A601(Pinokkio) 포팅 매뉴얼

1. 사용 도구

- **이슈 관리**: Jenkins
- **형상 관리**: GitLab
- **협업 툴**: MatterMost, Notion
- **CI/CD**: Jenkins

2. 개발 환경

백엔드 (Spring Boot, Gradle)

- **Spring Boot**: `3.2.7`
- **Spring Dependency Management**: `1.1.5`
- **Google Protobuf Plugin**: `0.8.19`
- **Java Language Version**: `21`
- **OpenVidu Java Client**: `2.20.0`
- **LiveKit Server**: `0.5.11`
- **Springdoc OpenAPI UI**: `2.0.2`
- **JSON Library**: `20230227`
- **Spring Cloud AWS**: `2.2.6.RELEASE`
- **JJWT API**: `0.11.5`

- **JJWT Impl:** `0.11.4`
- **JJWT Jackson:** `0.11.4`
- **gRPC Netty Shaded:** `1.57.2`
- **gRPC Protobuf:** `1.57.2`
- **gRPC Stub:** `1.57.2`
- **Apache Commons Math:** `3.6.1`
- **javax.annotation API:** `1.3.2`
- **Protobuf Java:** `3.23.4`
- **Protobuf Java Util:** `3.23.4`
- **Apache HttpClient 5:** `5.2.1`
- **Lombok**: 버전 명시되지 않음
- **MySQL Connector**: 버전 명시되지 않음
- **JUnit Platform Launcher**: 버전 명시되지 않음
- **Protobuf Compiler (protoc):** `3.23.4`
- **Protoc-gen-grpc-java:** `1.57.2`

프론트엔드 (React, npm/yarn)

- **React:** `^18.2.0`
- **Axios:** `^1.7.2`
- **Prettier:** `^2.8.8`

3. 환경 변수

- **BUCKET:** `pinokkio`
- **EMAIL:** `dltkdandl@naver.com`
- **EMAIL_PW:** `JD8K59PFV5C2`
- **JWT_SECRET:** `pinokkiopinokkiopinokkiopinokkiopinokkiopinokkio`
- **ENCRYPTION_KEY:** `XtRn3ABD4IwdM1EhiLsyJZaHwn04a9tEu3gbnQ9fP8E=`
- **LIVEKIT_API_KEY:** `devkey`

- **LIVEKIT_API_SECRET**: `pinokkiopinokkiopinokkiopinokkiopinokkiopinokkio`
- **REGION**: `ap-northeast-2`
- **S3_ACCESS_KEY**: `AKIA4MTWNYAX53KMP3WZ`
- **S3_SECRET_KEY**: `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`

4. 배포

Nginx Configuration

HTTP to HTTPS Redirection

```
server {
    listen       80;
    listen  [::]:80;
    server_name  i11a601.p.ssafy.io;

    # Redirect HTTP to HTTPS
    location / {
        return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl http2;
    server_name i11a601.p.ssafy.io;

    # SSL certificates
    ssl_certificate /etc/letsencrypt/live/i11a601.p.ssafy.i
o/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/i11a601.p.ssa
fy.io/privkey.pem;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers HIGH:!aNULL:!MD5;

    # Root location and SPA handling
    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # SPA 새로고침 처리
        try_files $uri $uri/ /index.html =404;
    }

    # WebSocket proxy for `/ws` location
    location /ws {
        proxy_pass http://localhost:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # API proxy and WebSocket for `/api/`
    location /api/ {
        proxy_pass http://localhost:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # wss(web-socket) 설정
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    # Fast server proxy for `/fast/`
    location /fast/ {
        proxy_pass http://localhost:5000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # CORS 설정
        add_header 'Access-Control-Allow-Origin' '*';
        add_header 'Access-Control-Allow-Methods' 'GET, POS
T, OPTIONS';
        add_header 'Access-Control-Allow-Headers' 'Origin,
Authorization, Accept, Content-Type, X-Requested-With';

        # WebSocket 설정
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";

        # OPTIONS 메소드에 대한 프리플라이트 요청 처리
        if ($request_method = 'OPTIONS') {
            add_header 'Access-Control-Allow-Origin' '*';
            add_header 'Access-Control-Allow-Methods' 'GET,
POST, OPTIONS';
            add_header 'Access-Control-Allow-Headers' 'Orig
in, Authorization, Accept, Content-Type, X-Requested-With';
            add_header 'Access-Control-Max-Age' 1728000;
            add_header 'Content-Type' 'text/plain charset=U
TF-8';
            add_header 'Content-Length' 0;
            return 204;
        }
    }

    # Error page handling
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
}
```

