포팅 메뉴얼

팀원명	수비
상태	진행 중

사용 도구

기본 도구

도구	버전	설명
Ubuntu	22.04.5 LTS	서버 운영체제
Docker	27.5.1	컨테이너 관리
Docker Compose	2.32.4	다중 컨테이너 관리
GitLab	17.6.2	버전 관리
Open JDK	17.0.14	Java 실행 환경
Node.js	22.13.1	JavaScript 실행 환경
Jira	-	이슈관리
Notion	-	회의관리
Figma	-	디자인

라이브러리/프레임 워크

이름	버전	용도
React.js	18.3.1	프론트엔드 프레임워크
Spring Boot	3.4.2	백엔드 프레임워크
Open Vidu	3.0.0	화상 회의 플랫폼
Nginx	1.18.0	웹 서버, 프록시 서버
Jenkins	2.492.1	CI/CD 자동화 도구

데이터베이스

이름	버전	용도
MySQL	22.04.1	관계형 데이터베이스
Redis	7.4.1	인메모리 데이터베이스

환경 변수

Frontend

```
VITE_BASE_URL=${backend-Url}
예) https://overthecam.site:15555/api
VITE_LIVEKIT_URL=${live-kit-Url}
예) wss://overthecam.site:7443
```

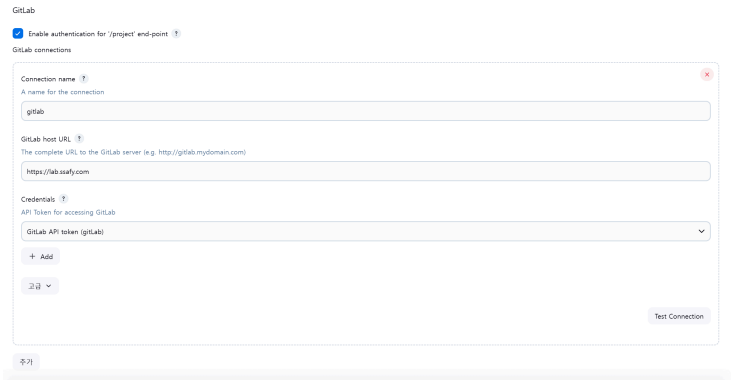
Backend

```
# Production configurations using environment variables
# MySQL
spring.datasource.url=${MYSQL_DB_URL}
spring.datasource.username=${MYSQL_DB_USERNAME}
spring.datasource.password=${MYSQL_DB_PASSWORD}
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
logging.level.org.springframework.web.client.RestTemplate=DEBUG
logging.level.org.apache.http=DEBUG
logging.level.javax.net.ssl=DEBUG
# JWT
jwt.secret=${JWT_SECRET}
# Openvidu
livekit.api.key=${LIVEKIT_API_KEY}
livekit.api.secret=${LIVEKIT_API_SECRET}
# Redis
spring.data.redis.host=spring-redis
spring.data.redis.port=6379
spring.data.redis.password=${SPRING_REDIS_PASSWORD}
# S3
cloud.aws.credentials.access-key=${accesskey-s3}
cloud.aws.credentials.secret-key=${secretkey-s3}
```

