

umm

관리자 가이드

설치, 설정, 계정과 권한, 운영과 장애 대응

버전 v0.71.6

작성일 2026-09-10

문서 관리자 가이드

umm 관리자 가이드

기준 버전 **v0.71.6**. 화면을 쓰는 사람을 위한 안내는 [사용자 가이드](#)에 있습니다. 이 문서의 화면은 모두 v0.71.6 을 실제로 띄워 1440x900 에서 찍은 것이며, 데이터는 가짜입니다. 환경 변수 표는 `internal/config/config.go` 에서, API 의 메서드와 경로는 `internal/httpapi/server.go` 의 라우트 등록에서 읽었습니다.

1. 구성 요소

구성 요소	무엇	주고받는 것
umm 컨테이너	Go 바이너리 하나에 React UI·한글 웹폰트·시간대·CA 인증서가 들어 있는 단일 이미지. non-root umm 사용자, :8080	브라우저·API·MCP 클라이언트의 HTTP
PostgreSQL 14 이상	유일한 저장소. 생각·연결·설정·감사 로그·오프라인 전달함이 모두 여기	POSTGRES_DSN . 협업 이벤트는 LISTEN/NOTIFY
TLS reverse proxy	앞단에서 TLS 종료	UMM_TRUSTED_PROXY_CIDRS 로 신뢰 대역을 지정할 때만 forwarding header 를 믿음
(선택) Keycloak	사내 SSO	OIDC Discovery. 관리자 화면에서 설정
(선택) AI Gateway	OpenAI 호환 채팅·임베딩 서버 (vLLM·Ollama·TGI 등)	Dream·AI 생각 도구·의미 검색. 없으면 내장 어휘 임베딩만
(선택) embeddings 컨테이너	compose.yaml 의 embeddings 프로파일 — Ollama 로 bge-m3 를 옆에 띄움	임베딩만. 완전 오프라인
(선택) Ptium	발표 자료를 만드는 별도 서비스	관리자 화면 Ptium 발표 자료 에서 주소·API 키
(선택) OTLP 수집기	추적	OTEL_EXPORTER_OTLP_* 가 있을 때만

실행 중에는 패키지 저장소나 CDN 에 접근하지 않습니다. 폐쇄망 반입은 이미지 하나면 됩니다.

2. 설치

릴리즈 자산은 `umm-v0.71.6.tar.gz` 와 `SHA256SUMS` 입니다.

2-1. 이미지 반입

```
sha256sum -c SHA256SUMS
./scripts/load-offline.sh umm-v0.71.6.tar.gz      # = gzip -dc umm-v0.71.6.tar.gz | docker load
docker image inspect umm:v0.71.6 --format '{{.Id}}'
```

2-2. compose 로 띄우기

저장소의 `compose.yaml` 이 PostgreSQL 17 과 umm 을 함께 올립니다. **비밀값은 그 파일의 개발용 값을 그대로 쓰지 말고** 아래처럼 바꿔 넣습니다. 암호화 키는 연결망 밖의 승인된 절차로 만들고 (`openssl rand -base64 32`) shell history·compose 파일·티켓·Git 에 남기지 마세요.

```
# 예시 값 — 모두 가짜입니다
export POSTGRES_PASSWORD='change-me-db-password'
export BOOTSTRAP_ADMIN_PASSWORD='change-me-admin-password'
export ENCRYPTION_KEY="$(openssl rand -base64 32)"

cat > compose.override.yaml <<'EOF'
services:
  postgres:
    environment:
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
  umm:
    image: umm:v0.71.6
    environment:
      POSTGRES_DSN: "postgres://umm:${POSTGRES_PASSWORD}@postgres:5432/umm?sslmode=disable&pool_max_conns=1"
      BOOTSTRAP_ADMIN: admin
      BOOTSTRAP_ADMIN_PASSWORD: ${BOOTSTRAP_ADMIN_PASSWORD}
      ENCRYPTION_KEY: ${ENCRYPTION_KEY}
EOF

docker compose up -d
curl -s http://127.0.0.1:8080/readyz      # {"status":"ready"}
```

