포팅 매뉴얼

📘 댕댕레인저 프로젝트 배포 가이드

```
[ 개발 환경 ]

VS Code : 1.81.1
IntelliJ : 17.0.7+10-b829.16 amd64
Remix IDE

spring boot : 2.7.13
JDK : OpenJDK 11.0.18
JVM : JDK와 동일
expo: 49.0.8
react-native: 0.72.4
mobx: 6.10.2
npm: 10.1.0
node.js: 18.16.1
express: 4.18.2
ethers.js: 6.7.1
recoil: 0.7.7
```

```
[ DB ]

mariaDB : 11.1.2-MariaDB, client 15.2
redis : 7.2.1
```

```
[ 서버 환경 ]

EC2 Ubuntu 20.04 LTS
nginx : 1.18.0 (Ubuntu)
certbot: 2.7.0
docker: 24.0.6
jenkins: 2.414.1
```

```
[ 외부 서비스 ]

AWS S3
NFT.Storage
PolygonScan
```

```
[ 협업 툴 ]

Notion
MatterMost
Jira
Discord
Gitlab
```

초기 세팅

git clone

```
git clone https://lab.ssafy.com/s09-blockchain-nft-sub2/S09P22A209.git
```

Nginx

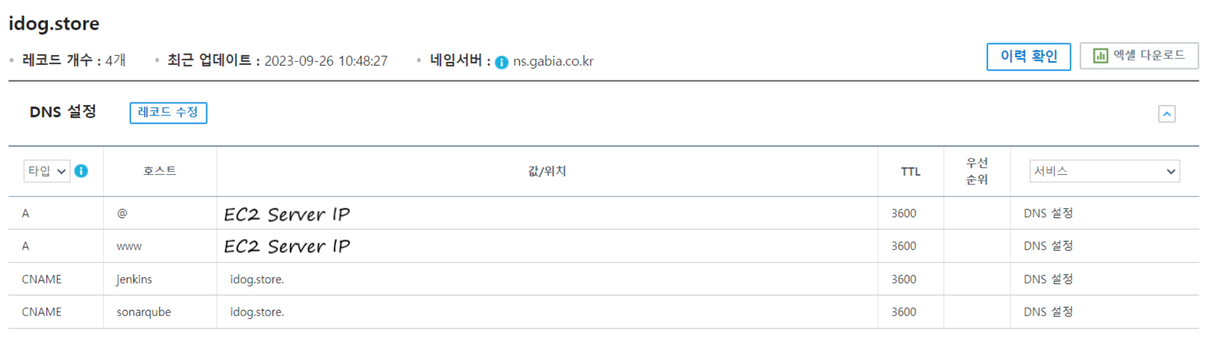
방화벽 설정

```
sudo ufw default deny incoming // 모든 인바운드 연결 차단
sudo ufw default allow outgoing // 모든 아웃바운드 연결 허용
sudo ufw allow ssh // 22번 포트 허용
sudo ufw allow http // 80번 포트 허용
sudo ufw allow https // 443번 포트 허용
sudo ufw allow 3310
sudo ufw allow 8002
sudo ufw enable

sudo ufw status
```

Nginx 설치

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install nginx
sudo service nginx start

# Nginx 삭제
sudo apt-get -y remove --purge nginx nginx-full nginx-common
```

도메인 적용


idog.store

• **레코드 개수 :** 4개 • **최근 업데이트 :** 2023-09-26 10:48:27 • **네임서버 :** ns.gabia.co.kr

DNS 설정

타입	호스트	값/위치	TTL	우선 순위	서비스
A	@	EC2 Server IP	3600		DNS 설정
A	www	EC2 Server IP	3600		DNS 설정
CNAME	jenkins	idog.store.	3600		DNS 설정
CNAME	sonarqube	idog.store.	3600		DNS 설정

SSL 적용

```
wget https://dl.eff.org/certbot-auto
# snap을 이용하여 core 설치 -> snap을 최신 버전으로 유지하기 위해 설치
sudo snap install core

# core를 refresh 해준다.
sudo snap refresh core

# 기존에 잘못된 certbot이 설치되어있을 수도 있으니 삭제 해준다.
sudo apt remove certbot

# certbot 설치
sudo snap install --classic certbot

# certbot 명령을 로컬에서 실행할 수 있도록 snap의 certbot 파일을 로컬의 cerbot과 링크(연결) 시켜준다. -s 옵션은 심볼릭링크를 하겠다는 것.
ln -s /snap/bin/certbot /usr/bin/certbot
```

