



포팅 매뉴얼

[작성 지침](#)

[개요](#)

[서버 환경](#)

[UFW 설정 및 포트](#)

[도메인 및 SSL 설정](#)

[주요 외부 서비스](#)

[프로젝트 매뉴얼](#)

[사전 필수 도구설치](#)

[운영 및 배포](#)

[Redis 7 \(Redis Insight UI\), PostgreSQL 17](#)

[Neo4J \(5.25\)](#)

[RabbitMQ](#)

[ElasticSearch, Kibana](#)

[SpringBoot, FastAPI, Web](#)

[Nginx](#)

[개발 환경\(유지보수\)](#)

[Jenkins](#)

[Grafana / Prometheus](#)

[구글 확장 및 MCP, 모바일, 워치](#)

[구글 확장](#)

[MCP](#)

[모바일](#)

[워치](#)

[프로젝트 구조도](#)

[시연 시나리오](#)

[웹](#)

[Chrome Extension](#)

[MCP](#)

[모바일](#)

[워치](#)

작성 지침

- 폴더명 'exec'

1. Gitlab 소스 클론 이후 빌드 및 배포할 수 있도록 정리한 문서
 - a. 사용한 JVM, 웹서버, WAS 제품 등의 종류와 설정 값, 버전(IDE 버전 포함) 기재
 - b. 빌드 시 사용되는 환경 변수 등의 내용 상세 기재
 - c. 배포 시 특이사항 기재
 - d. DB 접속 정보 등 프로젝트(ERD)에 활용되는 주요 계정 및 프로퍼티가 정의된 파일 목록
2. 프로젝트에서 사용하는 외부 서비스 정보를 정리한 문서
 - a. 소셜 인증, 포顿 클라우드, 코드 컴파일 등에 활용 된 '외부 서비스'가입 및 활용에 필요한 정보
3. DB 덤프 파일 최신본
4. 시연 시나리오
 - a. 시연 순서에 따른 site 화면별, 실행역(클릭 위치 등) 상세 설명

개요

- 서비스명 : **Second Brain**

- 팀명 : 유노 김이박?

- 주요 구성

BE : Spring Boot, FastAPI

FE : React.ts

AI : LangChain, LangGraph, ClovaTTS

INFRA : AWS EC2, S3, Jenkins, Docker, n8n, Grafana, Prometheus, Elasticsearch, Kibana, Nginx

DB : PostgreSQL, Redis, Milvus, Neo4j

Mobile, Wear : Android, Wear OS

서버 환경

- 주 서버

- OS : Ubuntu 22.04.5 LTS (Jammy)

- CPU : Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz, 4 vCPU

- 메모리: 16 GiB RAM

- 가상화: Xen (AWS EC2)

- 보조 서버(n8n 셀프 호스팅)

- OS : Ubuntu 22.04.5 LTS (Jammy)

- CPU : Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz, 2 vCPU

- 메모리: 4 GiB RAM

- 가상화: Xen (AWS EC2)

UFW 설정 및 포트

ufw status

- 22 (ALLOW)

- 80 (ALLOW)

- 7687 (ALLOW)

port(docker network) status

- 80 (ALLOW), 443 (ALLOW), 7687 (ALLOW) : nginx

- 8080 : springboot, cadvisor, jenkins

- 8000 : fastapi

- 5432 (ALLOW) : postgresql

- 6379 (ALLOW) : redis

- 5540 : redisinsight

- 9200 (ALLOW) : elasticsearch

- 5601 : kibana

- 7687 (ALLOW), 7473 : neo4j / UI

- 9090 (ALLOW) : prometheus

- 3000 : grafana

- 9100 (ALLOW) : node-exporter

- 9093 (ALLOW) : alertmanager

- 5672 (ALLOW), 4369 : rabbitmq / UI

도메인 및 SSL 설정

- 도메인
 - brainsecond.site
- SSL 인증서
 - Let's Encrypt (Certbot 활용)

주요 외부 서비스

서비스	용도	키 위치	연결 정보	발급
AWS S3	이미지 저장	.env	Springboot	https://aws.amazon.com/ko/
Naver Clova TTS	Text to Speech(음성 변환)	.env	Springboot	https://www.ncloud.com/product/aiService/clovaVoice
OpenAI (gpt, embedding model)	AI 노트 조회 및 유사도	.env	FastAPI	ssafy GMS
Anthropic (claude)	코드 리뷰, 리마인더	.env	gitlab, n8n	ssafy GMS

프로젝트 매뉴얼

사전 필수 도구설치

- EC2 환경 Ubuntu Pro 22.04 LTS 기준 (User 이름은 ubuntu 기준)

- 도커 설치

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# 도커 저장소 등록
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" |
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# 패키지 목록 업데이트
sudo apt-get update

# 설치
sudo apt-get install docker-ce=5:28.5.1~ubuntu.22.04~jammy docker-ce-cli=5:28.5.1~ubuntu.22.04~jammy containerd.io

# 버전 확인
docker --version

# docker 그룹 추가(굳이 안해도 됨 sudo 붙여서 써도 됨)

```

```
sudo usermod -aG docker ubuntu  
newgrp docker
```

- 도커 컴포즈 설치

```
# CLI 플러그인 디렉터리 생성  
mkdir -p ~/.docker/cli-plugins  
  
# 바이너리 다운  
curl -SL https://github.com/docker/compose/releases/download/v2.40.2/docker-compose-linux-x86_64 -o ~/.docker/cli-plugins/docker-compose  
  
# 권한 부여  
chmod +x ~/.docker/cli-plugins/docker-compose  
  
# 버전 확인(나오면 설치 완료)  
docker compose version
```

- 프로젝트 zip

```
# pem key, ec2-ip-address는 개인별 상이  
scp -i your-key.pem S13P31E107.zip ubuntu@ec2-ip-address:/home/ubuntu/  
  
unzip -v  
  
sudo apt-get update  
sudo apt-get install unzip -y  
  
unzip S13P31E107.zip -d /home/ubuntu/
```

- 프로젝트 클론

```
cd /home/ubuntu  
  
git --version  
  
## 위 버전이 나오면  
git https://lab.ssafy.com/s13-final/S13P31E107.git  
  
## 버전이 나오지 않으면 아래 cli 실행 후 다시 클론  
sudo apt-get update  
sudo apt-get install git -y  
git --version
```

- 도커 네트워크 설정 (필수)

