TheRamzee 포팅 매뉴얼

작성일시	@2025년 2월 20일 오전 9:16
강의 유형	OffLine
유형	인프라
복습	☐

📌 Project Skill Stack Version

Skill	Version
gradle	7.6.1
Java	17.0.2
SpringBoot	3.4.1
MySQL	9.2.0
Redis	7.4.2
React	18.3.1
Node.js	18.18.0
NPM	10.9.2
Jenkins	2.496
IntelliJ(IDE)	21.0.5

📌 EC2 포트 번호

Skill	EC2 Port	Container Port
Back	x (Nginx Proxy)	8080
Front	x (Nginx Proxy)	80
MySQL	3306	3306/33060
Redis	6379	6379
Nginx http	80	80
Nginx https	443	443
OpenVidu	3478/5349/8443	3478/5349/8443
Jenkins	8081/50000	8080/50000

📌 외부 프로그램

OpenVidu : openvidu 2.31.0

📌 빌드 방법

ufw 설정

```
sudo su

ufw allow 22
ufw allow 8989
ufw allow 443
ufw allow 22/tcp
ufw allow 80/tcp
ufw allow 443/tcp
```

```
ufw allow 3478/tcp
ufw allow 3478/udp
ufw allow 40000:57000/tcp
ufw allow 40000:57000/udp
ufw allow 57001:65535/tcp
ufw allow 57001:65535/udp
ufw allow 8443/tcp
ufw allow 8082/tcp
ufw allow 5443/tcp
ufw allow 5442/tcp
ufw allow 8081/tcp
ufw allow 3000/tcp
ufw allow 8080/tcp
ufw allow 3306/tcp
ufw allow 6379/tcp

ufw enable
```

서버 시간 설정

```
sudo timedatectl set-timezone Asia/Seoul
```

Docker 설치

```
sudo apt-get update
sudo apt-get install -y ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ub
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
sudo docker --version
```

- Docker 23.0 이후부터 docker compose 명령어가 기본 플러그인으로 포함되어 별도 설치가 필요 없습니다.

Jenkins 설치

```
docker pull jenkins/jenkins:lts

docker run -d \
  --name jenkins \
  -p 8081:8080 \
  -p 50000:50000 \
  -v jenkins_home:/var/jenkins_home \
  -v /var/run/docker.sock:/var/run/docker.sock \
  jenkins/jenkins:lts
```

```
docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword
```

- 브라우저에 표시된 초기 설정 페이지에 복사한 암호를 입력한 후, 제안된 플러그인을 설치하고 관리자를 생성하세요.

Jenkins GitLab Webhook

1. 필수 플러그인 설치

Jenkins 웹 UI에서 다음 경로로 이동합니다:

Manage Jenkins → Manage Plugins

설치할 플러그인은 다음과 같습니다:

- **GitLab -** GitLab 웹훅(Webhook) 트리거 지원.
- **Pipeline -** Jenkinsfile 실행에 필요.
- **Docker Pipeline -** Docker와의 연동에 필요.
- **SSH Pipeline Steps -** EC2 배포 시 사용.
- **Credentials Binding -** Docker Hub 및 SSH 키 인증에 필요.

2. GitLab Credentials 설정

2.1 GitLab에서 Access Token 발급

1. GitLab 웹사이트에 접속하여 **User Settings → Access tokens** 메뉴로 이동합니다.
2. **Add new token**을 클릭하고, 다음 정보를 입력합니다:
   - **Token name**: 원하는 이름 입력.
   - **Select scopes**: `api`, `read_repository`, `write_repository` 선택.
3. **Create personal access token** 버튼을 클릭한 후, 생성된 토큰을 복사합니다.