```
sudo certbot --nginx

# 2. 공개키 경로
/etc/letsencrypt/live/animaid.co.kr/fullchain.pem

# 3. 비밀키 경로
 /etc/letsencrypt/live/animaid.co.kr/privkey.pem
```

Nginx 설정

spring, express 서버 설정

```
upstream backend {
  server 127.0.0.1:8080;
}

upstream blockchain {
  server 127.0.0.1:3000;
}

server {
  listen 80;
  server_name 3.38.98.134 k9a103.p.ssafy.io;
  location / {
    return 301 $scheme://animaid.co.kr$request_uri;
  }
}

server {
  server_name animaid.co.kr www.animaid.co.kr;

  location /api {
    proxy_pass http://backend;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    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;
  }

  location /blockchain {
    proxy_pass http://blockchain;
    proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                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;

  }

  location /ws-stomp { # 백엔드 웹소켓
    proxy_pass http://backend/ws-stomp;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header HOST $http_host;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
  }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/animaid.co.kr/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/animaid.co.kr/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 ($host = www.animaid.co.kr) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    if ($host = animaid.co.kr) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


  listen 80;
  server_name animaid.co.kr www.animaid.co.kr;
    return 404; # managed by Certbot




}
```

jenkins 설정

```
upstream jenkins {
  keepalive 32;
  server 127.0.0.1:8001;
}

# Required for Jenkins websocket agents
map $http_upgrade $connection_upgrade {
  default upgrade;
  '' close;
}

server { # Listen on port 80 for IPv4 requests
  server_name jenkins.animaid.co.kr; # 이 부분을 자신의 주소로 변경하여야 합니다

    # this is the jenkins web root directory
    # (mentioned in the output of "systemctl cat jenkins")
    root /var/run/jenkins/war/;
    access_log /var/log/nginx/jenkins.access.log;
    error_log /var/log/nginx/jenkins.error.log;

    # pass through headers from Jenkins that Nginx considers invalid
    ignore_invalid_headers off;

    location ~ "^/static/[0-9a-fA-F]{8}\/(.*)$" {
        # rewrite all static files into requests to the root
        # E.g /static/12345678/css/something.css will become /css/something.css
        rewrite "^/static/[0-9a-fA-F]{8}\/(.*)" /$1 last;
    }

    location /userContent {
        # have nginx handle all the static requests to userContent folder
        # note : This is the $JENKINS_HOME dir
        root /var/lib/jenkins/;
        if (!-f $request_filename){
            # this file does not exist, might be a directory or a /**view** url
            rewrite (.*) /$1 last;
            break;
        }
        sendfile on;
    }

    location / {
        sendfile off;
  proxy_pass http://jenkins;
        proxy_redirect default;
        proxy_http_version 1.1;

        # Required for Jenkins websocket agents
        proxy_set_header Connection $connection_upgrade;
        proxy_set_header Upgrade $http_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;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_max_temp_file_size 0;
        #this is the maximum upload size
        client_max_body_size 10m;
        client_body_buffer_size 128k;
        proxy_connect_timeout 90;
        proxy_send_timeout 90;
        proxy_read_timeout 90;
        proxy_buffering off;
        proxy_request_buffering off; # Required for HTTP CLI commands
        proxy_set_header Connection ""; # Clear for keepalive
    }


    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/animaid.co.kr/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/animaid.co.kr/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 ($host = jenkins.animaid.co.kr) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


  server_name jenkins.animaid.co.kr;
    listen 80;
    return 404; # managed by Certbot


}
```

프로젝트 빌드 및 배포

SpringBoot

[Dockerfile]

```
FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

[SpringBoot Docker image Build]

```
cd S09P31A103/back/dangdangranger
docker build -t dognfta209/idog-back .
docker push dognfta209/idog-back:$BUILD_NUMBER
```

[SpringBoot 실행]

```
docker pull dognfta209/dangdang-back:$BUILD_NUMBER
docker run --name dangdang-back -d -p 8080:8080 dognfta209/dangdang-back:$BUILD_NUMBER
```

