포팅 메뉴얼

▼ 목차

1. Directory Structure

- docker compose / k8s 를 사용하는 웹 프로젝트로 `agent-operator/` / `agent/` / `backend/fastapi/` / `frontend/react/` / `k8s/agent-operator/` / `nginx/` 6개의 폴더로 구분.
- CI-CD 시 docker compose up 을 통해 웹서비스 배포
- agent-operator 와 agent 변경사항 있을 시 도커 허브에 이미지 빌드해 업로드
- k8s에 agent-operator 이미지 받아와 backend 서 deploy 로 요청 왔을 때에, agent deploy, pod 생성

```
📦 S12P31B107
├ 📜 README.md
├ 📂 agent
|  ├ 📜 Dockerfile
|  ├ 📂 app
|  |  └ 📜 main.py
|  └ 📜 pyproject.toml
├ 📂 agent-operator
```

```
│  ├ 📜 Dockerfile
│  ├ 📂 app
│  │  ├ 📜 main.py
│  │  ├ 📜 config.py
│  │  └ 📜 deploy.py
│  └ 📜 requirements.txt
├ 📂 backend
│  └ 📂 fastapi
│     ├ 📜 Dockerfile
│     ├ 📂 app
│     │  ├ 📜 main.py
│     │  ├ 📂 core
│     │  │  ├ 📜 config.py
│     │  │  ├ 📜 create_pod.py
│     │  │  └ 📜 security.py
│     │  ├ 📂 routers
│     │  │  ├ 📜 chat_bot.py
│     │  │  └ 📜 nosql_user.py
│     │  ├ 📂 crud
│     │  │  └ 📜 conversation.py
│     │  └ 📂 models
│     │     └ 📜 mcp_nosql.py
│     └ 📜 requirements.txt
├ 📂 frontend
│  └ 📂 react
│     ├ 📜 Dockerfile
│     ├ 📜 package.json
│     ├ 📂 src
│     │  ├ 📜 App.tsx
│     │  ├ 📂 pages
│     │  │  ├ 📂 login
│     │  │  │  └ 📜 page.tsx
│     │  │  ├ 📂 chat
│     │  │  │  └ 📜 page.tsx
│     │  │  └ 📂 mcp-setup
│     │  │     └ 📜 page.tsx
│     │  └ 📂 api
│     │     └ 📜 mcpService.ts
│     └ 📜 vite.config.ts
├ 📂 k8s
│  └ 📂 agent-operator
│     ├ 📜 00-namespace.yaml
│     ├ 📜 01-agent-operator-rbac.yaml
│     ├ 📜 02-agent-operator-deploy.yaml
│     └ 📜 03-agent-operator-svc.yaml
├ 📂 nginx
│  ├ 📜 Dockerfile
│  └ 📜 nginx.conf
└ 📜 docker-compose.yml
```

2. Tech Stack Version

01. EC2

001. Port & System Configuration

Skill	Port 번호
Nginx	80 / 443
React	5173 / 19000~19002
Jenkins	8080
SSH	22
FastAPI	8000
agent-operator FastAPI	8002
agent FastAPI	8001
k8s	30000:32767/tcp
Kubernetes API Server	6443

항목	내용
OS	Ubuntu 22.04.4 LTS (Jammy Jellyfish)
Kernel	6.8.0-1021-aws
Architecture	x86_64
CPU	Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz (4 cores, 1 thread/core)
Memory	총 15GiB (사용 2.7GiB / 사용 가능 12GiB)
Docker Engine	26.1.3
Docker Client	26.1.3
Docker Compose	v2.35.1
Docker Containers	4개 실행 중 (nginx, react, fastapi, jenkins)
Kubernetes	v1.29.15 (control-plane)
Container Runtime	containerd 1.7.24

002. Primary Packages

✅ FastAPI 주요 패키지

- **fastapi**: 타입 기반 비동기 Python 웹 프레임워크
- **uvicorn**: ASGI 서버
- **pydantic / pydantic-settings**: 데이터 검증 및 설정 관리
- **passlib / bcrypt**: 비밀번호 해싱
- **python-jose**: JWT 인증
- **python-dotenv**: `.env` 로딩
- **motor / pymongo**: MongoDB 드라이버
- **httpx / anyio / starlette**: 비동기 HTTP 및 서버 동작 기반
- **openai / openai-agents**: GPT API 호출용
- **kubernetes**: K8s 클러스터 제어용
- 기타: `email_validator`, `cryptography`, `cffi`, `rsa`

✅ React (Vite + Tailwind + Shadcn UI)