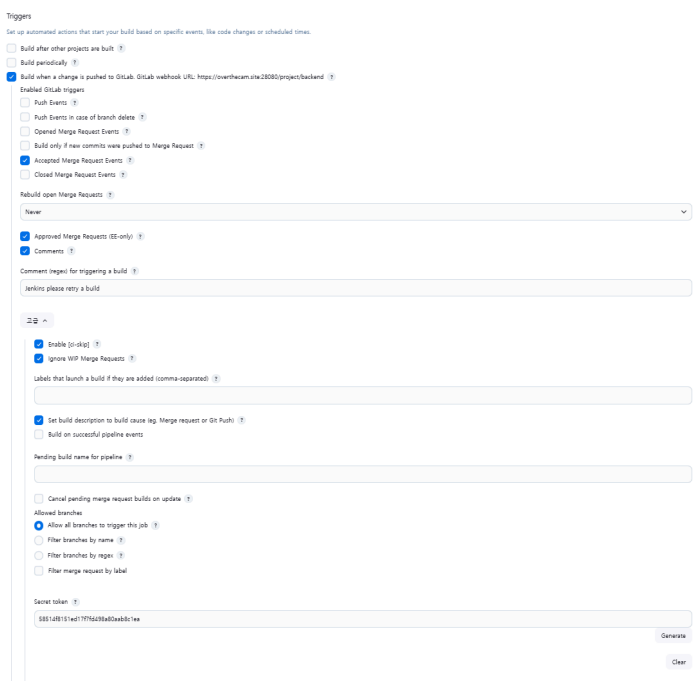
```
cloud.aws.region.static=ap-northeast-2
cloud.aws.stack.auto=false
cloud.aws.s3.bucket=overthecam
# OpenAI
openai.api.key=${OPENAI_API_KEY}
openai.api.url=${OPENAI_API_URL}
```

🦥 Jenkins CI/CD 구축

1. **Jenkins PlugIn**
    a. gitLab, gitLab api, gitLab athuntication
    b. docker, docker pipeline, docker api
    c. ssh, ssh agent
    d. node, python
2. **gitLab tool 추가**



3. **gitLab Webhook 연결**



4. **Item**

5. **DockerFile**
    a. front

```
FROM node:22 as build

ARG VITE_BASE_URL
ARG VITE_LIVEKIT_URL

ENV VITE_BASE_URL=${VITE_BASE_URL}
ENV VITE_LIVEKIT_URL=${VITE_LIVEKIT_URL}

WORKDIR /app

# package.json과 package-lock.json만 먼저 복사
COPY package*.json ./

# npm 캐시 정리 후 패키지 설치
RUN npm cache clean --force && \
    npm ci --production
```

```
# 나머지 파일들 복사
COPY . .

# React 빌드
RUN npm run build

FROM nginx:stable-alpine

COPY --from=build /app/dist /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

b. back

```
FROM openjdk:17-jdk-slim

# 이미지 작성자가 SSAFY임을 나타내는 메타데이터
LABEL authors="SSAFY"

# 환경 변수 정의
ENV SPRING_PROFILES_ACTIVE=prod

# 작업 디렉토리 설정
WORKDIR /app

# 빌드된 JAR 파일을 컨테이너로 복사
COPY build/libs/overthecam-0.0.1-SNAPSHOT.jar app.jar

# 헬스체크 추가
HEALTHCHECK --interval=30s --timeout=3s \
  CMD curl -f http://localhost:8080/actuator/health || exit 1

# 애플리케이션 실행
ENTRYPOINT ["java", "-jar", "app.jar"]
```

c. python

```
FROM python:3.9-slim

WORKDIR /python-analyzer

RUN apt-get update && apt-get install -y \
    build-essential \
    && rm -rf /var/lib/apt/lists/*

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 5001
CMD ["python", "api/app.py"]
```

6. **JenkinsFile**

a. front

```
pipeline {
    agent any

    environment {
        DOCKERHUB_CREDENTIALS = credentials('docker')
        DOCKER_IMAGE_FRONTEND = 'ghkdtnql/overthecam-frontend'
    }

    stages {
        stage('Git Clone') {
            steps {
                script {
                    try {
                        echo '====== Git Clone 시작 ======'
                        git branch: 'main',
                            url: 'https://lab.ssafy.com/s12-webmobile1-sub1/S12P11D204.git',
                            credentialsId: 'gitlab-user'
                        echo '====== Git Clone 성공 ======'
                    } catch (Exception e) {
                        echo '====== Git Clone 실패 ======'
                        error "Git clone failed: ${e.message}"
                    }
                }
            }
```