```
sudo docker network create e107_net
```

운영 및 배포

 설명에 앞서, Nginx는 대부분 툴 UI를 리버스 프록시하기 때문에 프로젝트만 서비스 하고 싶다면 conf 파일의 투닝이 필요. 본 프로젝트에서는 Nginx를 컨테이너로 띄움.

 sudo docker compose up -d 로 컨테이너를 run 시킬 때마다 docker ps 로 확인하는 걸 추천!

 모든 Docker compose 파일은 본 프로젝트의 도메인인 brainsecond.site 기준이므로 참고하세요!!

▶ EC2의 메모리, CPU의 한계로 하드 리밋을 걸어 놓았습니다. 환경의 리소스 상황에 따라 조절해주세요!

Redis 7 (Redis Insight UI), PostgreSQL 17

```
# Redis
cd /home/ubuntu/S13P31E107/Deploy/redis

cat <<EOF > .env
REDIS_PASSWORD=userpassword
REDIS_DB=16
EOF

sudo docker volume create redis_data
sudo docker volume create redisinsight_data

sudo docker compose up -d

# PostgreSQL
cd /home/ubuntu/S13P31E107/Deploy/postgresql

cat <<EOF > .env
POSTGRES_USER=username
POSTGRES_PASSWORD=userpassword
POSTGRES_DB_NAME=dbname
EOF

sudo docker volume create postgres_data

sudo docker compose up -d
```

Neo4J (5.25)

```
# Neo4J
cd /home/ubuntu/S13P31E107/Deploy/rabbitmq

cat <<EOF > .env
NEO4J_PASSWORD=userpassword
EOF

sudo docker volume create neo4j_main_data
sudo docker volume create neo4j_logs
sudo docker volume create neo4j_import
sudo docker volume create neo4j_plugins

sudo docker compose up -d
```

RabbitMQ

```
# RabbitMQ
cd /home/ubuntu/S13P31E107/Deploy/rabbitmq

cat <<EOF > .env
RABBITMQ_DEFAULT_USER=username
RABBITMQ_DEFAULT_PASS=userpassword
EOF
```

```
RABBITMQ_DEFAULT_VHOST=vhostname  
EOF
```

```
sudo docker volume create rabbitmq_data  
sudo docker compose up -d
```

ElasticSearch, Kibana

```
cd /home/ubuntu/S13P31E107/Deploy/ek  
  
# 출력 값 복사후 .env의 encryption key에 붙여넣기  
openssl rand -hex 32  
  
cat <<EOF > .env  
STACK_VERSION=8.7.1  
ES_PORT=9200  
KIBANA_PORT=5601  
CLUSTER_NAME=es-docker-cluster  
LICENSE=basic  
ELASTIC_PASSWORD=userpassword  
KIBANA_PASSWORD=userpassword  
ENCRYPTION_KEY=암호화 키(32자 이상 임의 문자열.)  
ES_MEM_LIMIT=2g  
KB_MEM_LIMIT=1g  
EOF  
  
sudo docker volume create certs  
sudo docker volume create esdata  
sudo docker volume create kibanadata  
sudo docker volume create esplugins  
sudo docker volume create erickor_plugin  
  
sudo docker compose up -d
```

SpringBoot, FastAPI, Web

```
cd /home/ubuntu/S13P31E107/Deploy  
  
# env 파일 예를 확인하고 모두 작성하세요  
cat .env.example  
  
# vi나 nano 등을 활용해서 .env 파일 작성하세요  
vi .env  
  
sudo docker compose up -d klp_back_blue klp_front_blue klp_ai_blue note_consumer
```

Nginx

개발 및 유지보수가 목적이라면 다음 단계인 개발 환경 챕터를 완료하셔야 합니다! 그렇지 않으면 nginx의 conf 파일이 컨테이너를 찾지 못해 무한 재실행 됩니다!

- CloudFlare 같은 DNS 관리 서비스 활용하여 도메인 등록 후 서브 도메인 설정
 - brainsecond.site

- api.brainsecond.site
- jenkins.brainsecond.site (유지보수 시)
- SSL 인증서 발급
 - `docker-compose.yml` 의 `services → nginx → volumes` 확인
 - `./nginx.conf:/etc/nginx/nginx.conf:ro` 를 `./get_ssl.conf:/etc/nginx/nginx.conf:ro` 로 변경

```
cd /home/ubuntu/S13P31E107/Deploy/nginx

sudo docker compose up -d nginx
sudo docker compose up -d certbot

# ssl 인증서 발급
sudo docker exec certbot certbot certonly --webroot \
--webroot-path=/var/www/certbot \
-d brainsecond.site -d api.brainsecond.site

sudo docker compose stop certbot

sudo docker restart nginx
```

- 서비스 실행, 운영 목적
 - 위에서 변경했던 `./get_ssl.conf:/etc/nginx/nginx.conf:ro` 를 `./nginx_so.conf:/etc/nginx/nginx.conf:ro` 로 변경
- 개발 및 유지 보수 목적
 - Jenkins, grafana, prometheus 활용 예정 시
 - 위에서 변경했던 `./get_ssl.conf:/etc/nginx/nginx.conf:ro` 를 `./nginx.conf:/etc/nginx/nginx.conf:ro` 로 변경

```
# 수정한 후
sudo docker compose up -d --build nginx

or

sudo docker restart nginx
```

개발 환경(유지보수)

Jenkins

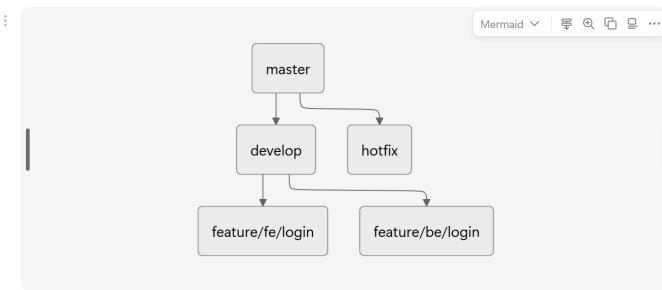
```
cd /home/ubuntu/S13P31E107/Deploy/jenkins

cat <<EOF > .env
JENKINS_ADMIN_USER=username
JENKINS_ADMIN_PASS=userpassword
JENKINS_URL=https://jenkins.brainsecond.site
EOF

