Spring Boot + Docker Compose Example with MySQL and NGINX

### Stack

* Docker
* Spring Boot
* MySQL
* NGINX
* Maven

## Init project structure and dependencies

### Project structure

├── app

│   ├── src

│   │   └── main

│   │   ├── java

│   │   │   └── com

│   │   │   └── hellokoding

│   │   │   └── springboot

│   │   │   ├── IndexController.java

│   │   │   └── WebApplication.java

│   │   └── resources

│   │   ├── static

│   │   │   ├── css

│   │   │   │   └── main.css

│   │   │   └── js

│   │   │   └── main.js

│   │   ├── templates

│   │   │   └── index.ftl

│   │   └── application.properties

│   ├── Dockerfile

│   └── pom.xml

├── nginx

│   └── conf.d

│   └── app.conf

└── docker-compose.yaml

### Application dependencies

[pom.xml](https://github.com/hellokoding/hellokoding-courses/blob/master/docker-examples/dockercompose-springboot-mysql-nginx/app/pom.xml)

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<artifactId>com.hellokoding</artifactId>

<name>dockercompose-springboot-mysql-nginx</name>

<description>dockercompose-springboot-mysql-nginx</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.5.8.RELEASE</version>

</parent>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-freemarker</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

## Define Controller, View Template and Config

### Controller

[IndexController.java](https://github.com/hellokoding/hellokoding-courses/blob/master/docker-examples/dockercompose-springboot-mysql-nginx/app/src/main/java/com/hellokoding/springboot/IndexController.java)

package com.hellokoding.springboot;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import java.util.logging.Logger;

@Controller

public class IndexController {

private final Logger logger = Logger.getLogger(this.getClass().getName());

@GetMapping("/")

public String index(Model model) {

return "index";

}

}

### FreeMarker View template

[index.ftl](https://github.com/hellokoding/hellokoding-courses/blob/master/docker-examples/dockercompose-springboot-mysql-nginx/app/src/main/resources/templates/index.ftl)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Docker Compose with Spring Boot, MySQL, NGINX</title>

</head>

<body>

<h1>Docker Compose with Spring Boot, MySQL, NGINX</h1>

<p></p>

</body>

</html>

### Application Configuration

[WebApplication.java](https://github.com/hellokoding/hellokoding-courses/blob/master/docker-examples/dockercompose-springboot-mysql-nginx/app/src/main/java/com/hellokoding/springboot/WebApplication.java)

package com.hellokoding.springboot;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class WebApplication {

public static void main(String[] args) throws Exception {

SpringApplication.run(WebApplication.class, args);

}

}

## Dockerize

### Dockerfile of Spring Boot web application

[Dockerfile](https://github.com/hellokoding/hellokoding-courses/blob/master/docker-examples/dockercompose-springboot-mysql-nginx/app/Dockerfile)

FROM maven:3.5-jdk-8

### NGINX config file

[app.conf](https://github.com/hellokoding/hellokoding-courses/blob/master/docker-examples/dockercompose-springboot-mysql-nginx/nginx/conf.d/app.conf)

server {

listen 80;

charset utf-8;

access\_log off;

location / {

proxy\_pass http://app:8080;

proxy\_set\_header Host $host:$server\_port;

proxy\_set\_header X-Forwarded-Host $server\_name;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

}

location /static {

access\_log off;

expires 30d;

alias /app/static;

}

}

### Docker Compose

[docker-compose.yaml](https://github.com/hellokoding/hellokoding-courses/blob/master/docker-examples/dockercompose-springboot-mysql-nginx/docker-compose.yaml)

version: '3'

services:

nginx:

container\_name: some-nginx

image: nginx:1.13

restart: always

ports:

- 80:80

- 443:443

volumes:

- ./nginx/conf.d:/etc/nginx/conf.d

depends\_on:

- app

mysql:

container\_name: some-mysql

image: mysql/mysql-server:5.7

environment:

MYSQL\_DATABASE: test

MYSQL\_ROOT\_PASSWORD: hellokoding

MYSQL\_ROOT\_HOST: '%'

ports:

- "3306:3306"

restart: always

app1:

restart: always

build: ./app1

working\_dir: /app1

volumes:

- ./app1:/app1

expose:

- "8080"

command: mvn clean spring-boot:run

depends\_on:

- mysql

### Test

* Run command docker-compose up
* Access to http://localhost

# Creating a simple PHP-FPM, Nginx and MySQL application with docker compose

You can use example below if you want to create a simple PHP-FPM, Nginx and MySQL application with docker compose.

