데이터 엔지니어링 기본

> *도커 엔진의 기본적인 동작 방식을 이해하기 위한 실습을 수행합니다*

- 목차
  -
  -
  -
  -
  -
  -

0 도커 기본 명령어 실습

도커 설치 및 서비스 기동

```
bash>
curl -fsSL https://get.docker.com/ | sudo sh     # 설치 스크립트 다운로드 및 설치
sudo usermod -a -G docker $USER     # sudo 없이 명령어를 실행하기 위해 현재 접속 중인 사용자 ($USER)에게 권한 주기
sudo usermod -a -G docker psyoblade    # 혹은 임의의 사용자 (psyoblade) 에게 권한 주기 - 다시 로그인 해야 적용됩니다

# sudo 없이 docker 명령이 실행이 가능한 지 테스트 합니다
docker info

# docker info 실행이 되지 않는다면 docker.sock 파일의 권한 때문일 수 있습니다
sudo chmod 666 /var/run/docker.sock
docker --version

# 반드시 별도로 경로를 지정하고 다운로드 받아야 1.20 이상 버전이 설치됩니다
sudo curl -L "https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version

# 기타 도구를 설치합니다
sudo apt-get install tree
```

코드 클론 및 환경설정

> *알파인 및 우분투 이미지를 통해 도커 컨테이너가 어떻게 동작하는지 이해하고, 기본 명령어를 통해 컨테이너 생성 및 삭제를 수행합니다*

```
mkdir work
cd work
git clone https://github.com/psyoblade/data-engineer-intermediate-training.git
cd data-engineer-intermediate-training/basic
```

도커 이미지 크기 비교

```
docker pull alpine
docker pull ubuntu
docker image ls
```

간단한 명령어 실습

```
docker run alpine top
docker run alpine uname -a
docker run alpine cat /etc/issue
```

bash 가 없기 때문에 /bin/sh 을 통해 --interactive --tty 를 생성해봅니다

```
docker run -it alpine /bin/sh
```

vim 도 없기 때문에 패키지 도구인 apk 및 apk add/del 를 통해 vim 을 설치 및 제거합니다

```
# apk { add, del, search, info, update, upgrade } 등의 명령어를 사용할 수 있습니다
$ apk update # 를 통해
$ apk add vim
$ apk del vim
```

각종 기본 도커 명령어 실습

```
docker inspect {CONTAINER_NAME}
docker image prune
docker stats {CONTAINER_NAME}
```

메모리 설정을 변경한 상태 확인

```
docker run -it -m 500m ubuntu
docker run -it --restart=on-failure:3 -m 500m ubuntu
docker run -it --restart=always -m 500m ubuntu
docker stats {CONTAINER_NAME}
docker container prune
```

도커 컨테이너 사용 후 삭제 여부

```
docker run -it ubuntu /bin/bash
docker run --rm -it ubuntu /bin/bash
```

1 도커 컨테이너 다운로드 및 실행

> *MySQL 기본 이미지를 베이스로 나만의 이미지를 생성합니다*

MySQL 데이터베이스 기동

```
$> docker run --name mysql
    -e MYSQL_ROOT_PASSWORD=rootpass \
    -e MYSQL_DATABASE=testdb \
```

```
    -e MYSQL_USER=user \
    -e MYSQL_PASSWORD=pass \
    -d mysql
$> docker exec -it mysql mysql -u user -p

mysql> use testdb;
mysql> create table foo (id int, name varchar(300));
mysql> insert into foo values (1, 'my name');
mysql> select * from foo;
```

볼륨을 추가하여 저장소 관리하기

- 아래와 같이 호스트의 현재 경로 혹은 절대경로를 넣으면 호스트와 쉐어할 수 있습니다

```
$> docker run --name mysql
  -e MYSQL_ALLOW_EMPTY_PASSWORD=yes
  -e MYSQL_DATABASE=testdb
  -e MYSQL_USER=user
  -e MYSQL_PASSWORD=pass
  -v ./data:/var/lib/mysql
  -d mysql
```

$> docker exec -it mysql mysql -u user -p mysql> use testdb; mysql> create table foo (id int, name varchar(300)); mysql> insert into foo values (1, 'my name'); mysql> select * from foo;

$> docker volume ls

$> docker run --name mysql -e MYSQL_ALLOW_EMPTY_PASSWORD=yes -e MYSQL_DATABASE=testdb -e MYSQL_USER=user -e MYSQL_PASSWORD=pass -v ./data:/var/lib/mysql -d mysql

```
* 일반적으로 볼륨은 파일이 많이 발생하므로 굳이 쉐어하지 않는 편이 좋습니다
```bash
$> docker run --name mysql
    -e MYSQL_ALLOW_EMPTY_PASSWORD=yes
    -e MYSQL_DATABASE=testdb
    -e MYSQL_USER=user
    -e MYSQL_PASSWORD=pass
    -v mysql_default:/var/lib/mysql
    -d mysql
```

2 도커 컴포즈를 통한 실행

- 도커 컴포즈를 통해서 커맨드라인 옵션을 설정을 통해 수행할 수 있습니다