sudo docker compose up -d jenkins
```

 프로젝트 내의 Jenkinsfile 과 수동 배포 파이프라인(dev.Jenkinsfile) 활용하려면 git branch strategy를 참고

Git Branch Strategy



• Jenkins Pipeline 기본 세팅

1. Jenkins GUI의 **Credentials**에서 4개의 Credentials를 작성해야함

- 모든 건 Global Scope
- value는 상황에 맞게, ID는 고정값

1. Username/password(Gitlab ID/Gitlab PASS) - ID(seok)
2. Username/password(DockerHub ID/DockerHub PASS) - ID(dockerhub-creds)
3. Secret test(발급 방법은 아래에서 설명되었음) - ID(MM_WEBHOOK)
4. Secret file(위에서 유일하게 vi, nano로 작성했던 .env파일) - ID(ENV_FILE)

2. Jenkins GUI의 **Plugins**에서 필요한 플러그인들 설치

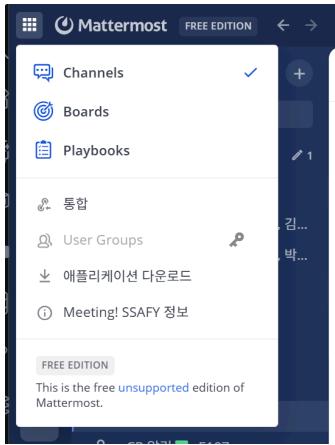
- AnsiColor
- Generic Webhook Trigger Plugin
- Git Plugin
- Pipeline
- Pipeline Utility Steps
- Pipeline: Groovy
- Pipeline: Job
- Timestamper

3. Jenkins GUI 메인 화면의 왼쪽 상단 **새로운 Item** 클릭

4. **item name** 입력 후 **Pipeline** 선택
5. **Triggers** 탭에서 **Generic Webhook Trigger** 체크
6. **Token** 입력란에 **uknow-mr** 입력
7. **Pipeline** 탭에서 **Definition**의 **Pipeline script from SCM** 설정
8. **SCM**은 **Git** 후에 Repo 정보 입력 후, 연결 표시 뜨면 **Script Path**에 **Deploy/Jenkinsfile** 입력 후 저장
> 한 번 Merge를 해서 진행시키면 파일을 읽고 세부 정보들은 자동 설정됨

• Jenkins Pipeline MatterMost Webhook 연결하기

1. Jenkins GUI의 **Plugins**에서 **Mattermost Notification Plugin** 설치 (필수)
2. 원하는 MatterMost 채널의 왼쪽 상단 메뉴를 클릭하고, **통합** 클릭



3. 전체 Incoming Webhook 클릭

통합

앱 디렉토리를 방문하여 Mattermost에 대한 자체 호스팅인 third-party 앱과 통합 기능을 찾으십시오.



전체 Incoming Webhook

Incoming Webhook은 외부 시스템에서 메시지를 받을 수 있게합니다.



전체 Outgoing Webhook

Outgoing Webhook은 외부 시스템에 메시지를 보내고 응답받을 수 있게합니다.



슬래시 명령어

슬래시 명령어는 외부에 연결한 서비스에 이벤트를 보냅니다.

4. Incoming Webhook 추가하기 클릭

5. 제목, 설명, 원하는 채널 선택 후 완료 클릭

6. 설정 완료 시 또는 URL 경로를 복사하거나 저장

7. S13P31E107/Deploy/Jenkinsfile 의 mmNotify에서 메세지의 url은 상황에 따라 변경해줘야 함

8. GitLab 프로젝트 메인 페이지 → 왼쪽 메뉴 Settings → Webhooks → Add new webhook

9. URL 입력란에 <https://jenkins.brainsecond.site/generic-webhook-trigger/invoke?token=uknow-mr> 입력

10. Trigger에서 Merge request events 체크, 저장 후 테스트 진행

Enable SSL verification은 ssl 인증서를 nginx로 적용을 했다면 체크, 안해도 무방(보안상 추천)

11. 결과 예시

유정석[부울경_1반_E107]팀원 BOT 오후 12:13

박진호 MR Generated!!!!!!!

[AI] graph서청 MCP 구현

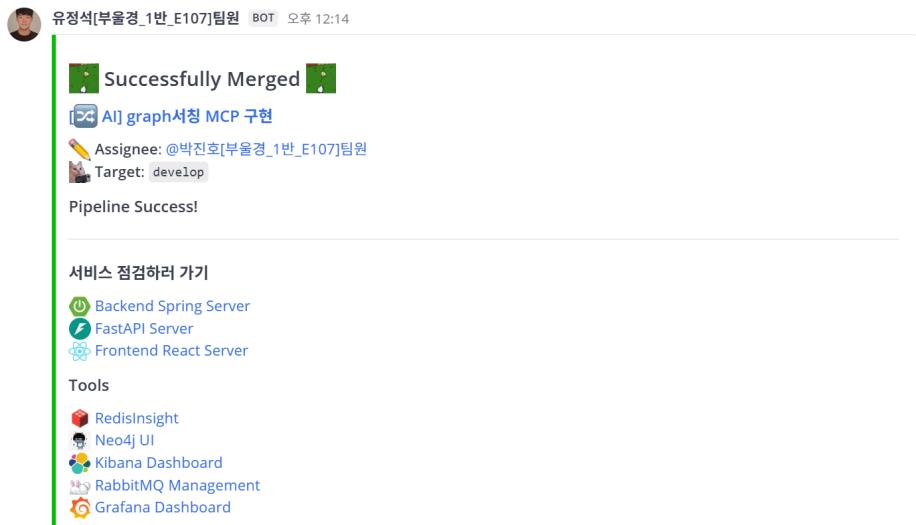
서둘러서 코드 리뷰 해주세요~! 수정 필요할 경우 작성자 태그해주세요!!

Assignee: @박진호[부울경_1반_E107]팀원

Reviewer: @김수민[부울경_1반_E107]팀원

Target: develop

Pipeline Success!



Grafana / Prometheus

