DA:ON 포팅 메뉴얼

1. 아키텍처 개요

- **클라이언트**: 브라우저 → `https://da-on.store`
- **Nginx(리버스 프록시/정적 제공)**: `/var/www/da-on.store/current`에서 프론트 정적 파일 제공, 백엔드/FASTAPI/웹소켓 프록시
- **Backend (Spring Boot, Docker 컨테이너)**: 내부 포트 `8080` → Nginx가 `/api/` 경로로 프록시
- **FastAPI (옵션: STT/OCR, Docker 컨테이너)**: 내부 포트 `8000` / `9000` 등 → Nginx가 `/stt/` `/ocr/`로 프록시
- **Redis (Docker 컨테이너)**: 내부 네트워크 통신만
- **MySQL**: Docker 로컬
- **도커 네트워크**: `daon-network` (모든 컨테이너 연결)
- **SSL**: Let's Encrypt + certbot 자동 갱신

```
[Browser] ⇄ HTTPS(443) ⇄ [Nginx] ⇄ /api → [Spring Boot 8080]
                                  └─ /stt  → [FastAPI 8000]
                                  └─ /ocr  → [FastAPI 9000]
                                  └─ /ws   → [Spring WebSocket(STOMP)]
                          └─ / (정적) → /var/www/da-on.store/current
[Spring Boot] ⇄ [Redis]
[Spring Boot] ⇄ [MySQL(Docker)]
```

2. 개발 환경

운영체제 & 서버

- Ubuntu 22.04 LTS
- Nginx 1.18.0
- Certbot 2.1

백엔드

- Java 17
- Spring Boot 3.5.4
- Gradle 8.7.0

프론트엔드

- Node.js 20.3
- npm 10.2
- Vue 3 (Vite)

AI 서비스

- FastAPI 0.115.x (Python 3.10)
- Uvicorn 0.30.
- TorchAudio 2.3.x

데이터베이스

- MySQL 8.0
- Redis 7.0

인프라

- Docker 27.0
- Docker Compose v2
- GitLab Runner 18.0

도메인 & 보안

- `도메인: da-on.store`
- `SSL: Let's Encrypt`
- `방화벽: UFW`

3. 체크 리스트

- AWS EC2(Ubuntu 22.04) 퍼블릭 IP 및 SSH Key (`ubuntu` 사용자)
- DNS(가비아) 관리 권한
- GitLab 리포지토리 접근 토큰
- Docker Hub 계정 및 토큰
- 보안 그룹: 22, 80, 443 포트 오픈

4. EC2 서버 초기세팅

```
sudo timedatectl set-timezone Asia/Seoul
sudo apt update && sudo apt -y upgrade
sudo apt -y install curl unzip git ufw nginx

# 스왑 설정 (RAM 부족시)
sudo fallocate -l 2G /swapfile
sudo chmod 600 /swapfile
```

```
sudo mkswap /swapfile
sudo swapon /swapfile
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab

# 방화벽 설정
sudo ufw allow OpenSSH
sudo ufw allow 80
sudo ufw allow 443
sudo ufw --force enable
```

Docker 설치

```
curl -fsSL https://get.docker.com | sudo sh
sudo usermod -aG docker $USER
newgrp docker

sudo apt -y install docker-compose-plugin

# 네트워크 생성
docker network create daon-network || true
```

Nginx 설치

- nginx 설치

```
sudo apt install nginx
```

- http, https 방화벽 허용

```

sudo ufw allow 'Nginx Full'
```

Certbot(SSL) 설치

1. **Certbot 설치**

Certbot을 사용해 SSL 인증서를 발급받기 위해 Certbot 패키지를 설치합니다.

```
sudo snap install --classic certbot
```

2. **Certbot 인증서 발급 과정 수행**

도메인 소유권 확인을 위해 수동 모드로 SSL 인증서를 발급받습니다.

```
sudo certbot certonly --manual
```

- 도메인 이름을 입력하고, IP 로그 동의 여부를 선택합니다.

3. **도전 파일 생성 및 Nginx 설정 확인**

   도전 파일을 생성하고, 이를 제공할 수 있도록 Nginx 설정을 확인하여 SSL 인증서를 적용합니다.