```
    }

    stage('Build FE') {
        steps {
            script {
                try {
                    echo '====== React Build 시작 ======'
                    dir('frontend/overthecam') {
                        sh '''
                            set -x
                            npm install
                            npm uninstall rollup
                            npm install rollup
                            CI=false npm run build
                        '''
                    }
                    echo '====== React Build 성공 ======'
                } catch (Exception e) {
                    echo '====== React Build 실패 ======'
                    error "Build failed: ${e.message}"
                }
            }
        }
    }

    stage('Docker Build & Push') {
        steps {
            script {
                try {
                    echo '====== Docker Build 시작 ======'

                    dir('frontend/overthecam') {
                        withCredentials([
                            string(credentialsId: 'front-base-url', variable: 'VITE_BASE_URL'),
                            string(credentialsId: 'front-livekit-url', variable: 'VITE_LIVEKIT_URL')
                        ]) {
                            sh '''
                                set -x
                                echo "== Docker 환경 확인 =="
                                docker version
                                docker info

                                echo "== 환경 변수 확인 =="
                                echo "VITE_BASE_URL=${VITE_BASE_URL}"
                                echo "VITE_LIVEKIT_URL=${VITE_LIVEKIT_URL}"

                                echo "== Docker Build 실행 (BuildKit 비활성화) =="
                                export DOCKER_BUILDKIT=0
                                docker build --no-cache \
                                    --build-arg VITE_BASE_URL=${VITE_BASE_URL} \
                                    --build-arg VITE_LIVEKIT_URL=${VITE_LIVEKIT_URL} \
                                    --pull \
                                    -t ${DOCKER_IMAGE_FRONTEND}:${BUILD_NUMBER} .

                                echo "== DockerHub 로그인 =="
                                echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin

                                echo "== Docker 이미지 푸시 =="
                                docker push ${DOCKER_IMAGE_FRONTEND}:${BUILD_NUMBER}

                                echo "== DockerHub 로그아웃 =="
                                docker logout
                            '''
                        }
                    }
                    echo '====== Docker Build & Push 완료 ======'
                } catch (Exception e) {
                    echo '====== Docker Build/Push 실패 ======'
                    sh "docker logout || true"
                    error "Docker build/push failed: ${e.message}"
                }
            }
        }
    }

    stage('Deploy to EC2') {
        steps {
            script {
                try {
                    echo '====== EC2 배포 시작 ======'
                    def dockerImageTag = "${DOCKER_IMAGE_FRONTEND}:${BUILD_NUMBER}"

                    sshagent(['ssh']) {
```

```
                withCredentials([
                    string(credentialsId: 'front-base-url', variable: 'VITE_BASE_URL'),
                ]) {
                    sh """
                        set -x
                        echo "== EC2 배포 환경 변수 확인 =="
                        echo "DOCKER_IMAGE_FRONTEND=${DOCKER_IMAGE_FRONTEND}"
                        echo "BUILD_NUMBER=${BUILD_NUMBER}"
                        echo "dockerImageTag=${dockerImageTag}"

                        echo "== EC2에 SSH 접속하여 Docker 컨테이너 실행 =="
                        ssh -o StrictHostKeyChecking=no ubuntu@i12d204.p.ssafy.io '
                            echo "VITE_BASE_URL=${VITE_BASE_URL}" >> .env &&
                            docker pull ${dockerImageTag} &&
                            docker stop frontend || true &&
                            docker rm frontend || true &&
                            docker run -d --name frontend -p 5557:80 --env-file .env ${dockerImageTag} &&
                            docker system prune -f'
                    """
                }
            }
            echo '====== EC2 배포 성공 ======'
        } catch (Exception e) {
            echo '====== EC2 배포 실패 ======'
            error "Deployment failed: ${e.message}"
        }
    }
}

post {
    always {
        echo '====== 파이프라인 종료 ======'
        sh "docker logout || true"
        cleanWs()
    }
    failure {
        echo '====== 파이프라인 실패 ======'
    }
    success {
        echo '====== 파이프라인 성공 ======'
    }
}
}
```

b. back