Express

[env]

```
RPC_URL=https://polygon-rpc.com/
SECRET_SALT=디지털 지갑 접근 비밀번호 SALT
NFT_STORAGE_KEY=NFT.Storage API key
BUCKET_NAME=ppobbi
ACCESS_KEY_ID=S3 ACCESS KEY
SECRET_ACCESS_KEY=S3 SECRET KEY
```

[Dockerfile]

```
FROM node:18-alpine

# package 내에 설정된 라이브러리들 현재 위치에 복사
COPY package*.json ./

RUN npm install

# 모든 파일들 현재 위치에 복사
COPY . .

CMD ["npm", "run", "dev"]
```

[Express Docker image Build]

```
cd S09P31A103/express
docker build -t dognfta209/dangdang-bc .
docker push dognfta209/dangdang-bc:$BUILD_NUMBER
```

[Express 실행]

```
docker pull dognfta209/idog-bc:$BUILD_NUMBER
docker run --name dangdang-bc -d -p 3000:3000 dognfta209/dangdang-bc:$BUILD_NUMBER
```

Android

[env]

```
# Google Login
WEB_CLIENT_ID=
# 구글 맵 위도경도 활용 API
GEOCODING_API_KEY=

# AWS S3
AWS_ACCESS_KEY=
AWS_REGION=ap-northeast-2
AWS_BUCKET=dangdangranger
AWS_SECRET_ACCESS_KEY=


# Node Express
RPC_URL=https://polygon-rpc.com/
```

```
SECRET_SALT=디지털 지갑 접근 비밀번호 SALT
NFT_STORAGE_KEY=NFT.Storage API key
MINT_DOG_TOKEN_ADDRESS=0xdB983532a92837Ee0faF0e67854993a858f621d2 // 컨트랙트 주소
ADMIN_WALLET_PRIVATE_KEY=관리자 지갑 개인키 // 민팅 가스비 충당용
POLYGON_API_KEY=폴리곤 API Key
MAPBOX_ACCESSTOKEN=지도 Access token
OBJECT_DETECT_API_KEY=객체 인식 api 키
OBJECT_DETECT_URL=http://aiopen.etri.re.kr:8000/ObjectDetect
```

[keystore 생성]

```
cd C:\Program Files\Java\jdkx.x.x_x\bin

# 관리자 계정으로 cmd 혹은 bash 실행
keytool -genkeypair -v -storetype PKCS12 -keystore "키스토어 이름".keystore -alias "키스토어 별칭" -keyalg RSA -keysize 2048 -validity 10000
```

[apk, aab 빌드]

```
cd S09P31A103/front/android

# apk 빌드
./gradlew app:assembleRelease

# aab 빌드
./gradlew bundleRelease
```

환경 변수, 계정, 프로퍼티 파일 목록

Spring

- application.yml
- application-release.yml
- applicaition-jwt.yml
- applicaition-appkey.yml
- application-oauth.yml
- application-mmlog.yml
- applicaition-redisrelease.yml

```
spring:
  profiles:
    group:
      release:
        - oauth
        - jwt
        - redisrelease
        - appkey
      develop:
        - oauth
        - jwt
        - redisdev
        - appkey
    # release(prod) / develop(dev) 중에 선택
    active: release
```

```
spring:
  config:
    activate:
      on-profile: "jwt"

jwt:
  secret: 시크릿키
  refresh-expired-in: 2_627_000_000
  access-expired-in: 1_800_000
```

```
spring:
  config:
    activate:
      on-profile: "appkey"

appkey:
  polygon:
    value: 폴리곤 API키
```

```
spring:
  security:
    oauth2:
      client:
        registration:
          google:
            client-id: 83063651083-fh2o1f6o59fna1pl74bb9kqr2fivjgf9.apps.googleusercontent.com
            client-secret: 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
            redirect-uri: http://localhost:8080/user/login/oauth2/code/google
            scope:
            - profile
            - email
```

```
server:
  port: 8080

spring:
  config:
    activate:
      on-profile: "release"

  initDb:
    enable: true

  datasource:
    url: jdbc:mariadb://animaid.co.kr:3310/dangdangranger?characterEncoding=UTF-8&serverTimezone=KST
    username: 계정이름
    password: 비밀번호
    driver-class-name: org.mariadb.jdbc.Driver
  jpa:
    hibernate:
    properties:
      hibernate:
        show_sql: true
        format_sql: true
        default_batch_fetch_size: 100

logging.level:
  com.haru.ppobbi: debug
  org.hibernate.SQL: debug
  org.hibernate.type: trace #스프링 부트 2.x, hibernate5
# org.hibernate.orm.jdbc.bind: trace #스프링 부트 3.x, hibernate6
```

```
spring:
  config:
    activate:
      on-profile: "redisrelease"

  redis:
    host: 서버IP
    port: 8002
    password: 비밀번호
    ttls:
      user-info: 3_600_000
```

