포팅매뉴얼 - 서버 세팅

3. Docker

4. Jenkins

5. Portainer

6. Jenkins ci/cd

7. NginX 리버스 프록시 설정

개발환경

💡 **프로젝트에서 사용한 프로그램들의 버전을 정리합니다.**

Java	`17`
Spring Boot	`3.3.1`
Gradle	`gradle-8.8`
Node.js	`v16.20.2`
MySQL	`8.0.37`
Jenkins	`2.454`
Nginx	`1.25.5`
Redis	`7.4.0`
VS Code	`1.90.2`
IntelliJ	`17.0.11+1-b1207.24 amd64`

프로젝트 사용 도구

💡 **프로젝트 과정에서 사용한 도구들 입니다.**

이슈 관리	**JIRA**
형상 관리	**Gitlab**
커뮤니케이션	**Notion,Mattermost**
디자인	**Figma**
UCC	**Movavi video editor**
CI/CD	**EC2,Jenkins,Docker**

배포환경

💡 **서버구성에 사용된 코드를 정리합니다.**

최초 서버 설정

```
0. sudo su  -> root권한얻기

1. 우분투 시스템 패키지 업데이트
sudo apt-get update
2. 필요한 패키지 설치
sudo apt-get install apt-transport-https ca-certificates curl


3. Docker의 공식 GPG키를 추가
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sud


4. Docker의 공식 apt 저장소를 추가
sudo add-apt-repository "deb [arch=amd64] https://download.do


5. 시스템 패키지 업데이트
sudo apt-get update
6. Docker 설치
sudo apt-get install docker-ce docker-ce-cli containerd.io

7. Docker가 설치 확인

7-1 도커 실행상태 확인
sudo systemctl status docker
```

```
7-2 도커 실행
sudo docker run hello-world

8. docker-compose 설치
sudo curl -L https://github.com/docker/compose/releases/lates

9. docker-compose 실행 권한 주기
chmod +x /usr/local/bin/docker-compose
```

JAVA 설치

```
1. java17 설치
    - sudo apt update
    - sudo apt install openjdk-17-jdk
    - sudo vi /etc/environment
    - JAVA_HOME="/usr/lib/jvm/java-17-openjdk-amd64"
2. mysql 설치
    - sudo apt update
    - sudo apt-get install mysql-server
    - mysql --version
3. mysql 루트 계정 비밀번호 설정
    - sudo mysql -u root -p
    - select user, Host, plugin from mysql.user; => root계정
의 plugin이 auto_socket인데 mysql_native_password로 변경해주면서
새로운 비밀번호를 설정해줘야 한다.
    - alter user 'root'@'localhost' identified with mysql_n
ative_password by '새로운 비밀번호';
    - grant all privileges on *.* to 'root'@'localhost';
    - flush privileges;
    - exit;
    - mysql -u root -p로 변경된 비밀번호로 접속가능한지 확인
4. mysql 작업용 유저 생성
    - create user ssafy;
    - alter user 'ssafy'@'%' identified by '새로운 비밀번호';
:%는 접속허용ip를 모두 허용하는 걸 의미
    - grant all privileges on *.* TO 'ssafy'@'%'; :모든 db에
권한부여
```

```
    - flush privileges;
5. mysql 외부접속 bind address 설정
    - sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf
    - bind-address=127.0.0.1 => bind-address=0.0.0.0
6. ufw 설정
    - sudo ufw status
    - sudo ufw allow 22
    - sudo ufw allow 3306
    - sudo ufw enable
7. aws 인바운드 설정

- https://jay-so.tistory.com/67 참고
- https://yes-admit.tistory.com/127 참고
```

MongoDB 설치

```
1. MongoDB 설치
https://www.mongodb.com/ko-kr/docs/manual/tutorial/install-
mongodb-on-ubuntu/#install-mongodb-community-edition
2. MongoDB 계정 생성 및 비밀번호 설정
    - mongosh
    - use admin
    - db.createUser({ user: '이름', pwd: '비밀번호', roles:
['root'] })
3. 외부 접속 허용
    - sudo vi /etc/mongod.conf
    - bindIp: 127.0.0.1 => 0.0.0.0
    - aws에서 27017 인바운드 설정하기
    - sudo ufw allow 27017
```