2.2 Jenkins에 GitLab Access Token 등록

1. Jenkins 웹 UI에서 **Manage Jenkins → Manage Credentials**로 이동합니다.
2. 전역(Global) 도메인에서 **Add Credentials**를 클릭합니다.
3. 아래와 같이 입력합니다:
   - **Kind**: `Username with password`
   - **Scope**: Global (Jenkins, nodes, items, all child items, etc)
   - **Username**: GitLab 아이디
   - **Password**: 앞서 복사한 GitLab Access Token
   - **ID**: (예) `gitlab-token`
4. **Create**를 클릭하여 등록합니다.

3. Docker Hub Credentials 설정

3.1 Docker Hub Access Token 생성

1. Docker Hub에 로그인합니다.
2. **Account Settings → Security → Personal access tokens** 메뉴에서 **Generate new token**을 선택합니다.
3. 토큰 정보를 입력합니다:
   - **Access token description**: 토큰 이름 입력.
   - **Access permissions**: `Read`, `Write`, `Delete` 선택 (Optional).
4. **Generate** 버튼을 클릭한 후, 생성된 토큰을 복사합니다.

3.2 Jenkins에 Docker Hub Credentials 등록

1. Jenkins 웹 UI에서 **Manage Jenkins → Manage Credentials**로 이동합니다.

2. 전역(Global) 도메인에서 **Add Credentials**를 클릭합니다.
3. 아래와 같이 입력합니다:
    - **Kind**: `Username with password`
    - **Scope**: Global (Jenkins, nodes, items, all child items, etc)
    - **Username**: Docker Hub 아이디
    - **Password**: 앞서 복사한 Docker Hub Access Token
    - **ID**: (예) `dockerhub-token`
4. **Create**를 클릭하여 등록합니다.

4. SSH 키 Credentials 설정

4.1 EC2에서 SSH 키 생성

EC2 인스턴스에서 아래 명령어를 실행하여 SSH 키를 생성합니다:

```
ssh-keygen -t rsa -b 4096 -C "jenkins"
```

- 기본 저장 경로: `/home/ubuntu/.ssh/id_rsa`
- 생성된 `id_rsa`는 **Private Key** (Jenkins에서 사용)
- `id_rsa.pub`는 **Public Key** (EC2 서버에 등록)

4.2 EC2 서버에 Public Key 등록

EC2 인스턴스에서 다음 명령어를 실행합니다:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

chmod 600 ~/.ssh/authorized_keys
```

- 첫 번째 명령은 현재 사용자의 공개 키를 `authorized_keys` 파일 끝에 추가합니다.
- 두 번째 명령은 파일 권한을 소유자에게 읽기 및 쓰기 권한만 부여하도록 설정합니다.

4.3 Jenkins에 Private Key 등록

1. Jenkins 웹 UI에서 **Manage Jenkins → Manage Credentials**로 이동합니다.
2. 전역(Global) 도메인에서 **Add Credentials**를 클릭합니다.
3. 아래와 같이 입력합니다:
    - **Kind**: `SSH Username with private key`
    - **Scope**: Global (Jenkins, nodes, items, all child items, etc)
    - **Username**: `ubuntu`
    - **Private Key**: "Enter directly" 선택 후, 생성된 `id_rsa` 파일의 내용을 복사하여 붙여넣기
    - **ID**: (예) `ec2-ssh-key`
4. **Create**를 클릭하여 등록합니다.

5. .env 파일을 Secret File로 등록

Jenkins Pipeline에서 환경 변수 파일을 안전하게 관리하기 위해, `.env` 파일을 Secret File Credential로 등록합니다.

```
MYSQL_HOST=theramzee-mysql-1
MYSQL_ROOT_PASSWORD=dleogus1
MYSQL_DATABASE=gradation_db
MYSQL_PORT=3306
MYSQL_USERNAME=root
MYSQL_PASSWORD=dleogus1
```

```
REDIS_PORT=6379
REDIS_HOST=theramzee-redis-1

MAIL_USERNAME=gd122572@gmail.com
MAIL_PASSWORD=fzbujhthgsrmwhwh

JWT_SECRET=theramzeejwtsecretkey
JWT_ACCESS_TOKEN_EXPIRATION=360000000
JWT_REFRESH_TOKEN_EXPIRATION=604800000