### Structure

1. api$ tree
2. .
3. ├── docker
4. │   ├── nginx
5. │   │   └── default.conf
6. │   └── php
7. │   ├── Dockerfile
8. │   └── www.conf
9. ├── .env
10. ├── docker-compose.yml
11. ├── index.html
12. ├── index.php
13. └── connect.php

### Files

#### .env

1. MYSQL\_ROOT\_PASSWORD=root
2. MYSQL\_DATABASE=dummy
3. MYSQL\_USER=dummy
4. MYSQL\_PASSWORD=dummy

#### docker-compose.yml

1. version: '3'
3. services:
5. php:
6. build:
7. context: ./docker/php
8. ports:
9. - 9000:9000
10. volumes:
11. - ./:/srv/www/api
12. - ./docker/php/www.conf:/usr/local/etc/php-fpm.d/www.conf
13. environment:
14. MYSQL\_USER: ${MYSQL\_USER}
15. MYSQL\_PASSWORD: ${MYSQL\_PASSWORD}
17. nginx:
18. image: nginx:1.13.8
19. ports:
20. - 80:80
21. volumes:
22. - ./:/srv/www/api
23. - ./docker/nginx/default.conf:/etc/nginx/conf.d/default.conf
24. depends\_on:
25. - php
27. mysql:
28. image: mysql:5.7
29. ports:
30. - 3306:3306
31. depends\_on:
32. - php
33. environment:
34. MYSQL\_ROOT\_PASSWORD: ${MYSQL\_ROOT\_PASSWORD}
35. MYSQL\_DATABASE: ${MYSQL\_DATABASE}
36. MYSQL\_USER: ${MYSQL\_USER}
37. MYSQL\_PASSWORD: ${MYSQL\_PASSWORD}

#### index.html

1. HTML

#### index.php

1. echo 'Welcome!';

#### connect.php

1. $host = 'mysql';
2. $user = getenv('MYSQL\_USER');
3. $pass = getenv('MYSQL\_PASSWORD');
5. $conn = mysqli\_connect($host, $user, $pass);
6. if (!$conn) {
7. exit('Connection failed: '.mysqli\_connect\_error().PHP\_EOL);
8. }
10. echo 'Successful database connection!'.PHP\_EOL;

#### docker/nginx/default.conf

1. server {
2. listen 80 default\_server;
4. server\_name api.com www.api.com;
6. root /srv/www/api;
8. location ~ \.php$ {
9. try\_files $uri =404;
10. fastcgi\_split\_path\_info ^(.+\.php)(/.+)$;
11. fastcgi\_pass php:9000;
12. fastcgi\_index index.php;
13. include fastcgi\_params;
14. fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;
15. fastcgi\_param PATH\_INFO $fastcgi\_path\_info;
16. }
18. error\_log /var/log/nginx/api\_error.log;
19. access\_log /var/log/nginx/api\_access.log;
20. }

#### docker/php/www.conf

1. [www]
3. user = www-data
4. group = www-data
6. listen = nginx:9000
8. pm = dynamic
9. pm.max\_children = 5
10. pm.start\_servers = 2
11. pm.min\_spare\_servers = 1
12. pm.max\_spare\_servers = 3

#### docker/php/Dockerfile

1. FROM php:7.2-fpm
3. RUN docker-php-ext-install mysqli

### Build

1. $ docker-compose up
2. Creating network "api\_default" with the default driver
3. Creating api\_php\_1 ... done
4. Creating api\_nginx\_1 ... done
5. Creating api\_mysql\_1 ... done
6. Attaching to api\_php\_1, api\_nginx\_1, api\_mysql\_1
7. ...
8. $ docker ps
9. CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
10. e517fbd9cb63 mysql:5.7 "docker-entrypoint.s…" 8 minutes ago Up 8 minutes 0.0.0.0:3306->3306/tcp api\_mysql\_1
11. 4caace5a0d52 nginx:1.13.8 "nginx -g 'daemon of…" 8 minutes ago Up 8 minutes 0.0.0.0:80->80/tcp api\_nginx\_1
12. c113fb3ba41e api\_php "docker-php-entrypoi…" 8 minutes ago Up 8 minutes 0.0.0.0:9000->9000/tcp api\_php\_1
13. $ docker-compose ps
14. Name Command State Ports
15. ----------------------------------------------------------------------------
16. api\_mysql\_1 docker-entrypoint.sh mysqld Up 0.0.0.0:3306->3306/tcp
17. api\_nginx\_1 nginx -g daemon off; Up 0.0.0.0:80->80/tcp
18. api\_php\_1 docker-php-entrypoint php-fpm Up 0.0.0.0:9000->9000/tcp

### Test

Obtain the IP address of the container network first then call it.

1. $ echo $(docker network inspect api\_default | grep Gateway | grep -o -E '[0-9\.]+')
2. 172.18.0.1
3. $ curl 172.18.0.1
4. HTML
5. $ curl 172.18.0.1/index.php
6. Welcome!
7. $ curl 172.18.0.1/connect.php
8. Successful database connection!