

Programming Assignment Task II (10 Marks)

Background:

With your docker file 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) is accepting 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.

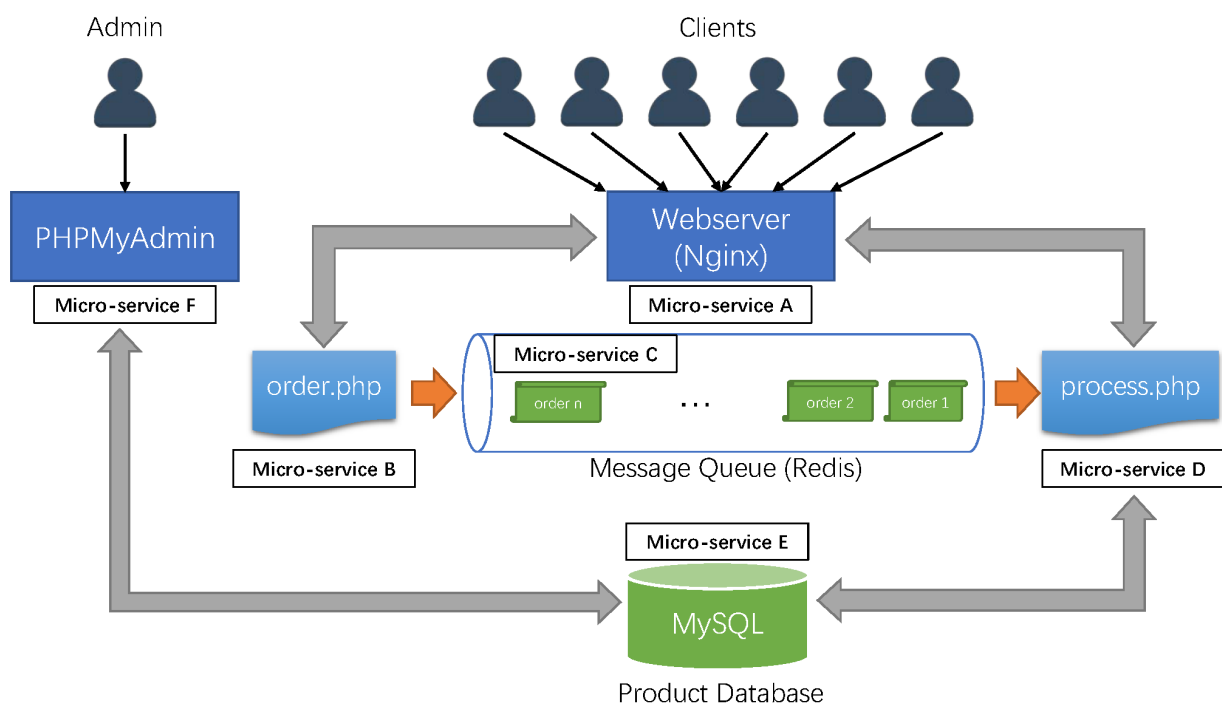


Figure 1: Micro-service architecture.

Task Description:

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 setting files and test web pages. In your docker-compose file, you should copy the given setting files for Nginx and PHP-FPM to the corresponding places inside of the docker containers by using docker-compose instructions.
2. Apart from the setting 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 testing 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 MariaDB 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. Name the container as “myphp” and expose port 9000.
 - b. 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`).
 - c. As PHP-FPM needs to communicate with MariaDB and Redis, this container (“myphp”) needs to depend on “mysql” and “myredis”.
 - d. 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 so that `order.php` and `process.php` can work properly.
 - e. 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 setting 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”.
10. The installation steps of Nginx, PHP-FPM, Redis, and MariaDB on a VM will be described in Appendix. 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 installation steps of phpMyAdmin will NOT be described. You need to investigate how to run a phpMyAdmin docker container by yourself.