4. **Nginx 재시작**

   도전 파일 설정이 완료되면 Nginx를 재시작하여 설정을 반영합니다.

```
sudo nginx -t
sudo systemctl reload nginx
```

디렉토리 구성

```
-프론트 환경
sudo mkdir -p /var/www/da-on.store/releases
sudo mkdir -p /var/www/da-on.store/shared

-백엔드 환경
sudo mkdir -p /opt/daon/env
sudo mkdir -p /opt/daon/logs
sudo chown -R ubuntu:ubuntu /var/www /opt/daon
```

5. DNS&SSL

DNS 설정

- A 레코드: `da-on.store` → EC2 IP
- A 레코드: `www.da-on.store` → EC2 IP
- A 레코드: `i13a706.p.ssafy.io` → EC2 IP

SSL 설정

```
sudo apt -y install certbot python3-certbot-nginx
sudo certbot --nginx \
  -d da-on.store -d www.da-on.store \
  -d i13a706.p.ssafy.io \
  --redirect --agree-tos -m admin@da-on.store
```

6. 환경변수

경로: `/opt/daon/env/`

backend.env

```
# Database
DB_URL=jdbc:mysql://i13a706.p.ssafy.io:3306/daon_db?serverTimezone=Asia/Seoul&characterEncoding=UTF-8
DB_USERNAME=ssafy
DB_PASSWORD=ssafy

# GMS API
GMS_API_KEY=S13P11A06-a891d0e-f242-4d11-80d6-3b70c39e46
GMS_API_URL=https://gms.ssafy.io/gmsapi/api.openai.com/v1/chat/completions
```

```
GMS_IMAGE_API_URL=https://gms.ssafy.io/gmsapi/api.openai.com/v1/images/generations

# AWS Credentials
AWS_ACCESS_KEY_ID=AKIAXKJSFOVNY2D3PV
AWS_SECRET_ACCESS_KEY=ZHDbXQgBCWL/XSLF38jVvVE7qBWe/Lz+bx/Dc

# Google OAuth2
GOOGLE_CLIENT_ID=222860421159-nagruv3rnmm40s4k8qebtu1mqjn4u.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=your_google_client_secret=GOCSPX-20K83JjKJ
GOOGLE_REDIRECT_URI=http://localhost:8080/login/oauth2/code/google
```

7. Nginx설정

파일 경로: `/etc/nginx/sites-available/default`

```
# ================================
# HTTP → HTTPS 리디렉션 (포트 80)
# ================================
server {
    if ($host = i13a706.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name i13a706.p.ssafy.io www.i13a706.p.ssafy.io da-on.store www.da-on.store;

    return 301 https://$host$request_uri;
}
```

```
# ================================
# HTTPS 서버 (포트 443)
# ================================
server {
    listen 443 ssl;
    server_name i13a706.p.ssafy.io www.i13a706.p.ssafy.io da-on.store ww
w.da-on.store;

    # --- SSL (Certbot) ---
    ssl_certificate /etc/letsencrypt/live/i13a706.p.ssafy.io/fullchain.pem; # m
anaged by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i13a706.p.ssafy.io/privkey.pem; #
managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    # --- 정적 파일 루트 ---
    root /var/www/i13a706.p.ssafy.io/current;
    index index.html;

    # ================================
    # Spring Boot API 프록시: /api/
    # ================================
    location /api/ {
        # --- CORS 동적 허용(두 오리진 화이트리스트) ---
        set $cors_origin "";
        if ($http_origin = https://da-on.store)       { set $cors_origin $http_origi
n; }
        if ($http_origin = https://www.da-on.store)   { set $cors_origin $http_or
igin; }

                add_header Access-Control-Allow-Origin $cors_origin always;
        add_header Vary "Origin" always;
        add_header Access-Control-Allow-Credentials "true" always;
        add_header Access-Control-Allow-Headers "Authorization,Content-Ty
pe,Accept,Origin,X-Requested-With" always;
        add_header Access-Control-Allow-Methods "GET,POST,PUT,PATCH,D
ELETE,OPTIONS" always;
```

```
        if ($request_method = OPTIONS) { return 204; }

        proxy_pass http://localhost:8080/api/;
        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;
    }

    # ================================
    # FastAPI STT 프록시: /stt/
    # ================================
    location /stt/ {
        # --- CORS 동적 허용(두 오리진 화이트리스트) ---
        set $cors_origin "";
        if ($http_origin = https://da-on.store)       { set $cors_origin $http_origi
n; }
        if ($http_origin = https://www.da-on.store)   { set $cors_origin $http_or
igin; }

        add_header Access-Control-Allow-Origin $cors_origin always;
        add_header Vary "Origin" always;
        add_header Access-Control-Allow-Credentials "true" always;
        add_header Access-Control-Allow-Headers "Authorization,Content-Ty
pe,Accept,Origin,X-Requested-With" always;
        add_header Access-Control-Allow-Methods "GET,POST,PUT,PATCH,D
ELETE,OPTIONS" always;

        if ($request_method = OPTIONS) { return 204; }
              if ($request_method = OPTIONS) { return 204; }

        proxy_pass http://127.0.0.1:8000/;
        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;
    }

    # ================================
    # STOMP + SockJS 웹소켓: /ws/
    # ================================
    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_read_timeout 3600;
        proxy_send_timeout 3600;

        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_buffering off;
    }

    # ================================
    # 정적 SPA 라우팅
    # ================================
    location / {
        try_files $uri $uri/ /index.html;
    }
}
```