```
pipeline {
    agent any

    environment {
        DOCKER_IMAGE_BACKEND = 'ghkdtnql/overthecam-backend'
        NETWORK_NAME = 'my_network'
    }

    stages {
stage('Git Clone') {
    steps {
        script {
            try {
                echo '====== Git Clone 시작 ======'
                git branch: 'main',
                    url: 'https://lab.ssafy.com/s12-webmobile1-sub1/S12P11D204.git',
                    credentialsId: 'gitlab-user'
                echo '====== Git Clone 성공 ======'
            } catch (Exception e) {
                echo '====== Git Clone 실패 ======'
                error "Git clone failed: ${e.message}"
            }
        }
    }
}

stage('Clean old JARs') {
    steps {
        sh 'rm -f build/libs/*.jar'
    }
}

stage('Build BE') {
    steps {
        script {
```

```
            try {
                echo '====== Gradle Build 시작 ======'
                sh '''
                    cd backend/overthecam
                    chmod +x gradlew
                    rm -rf build
                    ./gradlew clean
                    ./gradlew build -x test
                    ls -l build/libs/
                '''
                echo '====== Gradle Build 성공 ======'
            } catch (Exception e) {
                echo '====== Gradle Build 실패 ======'
                error "Build failed: ${e.message}"
            }
        }
    }
}

stage('Docker Build & Push') {
    steps {
        script {
            try {
                echo '====== Docker Build 시작 ======'
                dir('backend/overthecam') {
                    withCredentials([usernamePassword(credentialsId: 'docker', usernameVariable: 'DOCKERHUB_CREDENTIALS_USR', passwordVariable: 'DOCKERHUB_
                    sh """
                        docker images | grep ${DOCKER_IMAGE_BACKEND} | tr -s ' ' | cut -d ' ' -f 2 | xargs -I {} docker rmi ${DOCKER_IMAGE_BACKEND}:{} || true
                        docker build -t ${DOCKER_IMAGE_BACKEND}:${BUILD_NUMBER} .
                        echo '${DOCKERHUB_CREDENTIALS_PSW}' | docker login -u '${DOCKERHUB_CREDENTIALS_USR}' --password-stdin
                        docker push ${DOCKER_IMAGE_BACKEND}:${BUILD_NUMBER}
                        docker logout
                    """
                }
            }
            echo '====== Docker Build & Push 완료 ======'
        } catch (Exception e) {
            echo '====== Docker Build/Push 실패 ======'
            error "Docker build/push failed: ${e.message}"
        }
    }
  }
}

stage('Deploy to EC2') {
    steps {
        script {
            try {
                echo '====== EC2 배포 시작 ======'
                sshagent(['ssh']) {
                    withCredentials([
                        string(credentialsId: 'mysql-url', variable: 'MYSQL_DB_URL'),
                        usernamePassword(credentialsId: 'mysql-credentials', usernameVariable: 'MYSQL_DB_USERNAME', passwordVariable: 'MYSQL_DB_PASSWORD'),
                        string(credentialsId: 'jwt-secret', variable: 'JWT_SECRET'),
                        string(credentialsId: 'LIVEKIT_API_KEY', variable: 'LIVEKIT_API_KEY'),
                        string(credentialsId: 'LIVEKIT_API_SECRET', variable: 'LIVEKIT_API_SECRET'),
                        string(credentialsId: 'SPRING_REDIS_PASSWORD', variable: 'SPRING_REDIS_PASSWORD'),
                        string(credentialsId: 'secretkey-s3', variable:'secretkey-s3'),
                        string(credentialsId: 'accesskey-s3', variable:'accesskey-s3')
                    ]) {
                        sh '''
                            ssh -o StrictHostKeyChecking=no ubuntu@i12d204.p.ssafy.io "\
                            # Redis 네트워크 생성 (없는 경우에만)
                            docker network create redis-network || true && \