Docker 설치

```
1. Docker 설치

https://velog.io/@osk3856/Docker-Ubuntu-22.04-Docker-Instal
lation

1. Docker-compose 설치

https://soyoung-new-challenge.tistory.com/73
```

포트

React	`3000:3000`
SpringBoot Apl Server	`8080:8080`
Jenkins	`8090:8080`
Redis	`6379:6379`
MySQL	`3306:3306`

Jenkins 설정 및 파이프라인

> 💡 **Jenkins 도커 설정파일 및 CI/CD 파이프라인 스크립트 입니다**

Jenkins 설정

```
[Linux](https://www.jenkins.io/doc/book/installing/linux/)

1. 자바 설치
    - sudo apt update
    - java -version
        - openjdk version 17.0.12
2. 젠킨스 설치
```

```bash
    sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
      https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
    echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
      https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
      /etc/apt/sources.list.d/jenkins.list > /dev/null
    sudo apt-get update
    sudo apt-get install jenkins
```

3. 설치 확인
    - sudo service jenkins status
4. Jenkins port 변경(default(8080) → 9090)
    - sudo vi /etc/default/jenkins ⇒ jenkins의 환경변수설정파일
        - HTTP_PORT=9090 로 변경
    - sudo vi /etc/init.d/jenkins ⇒ jenkins의 시작, 중지, 재시작 등을 제어하는 스크립트 파일
        - 변경 전: check_tcp_port “http” “${HTTP_PORT}” “8080” ${HTTP_HOST}“ “0.0.0.0” || return 2
        - 변경 후: check_tcp_port “http” “${HTTP_PORT}” “9090” ${HTTP_HOST}“ “0.0.0.0” || return 2
    - sudo vi /lib/systemd/system/jenkins.service ⇒ Systemd 기반의 jenkins 서비스 관리 파일. Systemd에서 젠킨스를 관리할 수 있도록 설정을 정의한 파일
        - Environment=”JENKINS_PORT=9090”
    - sudo systemctl daemon-reload ⇒ 수정된 설정파일을 systemd가 로드하도록 한다.
    - sudo systemctl restart jenkins ⇒ 젠킨스 재시작
    - sudo lsof -i -P -n | grep LISTEN ⇒ 젠킨스가 9090으로 뜨는지 확인
5. ufw 포트 열기
    - sudo ufw allow 9090

1. 초기 비밀번호 확인 후 입력해서 접속

```
    - sudo cat /var/lib/jenkins/secrets/initialAdminPasswor
d
2. Install suggested plugins를 선택해 권장 플러그인 설치
3. 관리자 계정 생성(Create First Admin User)
4. 젠킨스 접속 url 설정(Instance Configuration)

**607의 유물**

[Jenkins 설치 및 세팅](https://www.notion.so/Jenkins-12dfb59873b08166a792eb9435c2f6b9?pvs=21)
```

Jenkins 파이프라인

cf) ./gradlew clean build ⇒ 기존의 빌드 산출물들을 지우고 다시 빌드한다.

cf) build가 완료되면, build/libs 디렉토리 아래에 .jar파일이 생성된다.

1. /home/ubuntu/.env 파일 생성
    - sudo vi .env
    - sudo chown jenkins:jenkins /home/ubuntu/.env
    - sudo chmod 644 /home/ubuntu/.env
2. 우분투 유저그룹에 Jenkins 유저를 추가 ⇒ 해당 단계를 스킵하면 .env파일의 권한을 777로 줘도 권한 에러가 발생할 수 있음
    - sudo usermod -aG ubuntu jenkins
3. 도커 유저그룹에 Jenkins 유저를 추가
    - sudo usermod -aG docker jenkins

