” and map the external port 8082 to the internal port 80.
    b. Specify a MySQL host in the phpMyAdmin container using the hostname of “mysql”. The details of the database “cloud_computing” created in the “mysql” container can be viewed on the phpMyAdmin page.
    c. Connect the phpMyAdmin container to the predefined network “mynet”.

The installation steps of Nginx, PHP-FPM, Redis, and MariaDB on a VM is describe in **Practical 5**. You can practise installing the entire framework on your VM first. Afterwards, you need to convert those installation steps into a docker-compose file with the correct instructions. Notice that the setup of PHP and the installation steps of phpMyAdmin is NOT described. You need to investigate how to create a Dockerfile to ensure `order.php` and `process.php` could be successfully running and run a phpMyAdmin docker container by yourself.

Task Description for Dockerfile:

To successfully start these services by the docker-compose file, you will need a dockerfile to properly setup all necessary extensions in the myphp container. Your task is to create a dockerfile (`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.

1. Base Image: Use an official PHP image with version 8.2 and FPM support (as the provided base image in the dockerfile).
2. Incorporate the `redis` extension to facilitate PHP-Redis communication.
3. Include the `mysqli` extension [4] for MySQL database support in PHP.
4. Note: Ensure that the `redis` extension is properly configured and loaded every time PHP 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 **Background** and **Task Descriptions** to understand the background, motivations, the task, and technical requirements.
- Secondly, you should practise docker commands and Linux commands to manually orchestrate the five containers.
- Thirdly, the port 8082 should be properly opened in http firewall rule of the default network.
- Lastly, you need to write a dockerfile and a docker-compose.yml by converting the docker commands and Linux commands into the docker compose instructions. **All technical requirements need to be fully met to achieve full marks.**

You can practise either on the GCP’s VM or your local machine with Oracle VirtualBox if you are unable to access GCP.

Assignment Submission:

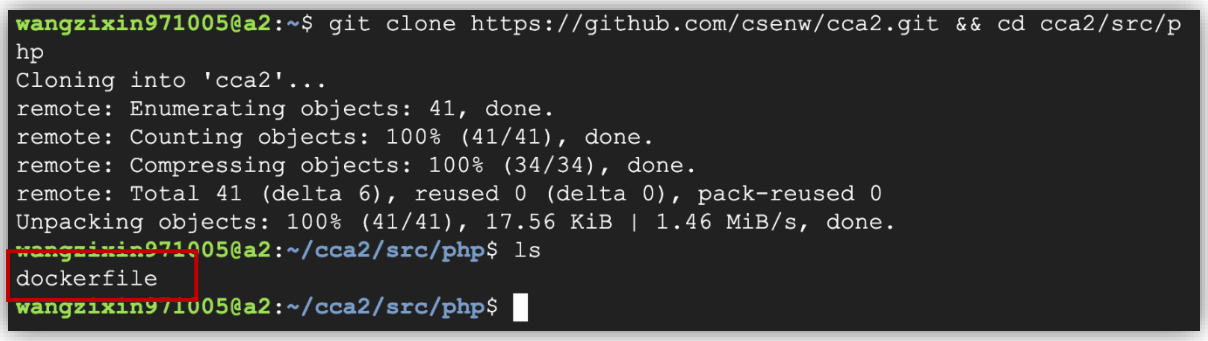
- You need to compress the **docker-compose.yml file, the dockerfile** and the supporting files (if they exist).
- The name of the compressed file should be named "FirstName_LastName_StudentNo.zip".
- You must make an online submission to Blackboard **before 1:00 PM on Friday, 15/09/2023**.
- Only one extension application could be approved due to medical conditions.

Main Steps:

Step 1:
Log in to your VM and change to your home directory.

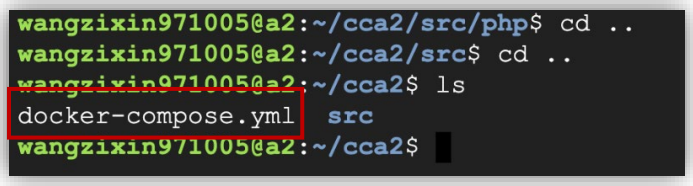
Step 2:
```
git clone https://github.com/csenw/cca2.git && cd cca2/src/php
```

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



In this folder cca2/src/php/dockerfile, please **write the dockerfile to ensure the order.php and process.php can be shown without any error. (2 marks)**