SERVER_PORT=8080

OPENVIDU_URL=https://ramzee.online:8443/
OPENVIDU_SECRET=dleogus1

BACKEND_IMAGE=murhyun2/theramzee-backend:latest
FRONTEND_IMAGE=murhyun2/theramzee-frontend:latest
NGINX_IMAGE=murhyun2/theramzee-nginx:latest
```

5.1 Jenkins에서 Secret File 등록

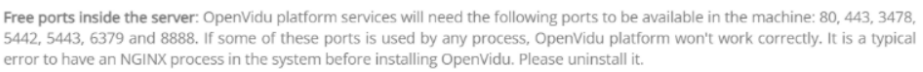
1. Jenkins 웹 UI에서 **Manage Jenkins → Manage Credentials**로 이동합니다.
2. 전역(Global) 도메인에서 **Add Credentials**를 클릭합니다.
3. 아래와 같이 설정합니다:
   - **Kind**: `Secret file`
   - **Scope**: Global (Jenkins, nodes, items, all child items, etc)
   - **File**: 등록할 `.env` 파일 선택
   - **ID**: (예) `env-file-content`
4. **Create**를 클릭하여 등록합니다.

5.2 Jenkins Pipeline에서 .env 파일 사용

Jenkins Pipeline 내에서 등록한 Secret File을 사용하려면, `withCredentials` 스텝을 이용하여 파일을 로드합니다. 예시는 다음과 같습니다:

```
pipeline {
    agent any
    stages {
        stage('Load Environment') {
            steps {
                withCredentials([file(credentialsId: 'env-file-content', variable: 'ENV_FILE')]) {
                    sh 'cat $ENV_FILE'
                    // .env 파일의 내용을 이용한 추가 작업 수행
                }
            }
        }
    }
}
```

openvidu 설치



⚠️openvidu 자체적으로 nginx 등을 사용하기 때문에, openvidu를 먼저 설치하지 않을 시 포트 충돌이 발생하여 원활한 실행이 불가능 할 수 있습니다.

- 오픈비두를 배포하기 root 권한을 얻어야 함

```
sudo su
```

- 오픈비두를 설치하기 위해 권장되는 경로인 `/opt`로 이동

```
cd /opt
```

- 오픈비두 설치

```
curl <https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh> | bash
```

- 설치 후 오픈비두가 설치된 경로로 이동

```
$ cd openvidu
```

- 도메인 또는 퍼블릭IP와 오픈비두와 통신을 위한 환경설정

```
$ nano .env

# OpenVidu configuration
# ----------------------
# 도메인 또는 퍼블릭IP 주소
DOMAIN_OR_PUBLIC_IP=ramzee.online

# 오픈비두 서버와 통신을 위한 시크릿
OPENVIDU_SECRET=dleogus1

# Certificate type
CERTIFICATE_TYPE=letsencrypt

# 인증서 타입이 letsencrypt일 경우 이메일 설정
LETSENCRYPT_EMAIL=user@example.com

# HTTP port
HTTP_PORT=8082

# HTTPS port(해당 포트를 통해 오픈비두 서버와 연결)
HTTPS_PORT=8443
```

- 설정 후 오픈비두 서버 실행(`ctrl + c`를 누르면 백그라운드로 실행됨)

```
$ ./openvidu start

[+] Running 5/5
 ✔ Container openvidu-nginx-1          Started
 ✔ Container openvidu-app-1            Started
 ✔ Container openvidu-coturn-1         Started
 ✔ Container openvidu-kms-1            Started
 ✔ Container openvidu-openvidu-server-1  Started

openvidu-server-1 | ----------------------------------------------------
openvidu-server-1 |
openvidu-server-1 |    OpenVidu is ready!
openvidu-server-1 |    ---------------------------
openvidu-server-1 |
openvidu-server-1 |    * OpenVidu Server URL: https://ramzee.online:8443/
```