Jenkins

```
pipeline {
    agent any

    environment {
        back_repository = "dognfta209/dangdang-back"  //docker hub id와 repository 이름
        blockchain_repository = "dognfta209/dangdang-bc"
        DOCKERHUB_CREDENTIALS = credentials('docker-hub') // jenkins에 등록해 놓은 docker hub credentials 이름
        dockerImage = ''
    }

    stages {
        // git clone
        stage('Clone repository') {
            steps {
                git branch: 'develop',
                    url: 'https://lab.ssafy.com/s09-final/S09P31A103.git',
                    credentialsId: 'Donggyeom'
            }

            post {
                success {
                    sh 'echo "Successfully Cloned Repository"'
                }
                failure {
                    sh 'echo "Fail Cloned Repository"'
                }
            }
        }

        // build
        stage('Build backend jar') {
            steps {
                dir('back') {
                    dir('dangdangranger') {
                        sh """
                        chmod +x gradlew
                        ./gradlew clean build --exclude-task test
                        """

                        sh 'ls -al ./build'
                    }
                }
            }
            post {
                    success {
                        echo 'gradle build success'
                    }

                    failure {
                        echo 'gradle build failed'
                    }
                }
        }

        stage('Build spring image'){
            steps{
                dir('back') {
                    dir('dangdangranger') {
                        sh 'echo " Backend Image Bulid Start"'
                        script {
                            dockerImage = docker.build back_repository + ":$BUILD_NUMBER"
                        }
                    }
                }
            }
            post {
                success {
                    sh 'echo "Bulid back Docker Image Success"'
                }

                failure {
                    sh 'echo "Bulid back Docker Image Fail"'
                }
            }
        }

        stage('Build blockchain') {
            steps {
                dir('blockchain') {
                    dir('express') {
                        sh 'echo " Front Image Build Start"'
                        script {
                            dockerImage = docker.build blockchain_repository + ":$BUILD_NUMBER"
                        }
                    }
                }
            }

            post {
                success {
                    sh 'echo "Build blockchain Docker Image Success"'
                }
                failure {
                    sh 'echo "Bulid blockchain Docker Image Fail"'
                }
            }
        }

        stage('Push Docker') {
            steps {
                echo 'Push Docker'
                sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin' // docker hub 로그인
                script {
                    sh 'docker push $back_repository:$BUILD_NUMBER' //docker push
                    sh 'docker push $blockchain_repository:$BUILD_NUMBER'
                }
            }
            post {
                success {
                    sh 'echo "Push Docker Image Success"'
                }
                failure {
                    sh 'echo "Push Docker Image Fail"'
                }
            }
        }

        stage('Stop Container') {
             steps {
                sh 'docker stop dangdang-back'
                sh 'docker stop dangdang-bc'
            }

            post {
                success {
                    echo 'stop container success'
            }

                failure {
                    echo 'stop container failed'
                }
            }
        }

        stage('Remove Container') {
            steps {
                sh 'docker rm dangdang-back'
                sh 'docker rm dangdang-bc'
            }

            post {
                success {
                    echo 'remove container success'
                }

                failure {
                    echo 'remove container failed'
                }
            }
        }

        stage('Deploy') {
            steps {
                sh 'docker run --name dangdang-back -d -p 8080:8080 -e TZ=Asia/Seoul --restart=always $back_repository:$BUILD_NUMBER'
                sh 'docker run --name dangdang-bc -d -p 3000:3000 -e TZ=Asia/Seoul --restart=always $blockchain_repository:$BUILD_NUMB
            }

            post {
                success {
                    echo 'deploy success'
                }

                failure {
                    echo 'deploy failed'
                }
            }
        }

        stage('Cleaning up') {
            steps {
                sh 'docker system prune -f -a'
            }

            post {
                success {
                    sh 'echo "clean docker success"'
                }
                failure {
                    sh 'echo "clean docker fail"'
                }
            }
        }
    }
}
```