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 Description** to understand the background, motivations, the task, and the technical requirements.
- Secondly, you should practise docker commands, and Linux commands to manually orchestrate the five containers.
- Lastly, you need to write 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 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, 16/09/2022.**
- 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
```

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

```
uqteaching@instance-1:~$ git clone https://github.com/csenw/cca2.git && cd cca2
Cloning into 'cca2'...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 32 (delta 4), reused 32 (delta 4), pack-reused 0
Unpacking objects: 100% (32/32), done.
uqteaching@instance-1:~/cca2$
```

In this folder (cca2), please write your docker-compose.yml.

```
uqteaching@instance-1:~/cca2$ ls
docker-compose.yml  src
```

Step 3:

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 4:

Test the containers on your VM.

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

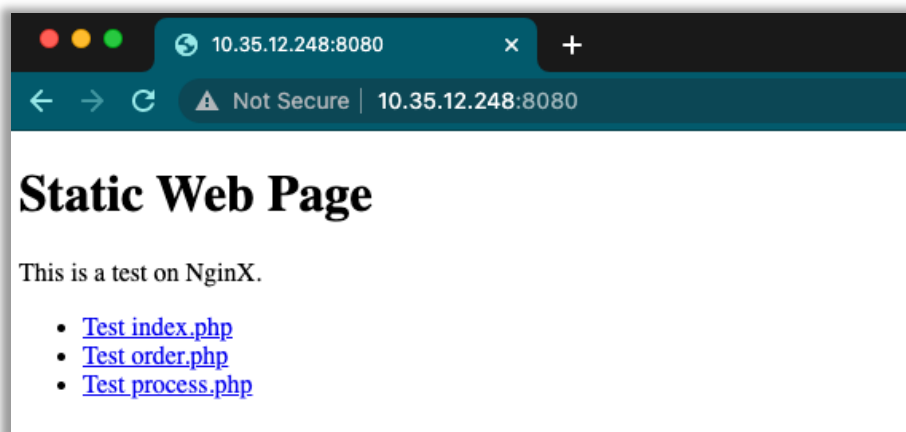


Figure 2: A Nginx test page.

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

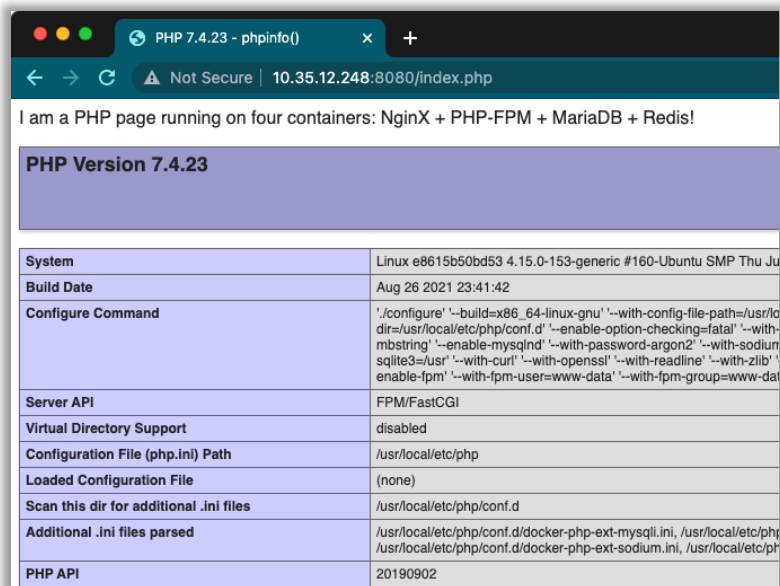


Figure 3: A PHP-FPM test page.

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

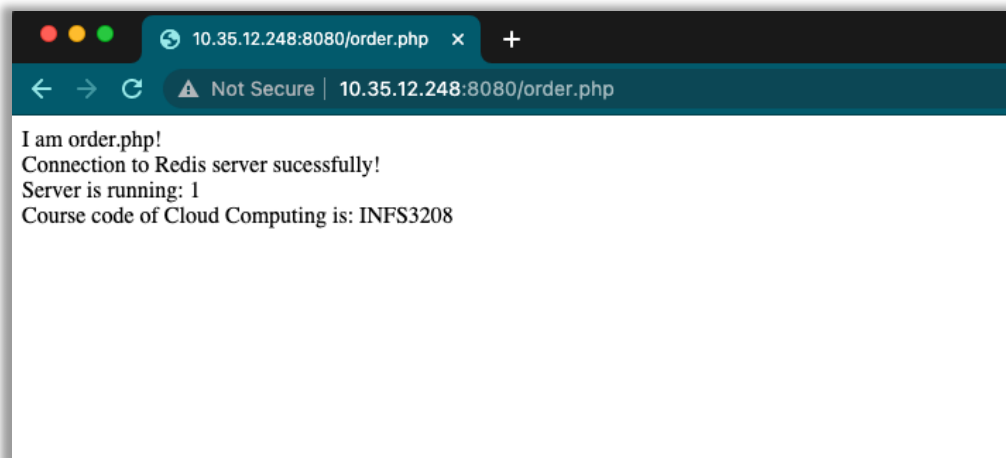


Figure 4: A Redis test page.

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



Figure 5: MariaDB test page.

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