8.프론트엔드 배포

```
mkdir -p /var/www/da-on.store/releases/$RELEASE
scp dist-$RELEASE.tar.gz ubuntu@<EC2>:/var/www/da-on.store/releases/$RELEASE/
ssh ubuntu@<EC2> "cd /var/www/da-on.store/releases/$RELEASE && tar -xzf dist-$RELEASE.tar.gz && ln -sfn /var/www/da-on.store/releases/$RELEASE/dist /var/www/da-on.store/current && sudo nginx -t && sudo systemctl reload nginx"
```

FastAPI 배포

- **파이썬 설치**

```
- 파이썬 설치

sudo apt update

# python3, pip3, venv 설치
sudo apt install -y python3 python3-pip python3-venv python-is-python3

# 버전 확인
python3 --version
pip3 --version

- 폴더생성 + 권한

# 프로젝트 배치 경로 생성
sudo mkdir -p /srv/fastapi-stt

# ubuntu 계정이 해당 디렉토리 접근 가능하게 소유권 변경
sudo chown ubuntu:ubuntu /srv/fastapi-stt

# 경로 이동
cd /srv/fastapi-stt
```

EC2에 FastAPI 프로젝트 git clone

해동 폴더 로 이동 후 레포지토리 깃 클론

Python 가상환경 생성 & 패키지 설치

💡

1) 가상환경 생성

```
bash
복사편집
# AI_Pronounciation 폴더에서 실행
cd /srv/fastapi-stt/S13P11A706/AI_Pronounciation

# venv 생성
python3 -m venv venv
```

2) 가상환경 활성화

```
bash
복사편집
source venv/bin/activate
```

> 활성화되면 터미널 프롬프트 앞에 (venv)가 붙어요.

3) pip 최신화 & 패키지 설치

```
bash
복사편집
pip install --upgrade pip
pip install -r requirements.txt
```

FastAPI 서버 임시 실행 테스트

💡 이 단계는 EC2에서 FastAPI 서버가 정상적으로 뜨는지, Spring Boot와도 연결이 되는지 확인하는 과정이에요.

아직 서비스 등록이나 Nginx 연동은 안 하고, 그냥 uvicorn으로 바로 실행

---

1) 가상환경 활성화

```
bash
복사편집
cd /srv/fastapi-stt/S13P11A706/AI_Pronounciation
source venv/bin/activate
```

2) uvicorn 실행

```
bash
복사편집
uvicorn main:app --host 0.0.0.0 --port 8000
```

> --reload는 개발용 옵션이라 운영 테스트에서는 빼고 실행해요.
>
> 이렇게 실행하면 EC2의 8000 포트로 접근 가능해집니다.

6단계) Nginx 리버스 프록시 설정

💡 이 단계에서 FastAPI가 8000에서 뜨고, Nginx가 `/stt`로 요청을 프록시합니다.

1. **Nginx 설정 파일 열기**

```
bash
복사편집
sudo nano /etc/nginx/sites-available/default
```

1. **server 블록 안에 location 추가**

```
nginx
복사편집
server {
    listen 80;
    server_name i13a706.p.ssafy.io;

    # Spring Boot 기본 API 프록시 예시
    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    # FastAPI STT 서비스 프록시
    location /stt/ {
        proxy_pass http://127.0.0.1:8000/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

1. **Nginx 테스트 & 재시작**

```
bash
복사편집
```

```
sudo nginx -t
sudo systemctl reload nginx
```

7단계) FastAPI를 systemd 서비스로 등록

💡 이걸 하면 `(venv)`를 매번 켜고 `uvicorn` 실행할 필요 없이 항상 살아있게 됩니다.

1) 서비스 파일 생성

```
bash
복사편집
sudo nano /etc/systemd/system/fastapi-stt.service
```

2) 아래 내용 붙여넣기

> 경로는 현재 프로젝트 기준으로 작성했어요.
>
> 가상환경 경로(`venv/bin/uvicorn`)와 `main:app` 위치 꼭 맞춰야 합니다.