- **react / react-dom**: UI 프레임워크
- **react-router-dom**: 라우팅
- **axios**: HTTP 요청
- **tailwindcss / @shadcn/ui**: 스타일링
- **lucide-react / react-hot-toast / react-markdown**: UI 구성 요소
- **vite**: 번들러
- **typescript**: 정적 타입 지원
- **eslint**: 린팅

3. Dockerfile & docker-compose.yml

00. Web docker-compose.yml

```
services:
 fastapi:
  build:
   context: ./backend/fastapi
   dockerfile: Dockerfile
  container_name: fastapi
  ports:
   - "8000:8000"
  volumes:
   - /home/ubuntu/.kube:/root/.kube:ro
  environment:
   # Agent Operator에서 던질 사용자별 ENV
   - OPENAI_API_KEY=${OPENAI_API_KEY}
   - NOTION_API_TOKEN=${NOTION_API_TOKEN}

   # MongoDB 연결 정보
   - MONGODB_URL=${MONGODB_URL}
   - MONGO_DB_PASSWORD=${MONGO_DB_PASSWORD}
   - MONGO_DB_USER_NAME=${MONGO_DB_USER_NAME}
```

```
    - DATABASE_NAME=${DATABASE_NAME}

    # JWT 설정
    - SECRET_KEY=${SECRET_KEY}
    - ALGORITHM=${ALGORITHM}
    - ACCESS_TOKEN_EXPIRE_MINUTES=${ACCESS_TOKEN_EXPIRE_MINUTES}
    - API_SECRET_KEY=${API_SECRET_KEY}

    # GMS 설정
    - GMS_API_KEY=${GMS_API_KEY}

    # AGENT 설정
    - AGENT_URL=${AGENT_URL}
    - DEPLOY_SERVER_URL=${DEPLOY_SERVER_URL}

    # CORS 설정
    - CORS_ORIGINS=${CORS_ORIGINS}

  react:
    build:
      context: ./frontend/react
      dockerfile: Dockerfile
    container_name: react
    ports:
      - "5173:5173" # Vite dev server 기본 포트
    environment:
      - HOST=0.0.0.0 # Vite 외부 접속 허용
    command: npm run dev


  nginx:
    build:
      context: ./nginx
      dockerfile: Dockerfile
    container_name: nginx
    depends_on:
      - fastapi
    ports:
      - "80:80"
      - "443:443"
    volumes:
      # SSL 인증서 마운트 (EC2 호스트 → 컨테이너 내부)
      - /etc/letsencrypt/live/p.ssafy.io/fullchain.pem:/etc/nginx/ssl/fullchain.pem:ro
      - /etc/letsencrypt/live/p.ssafy.io/privkey.pem:/etc/nginx/ssl/privkey.pem:ro
```

01. Fastapi

```
FROM python:3.13

WORKDIR /app

# 기본 툴 설치
RUN apt-get update && apt-get install -y \
    curl build-essential git gnupg ca-certificates \
  && rm -rf /var/lib/apt/lists/*

# kubectl 설치
```

```
RUN curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl" \
    && install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl \
    && rm kubectl

# requirements.txt 복사 후 설치
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# 앱 복사
COPY app/ ./app/

EXPOSE 8000

# 개발용 reload 포함 실행 명령
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000", "--reload"]
```

02. React

```
FROM node:22.13.1

# 시스템 타임존, git 등 필요한 패키지 설치 (선택)
RUN apt-get update && apt-get install -y git

# 앱 작업 디렉토리
WORKDIR /app

# 종속성 먼저 복사 후 설치
COPY package*.json ./
RUN npm install

# 전체 코드 복사
COPY . .

# 포트 노출 (Vite 기본: 5173)
EXPOSE 5173
```

03. nginx

```
# 1. HTTP 요청 → HTTPS로 리디렉션
server {
    listen 80;
    server_name k12b107.p.ssafy.io;

    # 모든 요청을 HTTPS로 리다이렉트
    return 301 https://$host$request_uri;
}

# 2. HTTPS 요청 처리
server {
    listen 443 ssl;
    # listen 443 ssl http2;
    server_name k12b107.p.ssafy.io;

    # SSL 인증서 경로
    ssl_certificate     /etc/nginx/ssl/fullchain.pem;
    ssl_certificate_key /etc/nginx/ssl/privkey.pem;
```

```
    ssl_session_cache   shared:SSL:10m;
    ssl_protocols       TLSv1.2 TLSv1.3;
    ssl_ciphers         HIGH:!aNULL:!MD5;

    # 공통 프록시 설정
    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;

    # React 앱
    location / {
        proxy_pass http://react:5173/;
        proxy_set_header Upgrade   $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    # FastAPI
    location /api/ {
        rewrite ^/api/(.*) /$1 break;
        proxy_pass http://fastapi:8000/;
    }

    # WebSocket 경로들
    location /hot {
        proxy_pass http://react:5173/hot;
        proxy_set_header Upgrade   $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    location /message {
        proxy_pass http://react:5173/message;
        proxy_set_header Upgrade   $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    # Vite 클라이언트 하위 경로
    location ~ ^/@vite/ {
        proxy_pass http://react:5173;
        proxy_http_version 1.1;
        proxy_set_header Upgrade   $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    # node_modules, src, favicon 처리
    location ~ ^/(node_modules|src)/ {
        proxy_pass http://react:5173;
    }

    location = /favicon.ico {
        proxy_pass http://react:5173/favicon.ico;
    }
}
```