                            # Redis 컨테이너 실행/재시작
                            echo '기존 Redis 컨테이너 정리 중...' && \
                            docker stop spring-redis || true && \
                            docker rm spring-redis || true && \

                            echo 'Redis 컨테이너 시작 중...' && \
                            docker run -d \
                                --name spring-redis \
                                --network redis-network \
                                -p 6379:6379 \
                                redis:7.4.1-alpine \
                                /bin/sh -c 'redis-server --requirepass "${SPRING_REDIS_PASSWORD}"'

                            echo '새 이미지 가져오는 중...' && \
                            docker pull ${DOCKER_IMAGE_BACKEND}:${BUILD_NUMBER} && \
                            echo '기존 컨테이너 중지 중...' && \
                            docker stop backend || true && \
                            docker rm backend || true && \
```

```
                    echo '새 컨테이너 시작 중...' && \
                    docker run -d --name backend \
                      --network redis-network \
                      -p 5556:8080 \
                      -e SPRING_PROFILES_ACTIVE=prod \
                      -e MYSQL_DB_URL='${MYSQL_DB_URL}' \
                      -e MYSQL_DB_USERNAME='${MYSQL_DB_USERNAME}' \
                      -e MYSQL_DB_PASSWORD='${MYSQL_DB_PASSWORD}' \
                      -e JWT_SECRET='${JWT_SECRET}' \
                      -e LIVEKIT_API_KEY='${LIVEKIT_API_KEY}' \
                      -e LIVEKIT_API_SECRET='${LIVEKIT_API_SECRET}' \
                      -e SPRING_REDIS_PASSWORD='${SPRING_REDIS_PASSWORD}' \
                      -e accesskey-s3='${accesskey-s3}'\
                      -e secretkey-s3='${secretkey-s3}'\
                      ${DOCKER_IMAGE_BACKEND}:${BUILD_NUMBER} && \
                    docker system prune -f && \
                    echo '컨테이너 로그:' && \
                    sleep 10 && \
                    docker logs backend"
                '''
              }
            }
            echo '====== EC2 배포 성공 ======'
          } catch (Exception e) {
            echo '====== EC2 배포 실패 ======'
            error "Deployment failed: ${e.message}"
          }
        }
      }
    }

    post {
      always {
        echo '====== 파이프라인 종료 ======'
        sh "docker logout || true"
        cleanWs()
      }
      failure {
        echo '====== 파이프라인 실패 ======'
        sh "docker rmi ${DOCKER_IMAGE_BACKEND}:${BUILD_NUMBER} || true"
      }
      success {
        echo '====== 파이프라인 성공 ======'
      }
    }
}
```

c. python

```
pipeline {
    agent any

    environment {
        DOCKER_IMAGE_ANALYZER = 'ghkdtnql/overthecam-analyzer'
    }

    stages {
        stage('Git Clone') {
            steps {
                script {
                    try {
                        echo '====== Git Clone 시작 ======'
                        git branch: 'main',
                            url: 'https://lab.ssafy.com/s12-webmobile1-sub1/S12P11D204.git',
                            credentialsId: 'gitlab-user'
                        echo '====== Git Clone 성공 ======'

                        // 📌 Git Clone 후 파일 목록 확인
                        sh "ls -l /var/jenkins_home/workspace/python/python-analyzer/models/saved/emotion_model"

                    } catch (Exception e) {
                        echo '====== Git Clone 실패 ======'
                        error "Git clone failed: ${e.message}"
                    }
                }
            }
        }

        stage('Docker Build & Push') {
            steps {
                script {
                    try {
```

