**Tugas Day 12**

Instalasi & Konfigurasi Docker

pada sistem operasi CentOS

**Intro**

Docker merupakan aplikasi open source yang memiliki berfungsi sebagai container/tempat membungkus sebuah software ke dalam *environment* yang terisolasi dari komputer host. Aplikasi ini juga dapat difungsikan untuk melakukan otomasi terhadap *software deployment*.

**Instalasi Docker**

Terdapat beberapa cara yang dapat digunakan untuk menginstall aplikasi Docker, baik secara manual maupun otomatis. Semua langkah tersebut telah terdokumentasi secara lengkap pada laman <https://docs.docker.com/engine/install/centos/>. Adapun berikut ini merupakan instalasi dengan bantuan **shebang script** sebagai berikut

|  |
| --- |
| $ curl -fsSL https://get.docker.com -o get-docker.sh  $ sudo sh get-docker.sh  $ sudo curl -L "https://github.com/docker/compose/releases/download/1.26.1/docker-compose-$(uname -s)-$(uname -m)" -o /bin/docker-compose |

**Konfigurasi User dan Aktivasi layanan**

Demi kemudahan penggunaan, terlebih dahulu kita masukkan user ke dalam group **docker**. Adapun perintahnya sebagai berikut:

|  |
| --- |
| $ sudo usermod -aG docker $USER |

Selanjutnya, kita aktifkan layanan docker dengan perintah:

|  |
| --- |
| $ sudo systemctl enable docker  $ sudo systemctl start docker  $ sudo systemctl status docker |
|  |

**Skenario Deployment**

Berdasarkan petunjuk soal yang kita terima, kita diharuskan untuk men*-deploy* sebuah aplikasi laravel (<https://gitlab.com/muhammadyaqin/laravel5_sample.git>) yang dijalankan dengan **Nginx, PHP,** dan **Mysql** pada sebuah docker container.

Dari sini, kita akan membaginya ke dalam dua buah skenario, yakni **single-container** dan **multi-container**. Pada **single-container** hanya akan dihasilkan satu buah ***container*** yang memuat ketiga layanan. Sedangkan **multi-container**akan membagi layanan ke unit-unit container dengan fungsionalitas masing-masing

**Persiapan project**

Sebelum melakukan *deployment*, terlebih dahulu kita unduh *project laravel* dengan perintah sebagai berikut

|  |
| --- |
| $ git clone https://gitlab.com/muhammadyaqin/laravel5\_sample src |

Karena repository tersebut tidak menyertakan folder *dependencies*, kita diharuskan untuk menginstalnya secara manual dengan bantuan ***composer***. Terlebih dahulu kita lakukan instalasi ***composer***

|  |
| --- |
| $ curl -sS https://getcomposer.org/installer | sudo php -- --install-dir=/usr/bin --filename=composer  $ sudo yum install php72 php72-php-mbstring php72-php-zip php72-php-xml -y |

Untuk mengecek apakah **composer** telah terinstal, dapat digunakan perintah:

|  |
| --- |
| $ composer -V |
|  |

Selanjutnya kita lakukan instalasi *dependencies* sebagai berikut

|  |
| --- |
| $ cd src && rm -rf vendor/ && composer install |
|  |

Kemudian, kita buat *environment file* bernama **.env** yang dapat kita ambil dari **.env.example.** Berikut perintah yang digunakan serta isi dari file tersebut

|  |
| --- |
| $ mv .env.example .env  $ cat .env |
| ..  ..  DB\_CONNECTION=mysql  DB\_HOST=127.0.0.1  DB\_PORT=3306  DB\_DATABASE=**homestead**  DB\_USERNAME=**homestead**  DB\_PASSWORD=**secret**  ..  .. |

Di sini, kita dapat melihat sebuah **db config** yang laravel untuk menginisiasi konektivitas dengan database server. Untuk keperluan lebih lanjut, terlebih dahulu kita ubah konfigurasinya sebagai berikut

|  |
| --- |
| $ vim .env |
| ..  ..  DB\_CONNECTION=mysql  DB\_HOST=127.0.0.1  DB\_PORT=3306  DB\_DATABASE=**laravel**  DB\_USERNAME=**devops**  DB\_PASSWORD=**12345**  ..  .. |

