

Menggunakan nodejs-mysql dengan docker

Tujuan Instruksional Khusus

- Menjalankan aplikasi nodejs dengan database mysql menggunakan arsitektur microservice
- Menjalankan server mysql di docker container.
- Menjalankan aplikasi nodejs sederhana pada docker container yang terpisah.
- Menghubungkan kedua container dan uji integrasi aplikasi mysql-nodejs.

Persyaratan

- docker dan docker-compose sudah terpasang pada Operating System yang digunakan.

Prosedur 1: Menjalankan MySQL Container

Clone repository

```
$ git clone https://github.com/xhartono/lab-nodejs-mysql.git
$ git checkout master
$ tree
.
├── docker-compose.yml
├── mysql
│   ├── Dockerfile
│   └── test-dump.sql
├── nodejs
│   ├── Dockerfile
│   ├── index.js
│   ├── package.json
│   ├── package-lock.json
│   ├── test_connection.js
│   └── wait-for-mysql.sh
├── README.md
└── values.yaml
```

Membangun docker Image

```
$ cd lab-nodejs-mysql/mysql
$ ls
Dockerfile  test-dump.sql
```

 Catatan:

- berkas Dockerfile berisi informasi yang digunakan docker untuk membangun image
- test-dump.sql, berisi perintah sql untuk membuat table dan mengisi dengan dummy data.

Lihat isi Dockerfile

```
$ more Dockerfile
## Pull the mysql:5.7 image
FROM mysql:5.7

## The maintainer name and email
MAINTAINER Inixindo (rbx.inixindo@gmail.com)

# database = test and password for root = password
ENV MYSQL_DATABASE=sistradb \
    MYSQL_ROOT_PASSWORD=inix2021

# when container will be started, we'll have `test` database created with this
schema
COPY ./test-dump.sql /docker-entrypoint-initdb.d/
```

Catatan:

- FROM: membangun berdasarkan image pada nilai FROM
- ENV: akan mengisi variable MYSQL_DATABASE dan MYSQL_ROOT_PASSWORD dengan nilai 'test' dan 'password'. Container akan membuat database dengan nama 'test' dan username root dengan password 'inix2021'.
- COPY: menyalin berkas test-dump.sql pada lokal direktori ke direktori docker-entrypoint-initdb.d pada direktori di image.
- Berkas test-dump.sql, karena diletakkan pada direktori docker-entrypoint-initdb.d, container dari image mysql:5.7 ini akan otomatis mengeksekusi perintah-perintah yang ada didalam berkas test-dump.sql.
- Pada saat di run, container akan membuat database dengan nama sistradb, dengan username: root dan password: inix2021

Lihat isi file test-dump.sql

```
DROP TABLE IF EXISTS `peserta`
CREATE TABLE `peserta` (
  `nopeserta` int(11) AUTO_INCREMENT,
  `nama` varchar(255) DEFAULT NULL,
  `alamat` varchar(255) DEFAULT NULL,
  `kota` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`nopeserta`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
LOCK TABLES `peserta` WRITE;
INSERT INTO `peserta` VALUES (1,'Azyva Giselle Kurniawan', 'Ciawi', 'Bogor');
INSERT INTO `peserta` VALUES (2,'Larasati Kirana', 'Permata Hijau',
'Jakarta');
UNLOCK TABLES;
```

Catatan:

- Berkas test-dump.sql: berisi perintah untuk menciptakan table peserta (CREATE) dan diisi (INSERT) dengan data dummy
- Table peserta akan dibuat secara otomatis, dan diisi dengan 2 record.

Membangun docker image

```
$ docker images
$ docker build -t tutorial/mysql .
$ docker images
```

Catatan:

- Perhatikan tanda titik (dot .) diakhir perintah docker build

Buat docker Volume untuk mysql data

```
$ docker volume create mysql_volume
$ docker volume list
```

Jalankan container

```
$ docker ps -a
$ docker run \
  -p 3306:3306 \
  -v mysql_volume:/var/lib/mysql \
  --name mysqlku \
  -d tutorial/mysql
$ docker ps -a
```

Catatan

- Bisakah menjelaskan opsi-opsi pada perintah diatas?
- Jika ada yang belum faham mengenai opsi perintah diatas, tanyakan fasilitator

Inspeksi Log

```
$ docker logs -f mysqlku
```

Catatan:

- Jika sukses, perhatikan pada log akan terdapat informasi seperti berikut:

```
2021-06-16T05:59:40.122523Z 0 [Note] mysqld: ready for connections.
```

- Lihat apakah terdapat error?
- Minta bantuan fasilitator jika tidak bisa memperbaiki error.
- Tekan `Ctrl+C` untuk keluar dari logs

Lihat apakah data dummy telah terbentuk pada database

```
$ docker exec -t mysqlku \
mysql -uroot -pinix2021 sistradb -e 'select * from peserta;
```

📌 Catatan:

- -p: diisi dengan password
- sistradb: nama database yang sebelumnya telah dibuat
- -e: perintah SQL yang digunakan untuk melihat data pada table peserta
- table peserta beserta datanya di ciptakan melalui file test-dump.sql pada pembahasan sebelumnya

Menjalankan NodeJS pada docker container

Aktifkan directory nodejs

```
$ cd ../nodejs
$ ls
Dockerfile  index.js  package.json  package-lock.json  test_connection.js
wait-for-mysql.sh
```

Catatan:

- Dockerfile: untuk membuat Docker Images
- package.json: Konfigurasi dan dependencies yang diperlukan aplikasi nodejs
- index.js: aplikasi nodejs untuk mengakses data pada mysql
- test_connection.js: nodejs script yang digunakan untuk menguji koneksi ke mysql
- wait-for-mysql.sh: digunakan container untuk menunggu proses mysql sehingga service mysql aktif dan bisa digunakan

Membangun image nodejs

```
$ docker images
$ docker build -t tutorial/nodejs .
$ docker images
```

Jalankan container berdasarkan image

```
docker run -d \
  -p 4000:4000 \
  -e MYSQL_USER=root \
  -e MYSQL_PASSWORD=password \
  -e MYSQL_DATABASE=test \
  -e MYSQL_HOST=db \
  --link mysqlku:db \
  --name nodejsku \
  -d tutorial/nodejs
```

Prosedur 3: Akses aplikasi

Akses homepage dari app:

```
$ curl -X GET localhost:4000
```

Tampilkan semua peserta:

```
$ curl -X POST localhost:4000/daftar
```

👉 Catatan:

- Agar tampilan hasil query diatas tersusun rapi, install jq
- di Ubuntu

```
$ sudo apt install -y jq
```

- di Centos

```
$ sudo dnf install -y jq
```

- Setelah instalasi jq selesai, tambahkan jq pada perintah get-students, seperti dibawah ini:

```
$ curl -X GET localhost:4000/daftar | jq
```

Tambahkan peserta

```
curl --header "Content-Type: application/json" \
  -d '{"nopeserta": 1130360, "nama": "Abizhar", "alamat": "jl. imam bonjol",
  "kota": "jakarta"}' \
  -X POST localhost:4000/tambah
```

Lihat kembali peserta

```
$ curl -X POST localhost:4000/get-students | jq
```

Terima kasih