Backend 배포 구성

```
pipeline{
    agent any
```

```
    parameters {
        string(name: 'AWS_ACCESS_KEY_ID', defaultValue:
'~', description: 'aws access key')
        string(name: 'AWS_SECRET_ACCESS_KEY', defaultValue:
'~', description: 'aws secret key')
        string(name: 'AWS_S3_BUCKET_NAME', defaultValue: 'j
dqr-aws-bucket', description: 'aws bucket name')
        string(name: 'AWS_REGION_STATIC', defaultValue: 'us
-east-1', description: 'aws region')
    }

    stages{

        stage('shutdown old process') {
            steps {
                sh """
                    pwd

                    #!/bin/bash
                    ls -al

                    # docker 내리기
                    echo "Stop docker container..."
                    docker compose down
                """
            }
        }

        stage('gitlab clone (BE)') {
            steps {
                script {
                    git branch: 'be-develop', credentialsI
d: 'a608',
                    url: 'https://lab.ssafy.com/s11-final/S
11P31A608.git'
                }
            }
        }
```

```
        stage('환경변수 값 주입') {
            steps {
                dir("./backend") {
                    sh """
                        echo 'application.yml 파일 위치로 이
동'
                        cd ./src/main/resources

                        echo 'application.yml 환경변수 주입'
                        sed -i 's#\${AWS_ACCESS_KEY_ID}#$AW
S_ACCESS_KEY_ID#' application.yml
                        sed -i 's#\${AWS_SECRET_ACCESS_KEY}
#$AWS_SECRET_ACCESS_KEY#' application.yml
                        sed -i 's#\${AWS_S3_BUCKET_NAME}#$A
WS_S3_BUCKET_NAME#' application.yml
                        sed -i 's#\${AWS_REGION_STATIC}#$AW
S_REGION_STATIC#' application.yml
                    """
                }

            }
        }

        stage('build (BE)') {
            steps {
                dir("./backend") {
                    sh """
                        chmod +x ./gradlew
                        ./gradlew clean build
                    """
                }
            }
        }

        stage('restart docker') {
            steps {
                sh """
```

```
                    # 다시 docker-compose 실행
                    echo "Starting Docker Compose..."
                    docker compose up -d

                    echo "Docker Compose started successful
ly."
                """
            }
        }
    }

    post {
        always {
            script {
                def message
                if (currentBuild.result == 'SUCCESS') {
                    message = "됐다!! 됐어!! : ${env.JOB_NAM
E} #${env.BUILD_NUMBER} \n(<${env.BUILD_URL}|Details>)"
                    mattermostSend (color: 'good',
                        message: message,
                        endpoint: 'https://meeting.ssafy.co
m/hooks/grkzh4nxyir6drb4oau8hn4awc',
                        channel: 'a608_jenkins_build'
                    )
                } else if (currentBuild.result == 'FAILUR
E') {
                    message = "X됐다!! X됐어!! : ${env.JOB_NA
ME} #${env.BUILD_NUMBER} \n(<${env.BUILD_URL}|Details>)\n"
                        mattermostSend (color: 'danger',
                        message: message,
                        endpoint: 'https://meeting.ssafy.co
m/hooks/grkzh4nxyir6drb4oau8hn4awc',
                        channel: 'a608_jenkins_build'
                    )
                } else {
                    message = "됐나?? 됐어?? : ${env.JOB_NAM
E} #${env.BUILD_NUMBER} \n(<${env.BUILD_URL}|Details>)"
                    mattermostSend (
```

```
                        message: message,
                        endpoint: 'https://meeting.ssafy.com/hooks/grkzh4nxyir6drb4oau8hn4awc',
                        channel: 'a608_jenkins_build'
                    )
                }
            }
        }
    }
}
```