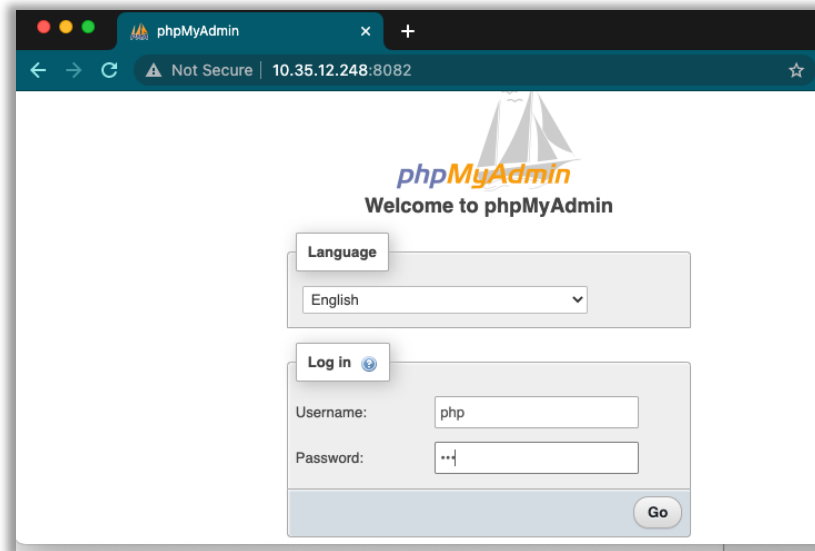


Figure 6: A phpMyAdmin login page.

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

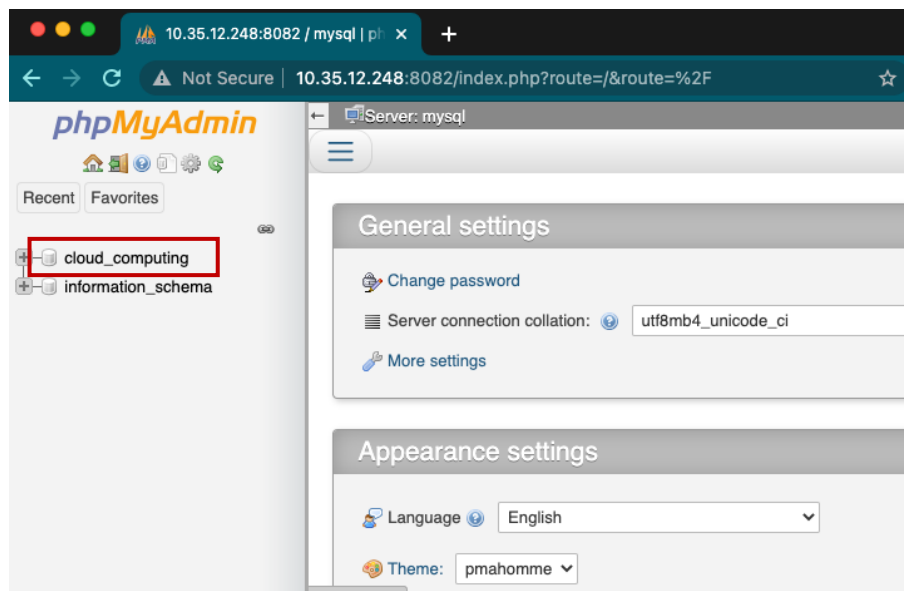


Figure 7: A phpMyAdmin test page.

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.

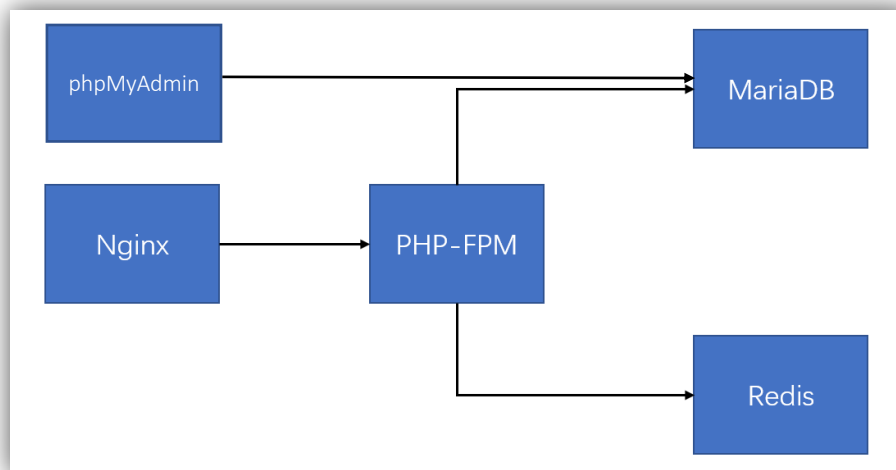
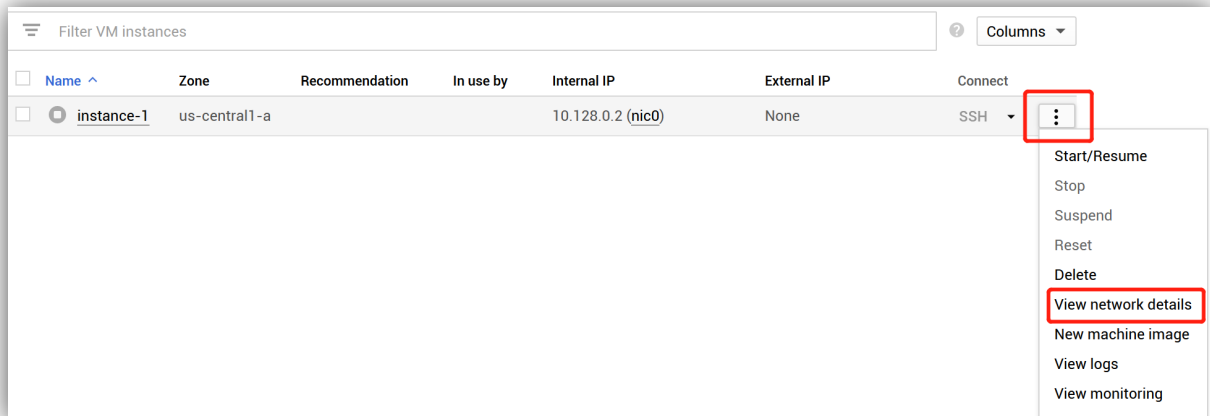


Figure 8. The dependencies among five containers: Nginx + PHP + MariaDB + phpMyAdmin + Redis.

Installation steps on a Linux VM:

- OS: Ubuntu 18.04,
- Network: using dynamical IP and Port 8080
- PHP & PHP-FPM version: 7.4
- MariaDB and Redis version: latest

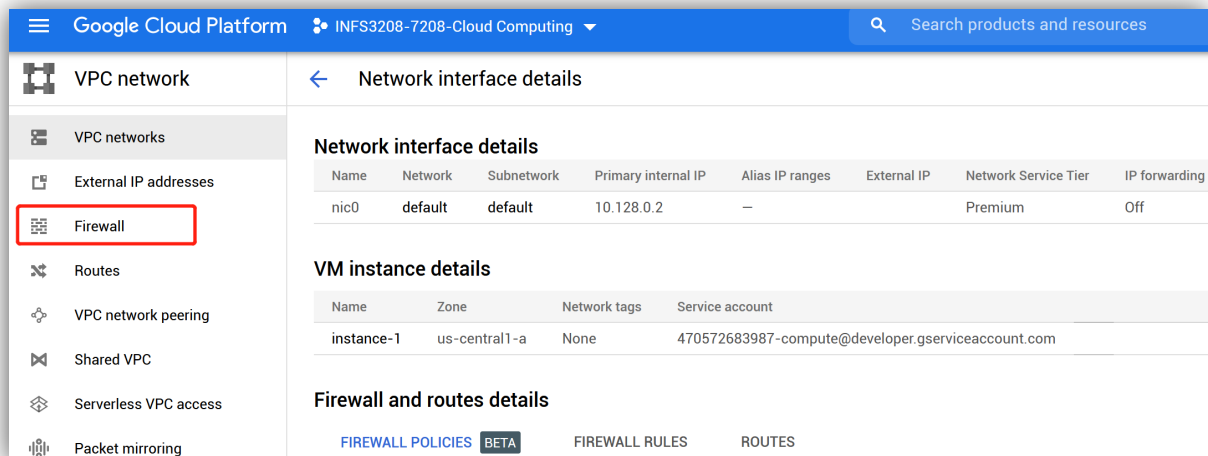
Step 1. Open a new port 8080



Filter VM instances

Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
instance-1	us-central1-a			10.128.0.2 (nic0)	None	SSH

