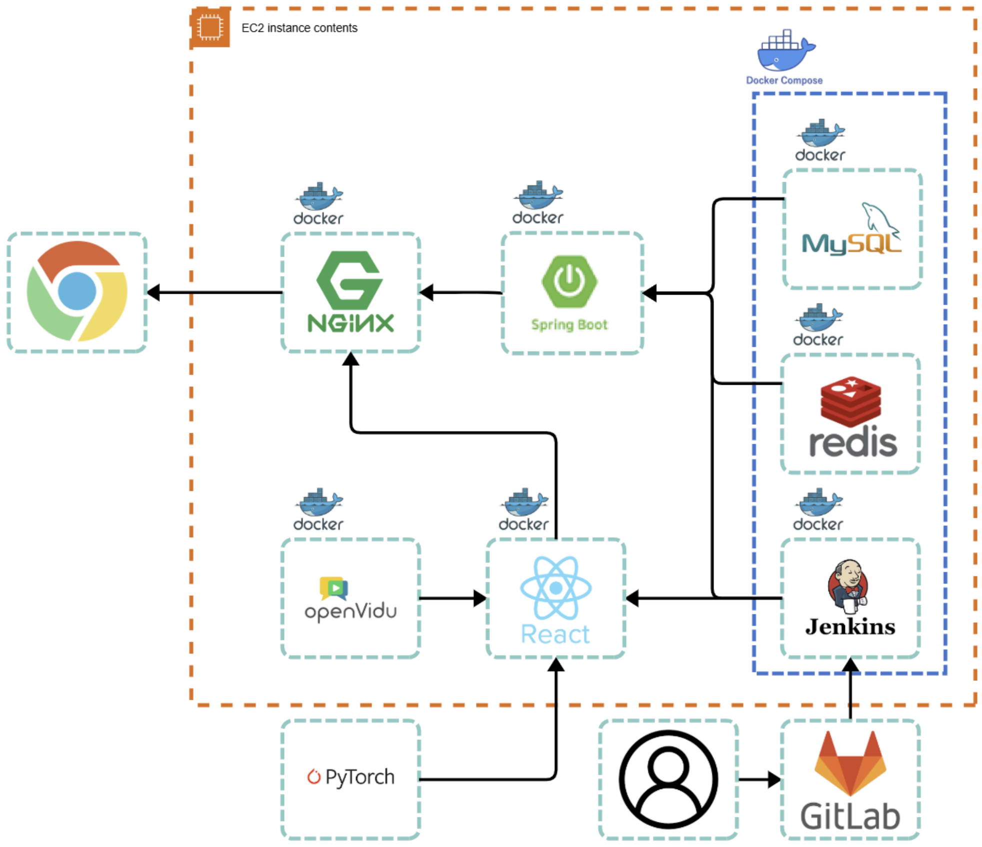
포팅 매뉴얼 - E106(철거왕 김주먹)

1. 개발 환경
2. Docker
3. Nginx 설정파일
4. Docker Compose.yml
5. Jenkins CI/CD pipeline



▼ 1. 개발 환경

- **Frontend**

Node.js	22.17.1
React.js	18.2.0
Pixi.js	7.4.3
MedeaPipe/tasks-vision	0.10.22-rc.20250304

- **Backend**

SpringBoot	3.5.3

Intellij	21.0.7
JDK	OpenJDK17
MySQL	21.0.7
MySQL Workbench	8.0.42
Redis	7.4.5

- Server

Jenkins	2.516.1
EC2	22.04.4LTS
Nginx	1.18.0
Openvidu3	3.3.0
docker	28.3.2
docker-compose	2.516.1

