배포

Docker

▼ Docker정보

버전 : 23.0.1

▼ 설치과정

1. Update the `apt` package index and install packages to allow `apt` to use a repository over HTTPS:

```
sudo apt-get update
sudo apt-get install ca-certificates
sudo apt-get install curl
sudo apt-get install gnupg
sudo apt-get install lsb-release
```

2. Add Docker’s official GPG key:

```
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

3. Use the following command to set up the repository:

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Install Docker Engine

1. Update the `apt` package index:

```
sudo apt-get update
```

if, Receiving a GPG error when running `apt-get update`? 위에 명령에서 에러 안나면 스킵!!

```
sudo chmod a+r /etc/apt/keyrings/docker.gpg
sudo apt-get update
```

2. Install Docker Engine, containerd, and Docker Compose.

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

3. Verify that the Docker Engine installation is successful by running the `hello-world` image: (잘 되는지 테스트)

```
sudo docker run hello-world
```

Docker Compose 설치

1. 설치

```
sudo curl -L https://github.com/docker/compose/releases/download/v2.15.1/docker-compose-linux-x86_64 -o /usr/local/bin/docker-comp
```

2. 권한 설정

```
sudo chmod +x /usr/local/bin/docker-compose
```

3. 심볼릭 링크 생성

```
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

4. 설치 확인

```
sudo docker-compose --version
```

Jenkins

▼ Jenkins정보

버전 : Jenkins 2.319.2 LTS

▼ 설치과정

1. 도커 허브에서 jenkins/jenkins:lts 이미지 pull

```
sudo docker pull jenkins/jenkins:lts
```

2. 젠킨스 컨테이너 실행

```
sudo docker run -d --name jenkins -p 8090:8080 -v /jenkins:/var/jenkins_home -v /usr/bin/docker:/usr/bin/docker -v /var/run/docker
```

계정명 : subsub1221

pw : enr05778642@

gitlab access token : KRY7FowgETYt3vVJSzqG

pipeline

- pipeline(nginx,react) 테스트용

Nginx

▼ Nginx정보

버전 : 1.18.0

▼ 설치과정

1. nginx 설치
    - 도커에 설치 하지 않고 EC2에 직접 설치

```
# 설치
sudo apt-get install nginx

# 설치 확인 및 버전 확인
nginx -v
```

2. letsencrypt 설치
    - HTTPS접속을 위해 인증서 발급 과정

```
sudo apt-get install letsencrypt

sudo systemctl stop nginx

sudo letsencrypt certonly --standalone -d i8b304.p.ssafy.io
```

3. 인증서 발급 후, /etc/nginx/sites-available로 이동 nginxec2.conf 파일 생성 후 아래 코드 작성
    a. touch명령어로 conf파일 생성
    b. vi로 접속 후 코드 작성
        - sudo로 vi접속하거나 conf파일의 write권한을 부여해 줘야함

        ▼ vi명령어

        i : 수정 모드 시작

        esc : 명령모드로 변경

        :(콜론) : 명령모드에서 마지막 줄로 이동

        q : vi 종료

        w : 저장하기
        - 소셜 로그인 사용 안할 시

```
server {

        location /{
                proxy_pass http://localhost:3000;
        }

        location /api {
                proxy_pass http://localhost:8080/api;
        }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/i8b304.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i8b304.p.ssafy.io/privkey.pem; # managed by Certbot
    # include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    # ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    if ($host = i8b304.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

        listen 80;
        server_name i8b304.p.ssafy.io;
    return 404; # managed by Certbot
}
```

- 소셜 로그인 사용 시

```
server {

    location /{
        proxy_pass http://localhost:3000;

        # proxy_http_version 1.1;
        # proxy_set_header Upgrade $http_upgrade;
        # proxy_set_header Connection "upgrade";
        # proxy_set_header Host $http_host;
        # proxy_set_header X-Real-IP $remote_addr;
        # proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location /api/ {
        proxy_pass http://localhost:8080/;

        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/i8b304.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i8b304.p.ssafy.io/privkey.pem; # managed by Certbot
    # include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    # ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    if ($host = i8b304.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

        listen 80;
        server_name i8b304.p.ssafy.io;
    return 404; # managed by Certbot
}
```

4. 마무리

```
sudo ln -s /etc/nginx/sites-available/nginxec2.conf /etc/nginx/sites-enabled/nginxec2.conf

#success 확인
sudo nginx -t

sudo systemctl restart nginx
```

MySQL(DB)

▼ mysql 정보

mysql 버전 : 8.0.32

mysql rootpassword : enr05778642@
mysql userid : B310_sub
mysql password : sub1221

▼ 설치과정

1. mysql 최신 버전의 이미지를 가져온다.

```
$ docker pull mysql:latest
```

2. Docker 컨테이너 볼륨 설정
    - Docker 컨테이너가 삭제되었을 시 DB가 통째로 날아가는 것을 방지

```
# volume 생성
$ docker volume create mysql-volume
```

```
# volume 확인
$ docker volume ls
```

3. 생성한 volume을 컨테이너에 마운팅하여 실행시킨다. 루트 비번 넣어줘야함

```
$ docker run -d --name mysql-container -p 3306:3306 -v mysql-volume:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=1234 mysql:latest
```

- -name : **mysql-container**라는 이름의 컨테이너 생성
- p : **호스트의 3306** 포트를 **컨테이너의 3306** 포트와 포트포워딩
- v : **호스트의 mysql-volume**의 volume에 **컨테이너의 /var/lib/mysql** volume을 **마운팅**
- e : MYSQL_ROOT_PASSWORD 환경변수 값을 '1234'로 지정
- d : daemon으로 실행

4. 컨테이너에 접속하기

```
$ docker exec -it mysql-container bash
```

5. MySQL 서버에 접속하기

```
$ mysql -u root -p
# 컨테이너 생성 시 입력했던 패스워드 입력
Enter password:
...
mysql>
```

6. 이제 새로운 user를 만든다.

```
# USER 생성, '%'는 모든 IP에서 접속 가능
mysql> CREATE USER test01@'%' identified by '1234';
# 생성한 USER에 모든 권한 부여
mysql> GRANT ALL PRIVILEGES ON *.* to test01@'%';
# 변경 사항 적용
mysql> FLUSH PRIVILEGES;
mysql> exit;
```

7. 생성한 user로 MySQL 서버에 접속한다.

```
$ mysql -u test01 -p
Enter password:
...
mysql>
```

8. 데이터베이스를 하나 생성해본다.

```
mysql> CREATE DATABASE test;
mysql> SHOW DATABASES;
```

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| test               |
+--------------------+
5 rows in set (0.00 sec)
```

9. 외부에서 접속하기

- MySQL workbench에서 외부 접속

: Hostname에 서버의 IP or 도메인을 입력하고, 외부접속을 위해 생성했던 사용자명과 비밀번호를 입력해준다.