- Start/Resume
- Stop
- Suspend
- Reset
- Delete
- View network details
- New machine image
- View logs
- View monitoring



Google Cloud Platform

INFS3208-7208-Cloud Computing

Search products and resources

VPC network

Network interface details

Network interface details

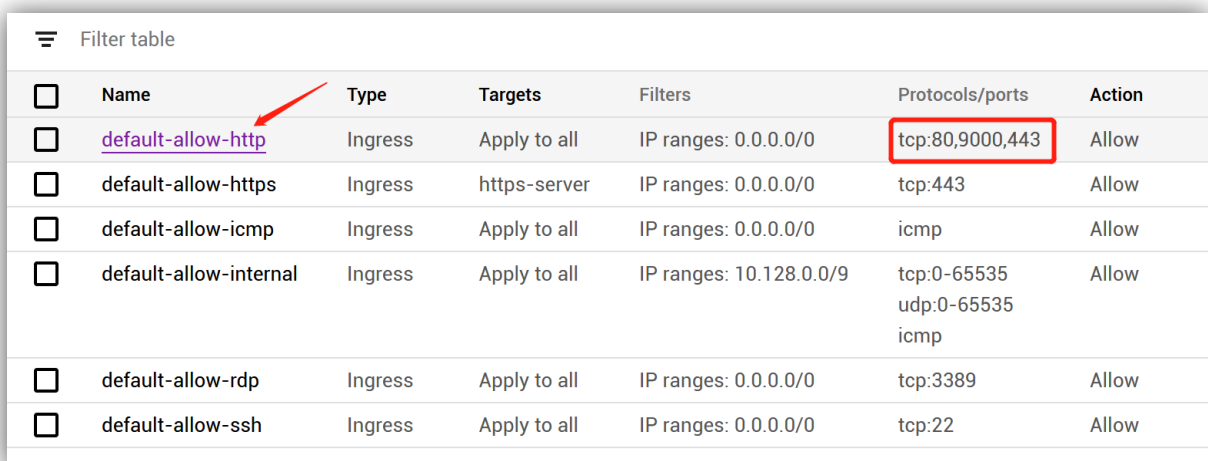
Name	Network	Subnetwork	Primary internal IP	Alias IP ranges	External IP	Network Service Tier	IP forwarding
nic0	default	default	10.128.0.2	—		Premium	Off

VM instance details

Name	Zone	Network tags	Service account
instance-1	us-central1-a	None	470572683987-compute@developer.gserviceaccount.com

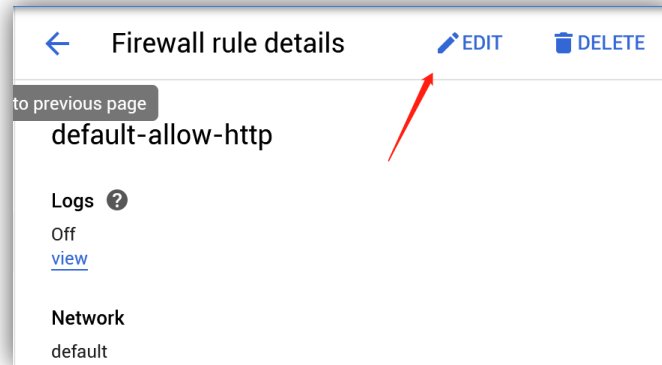
Firewall and routes details

FIREWALL POLICIES BETA FIREWALL RULES ROUTES



Filter table

Name	Type	Targets	Filters	Protocols/ports	Action
default-allow-http	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:80,9000,443	Allow
default-allow-https	Ingress	https-server	IP ranges: 0.0.0.0/0	tcp:443	Allow
default-allow-icmp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	icmp	Allow
default-allow-internal	Ingress	Apply to all	IP ranges: 10.128.0.0/9	tcp:0-65535 udp:0-65535 icmp	Allow
default-allow-rdp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:3389	Allow
default-allow-ssh	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:22	Allow



Protocols and ports ?

Firewall rule details

☒ Specified protocols and ports

☒ tcp :

☐ udp :

☐ Other protocols

[DISABLE RULE](#)

[SAVE](#) [CANCEL](#)

Protocols and ports

tcp:80
tcp:9000
tcp:443
tcp:8080

Step 2: Pull required docker images from the docker hub.