```
openvidu-server-1 |
openvidu-server-1 |    * OpenVidu Dashboard: https://ramzee.online:8443/dashboard
openvidu-server-1 |
openvidu-server-1 | ----------------------------------------------------
```

Git Clone

```
git clone https://lab.ssafy.com/s12-webmobile1-sub1/S12P11B204.git
```

- docker compose up -d 로 실행 가능

.env

```
MYSQL_HOST=theramzee-mysql-1
MYSQL_ROOT_PASSWORD=dleogus1
MYSQL_DATABASE=gradation_db
MYSQL_PORT=3306
MYSQL_USERNAME=root
MYSQL_PASSWORD=dleogus1

REDIS_PORT=6379
REDIS_HOST=theramzee-redis-1

MAIL_USERNAME=gd122572@gmail.com
MAIL_PASSWORD=fzbujhthgsrmwhwh

JWT_SECRET=theramzeejwtsecretkey
JWT_ACCESS_TOKEN_EXPIRATION=360000000
JWT_REFRESH_TOKEN_EXPIRATION=604800000

SERVER_PORT=8080

OPENVIDU_URL=https://ramzee.online:8443/
OPENVIDU_SECRET=dleogus1

BACKEND_IMAGE=murhyun2/theramzee-backend:latest
FRONTEND_IMAGE=murhyun2/theramzee-frontend:latest
NGINX_IMAGE=murhyun2/theramzee-nginx:latest
```

1. backend gradle 의존성

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.4.1'
    id 'io.spring.dependency-management' version '1.1.7'
}

group = 'com.gradation'
version = '0.0.1-SNAPSHOT'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
}
```

```
configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}


dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-websocket'
    compileOnly 'org.projectlombok:lombok'
    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'

    //sweager
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.2.0'
    //querydsl
    implementation 'com.querydsl:querydsl-jpa:5.0.0:jakarta'
    annotationProcessor 'com.querydsl:querydsl-apt:5.0.0:jakarta'
    annotationProcessor 'jakarta.annotation:jakarta.annotation-api'
    annotationProcessor 'jakarta.persistence:jakarta.persistence-api'
    //redis
    implementation 'org.springframework.boot:spring-boot-starter-data-redis'
    //jwt
    implementation 'javax.xml.bind:jaxb-api:2.3.1'
    implementation 'org.springframework.boot:spring-boot-starter-security'
    testImplementation 'org.springframework.security:spring-security-test'
    implementation 'org.springframework.boot:spring-boot-starter-mail'
    implementation 'io.jsonwebtoken:jjwt:0.9.1'
    //openvidu
    implementation "io.openvidu:openvidu-java-client:2.31.0"
    //validator
    implementation 'org.hibernate.validator:hibernate-validator:8.0.1.Final'
}

tasks.named('test') {
    useJUnitPlatform()
}
```

2. backend - application.yml

```
spring:
  datasource:
    url: jdbc:mysql://${MYSQL_HOST}:${MYSQL_PORT}/${MYSQL_DATABASE}
    username: ${MYSQL_USERNAME}
    password: ${MYSQL_PASSWORD}
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    hibernate:
      ddl-auto: none
```

```
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQLDialect
  data:
    redis:
      host: ${REDIS_HOST}
      port: ${REDIS_PORT}
  mail:
    host: smtp.gmail.com
    port: 587
    username: ${MAIL_USERNAME}
    password: ${MAIL_PASSWORD}
    properties:
      mail:
        smtp:
          auth: true
          starttls:
            enable: true

jwt:
  secret: ${JWT_SECRET}
  access-token:
    expiration: ${JWT_ACCESS_TOKEN_EXPIRATION}
  refresh-token:
    expiration: ${JWT_REFRESH_TOKEN_EXPIRATION}

server:
  port: ${SERVER_PORT}

springdoc:
  api-docs:
    path: /api-docs
  swagger-ui:
    path: /swagger-ui.html

openvidu:
  url: ${OPENVIDU_URL}
  secret: ${OPENVIDU_SECRET}
```

