🔧빌드 및 배포

🔨 빌드

1. back- end

> node.js 설치(16.14.2 LTS) 설치 `npm install` 을 통해 node-modules 다운
> `npm start`를 통해 node.js 실행

2. front-end

> `npm install`을 통해 node-modules 설치
> `npm start`를 통해 실행

💾 배포

👉설치

> mysql(8.0.0) 설치

> node(16.13.12)와 npm(8.4.0) 설치

> docker(20.10.12) 설치

> nginx(1.18.0) 설치

👉 DataBase

> mysql 접속
> ID : jeans
> pw : cjdcnsdmsqkfhwlrma

👉 back-end

> 생성해둔 Dockerfile을 이용
>
> - `docker build . t back:0.1` 를 통해 docker 이미지 생성
> - `docker run -i -t -d -p 3000:3000 --name nodeserver back:0.1` docker 컨테이너 실행
> - host:3000에 접속해 실행을 확인한다.

👉 front-end

> - `npm install` 을 이용해 node-modules와 package-lock.json 생성
> - `npm run build`를 실행 빌드 파일 생성
> - nginx conf 파일의 server-name 수정
> - `systemctl start nginx`를 이용 nginx 실행
> - https 적용을 위해 certbot을 이용해 ssl 인증서 발급
> - 다시 nginx conf 파일을 수정

```
server {
  root [front 빌드파일 경로];

  index index.html;

  server_name [도메인 주소];

  location / {
      try_files $uri $uri/ /index.html =404;
  }

      listen [::]:443 ssl ipv6only=on; # managed by Certbot
      listen 443 ssl; # managed by Certbot

      ssl_certificate /etc/letsencrypt/live/j6a405.p.ssafy.io/fullchain.pem;
# managed by Certbot
      ssl_certificate_key
/etc/letsencrypt/live/j6a405.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

  location /api {
          proxy_pass http://[도메인주소]:3000;
      }
  location /web3 {
          proxy_pass http://[BESU 네트워크];
      }
}
server {
   if ($host = [도메인주소]) {
       return 301 https://$host$request_uri;
    } # managed by Certbot

  listen 80;
  listen [::]:80;

  server_name [도메인주소];
    return 404; # managed by Certbot
}
```

위와 같이 nginx conf 수정

`systemctl reload nginx` 명령어 실행 nginx를 reload하여 https 적용과 실행을 확인한다.

외부 서비스

AWS S3

> 이미지를 저장하기 위한 backend/env 폴더의 json형식으로 필요한 정보 有