```
cd /home/ubuntu/S13P31E107/Deploy/grafana

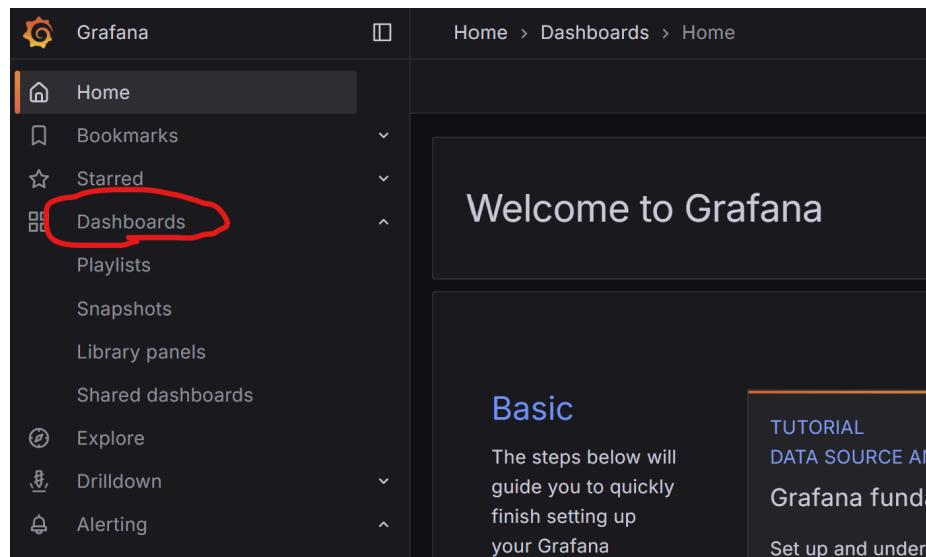
cat <<EOF > .env
GF_SECURITY_ADMIN_USER=username
GF_SECURITY_ADMIN_PASSWORD=userpassword
EOF

sudo docker volume create grafana_data
sudo docker volume create prometheus_data

sudo docker compose up -d
```

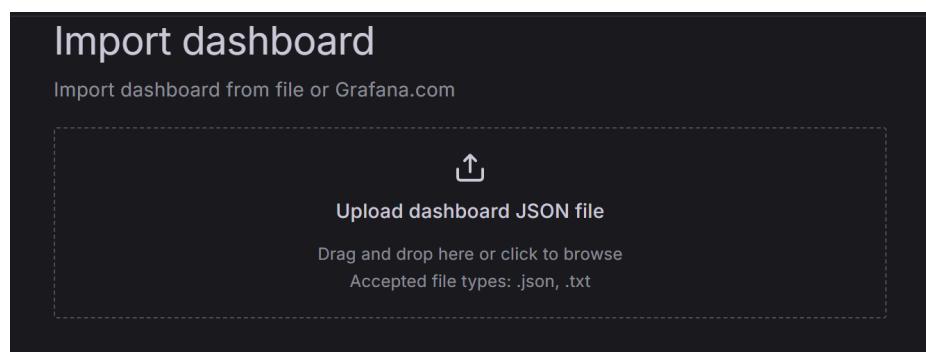
- **Grafana Dashboards 설정 (도커 특화 메트릭)**

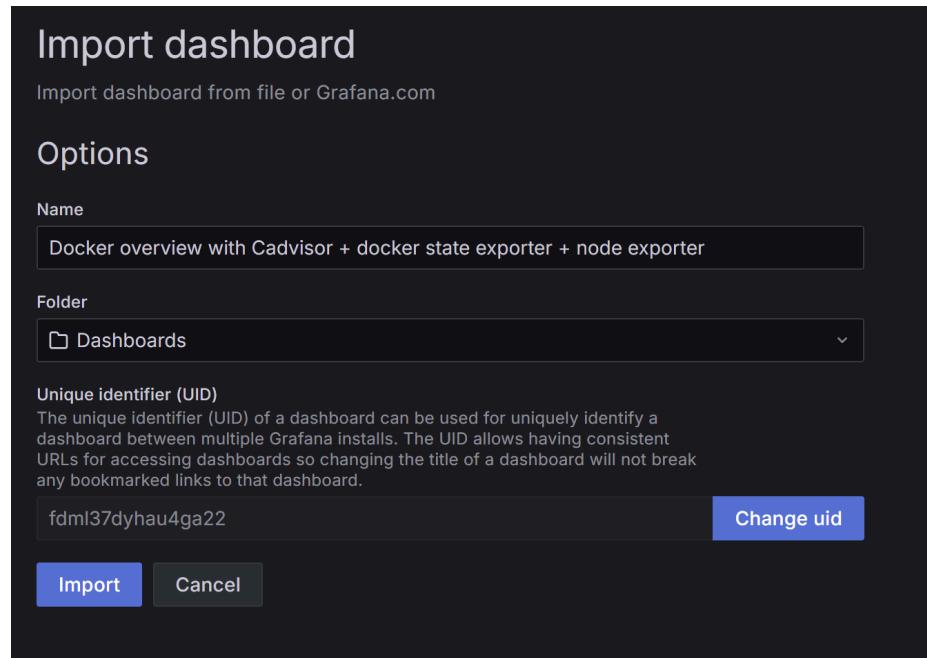
1. `/home/ubuntu/grafana/e107_dashboard.json` 파일을 확인
2. <https://brainsecond.site/grafana> 접속



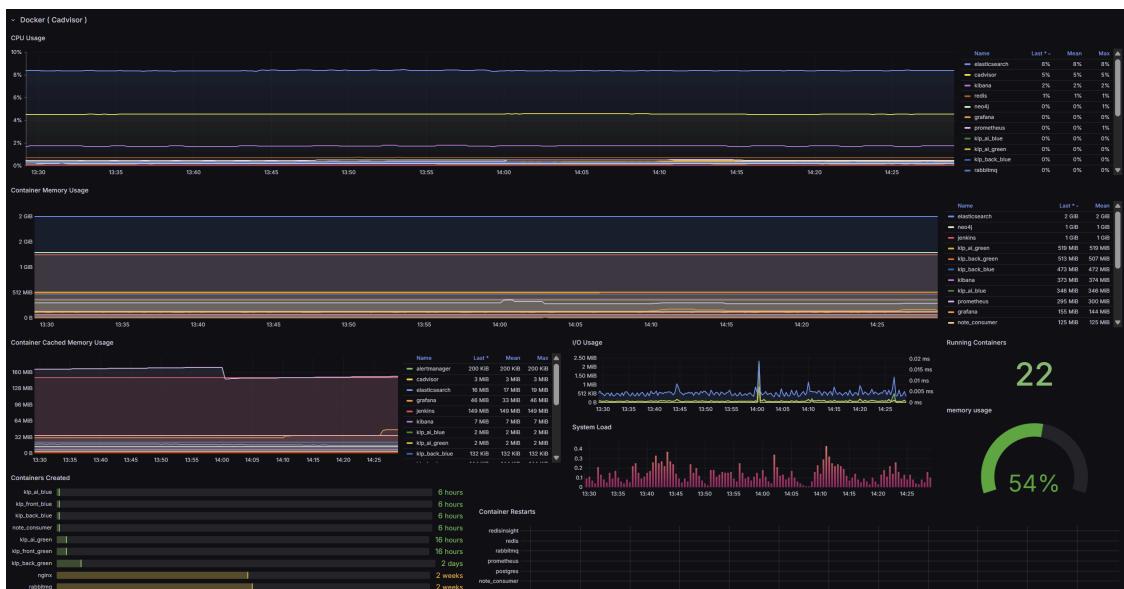
A screenshot of the 'Dashboards' page. It shows a list of dashboards with a search bar and filter options. In the top right, there is a 'New' dropdown menu with 'New dashboard', 'New folder', and 'Import' options. The 'Import' option is circled in red and has a red number '2' next to it.

3. [/home/ubuntu/grafana/e107_dashboard.json](#) 파일로 Dashboard 생성





4. 바로 Import 버튼 클릭 후 대쉬보드 확인



- EC2 Metric 상태 알람(MatterMost 활용) 적용

1. Webhook 생성은 Jenkins MR 알람 설정 때의 방법과 동일

2. `S13P31E107/Deploy/grafana/alertmanager/alertmanager.yml` 파일의 `api_url` 부분을 방금 저장한 URL로 변경, `channel`도 알맞게 변경.
`text` 의 `Buster Call` 부분도 @ 뒤에 팀원 아이디로 변경해주기
 > `channel` 은 메세지 채팅방의 링크 복사후 끝 부분이 채널명