3. mysql/init/schema.sql

```
-- 데이터베이스 생성 및 선택
CREATE DATABASE IF NOT EXISTS gradation_db;
USE gradation_db;

-- User 테이블
CREATE TABLE IF NOT EXISTS user (
    id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    username VARCHAR(20) NOT NULL UNIQUE,
    name VARCHAR(255) NOT NULL,
    nickname VARCHAR(10) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(50) NOT NULL,
    room_id INT UNSIGNED NULL,
    login_root VARCHAR(10) NOT NULL,
    user_status BOOLEAN NOT NULL,
    PRIMARY KEY (id)
);
```

```
-- Room 테이블
CREATE TABLE IF NOT EXISTS room (
    id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    title VARCHAR(100) NULL,
    password INT NULL,
    host_id INT UNSIGNED NOT NULL,
    game_status BOOLEAN NOT NULL,
    PRIMARY KEY (id),
    CONSTRAINT fk_user_name FOREIGN KEY (host_id) REFERENCES user(id) ON DELETE CASCADE
);

-- User 테이블 외래 키 추가
ALTER TABLE user
ADD CONSTRAINT fk_user_room FOREIGN KEY (room_id) REFERENCES room(id) ON DELETE SET NULL;

-- Friends 참가자 테이블
CREATE TABLE IF NOT EXISTS friends (
    id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    user_id INT UNSIGNED NOT NULL,
    friends_id INT UNSIGNED NOT NULL,
    status ENUM('REQUESTED', 'ACCEPTED', 'REJECTED') NOT NULL DEFAULT 'REQUESTED',
    PRIMARY KEY (id),
    CONSTRAINT fk_friends_user FOREIGN KEY (user_id) REFERENCES user(id) ON DELETE CASCADE,
    CONSTRAINT fk_friends_friend FOREIGN KEY (friends_id) REFERENCES user(id) ON DELETE CASCADE
);
```