04. agent-operator

```
FROM python:3.10-slim

WORKDIR /app

COPY ./app ./app
COPY ./requirements.txt .

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

CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8002"]
```

05. agent

```
FROM python:3.13

WORKDIR /app

# 기본 툴 설치 + Node.js 설치 (npx 포함)
RUN apt-get update && apt-get install -y \
    curl build-essential git gnupg ca-certificates \
 && curl -fsSL https://deb.nodesource.com/setup_20.x | bash - \
 && apt-get install -y nodejs \
 && curl -LsSf https://astral.sh/uv/install.sh | sh \
 && ln -s /root/.local/bin/uv /usr/local/bin/uv \
 && rm -rf /var/lib/apt/lists/*

# “글로벌 패키지”로 Node 모듈을 설치
RUN npm install -g @suekou/mcp-notion-server
RUN npm install -g @zereight/mcp-gitlab
RUN npm install -g @winterjung/mcp-korean-spell
RUN npm install -g duckduckgo-mcp-server
RUN npm install -g @modelcontextprotocol/server-sequential-thinking
RUN npm install -g @modelcontextprotocol/server-github
RUN npm install -g @openbnb/mcp-server-airbnb
RUN npm install -g figma-developer-mcp

### python 기반
# 논문 검색 서버 소스 클론 후 설치
RUN git clone https://github.com/openags/paper-search-mcp.git /srv/paper-search-mcp
WORKDIR /srv/paper-search-mcp
RUN pip install --no-cache-dir .
# 체스 서버 소스 클론 후 설치
# RUN git clone https://github.com/jiayao/mcp-chess.git /srv/mcp-chess
# WORKDIR /srv/mcp-chess
# RUN pip install --no-cache-dir .
# dart 서버 소스 클론 후 설치
RUN git clone https://github.com/2geonhyup/dart-mcp.git /srv/dart-mcp
WORKDIR /srv/dart-mcp
RUN pip install --no-cache-dir .

### npm 로컬
# 카카오 지도 서버 소스 클론 후 설치
RUN git clone https://github.com/cgoinglove/mcp-server-kakao-map.git /srv/mcp-server-kakao-map
```

```
WORKDIR /srv/mcp-server-kakao-map
RUN npm install
RUN npm run build


# Python 패키지 설치
WORKDIR /app
RUN uv pip install --system \
    fastapi uvicorn[standard] openai-agents openai


# 앱 복사
COPY app/ ./app/

EXPOSE 8001

# 개발용 reload 포함 실행 명령
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8001"]
```

4. How to Run

01. Web (link)

1. EC2 에 Jenkins 설치
2. Job 생성 및 Gitlab 에 올라와 있는 repo CI-CD 설정
3. Jenkins 크레덴셜 통해 .env 파일 각각의 경로에 삽입
4. agent, agent-operator