▼ 2. Docker

컨테이너 이름	이미지	포트 매핑	상태 (Up)	실행 명령어 (요약)
frontend	frontend	80/tcp	14 hours	java -jar app.jar
openviduback	openviduback	0.0.0.0:6080→6080/tcp, 8080/tcp	14 hours	java -jar app.jar
backend	backend	8080/tcp	14 hours	java -jar app.jar
nginx	nginx:stable	0.0.0.0:80→80/tcp, 0.0.0.0:443→443/tcp	2 days	nginx
redis_cumstom	redis:7	0.0.0.0:6379→6379/tcp	2 days	redis-server
grafana	grafana/grafana:11.6.2		2 days	grafana
prometheus	prom/prometheus:v3.4.0		2 days	prometheus
promtail	grafana/promtail:3.5.1		2 days	promtail
loki	grafana/loki:3.5.1		2 days	loki
redis	redis:7.4.4-alpine		2 days	redis-server
ingress	openvidu/ingress:3.3.0		2 days	ingress
minio	bitnami/minio:2025.5.24		2 days	minio
openvidu	openvidu/openvidu-server:3.3.0		2 days	openvidu-server
caddy	openvidu/openvidu-caddy:3.3.0		2 days	caddy
dashboard	openvidu/openvidu-dashboard:3.3.0		2 days	dashboard
app	openvidu/openvidu-call:3.3.0		2 days	openvidu-call

컨테이너 이름	이미지	포트 매핑	상태 (Up)	실행 명령어 (요약)
egress	livekit/egress:v1.9.1		2 days	livekit-server
mongo	mongo:8.0.9		2 days	mongo
operator	openvidu/openvidu-operator:3.3.0		2 days	operator
jenkins_custom	dev-jenkins	0.0.0.0:8081→8080/tcp, 50000/tcp	2 weeks	jenkins
mysql_custom	mysql:8	0.0.0.0:3306→3306/tcp, 33060/tcp	2 weeks	mysqld

3. Nginx 설정파일

Nginx.conf

```
e3user  nginx;
worker_processes  auto;

events {
  worker_connections 10240;
}

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

  # 성능/기본 튜닝
  sendfile on;
  tcp_nopush on;
  tcp_nodelay on;
  keepalive_timeout 65;
  types_hash_max_size 4096;
  client_max_body_size 50m;

  # WebSocket 업그레이드 헬퍼(서버 블록들이 사용)
  map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
  }

  # conf.d/*.conf 에 실제 서버 블록(=default.conf)이 들어감
  include /etc/nginx/conf.d/*.conf;
}
```