```

receivers:
  - name: 'mattermost'
    slack_configs:
      - api_url: 'https://meeting.ssafy.com/hooks/obryq4gt83fxpg06dq5t4ew3zw'
        channel: '#monitoring-e107'
        send_resolved: true
        title: |-
          [{{ .Status | toUpper }}] 그라파나에서 확인하기!
  
```

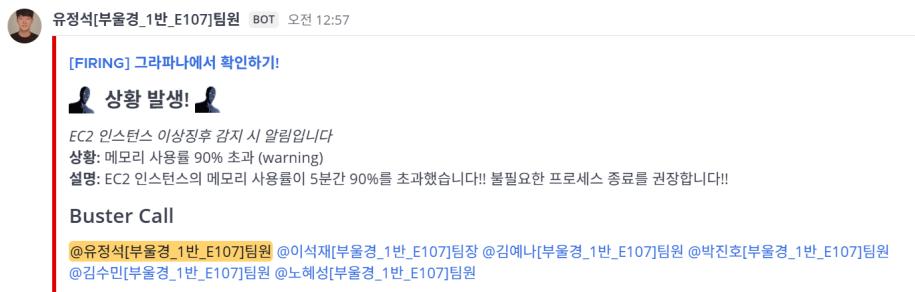
```

text: |-
{{ range .Alerts }}
{{ if eq .Status "firing" }}
##### :conan_hanzawa: **상황 발생!** :conan_hanzawa:
*EC2 인스턴스 이상진후 감지 시 알림입니다*
**상황:** {{ .Annotations.summary }} ({{ .Labels.severity }})
**설명:** {{ .Annotations.description }}
##### Buster Call
@9526yu @lsj6924 @yeneua.dev @dodwn48 @gkgkkgk22 @hro
{{ else if eq .Status "resolved" }}
##### :conan_machi: **상황 종료!** :conan_machi:

```

3. 모두 변경하면 `sudo docker restart alertmanager`, `sudo docker restart prometheus` 해줘야 함

4. 결과 예시 (현재 4가지 설정 상태 알람 수신 가능. `node_exporter on/off 여부`, `EC2 RAM 사용량`, `EC2 CPU 사용량`, `EC2 Storage 사용량`)



구글 확장 및 MCP, 모바일, 워치

이 네 가지는 EC2에서 실행하기보다 로컬 환경에서 따로 빌드해야 함

구글 확장

```

# 해당 디렉터리에서 Bash를 연 후
cp .env.example .env

pnpm install

pnpm build

# 개발 서버가 http://localhost:5174로 실행

```

1. 크롬 브라우저에서 `chrome://extensions` 열기
2. 우측 상단 "개발자 모드" 활성화
3. "압축해제된 확장 프로그램을 로드합니다" 클릭
4. `dist` 폴더 선택

MCP

```

# S13P31E107 디렉터리 상에서 Bash를 열고
cd agent-MCP

# 의존성 설치
uv sync

# .env 파일 생성
# API_KEY=your-api-key

```

```
# API_BASE_URL=https://api.brainsecond.site/
cp .env-example .env
```

- **Claude Desktop 연결**

- 방법 1

```
# FastMCP CLI 사용 (권장)
# 가상환경 활성화

source .venv/Script/activate

# 프로젝트 폴더에서 실행

fastmcp install claude-desktop main.py --name personal-notes
```

- 방법 2

```
# 수동 설정
# 설정 파일 열기

# Windows:
%APPDATA%\Claude\claude_desktop_config.json

# 설정 추가
uv run python generate_config.py

# 위 명령어로 현재 내 컴퓨터 경로로 config.json 생성
# 경로를 실제 프로젝트 위치로 변경
# 설정 후 Claude Desktop 종료 후 재시작
```

```
{
  "mcpServers": {
    "personal-notes": {
      "command": "uv",
      "args": [
        "--directory",
        "yout-directory-path",
        "run",
        "python",
        "main.py"
      ]
    }
  }
}
```

모바일

- https://drive.google.com/file/d/1lEer8lYdAvZ-2JVkiNU0V2DBT567OeJT/view?usp=drive_link
- 위 링크에서 apk 파일 다운로드 후 스마트폰에서 apk 파일로 설치

워치

- 워치는 개발자 모드를 켰 (설정 → 소프트웨어 정보 → 소프트웨어 버전 여러번 클릭)
- 안드로이드 스튜디오 다운한 뒤 워치 폴더 빌드 후 워치에 다운로드

프로젝트 구조도