Step 3:
In the folder cca2/, please **write your docker-compose.yml. (6 marks)**



Step 4:
Run all the containers: `docker-compose up -d`

```
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 containers on your VM. Check each of the pages. (2 marks)

Static webpage test: `http://external_ip:8080/index.html`

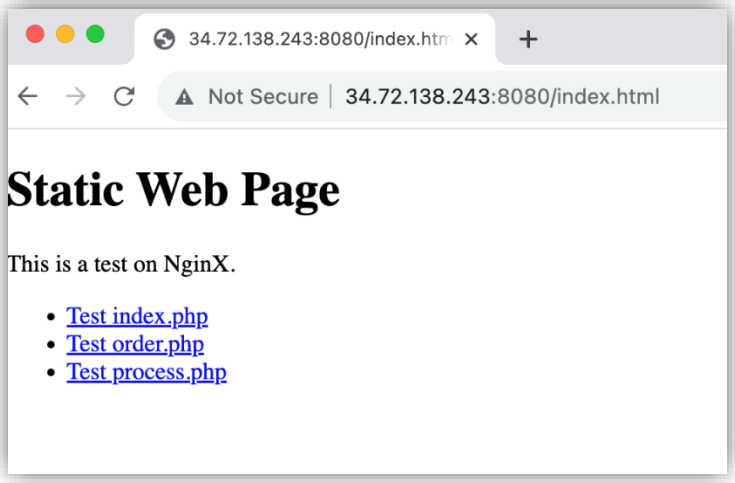


Figure 2: The Nginx test page.

`http://external_ip:8080/index.php`

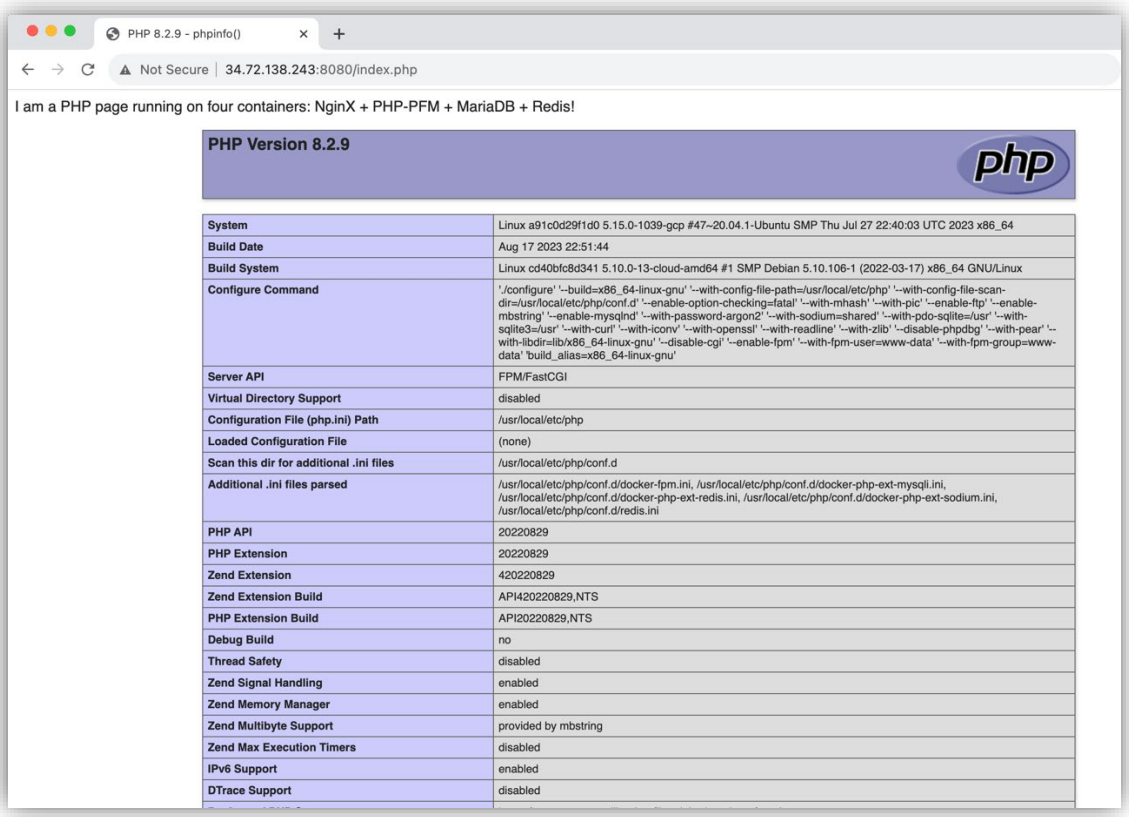


Figure 3: The PHP-FPM test page.

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

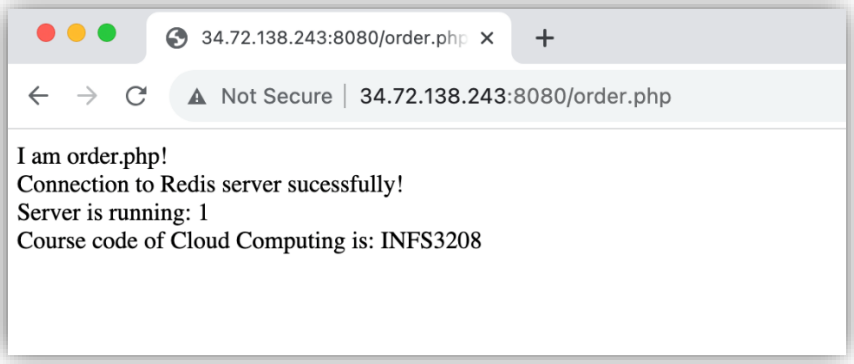


Figure 4: The Redis test page.

`http://external_ip:8080/process.php` (test the connection between PHP and MariaDB)

Figure 5: The MariaDB test page.