```
ini
복사편집
[Unit]
Description=FastAPI STT Service
After=network.target

[Service]
User=ubuntu
WorkingDirectory=/srv/fastapi-stt/S13P11A706/AI_Pronounciation
Environment="PATH=/srv/fastapi-stt/S13P11A706/AI_Pronounciation/venv/bin"
ExecStart=/srv/fastapi-stt/S13P11A706/AI_Pronounciation/venv/bin/uvicorn main:app --host 127.0.0.1 --port 8000
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
```

3) 서비스 등록 & 실행

```
bash
복사편집
# 변경 사항 적용
sudo systemctl daemon-reload

# 서비스 등록(부팅 시 자동 실행)
sudo systemctl enable fastapi-stt

# 서비스 시작
sudo systemctl start fastapi-stt

# 상태 확인
sudo systemctl status fastapi-stt
```

4) 실시간 로그 확인 (문제 있을 때)

```
bash
복사편집
sudo journalctl -u fastapi-stt -f
```

이렇게 하면 EC2 재부팅 후에도 FastAPI가 자동 실행되고,

Nginx `/stt` 경로를 통해 항상 접근할 수 있습니다.

9.GitLab CI/CD

파이프라인

```
# .gitlab-ci.yml — BE(도커) + FE(정적배포), deploy 브랜치 전용

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH == "deploy"'
    - when: never

stages: [build_be, deploy_be, build_fe, package_fe, deploy_fe]

default:
  interruptible: true
  tags: ["daon"]  # 모든 잡을 EC2 Shell 러너에서 실행

.ssh_setup: &ssh_setup
  - mkdir -p ~/.ssh
  - echo "$SSH_PRIVATE_KEY" > ~/.ssh/id_rsa
  - chmod 600 ~/.ssh/id_rsa
  - ssh-keyscan -H "$SSH_HOST" >> ~/.ssh/known_hosts

# ===== Backend =====
build-and-push-be:
  stage: build_be
  rules:
    - if: '$CI_COMMIT_BRANCH == "deploy"'
      changes:
        - BE/**/*
  before_script:
    - docker -v
  script:
    - echo "=== Docker Hub Login ==="
    - docker login -u "$DOCKERHUB_USERNAME" -p "$DOCKERHUB_TOKE
N"
    - echo "=== Docker Build ==="
    - docker build -t "$IMAGE_NAME:latest" BE
```

```
    - echo "=== Docker Push ==="
    - docker push "$IMAGE_NAME:latest"

deploy-backend:
  stage: deploy_be
  needs: ["build-and-push-be"]
  rules:
    - if: '$CI_COMMIT_BRANCH == "deploy"'
      changes:
        - BE/**/*
  script:
    - set -euxo pipefail
    - docker stop "$CONTAINER_NAME" || true
    - docker rm "$CONTAINER_NAME" || true
    - docker pull "$IMAGE_NAME:latest"
    - test -f "$ENV_FILE"
    - docker run -d --name "${CONTAINER_NAME}" --network daon-networ
k --env-file "${ENV_FILE}" -p "${HOST_PORT}:${CONTAINER_PORT}" --re
start=always "${IMAGE_NAME}:latest"

# ===== Frontend =====
build-fe:
  stage: build_fe
  rules:
    - when: never
  script:
    - echo "Skipping build-fe job (disabled)"

package-fe:
  stage: package_fe
  rules:
    - when: never
  script:
    - echo "Skipping package-fe job (disabled)"

deploy-frontend:
  stage: deploy_fe
  environment:
```

```
    name: production
    url: https://i13a706.p.ssafy.io
  rules:
    - if: '$CI_COMMIT_BRANCH == "deploy"'
      changes:
        - FE/my-vue-app/**/*
  before_script:
    - *ssh_setup
    - set -euxo pipefail
    - node -v
    - npm -v
    - cd FE/my-vue-app
  script:
    # 1) 로컬 빌드
    - npm ci
    - npm run build

    # 2) 릴리스 식별자 & 패키징
    - export RELEASE="${CI_PIPELINE_IID}-${CI_COMMIT_SHORT_SHA}"
    - tar -czf "dist-${RELEASE}.tar.gz" -C dist .