FrontEnd 배포 구성

```
pipeline{
    agent any

    tools {
        nodejs 'nodejs-20.15.0'
    }

    stages{

        stage('delete previous build') {
            steps {
                sh """
                    #!/bin/bash
                    whoami
                    rm -rf /usr/share/nginx/html/build
                """
            }
        }


        stage('github clone (FE)') {
```

```
            steps {
                script {
                    git branch: 'fe-develop', credentialsId: 'a608',
                    url: 'https://lab.ssafy.com/s11-final/S11P31A608.git'
                }
            }
        }

        stage('remove previous repo (FE)') {
            steps {
                dir("./frontend/jdqr-order") {
                    sh """
                        yarn cache clean
                        rm -rf node_modules yarn.lock
                        node --version  # 올바른 Node.js 버전이 출력되는지 확인
                        yarn install
                        CI=false yarn build
                    """
                }
            }
        }

        stage('build (FE)') {
            steps {
                dir("./frontend/jdqr-order") {
                    sh """
                        node --version  # 올바른 Node.js 버전이 출력되는지 확인
                        yarn install
                        CI=false yarn build
                    """
                }
            }
        }
```

```
        stage('move build files to nginx static') {
            steps {
                sh """
                    pwd
                    ls -al
                    mv frontend/jdqr-order/build /usr/shar
e/nginx/html/
                    echo "Deploy React App success."
                """
            }
        }
    }

  post {
        always {
            script {
                def message
                if (currentBuild.result == 'SUCCESS') {
                    message = "무슨 마술을 부린거야? : ${env.JO
B_NAME} #${env.BUILD_NUMBER} \n(<${env.BUILD_URL}|Details
>)"
                    mattermostSend (color: 'good',
                        message: message,
                        endpoint: 'https://meeting.ssafy.co
m/hooks/grkzh4nxyir6drb4oau8hn4awc',
                        channel: 'a608_jenkins_build'
                    )
                } else if (currentBuild.result == 'FAILUR
E') {
                    message = "쓰읍...사기를 쳐놨네: ${env.JOB
_NAME} #${env.BUILD_NUMBER} \n(<${env.BUILD_URL}|Details>)
\n"
                        mattermostSend (color: 'danger',
                        message: message,
                        endpoint: 'https://meeting.ssafy.co
m/hooks/grkzh4nxyir6drb4oau8hn4awc',
                        channel: 'a608_jenkins_build'
```

```
                    )
                } else {
                    message = "이건 뭐지 : ${env.JOB_NAME} #
${env.BUILD_NUMBER} \n(<${env.BUILD_URL}|Details>)"
                    mattermostSend (
                        message: message,
                        endpoint: 'https://meeting.ssafy.co
m/hooks/grkzh4nxyir6drb4oau8hn4awc',
                        channel: 'a608_jenkins_build'
                    )
                }
            }
        }
    }
}
```

환경변수 및 설정파일

💡 **개발 과정에서 사용한 설정파일 및 .env 파일 내용입니다**

docker-compose 및 Dockerfile

Backend

```
# docker-compose.yml

services:
  app:
    image: openjdk:17-jdk-slim
    container_name: springboot-app
```

```
    volumes:
      - ./backend/build/libs/backend-0.0.1-SNAPSHOT.jar:/app.
      - /home/ubuntu/pinpoint-agent-2.5.1:/pinpoint
    ports:
      - "8080:8080"
    command: ["java", "-jar","-javaagent:./pinpoint/pinpoint-
    environment:
      - TZ=Asia/Seoul
    depends_on:
      - redis
    restart: always

  redis:
    image: "redis:alpine"
    container_name: redis
    ports:
      - "6379:6379"
    volumes:
      - ./redis-data:/data
    restart: always
```

API Server application.yml

---

application.yml

```
spring:
  application:
    name: JDQR

  datasource:
    url: jdbc:mysql://13.124.221.144:3306/product?useUnicode=
    username: ssafy
    password: zeroticket608
```

```
    driver-class-name: com.mysql.cj.jdbc.Driver

  jpa:
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        format_sql: true
        use_sql_comments: true
        dialect: org.hibernate.dialect.MySQLDialect
        jdbc:
          time_zone: Asia/Seoul


  devtools:
    restart:
      additional-paths: .

  data:
    redis:
      host: redis
      port: 6379


    mongodb:
      uri: mongodb://ssafy:zeroticket608@13.124.221.144:27017
      database: product

  logging.level:
    org.hibernate.SQL: debug
    org.mongodb.driver: debug

  jwt:
    secret-key: Z29nby10bS1zZXJ2ZXItZGxyamVvYW9yb3JodG9kZ290c

external:
  token:
    toss: test_sk_Z61JOxRQVEGdzZDnPgkQ8W0X9bAq
```

```
  url:
    toss: https://api.tosspayments.com

cloud:
  aws:
    credentials:
      accessKey:
      secretKey:
    s3:
      bucketName: jdqr-aws-bucket
    region:
      static: us-east-1
    stack:
      auto: false
```

모니터링 설정파일

💡 **서버 모니터링을 위해 설정한 파일 입니다.**

Portainer 설정

```
1. Potainer 데이터를 저장할 디렉토리 생성
    - mkdir -p /data/portainer
2. Potainer컨테이너의 데이터를 저장할 Docker 볼륨 생성
    - docker volume create portainer_data
3. Potainer 컨테이너 실행
- docker run -d -p 55555:9000 --name=portainer --restart=al
ways -v /var/run/docker.sock:/var/run/docker.sock -v portai
ner_data:/data portainer/portainer-ce
    - -d ⇒ 컨테이너를 백그라운드에서 실행
    - -p 55555:9000 ⇒ 호스트의 55555번 포트를 컨테이너의 9000번
```