Selanjutnya, kita generate sebuah APP\_KEY yang dibutuhkan oleh laravel. Adapun perintahnya sebagai berikut

|  |
| --- |
| $ php artisan key:generate |

Selanjutnya ubah *permission* folder agar dapat diakses oleh php-fpm

|  |
| --- |
| $ chown -R 33:33 .  $ cd .. |

**Konfigurasi Docker**

Setelah selesai melakukan konfigurasi *project*, saatnya merancang arsitektur docker yang kita butuhkan. Pertama-tama kita buat konfigurasi **php-fpm** sebagai berikut

|  |
| --- |
| $ vim conf/php/www.conf |
| [www]  user = www-data  group = www-data  listen = 127.0.0.1:9000  listen.mode = 0660    pm = dynamic  pm.max\_children = 5  pm.start\_servers = 2  pm.min\_spare\_servers = 1  pm.max\_spare\_servers = 3 |

Dari sini kita akan membedakan konfigurasi selanjutnya berdasarkan skenario yang telah disebutkan sebelumnya

**A. Single-container**

Pada skenario ini, kita definisikan virtual host **nginx** sebagaimana berikut ini

|  |
| --- |
| $ vim conf/nginx/default.conf |
| server {  listen 80 default\_server;  index index.php index.html;  root /var/www/html/public;  server\_name \_;  location / {  try\_files $uri /index.php?$args;  }  location ~ \.php$ {  fastcgi\_split\_path\_info ^(.+\.php)(/.+)$;  fastcgi\_pass 127.0.0.1:9000;  fastcgi\_index index.php;  include fastcgi\_params;  fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;  fastcgi\_param PATH\_INFO $fastcgi\_path\_info;  }  } |

Berdasarkan konfigurasi tersebut, kita mendefinisikan sebuah default web server yang akan memproses **php file** yang diproses oleh **php-fpm** yang berjalan pada localhost:9000.

Kemudian, kita buat sebuah Dockerfile seperti berikut ini:

|  |
| --- |
| $ vim Dockerfile |
| # Pull docker image of php 7.2  FROM php:7.2  # Unlock php banned-list & allow additional php module  RUN rm /etc/apt/preferences.d/no-debian-php  # Allow services from starting automatically  RUN echo "exit 0" > /usr/sbin/policy-rc.d  # Install other depencies (nginx, mysql)  RUN apt update && apt install nginx -y  RUN apt install net-tools mariadb-server php-mysql -y  RUN apt install php-fpm -y  # Set owner of directory to www-data  RUN chown -R www-data:www-data /var/www/html  # Set working directory  WORKDIR /var/www/html  # Set environment variable to populate database  ENV MYSQL\_DATABASE=laravel  ENV MYSQL\_USER=devops  ENV MYSQL\_PASSWORD=12345  # Set default action when starting a container  CMD service mysql start && service php7.3-fpm start && nginx -g "daemon off;"  # Expose internal port 80 to docker host  EXPOSE 80 |

Lalu kita buat file *docker-compose.yml* untuk membuat container dari Image hasil pemrosesan Dockerfile

|  |
| --- |
| $ vim docker-compose.yml |
| version: "3"  services:  # service\_name  web:  # build Dockerfile on current directory  build: .  # Link local directory to docker volume  volumes:  # script for db automation  - ./init-db:/var/www/html/init-db  # laravel code  - ./src:/var/www/html/  # virtualhost  - ./conf/nginx/default.conf:/etc/nginx/sites-enabled/default  # php-fpm config  - ./conf/php/www.conf:/etc/php/7.3/fpm/pool.d/www.conf  ports:  # external-port : internal-port  - "4000:80"  # Set default network to bridge  networks:  default:  driver: bridge |

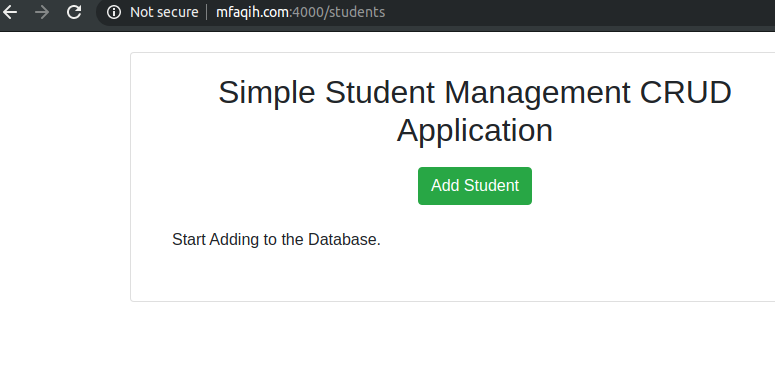
Terakhir, kita buat script untuk melakukan inisialisasi *database* serta *database* migration