```
S13P31E107/
├── backend/secondbrain
│   ├── Dockerfile
│   └── main.py
│
├── agent-MCP/
│   ├── services/...
│   ├── .env-example
│   └── main.py
│
├── knowledge-graph-service/
│   ├── app/...
│   ├── Dockerfile
│   └── main.py
│
├── extension/
│   ├── src/
│   ├── Dockerfile
│   ├── package.json
│   └── README.md
│
├── frontend/
│   └── secondbrain/
│       ├── Dockerfile
│       ├── src/...
│       ├── docs/...
│       ├── package.json
│       └── public/...
│
├── mobile_watch/
│   └── secondbrain/
│       ├── build.gradle.kts
│       ├── mobile/...
│       |   └── src/...
│       └── wear/...
│           ├── src/...
│           └── build.gradle.kts
│
└── Deploy/
    ├── docker-compose.dev.yml
    ├── docker-compose.prod.yml
    ├── docker-compose.ai.yml
    └── nginx/conf.d/
        └── default.conf
```

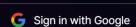
시연 시나리오

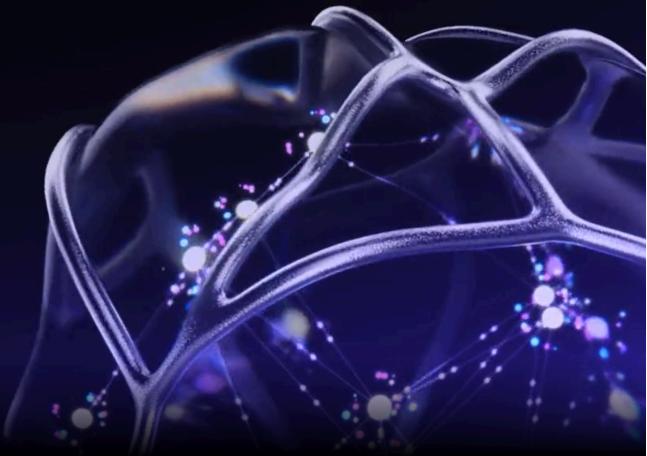
웹

1. 메인 화면 접속 후 소셜 로그인

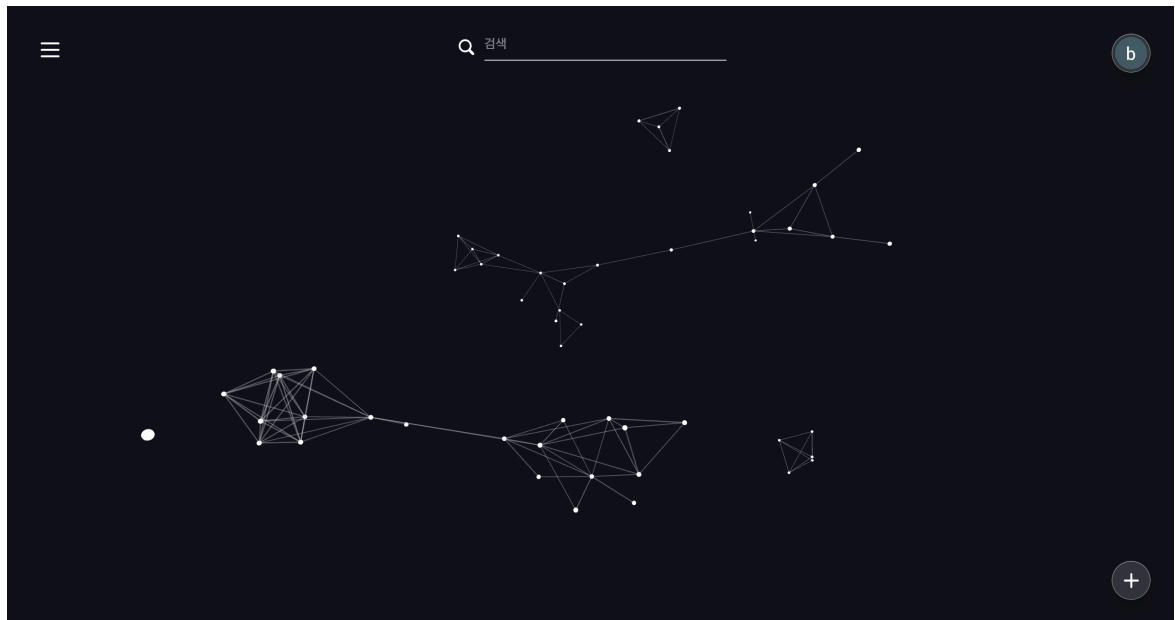
Second Brain

새로운 세상을 위한 새로운 두뇌

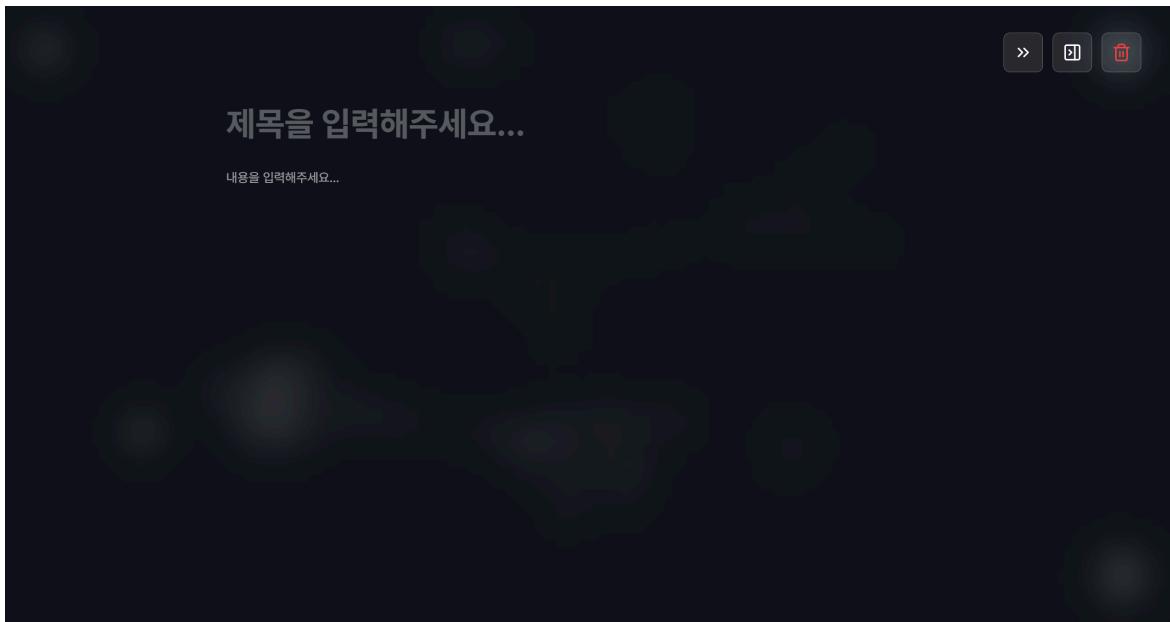
 Sign in with Google



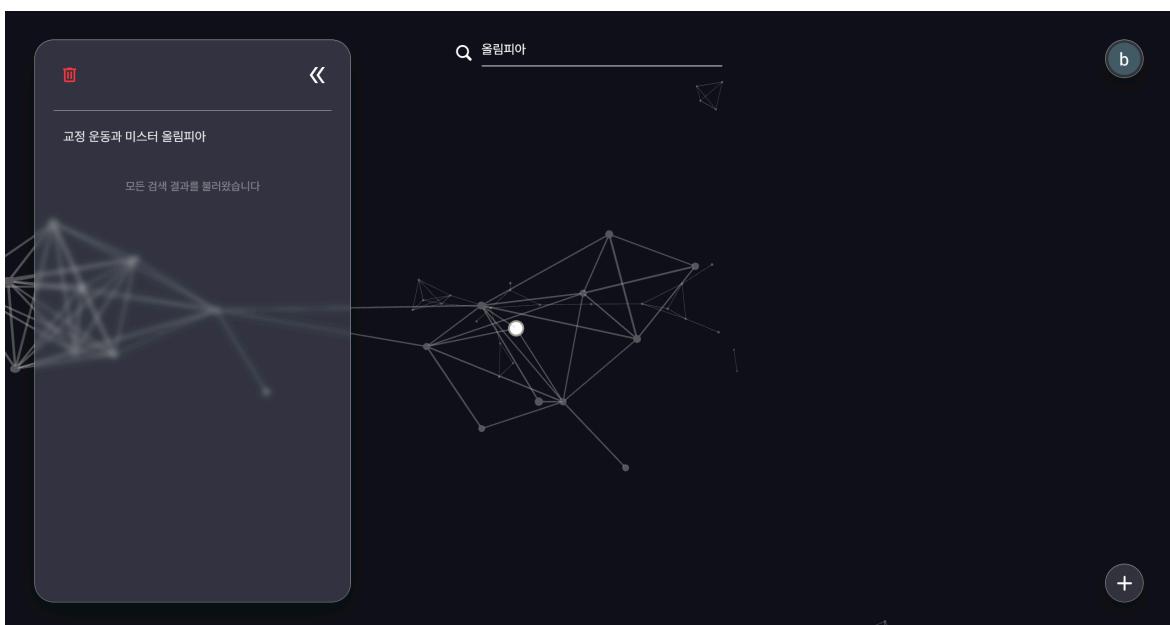
2. 로그인 후 메인 화면에서 노트 노드 조회 가능



3.  버튼을 클릭하여 노트 생성 가능

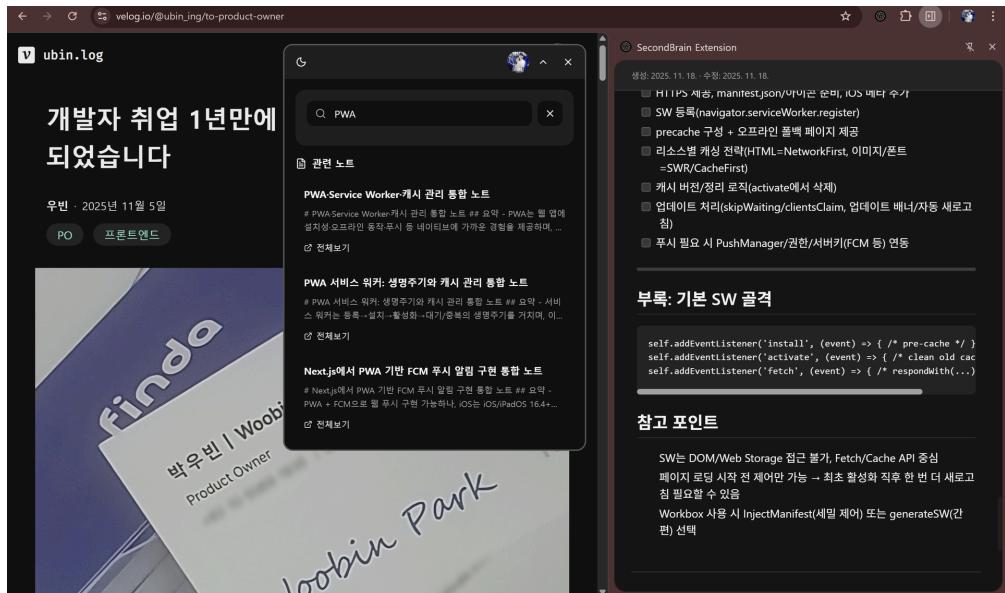


4. 검색 시 그에 맞는 노드 확인 가능 (노드 임팩트 발현)
물론 조회도 가능

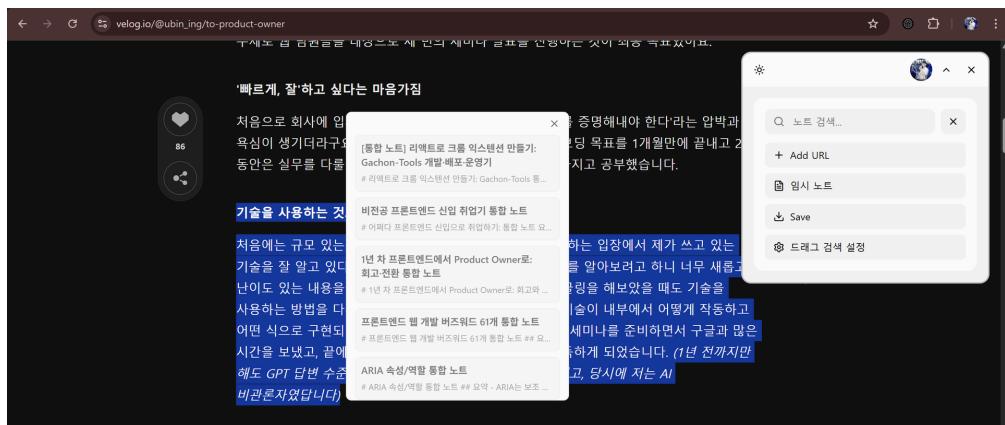


Chrome Extension

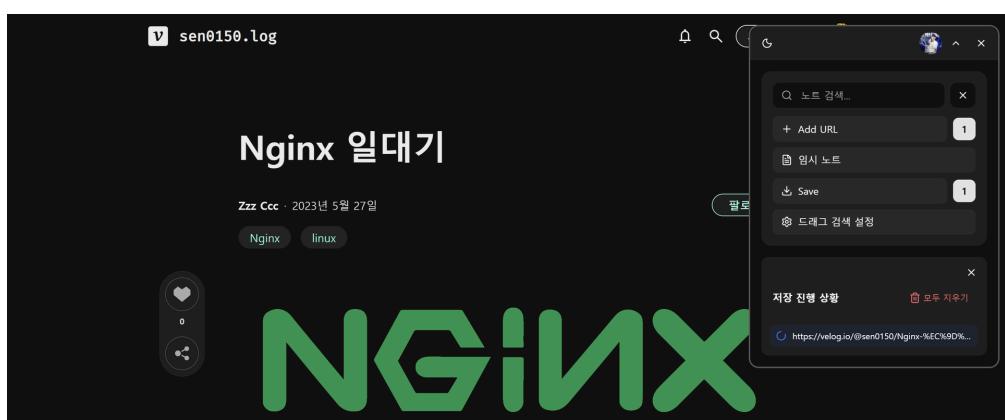
1. 해당 키워드와 관련된 노트들을 사이드 패널에서 검색 가능



2. 드래그 및 하이라이트 한 내용 저장할 콘텐츠로 지정 가능



3. 이때까지 저장한 내용들 내 노트에 자동 저장!

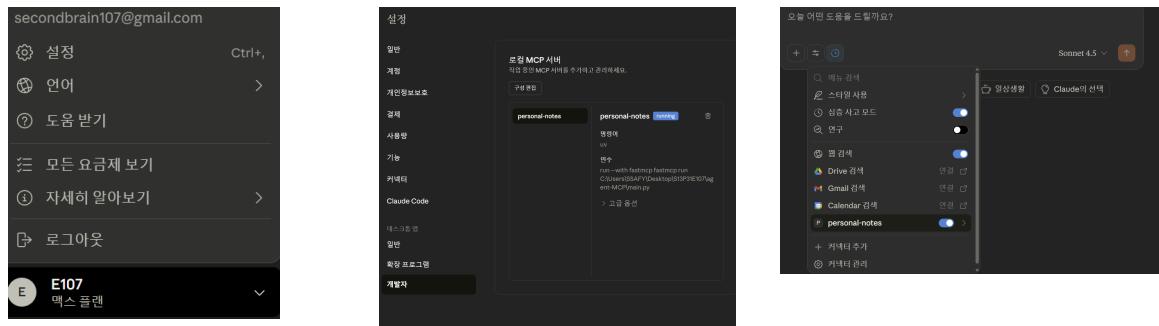


MCP

- API KEY는 웹 메인화면 오른쪽 상단의 프로필 클릭 후 API 발급 받기!
- Claude Desktop에서 아래 그림 따라 설정 후, 평소대로 활용하면 됨
- MCP 기능은 GPT(로컬호스트로 열어야 함) 등 여러 곳에서 활용 가능합니다!

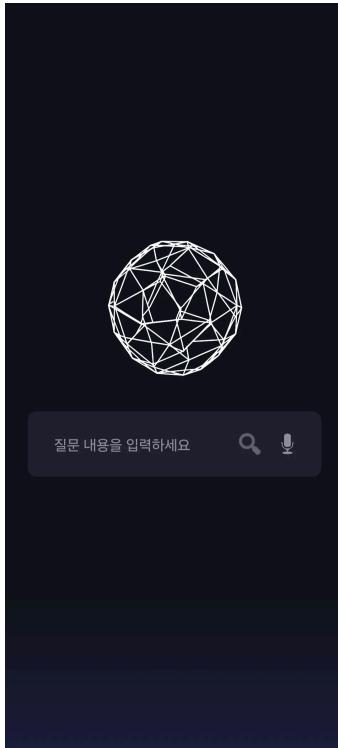
```
# GPT의 경우
# 개인 설정의 커넥터 접속 후 개발자 모드 on
# 만들기 후 URL을 localhost로 연결
```