    # 3) 서버 경로 설정
    - export TARGET="${SSH_TARGET_DIR}"
    - export RELEASE_DIR="${TARGET}/releases/${RELEASE}"

    # 4) 서버 폴더 생성 + 권한 부여 후 업로드
    - ssh ${SSH_USER}@${SSH_HOST} "sudo mkdir -p '${RELEASE_DIR}' &
& sudo chown -R ${SSH_USER}:${SSH_USER} '${TARGET}'"
    - scp "dist-${RELEASE}.tar.gz" ${SSH_USER}@${SSH_HOST}:"${RELEA
SE_DIR}/"
    - ssh ${SSH_USER}@${SSH_HOST} "cd '${RELEASE_DIR}' && tar -xzf di
st-${RELEASE}.tar.gz && rm dist-${RELEASE}.tar.gz"

    # 5) 심볼릭 링크 스위칭
    - ssh ${SSH_USER}@${SSH_HOST} "ln -sfn '${RELEASE_DIR}' '${TARGE
T}/current'"

    # 6) 퍼미션 & Nginx 재로드
```

```
    - ssh ${SSH_USER}@${SSH_HOST} "sudo chown -R www-data:www-data '${TARGET}' || true"
    - ssh ${SSH_USER}@${SSH_HOST} "sudo nginx -t && sudo systemctl reload nginx"

  allow_failure: false
```

10.도커 파일

```
# 1단계: 빌드용 이미지
FROM gradle:8.7.0-jdk17 AS build

# 작업 디렉토리 설정
WORKDIR /app

# 의존성 캐시를 위해 gradle 파일만 먼저 복사
COPY build.gradle .
COPY settings.gradle .
RUN gradle dependencies --no-daemon
# 나머지 소스 복사 후 빌드
COPY . .
RUN gradle build --no-daemon -x test

# 2단계: 경량화된 JRE 이미지로 실행
FROM eclipse-temurin:17-jdk
WORKDIR /app

RUN apt-get update && apt-get install -y \
    tesseract-ocr \
    libtesseract-dev \
    libleptonica-dev \
 && rm -rf /var/lib/apt/lists/*



# 빌드 결과 jar 파일 복사 (target 폴더 내 jar 이름 확인 필요)
COPY --from=build /app/build/libs/*.jar app.jar
```

```
# 포트 노출
EXPOSE 8080

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

#실행하려면
# docker build -t my-spring-app .
# docker run -p 8080:8080 my-spring-app
```

11.백엔드 설정(Spring Boot)

application-prod.yml

```
spring:
  application:
    name: BE

  datasource:
    url: ${DB_URL}
    username: ${DB_USERNAME}
    password: ${DB_PASSWORD}
    driver-class-name: com.mysql.cj.jdbc.Driver

  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true
    properties:
      hibernate:
        format_sql: true
        jdbc:
          time_zone: Asia/Seoul
    open-in-view: false
```

```
  jackson:
    time-zone: Asia/Seoul
    serialization:
      write-dates-as-timestamps: false

  data:
    redis:
      host: redis
      port: 6379

  security:
    oauth2:
      client:
        registration:
          google:
            client-id: ${GOOGLE_CLIENT_ID}
            client-secret: ${GOOGLE_CLIENT_SECRET}
            redirect-uri: ${GOOGLE_REDIRECT_URI}
            scope:
              - email
              - profile
        provider:
          google:
            authorization-uri: https://accounts.google.com/o/oauth2/v2/auth
            token-uri: https://oauth2.googleapis.com/token
            user-info-uri: https://www.googleapis.com/oauth2/v3/userinfo
            user-name-attribute: sub

    servlet:
      multipart:
        max-file-size: 15MB
        max-request-size: 15MB

gms:
  api:
    key: ${GMS_API_KEY}
    url: ${GMS_API_URL}
    model: gpt-4o
```

```
  image:
    api:
      url: ${GMS_IMAGE_API_URL}

cloud:
  aws:
    region: ap-northeast-2
    stack:
      auto: false
    s3:
      bucket: daon-image
    credentials:
      accessKey: ${AWS_ACCESS_KEY_ID}
      secretKey: ${AWS_SECRET_ACCESS_KEY}

frontend:
  origin: http://localhost:5173

tesseract:
  path: "/usr/share/tesseract-ocr/5/tessdata"

logging:
  level:
    root: INFO
    com.daon.be: DEBUG
    org.springframework.web: DEBUG
```