|  |
| --- |
| $ vim init-db |
| # Initialize database for laravel  mysql -e "CREATE DATABASE ${MYSQL\_DATABASE} /\*\!40100 DEFAULT CHARACTER SET utf8 \*/;"  # Create additional mysql user with privileges  mysql -e "CREATE USER ${MYSQL\_USER}@localhost IDENTIFIED BY '${MYSQL\_PASSWORD}';"  mysql -e "GRANT ALL PRIVILEGES ON ${MYSQL\_DATABASE}.\* TO '${MYSQL\_USER}'@'localhost';"  mysql -e "FLUSH PRIVILEGES;"  # Replace default .env file with new db-name, username & password  sed -i "s/DB\_DATABASE=homestead/DB\_DATABASE=$MYSQL\_DATABASE/g" .env  sed -i "s/DB\_USERNAME=homestead/DB\_USERNAME=$MYSQL\_USER/g" .env  sed -i "s/DB\_PASSWORD=secret/DB\_PASSWORD=$MYSQL\_PASSWORD/g" .env  # Do database migration  php7.3 artisan migrate |

Setelah itu, saatnya melakukan deployment

|  |
| --- |
| $ docker-compose build  $ docker-compose up -d  $ docker-compose exec web sh init-db  $ docker ps |
|  |

Hasilnya layanan web app akan berjalan pada http://ip-addr:4000



\* NB : Apabila hendak melanjutkan skenario selanjutnya, diharapkan untuk melakukan restart layanan docker, dikarenakan kedua config menggunakan deklarasi docker volume yang sama

**B. Multi-container**

Pada skenario ini, kita definisikan virtual host **nginx** sebagaimana berikut ini

|  |
| --- |
| $ vim conf/nginx/default.conf |
| server {  listen 80 default\_server;  index index.php index.html;  root /var/www/html/public;  server\_name \_;  location / {  try\_files $uri /index.php?$args;  }  location ~ \.php$ {  fastcgi\_split\_path\_info ^(.+\.php)(/.+)$;  fastcgi\_pass php:9000;  fastcgi\_index index.php;  include fastcgi\_params;  fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;  fastcgi\_param PATH\_INFO $fastcgi\_path\_info;  }  } |

Berdasarkan konfigurasi tersebut, kita mendefinisikan sebuah default web server yang akan memproses **php file** yang diproses oleh **php-fpm** yang berjalan pada **php:9000** yang menunjuk alamat IP service php-fpm.

Berbeda halnya dengan konfigurasi **single-container**, disini kita membagi layanan ke dalam unit-unit kecil sesuai dengan fungsionalitasnya. Selain itu, karena disini tidak terlalu membutuhkan instalasi *package*, kita dapat menghemat storage dengan menggunakan *image* ***alpine***

Berikut merupakan konfigurasi dua buah dockerfile, sebut saja **web.dockerfile (nginx)** dan **php.dockerfile (php-fpm)**

|  |
| --- |
| $ vim web.dockerfile |
| FROM nginx:alpine  WORKDIR /var/www/html  EXPOSE 80 |

|  |
| --- |
| $ vim php.dockerfile |
| FROM php:7.2-fpm-alpine  WORKDIR /var/www/html  RUN docker-php-ext-install mysqli  RUN docker-php-ext-install -j$(nproc) pdo\_mysql  EXPOSE 9000  CMD php-fpm |

Selanjutnya, kita buat file *docker-compose.yml* untuk membuat model multi container

|  |
| --- |
| $ vim docker-compose.yml |
| version: '3'  services:  web:  build:  context: .  dockerfile: web.dockerfile  volumes:  - ./src:/var/www/html  - ./conf/nginx/default.conf:/etc/nginx/conf.d/default.conf  ports:  - "8000:80"  links:  - php  php:  build:  context: .  dockerfile: php.dockerfile  volumes:  - ./src:/var/www/html  - ./conf/php/www.conf:/usr/local/etc/php-fpm.d/www.conf  environment:  - "DB\_PORT=3306"  - "DB\_HOST=database"  links:  - database  database:  image: mysql:latest  hostname: database  command: --default-authentication-plugin=mysql\_native\_password  environment:  MYSQL\_ROOT\_PASSWORD: secret  MYSQL\_DATABASE: laravel  MYSQL\_USER: devops  MYSQL\_PASSWORD: 12345  ports:  - "33061:3306" |

Setelah itu, saatnya melakukan deployment.

Perlu diketahui bahwa image **alpine** menggunakan **82:www-data** sebagai ganti dari **www-data:www-data.** Untuk itu, kita perlu melakukan perubahan *permission* kembali pada directory **/var/www/html**.

|  |
| --- |
| $ docker-compose build  $ docker-compose up -d  $ docker-compose exec web chown -R 82:www-data .  $ docker-compose exec php php artisan migrate  $ docker ps |
|  |

Hasilnya, kita dapat melihat tiga buah container yang dijalankan (Nginx, PHP, MySQL) dari hasil deployment. Layanan tersebut dapat diakses pada http://ip-addr:8000