```
docker pull nginx
```

Pull Nginx image from docker hub.

```
docker pull php:7.4-fpm
```

Pull php7.4 and php-fpm from docker hub.

```
docker pull mariadb:latest
```

Pull MariaDB from docker hub.

```
docker pull redis:latest
```

Pull Redis from docker hub.

```
docker images
```

Check pulled images

Step 3: Run four images in dependency order and link them.

```
docker run \  
--name mysql \  
-e MYSQL_ROOT_PASSWORD=MyDBRoot123 \  
-e MYSQL_DATABASE=cloud_computing \  
-e MYSQL_USER=php \  
-e MYSQL_PASSWORD=php \  
-d mariadb:latest
```

Run MariaDB image.

```
docker run --name myredis -d redis:latest
```

Run Redis image.

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

[OPTIONAL]: if you have installed php7.4-fpm on your VM, then port 9000 is being taken by PHP-FPM. You can stop the service and make port 9000 available with the above commands.

```
docker run --name myphp -p 9000:9000 -v $HOME/cca2/src:/var/www/html --link  
mysql:mysql --link myredis:myredis -d php:7.4-fpm
```

Run php:7.4-fpm image.

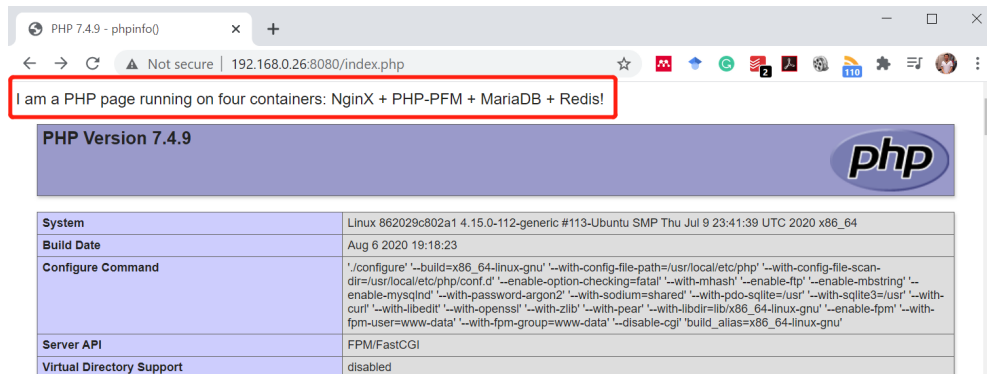
```
docker run \  
--name mynginx \  
-p 8080:80 \  
-v $HOME/cca2/src:/var/www/html \  
-v $HOME/cca2/src/nginx.ini:/etc/nginx/conf.d/default.conf \  
--link myphp:myphp \  
-d nginx:latest
```

Run Nginx image.

Test the static webpage and PHP webpage with the address (your GCP external IP):

- http://external_ip/index.html
- http://external_ip/index.php

All done, you have completed this installation and can run any HTML/PHP pages on your VM.



Environment

Variable	Value
MYREDIS_ENV_GOSU_VERSION	1.12
PHP_EXTRA_CONFIGURE_ARGS	--enable-fpm --with-fpm-user=www-data --with-fpm-group=www-data --disable-cgi
MYSQL_ENV_GPG_KEYS	177F4010FE56CA336300305F1656F24C74CD1D8
HOSTNAME	862029c802a1
MYSQL_ENV_MARIADB_VERSION	1:10.5.5+maria-focal
PHP_INI_DIR	/usr/local/etc/php
HOME	/var/www
MYSQL_ENV_MYSQL_DATABASE	php
MYREDIS_PORT	tcp://172.17.0.3:6379
PHP_LDFLAGS	-Wl,-O1 -pie
MYREDIS_PORT_6379_TCP_ADDR	172.17.0.3
PHP_CFLAGS	-fstack-protector-strong -fpic -fpie -O2 -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64
PHP_MD5	no value
MYREDIS_NAME	/myphp/myredis
PHP_VERSION	7.4.9
MYREDIS_PORT_6379_TCP_PORT	6379
GPG_KEYS	42670A7FE4D0441C8E4632349E4FDC074A4EF02D 5A52880781F755608BF815FC910DEB46F53EA312
MYREDIS_PORT_6379_TCP_PROTO	tcp
PHP_CPPFLAGS	-fstack-protector-strong -fpic -fpie -O2 -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64
PHP_ASC_URL	https://www.php.net/distributions/php-7.4.9.tar.xz.asc
MYREDIS_ENV_REDIS_DOWNLOAD_URL	http://download.redis.io/releases/redis-6.0.6.tar.gz
PHP_URL	https://www.php.net/distributions/php-7.4.9.tar.xz
MYREDIS_ENV_REDIS_VERSION	6.0.6
MYSQL_PORT_3306_TCP_ADDR	172.17.0.2
MYSQL_ENV_MYSQL_ROOT_PASSWORD	MyDBRoot123
PATH	/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
MYSQL_ENV_GOSU_VERSION	1.12
MYSQL_PORT_3306_TCP_PORT	3306
MYSQL_PORT_3306_TCP_PROTO	tcp
MYREDIS_PORT_6379_TCP	tcp://172.17.0.3:6379
MYSQL_ENV_MARIADB_MAJOR	10.5
MYSQL_PORT	tcp://172.17.0.2:3306
MYSQL_ENV_MYSQL_PASSWORD	php
MYSQL_PORT_3306_TCP	tcp://172.17.0.2:3306
MYREDIS_ENV_REDIS_DOWNLOAD_SHA	12ad49b163af5ef39466e8d2f7d212a58172116e5b441eebecb4e6ca22363d94
MYSQL_NAME	/myphp/mysql
PHPIZE_DEPS	autoconf dpkg-dev file g++ gcc libc-dev make pkg-config re2c
PWD	/var/www/html
MYSQL_ENV_MYSQL_USER	php

You should find information about MariaDB (MySQL) and Redis in Environment!
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>