4. nginx/nginx.conf

```
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

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

    limit_req_zone $binary_remote_addr zone=bot_limiter:10m rate=10r/s;

    upstream backend {
        server theramzee-backend-1:8080;
    }

    upstream frontend {
        server theramzee-frontend-1:80;
    }

    # 로깅 설정 추가
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';
    access_log /var/log/nginx/access.log main;
    error_log /var/log/nginx/error.log warn;

    server {
        listen 80;
```

```
    server_name ramzee.online i12b204.p.ssafy.io;
    return 301 https://ramzee.online$request_uri;
}

server {
    listen 443 ssl;
    server_name i12b204.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/ramzee.online/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/ramzee.online/privkey.pem;

    return 301 https://ramzee.online$request_uri;
}

server {
    listen 443 ssl;
    server_name ramzee.online;

    # SSL 설정
    ssl_certificate /etc/letsencrypt/live/ramzee.online/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/ramzee.online/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers 'TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:ECDHE-RSA-AES256-GCM-SHA384
    ssl_prefer_server_ciphers on;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;
    ssl_stapling on;
    ssl_stapling_verify on;
    resolver 8.8.8.8 8.8.4.4 valid=300s;
    resolver_timeout 5s;

    # 기본 보안 헤더
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header Referrer-Policy "no-referrer-when-downgrade" always;

    # 악성 봇 차단
    if ($http_user_agent ~* "(curl|wget|python|scrapy|scan|WordPress|wordpress|wp|WordPressScanner|java|bot|craw
        return 444;  # 연결 종료로 변경
    }

    # XDebug 세션 파라미터 차단
    if ($query_string ~* "XDEBUG_SESSION_START=phpstorm") {
        return 444;
    }

    # 숨김 파일 (.env 등) 접근 차단
    location ~ /\.(?!well-known) {
        deny all;
        access_log off;
        log_not_found off;
    }

    # 압축 설정
    gzip on;
    gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss tex

    # WordPress 공격 패턴 차단
```

```
    location ~* (wp-admin|wp-login|setup-config\.php) {
        deny all;
        return 444;
    }

    # PHP 파일 접근 차단
    location ~* \.php$ {
        deny all;
        return 444;
    }

    # Swagger UI 경로를 백엔드로 프록시 (prefix match)
    location /swagger-ui/ {
        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_cache_bypass $http_upgrade;
        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;
    }

    # 추가: api-docs 경로를 백엔드로 프록시
    location /api-docs/ {
        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_cache_bypass $http_upgrade;
        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;
    }

    # 프론트엔드 프록시
    location / {
        root /usr/share/nginx/html;  # 프론트엔드 빌드 파일 경로
        index index.html;
        try_files $uri $uri/ /index.html;

        proxy_pass http://frontend;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        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;

        # 디버깅을 위한 추가 헤더
        # add_header X-Debug-Message "Proxying to frontend" always;
    }

    # 정적 파일 캐싱 설정 (선택 사항)
    location /static/ {
```

```
        alias /usr/share/nginx/html/static/;
        expires 1y;
        add_header Cache-Control "public";
    }

    location /api/v1/email/ {
        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_cache_bypass $http_upgrade;
        add_header 'Access-Control-Allow-Origin' '*';
    }

    # 백엔드 API 프록시
    location /api/ {

        limit_req zone=bot_limiter burst=20 nodelay;

        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_cache_bypass $http_upgrade;

        # CORS 헤더 추가
        add_header 'Access-Control-Allow-Origin' '*' always;
        add_header 'Access-Control-Allow-Methods' 'GET, POST, PUT, DELETE, OPTIONS' always;
        add_header 'Access-Control-Allow-Headers' 'DNT, User-Agent, X-Requested-With, If-Modified-Since, Cache-Co
        add_header 'Access-Control-Allow-Credentials' 'true' always;

        # OPTIONS 처리
        if ($request_method = 'OPTIONS') {
            add_header 'Access-Control-Max-Age' 1728000;
            add_header 'Content-Type' 'text/plain; charset=utf-8';
            add_header 'Content-Length' 0;
            return 204;
        }
    }

    # OpenVidu 프록시
    location /openvidu/ {
        proxy_pass https://ramzee.online:8443/;
        proxy_ssl_verify off;
        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/ {
        proxy_pass http://backend/ws/;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_set_header Host $host;
```

```
            # 추가 보안 헤더
            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;

            # 연결 유지 시간 설정
            proxy_read_timeout 3600s;
            proxy_send_timeout 3600s;
            proxy_buffering off;
        }
    }
}
```

5. Dockerfile

- **backend/Dockerfile**

```
# Build stage
FROM gradle:7.6.1-jdk17 AS builder
WORKDIR /app
COPY . .
RUN chmod +x ./gradlew
RUN ./gradlew clean build -x test

# Run stage
FROM openjdk:17
WORKDIR /app
COPY --from=builder /app/build/libs/backend-0.0.1-SNAPSHOT.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

- **frontend/Dockerfile**

```
FROM node:18 AS builder
WORKDIR /app
COPY package*.json ./
RUN npm install --save-dev @babel/plugin-proposal-private-property-in-object
RUN npm install
COPY . .
RUN npm run build

FROM nginx:alpine
COPY --from=builder /app/build /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

- nginx/Dockerfile

```
FROM nginx:alpine
COPY --from=murhyun2/theramzee-frontend:latest /usr/share/nginx/html /usr/share/nginx/html
COPY nginx.conf /etc/nginx/nginx.conf
#COPY nginx.conf /etc/nginx/conf.d/nginx.conf
RUN mkdir -p /etc/letsencrypt
EXPOSE 80 443
CMD ["nginx", "-g", "daemon off;"]
```