Default.conf

```
  GNU nano 6.2
default.conf
```

```
# ---------------- HTTP → HTTPS redirect ----------------
server {
  listen 80;
  server_name i13e106.p.ssafy.io;
  return 301 https://$host$request_uri;
}

# ---------------- Main HTTPS server ----------------
server {
  listen 443 ssl;
  server_name i13e106.p.ssafy.io;

  # 인증서 (컨테이너에 /etc/letsencrypt 마운트)
  ssl_certificate     /etc/letsencrypt/live/i13e106.p.ssafy.io/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/i13e106.p.ssafy.io/privkey.pem;
  ssl_protocols       TLSv1.2 TLSv1.3;
  ssl_ciphers         HIGH:!aNULL:!MD5;

  # ==========================
  # 1) LiveKit / OpenVidu HTTP(WS) → host의 caddy:7880
  #    (/livekit, /openvidu 등 HTTP 계열은 전부 7880으로)
  # ==========================

  # 정확히 /livekit (SDK 헬스체크/프리플라이트)
  location = /livekit {
    proxy_pass http://172.26.14.249:7880/;
    proxy_http_version 1.1;

    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection $connection_upgrade;
    proxy_set_header Sec-WebSocket-Protocol $http_sec_websocket_protocol;

    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_read_timeout 3600s;
    proxy_send_timeout 3600s;
    proxy_buffering off;
  }

  # /livekit/* (예: /livekit/rtc?access_token=...)
  location ^~ /livekit/ {
    proxy_pass http://172.26.14.249:7880/;
    proxy_http_version 1.1;

    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection $connection_upgrade;
    proxy_set_header Sec-WebSocket-Protocol $http_sec_websocket_protocol;
```

```
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_read_timeout 3600s;
        proxy_send_timeout 3600s;
        proxy_buffering off;
    }

    # OpenVidu Dashboard / Default App 등 HTTP 엔드포인트
    location ^~ /openvidu/ {
        proxy_pass http://172.26.14.249:7880/;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_read_timeout 3600s;
        proxy_send_timeout 3600s;
    }

    # ==========================
    # 2) 백엔드(API) 프록시
    # ==========================
    # SSE (버퍼링 금지)
    location ^~ /api/sse/ {
        proxy_pass http://backend:8080/api/sse/;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;

        proxy_buffering off;
        proxy_cache off;
        gzip off;
        proxy_set_header Connection "";

        proxy_read_timeout 1h;
        proxy_send_timeout 1h;

        add_header Cache-Control "no-cache" always;
        add_header X-Accel-Buffering "no" always;
    }

    location ^~ /oauth2/ {
        proxy_pass http://backend:8080/;
        proxy_http_version 1.1;
        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;
        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Port $server_port;
    }

    location ^~ /login/oauth2/ {
        proxy_pass http://backend:8080/;
        proxy_http_version 1.1;
        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;
        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Port $server_port;
    }

    # 전체 API
    location /api/ {
        proxy_pass http://backend:8080/api/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    # Swagger
    location /swagger-ui/ {
        proxy_pass http://backend:8080/swagger-ui/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
    location /v3/api-docs {
        proxy_pass http://backend:8080/v3/api-docs;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    # =========================
    # 3) (선택) 우리 openviduback 서비스 (6080)
    # =========================
    location ^~ /openviduback/ {
        proxy_pass http://openviduback:6080/;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
    location = /openvidu {
        proxy_pass http://openviduback:6080/;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    # ==========================
    # 4) 프론트 (SPA)
    # ==========================
    # 정적 리소스(캐시)
    location ^~ /mediapipe/ {
        proxy_pass http://frontend:80;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;

        proxy_intercept_errors off;
        expires 1y;
        add_header Cache-Control "public, immutable" always;
    }
    location /assets/ {
        proxy_pass http://frontend:80;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;

        proxy_intercept_errors off;
        expires 1y;
        add_header Cache-Control "public, immutable" always;
        access_log off;
    }
    location ~* \.(woff2?|ttf|otf|eot|png|jpe?g|gif|webp|svg)$ {
        proxy_pass http://frontend:80;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;

        proxy_intercept_errors off;
        expires 1y;
        add_header Cache-Control "public, immutable" always;
        access_log off;
    }

    # 메인 프론트 + SPA fallback
    location / {
        add_header Content-Security-Policy "
          default-src 'self';
          script-src 'self' 'unsafe-inline' 'unsafe-eval' https://cdn.jsdelivr.net https://unpkg.
com;
          style-src  'self' 'unsafe-inline' https://cdn.jsdelivr.net https://unpkg.com;
          img-src    'self' data: blob: https://cdn.jsdelivr.net https://unpkg.com;
          font-src   'self' data:;
          connect-src 'self'
                   https://cdn.jsdelivr.net
                   https://unpkg.com
                   https://storage.googleapis.com
                   https://i13e106.p.ssafy.io
                   wss://i13e106.p.ssafy.io
                   https://i13e106.p.ssafy.io/livekit
                   wss://i13e106.p.ssafy.io/livekit;
          media-src 'self' blob:;
          worker-src 'self' blob:;
          frame-ancestors 'self';
        " always;

        proxy_pass http://frontend:80;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;

        proxy_intercept_errors on;
        error_page 404 = /index.html;
    }
}
```

▼ 4. Docker Compose.yml

```yaml
version: "3.8"

services:
  mysql_temp:
    image: mysql:8
    container_name: mysql_temp
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: ssafy
      MYSQL_DATABASE: demolition_db
    ports:
      - "3306:3306"
    volumes:
      - ./mysql_data_temp:/var/lib/mysql

  redis_temp:
    image: redis:7
    container_name: redis_temp
    restart: always
    ports:
      - "6379:6379"
    volumes:
      - ./redis_data_temp:/data
    command:
      - redis-server
      - --requirepass
      - ssafyel06

  jenkins:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: jenkins_custom
    ports:
      - "8081:8080"
    volumes:
      - jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock   # ✅ 호스트 Docker 접근 허용
    privileged: true                           # ✅ 내부에서 Docker 명령 실행 가능하게
    user: root                                # ✅ 퍼미션 이슈 방지

volumes:
  jenkins_home:
```