phpMyAdmin test: `http://external_ip:8082`

Figure 6: The phpMyAdmin login page.

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

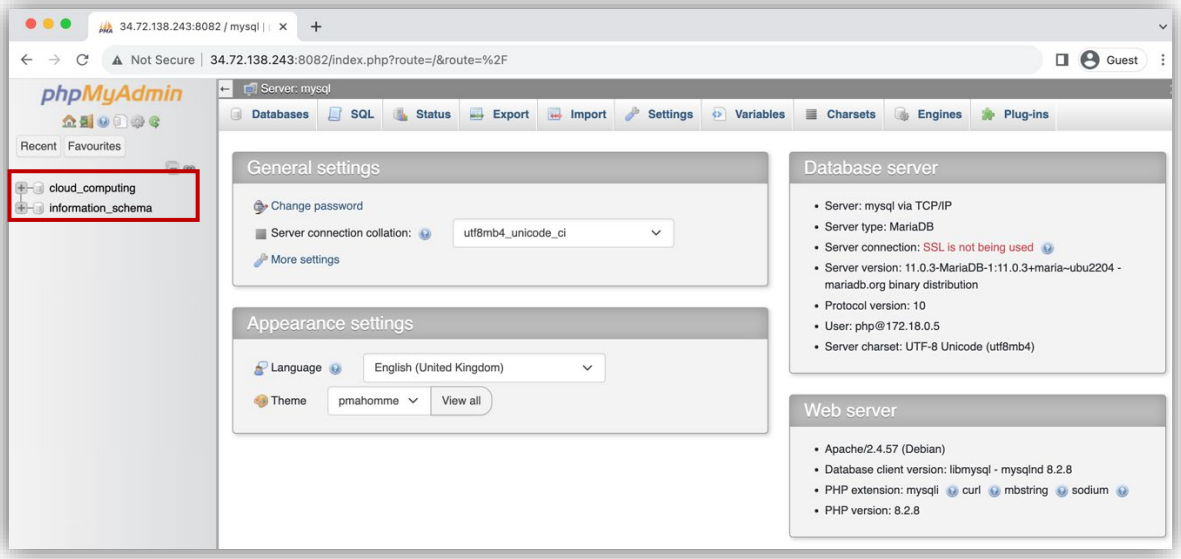


Figure 7: The phpMyAdmin test page.

Change Log:
Version 1.1: In Task Description for Dockerfile, at step 4:
Add configuration extension=redis.so → Ensure the redis extension is properly configured and loaded every time PHP starts.

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.

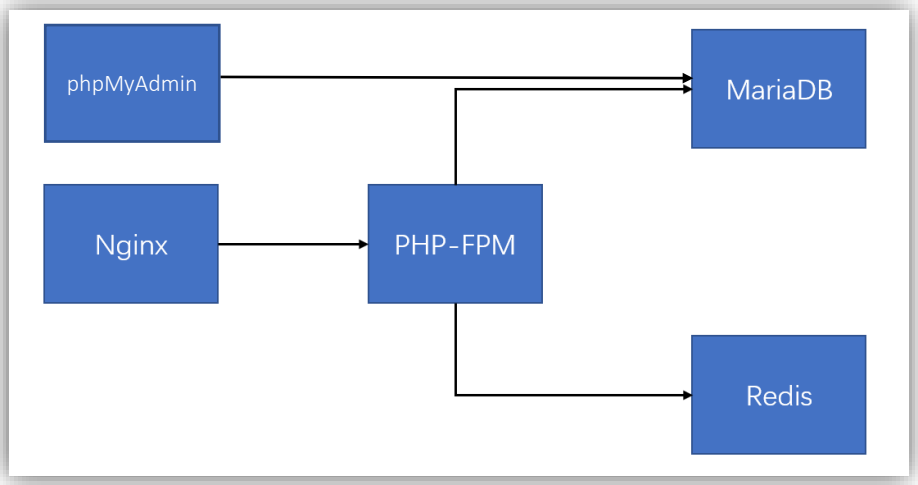


Figure 8. The dependencies among five containers: Nginx + PHP + MariaDB + phpMyAdmin + Redis.

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
[4] MySQLi in php: https://www.php.net/manual/en/book.mysqli.php