6. docker-compose.yml

```
services:
  mysql:
    image: mysql:latest
    container_name: theramzee-mysql-1
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
    ports:
      - "${MYSQL_PORT}:3306"
    volumes:
      - theramzee-mysql-data:/var/lib/mysql
      - ./mysql/init:/docker-entrypoint-initdb.d
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
    restart: unless-stopped
    networks:
      - theramzee-network

  redis:
    command: redis-server --bind 0.0.0.0
    image: redis:latest
    container_name: theramzee-redis-1
    ports:
      - "${REDIS_PORT}:6379"
    volumes:
      - theramzee-redis-data:/data
    restart: unless-stopped
    networks:
      - theramzee-network

  backend:
    build: ./backend
    image: ${BACKEND_IMAGE}
    container_name: theramzee-backend-1
    env_file:
      - .env
    depends_on:
      - mysql
      - redis
    restart: unless-stopped
    networks:
      - theramzee-network

  frontend:
    build: ./frontend
    image: ${FRONTEND_IMAGE}
    container_name: theramzee-frontend-1
    depends_on:
      - backend
    restart: unless-stopped
    networks:
      - theramzee-network

  nginx:
    build: ./nginx
    image: ${NGINX_IMAGE}
```

```
    container_name: theramzee-nginx-1
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf
      - /etc/letsencrypt:/etc/letsencrypt
      - ./certbot/www:/var/www/certbot
    depends_on:
      - frontend
      - backend
    restart: unless-stopped
    networks:
      - theramzee-network

volumes:
  theramzee-mysql-data:
  theramzee-redis-data:

networks:
  theramzee-network:
    driver: bridge
```