```
# 방법 : agent-MCP 디렉터리 배수 내에서
uv sync
python main.py
```

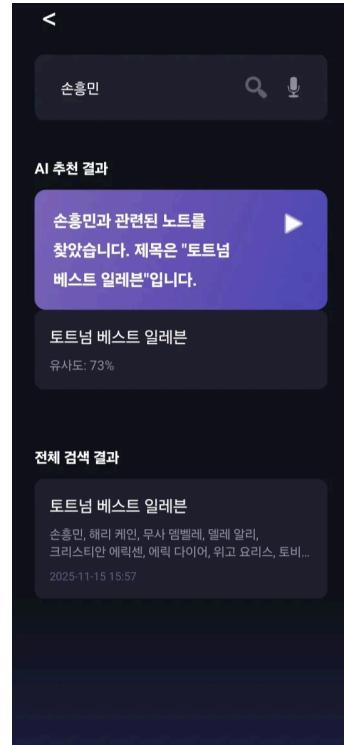


모바일

1. 메인 화면

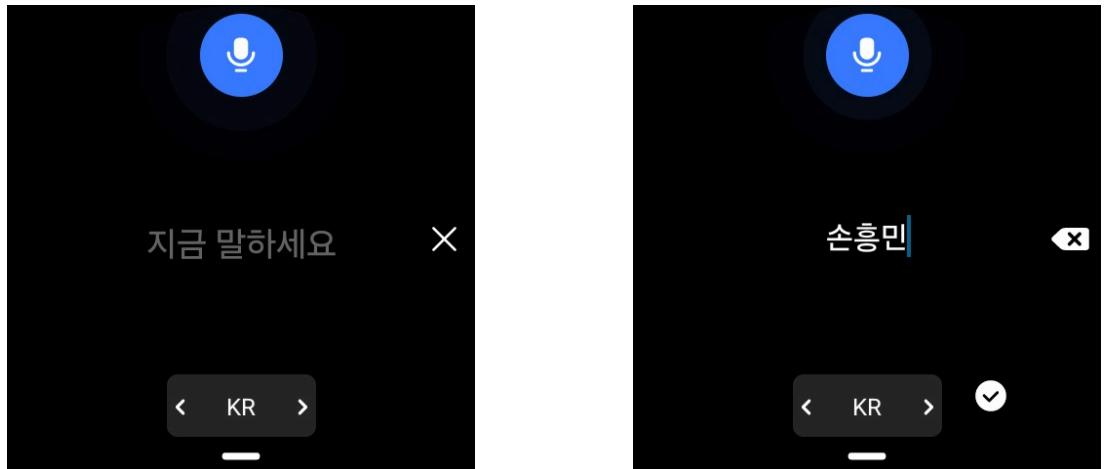


2. 검색 시 화면 (일반 검색, AI 검색, AI 검색 답변 TTS 지원)



위치

- 설정 → 버튼 및 제스처 → 두 번 누르기 → SecondBrain 설정
- 홈 화면 2회 클릭 후 STT 질문 실행



3. 질문 답 도착 시 알람을 통해 핸드폰으로 즉시 노트 조회