포트에 매핑. 웹브라우저에서 http://호스트ip:55555 시, 포테이너 접근가능
    - --name = portainer ⇒ 컨테이너 이름을 portainer로 지정
    - --restart=always ⇒ Docker 데몬이 재시작되면 Portainer 컨테이너가 자동 재시작되도록 설정
    - -v /var/run/docker.sock:/var/run/docker.sock ⇒ 호스트의 Docker 소켓파일을 portainer 컨테이너 내부의 동일경로로 마운트. 이를 통해 portainer가 호스트의 docker데몬에 직접 접근하고 제어가능.
    - -v portainer_data:/data ⇒ portainer_data 볼륨을 컨테이너 안의 /data 디렉토리에 마운트. 컨테이너 안에서 /data 디렉토리에 저장되는 내용은 portainer_data볼륨에도 저장되어 컨테이너가 죽어도 소실x
    - portainer/portainer-ce ⇒ Portainer의 커뮤니티 에디션 이미지를 사용해 컨테이너를 생성
1. 외부에서 Potainer에 접속할 수 있도록 방화벽 허용
    - sudo ufw allow 55555
2. portainer 컨테이너의 재시작이 필요하면,
    - sudo docker restart portainer

PINPOINT 설정

```
#wegt 명령어를 이용해 압축파일 다운로드
wget https://archive.apache.org/dist/hbase/1.2.7/hbase-1.2.7-bin.tar.gz

#압축해제
tar xzvf hbase-1.2.7-bin.tar.gz

#hbase 설정파일 확인
vi .../hbase-1.2.7/conf/hbase-env.sh
```

```
cd .../hbase-1.2.7/bin

#Hbase 실행
```

```
./start-hbase.sh

#HBase 중지
./stop-hbase.sh

# 스크립트 다운
wget https://raw.githubusercontent.com/pinpoint-apm/pinpoint/master/hbase/scripts/hbase-create.hbase

# 스크립트 실행
hbase/bin/hbase shell ../hbase-create.hbase

#Web
wget https://github.com/pinpoint-apm/pinpoint/releases/download/v2.5.1/pinpoint-web-boot-2.5.1.jar

#Collector
wget https://github.com/pinpoint-apm/pinpoint/releases/download/v2.5.1/pinpoint-collector-boot-2.5.1.jar

# 실행권한 부여하기
chmod +x pinpoint-collector-boot-2.5.1.jar
chmod +x pinpoint-web-boot-2.5.1.jar

#Web 실행
nohup java -jar -Dpinpoint.zookeeper.address=localhost pinpoint-web-boot-2.5.1.jar >/dev/null 2>&1 &

#Collector 실행
nohup java -jar -Dpinpoint.zookeeper.address=localhost pinpoint-collector-boot-2.5.1.jar >/dev/null 2>&1 &
```

여기서부터 스프링 서버에 설치

```
# agent 설치
wget https://github.com/pinpoint-apm/pinpoint/releases/download/v2.5.1/pinpoint-agent-2.5.1.tar.gz
```

```
# 압축 해제
tar xvzf pinpoint-agent-2.5.1.tar.gz

# agnet 설정 파일 수정
vi.../pinpoint-agent-2.5.1/pinpoint-root.config
```