```
                echo '====== Docker Build 시작 ======'
                dir('python-analyzer') {
                    writeFile file: '.dockerignore', text: '''
                        __pycache__/
                        *.pyc
                        venv/
                        .git/
                        .env
                        temp/
                        *.pth
                        checkpoints/
                    '''

                    withCredentials([usernamePassword(credentialsId: 'docker', usernameVariable: 'DOCKERHUB_CREDENTIALS_USR', passwordVariable: 'DOCKERHUB_
                        sh """
                            docker images | grep ${DOCKER_IMAGE_ANALYZER} | tr -s ' ' | cut -d ' ' -f 2 | xargs -I {} docker rmi ${DOCKER_IMAGE_ANALYZER}:{} || true
                            docker build -t ${DOCKER_IMAGE_ANALYZER}:${BUILD_NUMBER} .
                            echo '${DOCKERHUB_CREDENTIALS_PSW}' | docker login -u '${DOCKERHUB_CREDENTIALS_USR}' --password-stdin
                            docker push ${DOCKER_IMAGE_ANALYZER}:${BUILD_NUMBER}
                            docker logout
                        """
                    }
                }
                echo '====== Docker Build & Push 완료 ======'
            } catch (Exception e) {
                echo '====== Docker Build/Push 실패 ======'
                error "Docker build/push failed: ${e.message}"
            }
        }
    }
}

stage('Deploy to EC2') {
    steps {
        script {
            try {
                echo '====== EC2 배포 시작 ======'
                sshagent(['ssh']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no ubuntu@i12d204.p.ssafy.io "\
                        echo '새 이미지 가져오는 중...' && \
                        docker pull ${DOCKER_IMAGE_ANALYZER}:${BUILD_NUMBER} && \
                        echo '기존 컨테이너 중지 중...' && \
                        docker stop analyzer || true && \
                        docker rm analyzer || true && \
                        echo '새 컨테이너 시작 중...' && \
                        docker run -d --name analyzer \
                            -v /var/jenkins_home/workspace/python/python-analyzer/models/saved/emotion_model:/python-analyzer/models/saved/emotion_model \
                            -p 5001:5001 \
                            ${DOCKER_IMAGE_ANALYZER}:${BUILD_NUMBER} && \
                        docker system prune -f && \
                        echo '컨테이너 로그:' && \
                        sleep 10 && \
                        docker logs analyzer"
                    '''
                }
                echo '====== EC2 배포 성공 ======'
            } catch (Exception e) {
                echo '====== EC2 배포 실패 ======'
                error "Deployment failed: ${e.message}"
            }
        }
    }
}

post {
    always {
        echo '====== 파이프라인 종료 ======'
        sh "docker logout || true"
        cleanWs()
    }
    failure {
        echo '====== 파이프라인 실패 ======'
        sh "docker rmi ${DOCKER_IMAGE_ANALYZER}:${BUILD_NUMBER} || true"
    }
    success {
        echo '===== 파이프라인 성공 ====='
    }
}
}
```

7. Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	gitLab	GitLab API token (gitLab)
		System	(global)	ssh	ubuntu (ssh)
		System	(global)	docker	ghkdtnql/****** (docker)
		System	(global)	mysql-credentials	root/****** (mysql)
		System	(global)	mysql-url	mysql-url
		System	(global)	jwt-secret	jwt-secret
		System	(global)	gitlab-user	ghkdtnql/****** (gitlab-user)
		System	(global)	front-base-url	front-base-url
		System	(global)	LIVEKIT_API_KEY	LIVEKIT_API_KEY
		System	(global)	LIVEKIT_API_SECRET	LIVEKIT_API_SECRET
		System	(global)	front-livekit-url	front-livekit-url
		System	(global)	SPRING_REDIS_PASSWORD	스프링에서 사용하는 redis 비밀번호
		System	(global)	accesskey-s3	accesskey-s3
		System	(global)	secretkey-s3	secretkey-s3

Nginx 설정

```
################################################################################
# Nginx Example Configuration
# (default site)
#
# - HTTP: 10080 포트 사용
# - HTTPS: 443 / 15555 / 7443
#
# 도메인: overthecam.site
#
# Let's Encrypt 인증서 (이미 발급된 상태):
#   /etc/letsencrypt/live/overthecam.site/fullchain.pem
#   /etc/letsencrypt/live/overthecam.site/privkey.pem
################################################################################