Docker Compose Configuration (version: '3')

```
version: '3'

services:

  # Backend Service
  backend:
    container_name: pinokkio-backend
    build:
      context: ./backend/pinokkio
      dockerfile: Dockerfile
    network_mode: host
    ports:
      - "8080:8080"
      - "3333:3333"
      - "3334:3334"
      - "465:465"
      - "587:587"
    environment:
      - SPRING_DATASOURCE_URL=jdbc:mysql://localhost:3306/p
inokkio?serverTimezone=Asia/Seoul
      - SPRING_DATASOURCE_USERNAME=root
      - SPRING_DATASOURCE_PASSWORD=ssafy
      - SPRING_JPA_HIBERNATE_DDL_AUTO=create
      - SPRING_REDIS_HOST=localhost
      - SPRING_REDIS_PORT=6380
      - SPRING_MAIL_USERNAME=${EMAIL}
      - SPRING_MAIL_PASSWORD=${EMAIL_PW}
      - JWT_SECRET=${JWT_SECRET}
      - BUCKET=${BUCKET}
      - S3_ACCESS_KEY=${S3_ACCESS_KEY}
      - S3_SECRET_KEY=${S3_SECRET_KEY}
      - REGION=${REGION}
      - LIVEKIT_API_KEY=${LIVEKIT_API_KEY}
      - LIVEKIT_API_SECRET=${LIVEKIT_API_SECRET}
      - ENCRYPTION_KEY=${ENCRYPTION_KEY}
    depends_on:
      - mysql
      - redis

  # Fast Pinokkio Service
  fast_pinokkio:
    container_name: fast-pinokkio-backend
    build:
      context: ./backend/fast_pinokkio
      dockerfile: Dockerfile
    network_mode: host
    ports:
      - "5000:5000"
    environment:
      - REDIS_HOST=localhost
      - REDIS_PORT=6380
    depends_on:
      - redis

  # Frontend Service
  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    network_mode: host
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /etc/letsencrypt:/etc/letsencrypt:ro
    depends_on:
      - backend

  # MySQL Service
  mysql:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: ssafy
      MYSQL_DATABASE: pinokkio
    network_mode: host
    ports:
      - "3306:3306"
    volumes:
      - mysql-data:/var/lib/mysql

  # Redis Service
  redis:
    image: redis:latest
    network_mode: host
    ports:
      - "6380:6380"

volumes:
  mysql-data:
```

5. Jenkins CI/CD 파이프라인

1. Jenkins Job 생성

- Jenkins 대시보드에서 "새 작업(New Item)"을 클릭하고 작업 이름을 입력한 후 "Freestyle 프로젝트"를 선택

2. 소스 코드 관리 설정

- "소스 코드 관리"에서 Git을 선택하고 다음과 같이 설정합니다:
  - **Repository URL**: `https://lab.ssafy.com/s11-webmobile1-sub2/S11P12A601.git`
  - **Branch Specifier**: `develop`

  - **Credentials**: 저장소 접근을 위한 자격증명 추가 (GitHub/GitLab Personal Access Token 또는 사용자/비밀번호)

```bash
#!/bin/bash

# 작업 디렉토리로 이동
cd /home/ubuntu/S11P12A601

# Git 저장소를 안전한 디렉토리로 설정
git config --global --add safe.directory /home/ubuntu/S11P12A601

# Jenkins 사용자가 디렉토리에 대한 권한을 갖도록 설정
sudo chown -R jenkins:jenkins /home/ubuntu/S11P12A601
sudo chmod -R 755 /home/ubuntu/S11P12A601

# Git 저장소 갱신
git pull <https://97choijw%40gmail.com:yUB7CuNM2zP7KoeYLpLQ@lab.ssafy.com/s11-webmobile1-sub2/S11P12A601.git> develop

# .env 파일 로드
if [ -f .env ]; then
    export $(cat .env | xargs)
fi

# 기존 컨테이너 중단 및 제거
sudo docker-compose down

# Docker 이미지 빌드 및 컨테이너 시작
sudo docker-compose up --build -d
```