참고 : Jenkins CI/CD pipeline

```
pipeline {
    agent any

    options {
        disableConcurrentBuilds()
    }
    environment {
        DOCKER_IMAGE_PREFIX = "murhyun2/theramzee"
        EC2_HOST = "i12b204.p.ssafy.io"
        COMPOSE_PROJECT_NAME = "theramzee"
        EC2_SSH_CREDENTIALS_ID = "ec2-ssh-key"
        GIT_CREDENTIALS_ID = "gitlab-credentials"
        GIT_REPOSITORY_URL = "https://lab.ssafy.com/s12-webmobile1-sub1/S12P11B204"
        PROJECT_DIRECTORY = "jenkins"
        EC2_USER = "ubuntu"
        DOCKER_HUB_CREDENTIALS_ID = "docker-hub-credentials"
    }
    stages {
        stage('Checkout') {
            steps {
                git branch: "develop", credentialsId: "${GIT_CREDENTIALS_ID}", url: "${GIT_REPOSITORY_URL}"
            }
        }
        stage('Prepare Environment') {
            steps {
                withCredentials([file(credentialsId: 'env-file-content', variable: 'ENV_FILE_PATH')]) {
                    script {
                        def envContent = readFile(ENV_FILE_PATH)
                        dir("${PROJECT_DIRECTORY}") {
                            writeFile file: '.env', text: envContent
                        }
                    }
```

```
        }
      }
    }
    stage('Build Docker Images') {
      steps {
        script {
          docker.withRegistry('https://index.docker.io/v1/', "${DOCKER_HUB_CREDENTIALS_ID}") {
            dir("${PROJECT_DIRECTORY}") {
              sh """
                pwd
                ls -al
                docker --version
                export DOCKER_IMAGE_PREFIX=${DOCKER_IMAGE_PREFIX}
                docker compose build
              """
            }
          }
        }
      }
    }
    stage('Docker Push') {
      steps {
        script {
          docker.withRegistry('https://index.docker.io/v1/', "${DOCKER_HUB_CREDENTIALS_ID}") {
            dir("${PROJECT_DIRECTORY}") {
              sh """
                export DOCKER_IMAGE_PREFIX=${DOCKER_IMAGE_PREFIX}
                docker compose push
              """
            }
          }
        }
      }
    }
    stage('Deploy to EC2') {
      options {
        lock('ec2-deployment-lock') // 🔒 동일한 리소스에 대한 배포 작업 직렬화
      }
      steps {
        withCredentials([sshUserPrivateKey(credentialsId: "${EC2_SSH_CREDENTIALS_ID}", keyFileVariable: 'SSH_KEY
          // Create directories on EC2
          sh """
            ssh -o StrictHostKeyChecking=no -i ${SSH_KEY_FILE} ${EC2_USER}@${EC2_HOST} '
              mkdir -p /home/${EC2_USER}/${COMPOSE_PROJECT_NAME}/nginx
              mkdir -p /home/${EC2_USER}/${COMPOSE_PROJECT_NAME}/mysql/init
            '
          """

          // Copy files to EC2
          sh """
            scp -o StrictHostKeyChecking=no -i ${SSH_KEY_FILE} ${PROJECT_DIRECTORY}/docker-compose.yml ${
            scp -o StrictHostKeyChecking=no -i ${SSH_KEY_FILE} ${PROJECT_DIRECTORY}/.env ${EC2_USER}@${E
            scp -o StrictHostKeyChecking=no -i ${SSH_KEY_FILE} ${PROJECT_DIRECTORY}/nginx/nginx.conf ${EC2_
            scp -r -o StrictHostKeyChecking=no -i ${SSH_KEY_FILE} ${PROJECT_DIRECTORY}/mysql/init/* ${EC2_US
          """

          // Deploy the app on EC2
          sh """
            ssh -o StrictHostKeyChecking=no -i ${SSH_KEY_FILE} ${EC2_USER}@${EC2_HOST} '
```

```
                    cd /home/${EC2_USER}/${COMPOSE_PROJECT_NAME}
                    docker compose down
                    export DOCKER_IMAGE_PREFIX=${DOCKER_IMAGE_PREFIX}
                    sudo usermod -aG docker ${EC2_USER}
                    sudo chmod 666 /var/run/docker.sock
                    docker compose pull
                    docker compose up -d --no-recreate
                '
            """
        }
    }
  }
  stage('Cleanup') {
    steps {
      script {
        sh 'docker image prune -f'
      }
    }
  }
}

// 최상위 post 블록은 파이프라인 전체가 끝난 후 항상 실행됩니다.
post {
  always {
    script {
      // Git 관련 정보 수집
      def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
      def Author_Email = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
      def Commit_Message = sh(script: "git log -1 --pretty=%s", returnStdout: true).trim()
      def Branch_Name = sh(script: "git rev-parse --abbrev-ref HEAD", returnStdout: true).trim()

      // 빌드 시작 시간 및 소요 시간 포맷팅
      def Build_Time = new Date(currentBuild.startTimeInMillis)
                .format("yyyy년 MM월 dd일 HH시 mm분 ss초", TimeZone.getTimeZone("Asia/Seoul"))
      def Duration = currentBuild.durationString.replace(' and counting', '')

      // 빌드 결과 및 표시 색상/아이콘 결정
      def Status = currentBuild.result ?: "SUCCESS"
      def Color = (Status == "SUCCESS") ? 'good' : 'danger'
      def Icon = (Status == "SUCCESS") ? "✅" : "❌"

      // 메시지 구성
      def Message = """\
      ${Icon} *BUILD ${Status}*
      - *Job:* ${env.JOB_NAME} #${env.BUILD_NUMBER}
      - *Branch:* ${Branch_Name}
      - *Author:* ${Author_ID} (${Author_Email})
      - *Commit:* ${Commit_Message}
      - *시작 시간:* ${Build_Time}
      - *소요 시간:* ${Duration}
      [🔍 *Details*](${env.BUILD_URL})
      """.stripIndent()

      // Mattermost로 알림 전송
      mattermostSend(
        color: Color,
        message: Message,
        endpoint: 'https://meeting.ssafy.com/hooks/pdzq6qnza7yrjecuho6xx8418o',
        channel: 'B204-Jenkins-Result'
```

```
        )
      }
    }
  }
}
```