##
# 1) 10080 포트 기본 서버 블록 (default_server)
#    - 특정 도메인을 갖지 않은 나머지 요청은 여기로 와서 404 응답
##
server {
    listen 10080 default_server;
    listen [::]:10080 default_server;

    server_name _;  # 와일드카드(기본)

    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;

    location / {
        try_files $uri $uri/ =404;
    }
}

##
# 2) 10080 포트 + 도메인 overthecam.site → HTTPS(443) 리다이렉트
##
server {
    listen 10080;
    listen [::]:10080;

    server_name overthecam.site;

    location / {
        # http://overthecam.site:10080 → https://overthecam.site
        return 301 https://overthecam.site$request_uri;
    }
}

##
# 3) HTTPS(443) → localhost:5080 프록시 (Front)
##
server {
    listen 443 ssl http2;       # IPv4
    listen [::]:443 ssl http2;  # IPv6

    server_name overthecam.site;

    #--- SSL 인증서 설정 (Let's Encrypt 발급분)
    ssl_certificate     /etc/letsencrypt/live/overthecam.site/fullchain.pem;
```

```
    ssl_certificate_key /etc/letsencrypt/live/overthecam.site/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    # location /tutorial {
    #     proxy_pass http://127.0.0.1:5080/;
    #     # 필요 시 header 설정
    #     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;
    # }
    # React 애플리케이션의 assets 처리
    location /assets/ {
        proxy_pass http://127.0.0.1:5557/assets/;
        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 / {
        proxy_pass http://127.0.0.1:5557/;

        # 필요 시 header 설정
        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;

        try_files $uri $uri/ /index.html;  # React Router용

        # MIME 타입 보존을 위한 설정
        proxy_hide_header Content-Type;
        proxy_pass_header Content-Type;

    }

    # 정적 파일 캐싱 설정
    location ~* \.(css|js|jpg|jpeg|png|gif|ico|svg|woff2?|ttf|eot)$ {
        proxy_pass http://127.0.0.1:5557;
        proxy_set_header Host $host;
        expires 1y;
        add_header Cache-Control "public, no-transform";
    }

}

##
# 4) HTTPS(15555) → localhost:8080 프록시 (Back)
##
server {
    listen 15555 ssl http2;
    listen [::]:15555 ssl http2;

    server_name overthecam.site;

    ssl_certificate     /etc/letsencrypt/live/overthecam.site/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/overthecam.site/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location /api {
        proxy_pass http://127.0.0.1:5556;
        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;
    }
}

##
# 5) HTTPS(7443) → localhost:7880 WebSocket 프록시
##
server {
    listen 7443 ssl http2;
    listen [::]:7443 ssl http2;

    server_name overthecam.site;

    ssl_certificate     /etc/letsencrypt/live/overthecam.site/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/overthecam.site/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
```

```
    location / {
        proxy_pass http://127.0.0.1:7780/;

        # WebSocket 필수 헤더
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";

        # 일반 proxy 헤더
        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;
    }
}

server{
    listen 28080 ssl http2;
    listen [::]:28080 ssl http2;

    server_name overthecam.site;

    ssl_certificate     /etc/letsencrypt/live/overthecam.site/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/overthecam.site/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location / {
        proxy_pass http://127.0.0.1:18080/;
        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;
    }
}
```

Docker 설정

전체 도커 컨테이너