- Jenkins Job 실행 시 GitLab 커밋 해시 기반으로 agent 및 agent-operator 이미지를 각각 빌드
- 최신 태그(`latest`)와 커밋 기반 태그(`<short_sha>`)를 함께 생성
- Docker Hub에 로그인 후 두 이미지를 모두 푸시
- 이후 `kubectl`을 사용해 `agent-operator` 리소스들을 Kubernetes에 배포 (Namespace, RBAC, Deployment 등)
- 기존 `agent-operator` 디플로이먼트에는 `set image` 명령어를 통해 롤링 업데이트 적용
  → 무중단 방식으로 새 이미지 배포

```
#!/usr/bin/env bash
set -euo pipefail

################################################
# 0. 변수
SHORT_SHA="${GIT_COMMIT:0:7}"          # 짧은 SHA
AGENT_IMAGE="$DOCKER_USER/agent:$SHORT_SHA"
AGENT_OPERATOR_IMAGE="$DOCKER_USER/agent-operator:$SHORT_SHA"

################################################
# 1. agent 이미지 빌드 & 푸시
docker build -t "$AGENT_IMAGE" -t "$DOCKER_USER/agent:latest" ./agent

################################################
# 2. agent-operator 이미지 빌드 & 푸시
docker build -t "$AGENT_OPERATOR_IMAGE" -t "$DOCKER_USER/agent-operator:latest" ./agent-operator

################################################
# 3. Docker Hub 로그인 및 Push
echo "$DOCKER_PASS" | docker login -u "$DOCKER_USER" --password-stdin

docker push "$AGENT_IMAGE"
docker push "$DOCKER_USER/agent:latest"

docker push "$AGENT_OPERATOR_IMAGE"
docker push "$DOCKER_USER/agent-operator:latest"

docker logout

################################################
# 4. agent-operator Kubernetes 배포
# agent-operator Deployment 에 새 이미지 적용
# namespace 먼저
kubectl apply -f k8s/agent-operator/00-namespace.yaml

# RBAC·SA 등 한번에
kubectl apply -f k8s/agent-operator/

# 그 뒤에 이미지만 롤링 업데이트
kubectl -n agent-env set image deployment/agent-operator \
  operator="$AGENT_OPERATOR_IMAGE"
```

```
################################################
# 5. .env 교체
rm -f .env
cp "$PROJECT_ENV" .env

rm -f frontend/react/.env
cp "$FRONTEND_ENV" frontend/react/.env

################################################
# 6. 웹서비스(Compose) 무중단 롤링
#docker compose pull
#docker compose up -d --build --force-recreate

docker compose down
docker compose up -d --build
# run --rm react sh -c "ls -al /app"

echo "✅ 배포 완료 → Agent: $AGENT_IMAGE, Agent-Operator: $AGENT_OPERATOR_IMAGE"
```

- env 관련은 아래 기입해 두었습니다.

환경변수 설명

🖥️ 환경 변수 (`./.env`)

구분	변수명	예시 값
MongoDB	`MONGODB_URL`	mongodb+srv://S12P31B107:JODs7HbsLN@ssafy.ngivl.mongodb.net/S12P31B107?authSource=admin
	`MONGO_DB_USER_NAME`	S12P31B107
	`MONGO_DB_PASSWORD`	JODs7HbsLN
	`DATABASE_NAME`	S12P31B107
JWT 설정	`SECRET_KEY`	c9d63f62ddffe45756ef11246059be218309079b6a2ef1b409529b52a23287ff
	`API_SECRET_KEY`	aa98fc4572549f4e8f9da0bcf7d2fc2b29671d08e0d5415d7c5e0825deaa433b
	`ALGORITHM`	HS256
	`ACCESS_TOKEN_EXPIRE_MINUTES`	30
Kubernetes 관련	`DEPLOY_SERVER_URL`	http://3.35.167.118:30082
	`AGENT_URL`	http://localhost:8001/agent-query
CORS 설정	`CORS_ORIGINS`	["http://localhost:5173", "http://k12b107.p.ssafy.io", ...]
토큰 설정	`GMS_API_KEY`	S12P31B107-377ba571-638a-4e3c-959e-b29e3b00e7df
	`OPENAI_API_KEY`	sk-...

🔷 Frontend 관련 환경 변수 (`./frontend/react/.env`)

환경 변수	설정 값	설명
`VITE_API_BASE`	https://k12b107.p.ssafy.io	웹 서버 DNS

03. Jenkins를 활용한 자동 CI/CD

📥 Jenkins 컨테이너 생성 및 실행

```
# Jenkins 컨테이너를 EC2의 Docker 환경을 공유하는 방식으로 실행
cd /home/ubuntu && mkdir jenkins-data

docker run -d \
  -p 8080:8080 \
  -p 50000:50000 \
  -v /home/ubuntu/jenkins-data:/var/jenkins_home \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v /usr/bin/docker:/usr/bin/docker \
  -v /usr/libexec/docker/cli-plugins:/usr/libexec/docker/cli-plugins \
  --name jenkins \
  jenkins/jenkins:2.501

# 인증서 등록 및 업데이트 경로 변경
cd /home/ubuntu/jenkins-data
mkdir update-center-rootCAs
wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-center/rootCA/update-center.crt -O ./update-center-rootCAs/u

sudo sed -i 's#https://updates.jenkins.io/update-center.json#https://raw.githubusercontent.com/lework/jenkins-updat

docker restart jenkins
```

🔧 Jenkins 초기 설정

```
# EC2 Public DNS 확인
curl http://169.254.169.254/latest/meta-data/public-hostname

# Jenkins 초기 비밀번호 확인
docker exec -it jenkins cat /var/jenkins_home/secrets/initialAdminPassword
```