▼ 5. Jenkins CI/CD pipeline

▼ FrontEnd

```
pipeline {
  agent any

  options {
    disableConcurrentBuilds()
  }

  environment {
    DOCKER_IMAGE = "frontend"
    CONTAINER_NAME = "frontend"
  }

  stages {
    stage('Clean Workspace') {
      steps {
        deleteDir()  // ✅ 먼저 워크스페이스 전체 삭제
      }
    }

    stage('Git Clone') {
      steps {
```

```
      git credentialsId: 'e106_Token',
        url: 'https://lab.ssafy.com/s13-webmobile1-sub1/S13P11E106.git',
        branch: 'main'
    }
  }

  stage('NPM Cleanup') {
    steps {
      sh '''
        echo "🧼 Removing node_modules and lock file"
        rm -rf ./Front/node_modules || true
      '''
    }
  }

  stage('Docker Build') {
    steps {
      sh '''
        echo "🛠 Cleaning old Docker artifacts..."
        docker stop $CONTAINER_NAME || true
        docker rm $CONTAINER_NAME || true
        docker rmi $DOCKER_IMAGE || true

        echo "🔨 Building Docker image..."
        docker build --no-cache -t $DOCKER_IMAGE -f ./Front/Dockerfile ./Front
      '''
    }
  }

  stage('Docker Run') {
    steps {
      sh '''
        echo "🚀 Starting container..."
        docker run -d \
          --name $CONTAINER_NAME \
          --network backend-net \
          $DOCKER_IMAGE
      '''
    }
  }
}

post {
  success {
    echo '✅ Frontend deployed successfully.'
    sh '''
      echo "🔄 Reloading NGINX configuration..."
```

```
        docker exec nginx nginx -s reload || echo "❗ NGINX reload failed"
      '''
    }
    failure {
      echo '❌ Frontend build or deployment failed.'
    }
  }
}
```

▼ BackEnd

```
pipeline {
  agent any

  options {
    disableConcurrentBuilds()
  }

  environment {
    BACK_IMAGE = "backend"
    BACK_CONTAINER = "backend"
    OPENVIDU_IMAGE = "openviduback"
    OPENVIDU_CONTAINER = "openviduback"
  }

  stages {
    stage('Git Clone') {
      steps {
        git credentialsId: 'e106_Token',
            url: 'https://lab.ssafy.com/s13-webmobile1-sub1/S13P11E106.git',
            branch: 'main'
      }
    }

    stage('Build Backend (Gradle)') {
      steps {
        sh '''
          cd Back
          chmod +x ./gradlew
          ./gradlew clean build
        '''
      }
    }

    stage('Build OpenViduBack (optional)') {
      when {
        expression { fileExists('openviduback/Dockerfile') }
      }
      steps {
        echo '📦 openviduback Dockerfile exists, proceeding with build.'
      }
    }

    stage('Docker Build & Run - Backend') {
      steps {
        sh '''
          docker stop $BACK_CONTAINER || true
          docker rm $BACK_CONTAINER || true
          docker rmi $BACK_IMAGE || true
          docker build -t $BACK_IMAGE ./Back
          docker run -d --name $BACK_CONTAINER --network backend-net $BACK_IMAG
E
        '''
      }
    }

    stage('Docker Build & Run - OpenViduBack') {
     steps {
    sh '''
      docker stop $OPENVIDU_CONTAINER || true
      docker rm $OPENVIDU_CONTAINER || true
      docker rmi $OPENVIDU_IMAGE || true

      docker build -t $OPENVIDU_IMAGE ./openviduback

      docker run -d --name $OPENVIDU_CONTAINER \
        -p 6080:6080 \
        --network backend-net \
        --restart unless-stopped \
        -e LIVEKIT_API_KEY=e106e106e106e106e106e106e106e106 \
        -e LIVEKIT_API_SECRET=e106e106e106e106e106e106e106e106 \
        $OPENVIDU_IMAGE

      # LiveKit(caddy-proxy) 이름 해석 위해 추가 네트워크 연결
      docker network connect openvidu-community $OPENVIDU_CONTAINER
    '''
   }
    }
  }

  post {
    failure {
      echo '❌ Build or deployment failed.'
    }
    success {
      echo '✅ Both backend and openviduback deployed successfully.'   }

    }
  }
```