```
$ cat docker-compose.yml
version: "3"
```

services: mysql: container_name: mysql image: mysql restart: always environment: MYSQL_ROOT_PASSWORD: rootpass MYSQL_DATABASE: testdb MYSQL_USER: user MYSQL_PASSWORD: pass volumes: - mysql_default:/var/lib/mysql

```
```
```

```
## 3 외부 볼륨을 통한 환경설정
> 외부 볼륨을 통한 환경설정 제공 및 설정을 실습합니다

### 캐릭터셋 변경 적용하기
```bash
$> cat custom/my.cnf
[client]
default-character-set=utf8

[mysqld]
character-set-client-handshake=FALSE
init_connect="SET collation_connection = utf8_general_ci"
init_connect="SET NAMES utf8"
character-set-server=utf8
collation-server=utf8_general_ci

[mysql]
default-character-set=utf8
```

MySQL 설정파일 사용

> *지정한 설정파일을 사용하고, 내부 볼륨을 통한 MySQL 기동으로 변경합니다*

```
$> cat docker-compose.yml
version: "3"

services:
  mysql:
    container_name: mysql
    image: mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
      MYSQL_DATABASE: testdb
      MYSQL_USER: user
      MYSQL_PASSWORD: pass
    volumes:
      - ./custom:/etc/myslq/conf.d
      - mysql_utf8:/etc/myslq/conf.d
```

4 도커 이미지 빌드 및 실행

- 아래와 같이 초기화 파일을 생성합니다

```
$> cat init/testdb.sql
DROP TABLE IF EXISTS `seoul_popular_trip`;
CREATE TABLE `seoul_popular_trip` (
`category` int(11) NOT NULL,
`id` int(11) NOT NULL,
`name` varchar(100) DEFAULT NULL,
`address` varchar(100) DEFAULT NULL,
`naddress` varchar(100) DEFAULT NULL,
```
```

```
`tel` varchar(20) DEFAULT NULL,
`tag` varchar(500) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
LOCK TABLES `seoul_popular_trip` WRITE;
INSERT INTO `seoul_popular_trip` VALUES (0,25931,'통인가게 ','110-300 서울 종로구 관훈동 16 ','03148 서울 종로구 인사동길 32 (관훈동) ','02-733-4867','오래된가게,고미술,통인정신,통인가게,공예샵,현대공예');
UNLOCK TABLES;
```

- 도커파일을 생성합니다

```
$> cat Dockerfile
ARG BASE_CONTAINER=mysql:5.7
FROM $BASE_CONTAINER
LABEL maintainer="student@lg.com"
```

ADD ./init /docker-entrypoint-initdb.d

EXPOSE 3306

CMD ["mysqld"]

```
* 로컬에서 도커 이미지를 빌드합니다
```bash
$> docker build -t local/mysql:5.7 .
```

빌드된 이미지로 다시 테스트

```
$ docker image ls
$ cat docker-compose.yml
version: "3"

services:
  mysql:
    container_name: mysql
    image: local/mysql:5.7
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
      MYSQL_DATABASE: testdb
      MYSQL_USER: user
      MYSQL_PASSWORD: pass
    volumes:
      - ./custom:/etc/mysql/conf.d
      - mysql_utf8:/var/lib/mysql

$> docker exec -it mysql mysql -u user -p
$> use testdb;
$> select * from seoul_popular_trip;
```

5 도커 컴포즈 통한 여러 이미지 실행
```

> *MySQL 과 phpMyAdmin 2가지 서비스를 실행합니다*

도커 컴포즈를 통해 phpMyAdmin 추가 설치

```
$> cat docker-compose.yml
version: "3"

services:
  mysql:
    image: psyoblade/mysql:5.7
    container_name: mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
      MYSQL_DATABASE: testdb
      MYSQL_USER: user
      MYSQL_PASSWORD: pass
    volumes:
      - ./custom:/etc/mysql/conf.d
      - mysql_utf8:/var/lib/mysql
  php:
    image: phpmyadmin/phpmyadmin
    container_name: phpmyadmin
    links:
      - mysql
    environment:
      PMA_HOST: mysql
      PMA_PORT: 3306
      PMA_ARBITRARY: 1
    restart: always
    ports:
      - 8183:80
```

- phpMyAdmin 사이트에 접속하여 mysql/user/pass 로 접속합니다

MySQL 이 정상적으로 로딩된 이후에 접속하도록 설정합니다

- 테스트 헬스체크를 통해 MySQL 이 정상 기동되었을 때에 다른 어플리케이션을 띄웁니다

```
$> cat docker-compose.yml
version: "3"
```

services: mysql: image: psyoblade/mysql:5.7 container_name: mysql restart: always environment: MYSQL_ROOT_PASSWORD: rootpass MYSQL_DATABASE: testdb MYSQL_USER: user MYSQL_PASSWORD: pass healthcheck: test: ["CMD", "mysqladmin" ,"ping", "-h", "localhost"] interval: 3s timeout: 1s retries: 3 volumes: - ./custom:/etc/mysql/conf.d - mysql_utf8:/var/lib/mysql php: image: phpmyadmin/phpmyadmin container_name: phpmyadmin depends_on: - mysql links: - mysql environment: PMA_HOST: mysql PMA_PORT: 3306 PMA_ARBITRARY: 1 restart: always ports: - 8183:80

```
```