- 초기 비밀번호로 Jenkins 로그인 후 필요한 플러그인 설치
  오류 발생 시 `/jenkins-data/plugins` 폴더 삭제 후 수동 설치 가능

```
**기본 로그인 정보**
- ID: admin
- PW: B107_MCP
```

🔗 GitLab Webhook 설정

- Webhook URL: http://3.35.167.118:8080/project/deploy-develop

- 트리거 체크:
  - ✅ Push events
  - ✅ Merge request events

> Webhook 설정 후 “Test” 버튼으로 연결 확인

> GitLab Access Token 대신 Jenkins Job의 Secret Token 사용 권장

🔌 필수 Jenkins 플러그인

- GitLab Plugin → GitLab Webhook 요청 처리 및 인증
- Matrix Authorization Strategy → 익명 사용자에게도 Job 빌드 권한 허용
- 설치 후 : `Jenkins 관리 > 플러그인 > 설치됨` 목록에서 확인, 없으면 `사용 가능` 탭에서 설치

⚙️ Jenkins Job 생성 및 설정

1. Job 이름: `deploy-develop` (Freestyle Project)
2. 프로젝트 설정 :
   a. GitLab URL : `https://lab.ssafy.com/s12-final/S12P31B107/`
   b. Git 리포지토리 URL: `https://lab.ssafy.com/s12-final/S12P31B107.git`
      - Credentials:
        - Kind: `Username With Password`
        - Username: `정찬호`
        - Password: `GitLab Access Token`
   c. Branch to Build: `develop`
   d. Build Triggers : `Build when a change is pushed to GitLab`
      → GitLab Webhook URL: `http://3.35.167.118:8080/project/deploy-develop`
   e. Secrets 파일 등록:
      - Variable: `PROJECT_ENV`, `DOCKER_USER / DOCKER_PASS`, `FRONTEND_ENV`
      - Credentials Type: `Secret file`
   f. 빌드 스텝 (Shell Script) : `위에 기입해 두었습니다.`

---

5. 여봐라 시연 시나리오 & 화면 구조

로그인 플로우

- 웹 접속 후, `시작하기` 버튼 클릭 시 `로그인 페이지`로 이동합니다.
- 아이디와 비밀번호(ID 예: test00@gmail.com, PW: qwer1234!)를 입력하여 로그인합니다.
- 로그인 성공 시 `채팅 페이지`로 이동합니다.

회원가입 플로우

- 계정이 없는 경우 하단에 `회원가입` 버튼을 클릭하여 `회원가입 페이지`로 이동합니다.
- 이메일과 비밀번호(영문/숫자/특수문자 각 1개 이상 포함 8자 이상), 이름을 입력하고 `가입하기` 버튼 클릭하여 회원가입합니다.

MCP 서비스 설정 플로우

- `채팅 페이지` 에서 우측 상단 `톱니바퀴` 아이콘 클릭 시 `MCP 설정 페이지` 로 이동할 수 있습니다. 해당 페이지에서 MCP 환경변수 등록을 통해 서비스를 이용할 수 있습니다.

- 서비스 카드 클릭 시 각 서비스에 대한 간단한 설명과 서비스를 이용하기 위해 필요한 환경변수 등록 가이드라인이 툴팁으로 제공됩니다.
- 사용자는 환경변수를 모두 입력하고 저장 버튼을 클릭하여 환경변수를 등록합니다.
- 환경변수를 모두 입력한 서비스와 환경변수가 필요없는 서비스의 경우 토글을 통해 서비스 사용을 선택할 수 있습니다.
- 저장하고 계속하기 버튼 클릭 시 선택한 MCP 서비스를 사용할 수 있게 됩니다.

채팅 페이지

- 좌측 사이드바에서 새 채팅 버튼을 통해 새 채팅을 생성할 수 있으며, 이전 채팅 내역을 확인할 수 있습니다. 좌측 사이드바를 닫을 시 상단바에서 새 채팅 버튼을 사용할 수 있습니다.
- 우측 사이드바에서 각 서비스에 대한 채팅 가이드라인이 제공됩니다. 채팅 가이드라인에는 해당 MCP를 통해 사용할 수 있는 툴과 MCP github 링크를 확인할 수 있습니다.
- 하단 입력창을 통해 채팅을 입력하여 선택한 MCP 서비스를 이용할 수 있습니다.

마이페이지 페이지

- 상단바 톱니바퀴 아이콘 클릭 시 마이페이지 로 이동합니다.
- 마이페이지 에서는 비밀번호 수정 및 회원탈퇴를 진행할 수 있습니다.