```
ubuntu@ip-172-26-3-112:~$ docker ps
CONTAINER ID   IMAGE                                    COMMAND                  CREATED          STATUS                       PORTS
                                                                                 NAMES
a40ef225f6d4   ghkdtnql/overthecam-backend:313          "java -jar app.jar"      2 minutes ago    Up 2 minutes (unhealthy)     0.0.0.0:5556->8080/tcp
, [::]:5556->8080/tcp                                                            backend
1c5499160bff   redis:7.4.1-alpine                       "docker-entrypoint.s…"   2 minutes ago    Up 2 minutes                 0.0.0.0:6379->6379/tcp
, :::6379->6379/tcp                                                              spring-redis
18b5374114a7   ghkdtnql/overthecam-frontend:356         "/docker-entrypoint.…"   8 minutes ago    Up 8 minutes                 0.0.0.0:5557->80/tcp,
[::]:5557->80/tcp                                                                frontend
843407d434e8   grafana/grafana:11.3.0                   "/bin/sh -c '\n  if !…"   26 hours ago     Up 26 hours
                                                                                 grafana
6d6e19bdb98c   grafana/promtail:3.2.1                   "/bin/sh -c '\n  if !…"   26 hours ago     Up 26 hours
                                                                                 promtail
1ba90fbbff8e   prom/prometheus:v2.55.0                  "/bin/sh -c '\n  if !…"   26 hours ago     Up 26 hours
                                                                                 prometheus
897dc15c575f   livekit/ingress:v1.4.2                   "ingress"                26 hours ago     Up 26 hours
                                                                                 ingress
6e3788218588   openvidu/openvidu-call:3.0.0             "docker-entrypoint.s…"   26 hours ago     Up 26 hours
                                                                                 app
8c014ec3bae0   redis:7.4.1-alpine                       "/bin/sh -c '\n  . /c…"   26 hours ago     Up 26 hours
                                                                                 redis
a387025a91b4   bitnami/minio:2024.10.13-debian-12-r1    "/bin/sh -c '\n  . /c…"   26 hours ago     Up 26 hours
                                                                                 minio
75156685b72c   grafana/loki:3.2.1                       "/bin/sh -c '\n  if !…"   26 hours ago     Up 26 hours
                                                                                 loki
042755e4f425   livekit/egress:v1.8.4                    "/entrypoint.sh"         26 hours ago     Up 26 hours
                                                                                 egress
5ff91ff0e88b   openvidu/openvidu-dashboard:3.0.0        "./openvidu-dashboard"   26 hours ago     Up 26 hours
                                                                                 dashboard
0b3103b0fc7c   openvidu/openvidu-caddy:3.0.0            "/bin/caddy run --co…"   26 hours ago     Up 26 hours
                                                                                 caddy
0516eb2918b4   openvidu/openvidu-server:3.0.0           "/livekit-server --c…"   26 hours ago     Up 26 hours
                                                                                 openvidu
98b22444c78e   mongo:7.0.15                             "/bin/sh -c '\n  . /c…"   26 hours ago     Up 26 hours
                                                                                 mongo
c349c62e2529   jenkins/jenkins:lts                      "/usr/bin/tini -- /u…"   5 days ago       Up 26 hours                  0.0.0.0:18080->8080/tc
p, [::]:18080->8080/tcp, 0.0.0.0:40000->50000/tcp, [::]:40000->50000/tcp   jenkins
```

Openvidu 설정

1. 오픈비두 설치 (공식문서 참고)

https://openvidu.io/latest/docs/self-hosting/single-node/on-premises/install/#guided-installation

2. caddy 포트 바꾸기 (openvidu.env)

```
# ------------------------------------
# Caddy proxy related ports
# ------------------------------------
```

```
# This port is used by caddy to serve all the services
# at the specified port through HTTPS and to expose RTMPS and TURN with TLS.
# If you change it, you must also change the port binding
# in the docker-compose file of the Caddy service.
CADDY_HTTPS_PUBLIC_PORT=5443

# This port is used by caddy to redirect from HTTP to HTTPS.
# If you change it, you must also change the port binding
# in the docker-compose file of the Caddy service.
CADDY_HTTP_PUBLIC_PORT=5442
```

SSL 인증서 설정

1. 도메인 준비
2. Let’s Encript 인증서 발급
3. Nginx 적용