PINPOINT가 적용된 docker-compose.yml

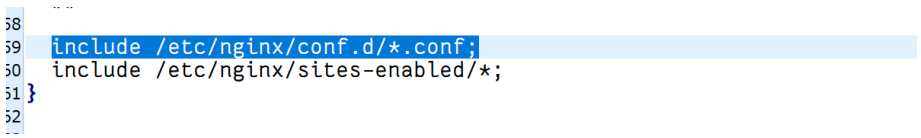
```
# docker-compose.yml

services:
  app:
    image: openjdk:17-jdk-slim
    container_name: springboot-app
    volumes:
      - ./backend/build/libs/backend-0.0.1-SNAPSHOT.jar:/ap
p.jar
      - /home/ubuntu/pinpoint-agent-2.5.1:/pinpoint
    ports:
      - "8080:8080"
    command: ["java", "-jar","-javaagent:./pinpoint/pinpoin
t-bootstrap-2.5.1.jar", "-Dpinpoint.agentId=adventcalendarD
ev","-Dpinpoint.applicationName=adventcalendar","-Dpinpoin
t.config=./pinpoint/pinpoint-root.config","-Dspring.profile
s.active=${SERVER_MODE}","-Duser.timezone=Asia/Seoul", "/ap
p.jar"]
    environment:
      - TZ=Asia/Seoul
    depends_on:
      - redis
    restart: always

  redis:
    image: "redis:alpine"
    container_name: redis
```

```
    ports:
      - "6379:6379"
    volumes:
      - ./redis-data:/data
    restart: always
```

Nginx 설정파일

---

1. DuckDNS로 서버 도메인 생성
2. NginX 설치
   - sudo apt update
   - sudo apt install nginx
3. NginX가 리버스 프록시 역할을 수행하도록 conf파일 작성
   - sudo vi /etc/nginx/conf.d/jdqr.conf
     - /etc/nginx/nginx.conf 파일안에 include /etc/nginx/conf.d/*.conf;가 있어서 conf.d 디렉토리 안에 nginx설정 파일 추가하면 반영됨




```
server {
  listen 8081 ssl http2;
  server_name jdqr608.duckdns.org;

  ssl_certificate /etc/letsencrypt/live/jdqr608-duckdns-or
  ssl_certificate_key /etc/letsencrypt/live/jdqr608-duckdn

  location / {
    root /usr/share/nginx/html/owner;
```

```
    try_files $uri $uri/ /index.html;
    add_header Access-Control-Allow-Origin "https://jdqr60
    add_header Access-Control-Allow-Credentials "true";
  }

}

server {
  listen 80;
  server_name jdqr608.duckdns.org;
  return 301 https://jdqr608.duckdns.org$request_uri;

  location /project/be-release {
    proxy_pass http://localhost:9090/project/be-release;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarde
    proxy_set_header X-Forwarded-Proto $scheme;
  }
}

server {
  listen 443 ssl http2;
  server_name jdqr608.duckdns.org;

  # ssl 인증서 설정하기
  ssl_certificate /etc/letsencrypt/live/jdqr608-duckdns-or
  ssl_certificate_key /etc/letsencrypt/live/jdqr608-duckdn

  location /api/v1 {
    proxy_pass http://localhost:8080/api/v1;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarde
    proxy_set_header X-Forwarded-Proto $scheme;

  }
```

```
    # WebSocket
    location /ws {
        proxy_pass http://localhost:8080/ws;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_set_header Host $host;
        add_header Access-Control-Allow-Origin "https://jdqr
        add_header Access-Control-Allow-Credentials "true";

        proxy_read_timeout 21600000; # 6 * 60 * 60 * 1000
        proxy_send_timeout 21600000; # 6 * 60 * 60 * 1000
    }

    location /v3 {
      proxy_pass http://localhost:8080/v3;
      proxy_set_header Host $http_host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarde
      proxy_set_header X-Forwarded-Proto $scheme;
    }

    location / {
      root /usr/share/nginx/html/order;
      try_files $uri $uri/ /index.html;
      add_header Access-Control-Allow-Origin "https://jdqr60
      add_header Access-Control-Allow-Credentials "true";
    }

  }
```

4. NginX 재시작
    - sudo nginx -t
    - sudo service nginx restart
5. 방화벽 설정 확인하기
    - sudo ufw allow ‘Nginx Full’

6. SSL 인증서 설정
    - sudo apt install certbot python3-certbot-nginx
    - sudo certbot --nginx -d your-domain.com
        - 도메인이 duckdns로 생성한 것이라 도메인 이름에 인증서 경로로 사용할 수 없는 문자가 포함되었다는 에러가 발생 ⇒ --cert-name 옵션으로 다시 수행
    - sudo certbot --nginx -d jdqr608.duckdns.org --cert-name jdqr608-duckdns-org

외부 서비스

KakaoMap API	사용자 지도에 사용

DB 덤프 파일 최신본

Dump20241119.sql