배포 정리 문서

🌐 배포 설정

프론트엔드 빌드 및 배포

1. **git clone**

```
git clone -b frontend https://lab.ssafy.com/s08-blockchain-contract-sub2/S08P22A208.git
```

2. FE **폴더로 이동**

```
cd ./S08P22A208/FE
```

3. **Dockerfile 작성**

```
# base image
FROM node:16-alpine

# set working directory
WORKDIR /app

# copy package.json and package-lock.json
COPY package*.json ./

# install dependencies
RUN npm install

# copy source code
COPY . .

# build Next.js app
RUN npm run build

# expose port 3000
EXPOSE 3000

# start the app
CMD ["npm", "start"]
```

4. **빌드**

```
npm install
npm run build
```

5. **도커 이미지 생성**

```
sudo docker build --tag {도커 이미지 이름} .
```

6. **도커 컨테이너 생성**

```
sudo docker run -d -p {외부에서 연결할 포트번호}:3000 --name {도커 컨테이너 이름} -v /etc/localtime:/etc/localtime:ro -e TZ=
Asia/Seoul {도커 이미지 이름}
```

백엔드 빌드 및 배포

1. **git clone**

```
git clone -b backend https://lab.ssafy.com/s08-blockchain-contract-sub2/S08P22A208.git
```

2. **backend/b612 폴더로 이동**

```
cd ./S08P22A208/backend/b612
```

3. **Dockerfile 작성**

```
FROM openjdk:8-jre
COPY build/libs/*.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

4. **빌드**

```
chmod +x ./gradlew
./gradlew build
```

5. **도커 이미지 생성**

```
sudo docker build --tag {도커 이미지 이름} .
```

6. **도커 컨테이너 생성**

```
sudo docker run -d -p {외부에서 연결할 포트번호}:8080 --name {도커 컨테이너 이름} -v /etc/localtime:/etc/localtime:ro -e TZ=
Asia/Seoul {도커 이미지 이름}
```

---

nginx.conf 파일

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
        worker_connections 768;
        # multi_accept on;
}

http {

        ##
        # Basic Settings
        ##

        sendfile on;
        tcp_nopush on;
        tcp_nodelay on;
        keepalive_timeout 65;
        types_hash_max_size 2048;
        # server_tokens off;

        # server_names_hash_bucket_size 64;
        # server_name_in_redirect off;

        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        ##
        # SSL Settings
        ##

        ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
        ssl_prefer_server_ciphers on;

        ##
        # Logging Settings
        ##

        access_log /var/log/nginx/access.log;
        error_log /var/log/nginx/error.log;

        ##
        # Gzip Settings
        ##

        gzip on;

        # gzip_vary on;
        # gzip_proxied any;
        # gzip_comp_level 6;
        # gzip_buffers 16 8k;
        # gzip_http_version 1.1;
        # gzip_types text/plain text/css application/json application/javascript text/xml application/xml applicat
ion/xml+rss text/javascript;

        ##
        # Virtual Host Configs
        ##

        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
}
```

nginx 설정

1. **nginx 설치**

```
sudo apt-get install nginx
```

2. **/etc/nginx/sites-available로 이동후 새로운 이름의 파일 생성(ex: b612.conf)**

```
server {
        # 프론트 연결(포트 번호는 본인의 프론트 포트번호를 입력)
        location / {
                proxy_pass http://localhost:3000;
        }

        # 백엔드 연결(포트 번호는 본인의 백엔드 포트번호를 입력)
        location /api {
                proxy_pass http://localhost:8080/api;
        }
        # 광장에서 실시간 채팅 웹소켓 연결
        location /ws/chat {
                proxy_pass http://localhost:8080/ws/chat;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_set_header Host $host;
        }
        # 광장에서 실시간 이동 웹소켓 연결
        location /ws/move {
                proxy_pass http://localhost:8080/ws/move;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_set_header Host $host;
        }
        # Swagger
        location ~ ^/(swagger|webjars|configuration|swagger-resources|v2|csrf) {
               proxy_pass http://localhost:8080;
               proxy_set_header Host $host;
               proxy_set_header X-Real-IP $remote_addr;
               proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
               proxy_set_header X-Forwarded-Proto $scheme;
        }
    listen 443 ssl; # managed by Certbot
    # 도메인 이름을 써줘야함
    ssl_certificate /etc/letsencrypt/live/j8a208.p.ssafy.io//fullchain.pem; # managed by Certbot
    # 도메인 이름을 써줘야함
    ssl_certificate_key /etc/letsencrypt/live/j8a208.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 {
    # 도메인 이름을 입력
    if ($host = j8a208.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

        listen 80;
        server_name j8a208.p.ssafy.io;
    return 404; # managed by Certbot
}
```

3. nginx 실행

```
sudo service nginx start
```

최종 master branch jenkins 자동배포 설정

Execute Shell

```
cd backend
cd b612
chmod +x ./gradlew
./gradlew build
docker build --tag b612_master_be .
docker stop b612_master_be
docker rm b612_master_be
docker run -d -p 8080:8080 --name b612_master_be  -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seoul b612_master_be

cd ..
cd ..
cd FE
npm install
npm run build
docker build --tag b612_master_fe .
docker stop b612_master_fe
docker rm b612_master_fe
docker run -d -p 3000:3000 --name b612_master_fe  -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seoul b612_master_fe
```

---

solidity 배포

1. https://remix.ethereum.org/ 접속
2. vscode에서 파일 실행 후 다음 명령어로 remix에 접속

   ```
   remixd
   ```

3. remixd에서 default_workspace를 connect to localhost로 변경
4. 파일 선택 후 Compiler 눌러서 컴파일
5. 파일 선택 후 Deploy & run transactions에서 배포에 필요한 변수 입력 후 Deploy
6. 배포 후 생성된 배포주소와 abi 프론트에게 전달