이미 PostgreSQL 이 있다면 umm 만 `docker run` 합니다. 필요한 환경 변수는 아래 표의 필수 네 개뿐입니다.

```
docker run -d --name umm --restart unless-stopped -p 8080:8080 \
-e POSTGRES_DSN='postgres://umm:change-me-db-password@postgres.internal:5432/umm?sslmode=require&pool_max
-e BOOTSTRAP_ADMIN='admin' \
-e BOOTSTRAP_ADMIN_PASSWORD='change-me-admin-password' \
-e ENCRYPTION_KEY='<openssl rand -base64 32 의 출력>' \
umm:v0.71.6
```

PostgreSQL 사용자는 대상 데이터베이스에 schema·table·extension(pgcrypto , citext , pg_trgm)을 만들 권한이 있어야 합니다. 마이그레이션은 기동 시 트랜잭션으로 자동 적용됩니다.

항목	값
포트	8080/tcp (컨테이너 안). UMM_HTTP_ADDR 로 바꿀 수 있음
볼륨	umm 컨테이너에는 없음(상태는 전부 PostgreSQL). compose 의 umm-postgres , 선택 umm-embeddings
DB 연결	인스턴스당 request pool 기본 최대 16 + 외부 호출용 lease 최대 5 → 전역 예산은 replica 마다 pool_max_conns + 5
외부 접근	실행 중 네트워크 접근 없음. AI Gateway·Ptium·OTLP 는 설정했을 때만

2-3. 최초 관리자 계정

BOOTSTRAP_ADMIN / BOOTSTRAP_ADMIN_PASSWORD 로 첫 관리자가 만들어집니다. 이미 그 계정이 DB 에 있으면 재시작해도 비밀번호를 덮어쓰지 않습니다. 로그인 뒤 순서대로:

1. 서비스 관리자 → 일반: 서비스 이름, 공개 URL, Asia/Seoul 같은 IANA 시간대.
2. 키·권한: 허용 API/MCP 스코프, 기본 만료, 회전 중첩, 남용 방지 한도 확인.
3. 필요할 때만 Keycloak SSO 를 저장하고 연결 시험 뒤 활성화.
4. 필요할 때만 AI Gateway 를 저장하고 임베딩 품질 측정에서 ● 확인.
5. Dream Layer 는 Gateway 확인 뒤 기능과 자동 생성을 각각 켭니다.
6. 검토 프로세스에서 실제로 승인이 필요한 작업만 켭니다. 꺼져 있으면 승인 단계는 생기지 않습니다.

← 내 공간

um Service Admin
umm control room

운영 현황

일반

Keycloak SSO

Dream Layer

AI Gateway

Ptium 발표 자료

유사도 기준

AI 품질 평가

키 · 권한

검토 프로세스

사용자

공간과 참여자

웹훅 상태

감사 로그

모든 변경은 감사 로그에 기록됩니다.

SERVICE ADMINISTRATION

빠른 이동ctrl K

↺

운영 현황

전체 사용자1

활성 사용자1

생각5

공간3

Dream 생성0

AI 호출0

실시간 협업

열려 있는 이벤트 구독과 PostgreSQL 수신 상태

수신 대기

연결된 구독0

공간0

전달한 신호0

Dream 운영

이번 달 채택·후속 활용과 현재 설정 기준 사전 예측

지금 큐 생성

생성 조건 충족 사용자1

예상 월 호출30

예상 월 비용\$0.00

검토 완료0

평균 내부 품질0%

Dream 채택률0%

운영 현황 — 사용자·공간·생각 수, 실시간 협업 상태, 최근 웹훅 실패, AI 사용 통계

← 내 공간

um Service Admin
umm control room

운영 현황

일반

Keycloak SSO

Dream Layer

AI Gateway

Ptium 발표 자료

유사도 기준

AI 품질 평가

키 · 권한

검토 프로세스

사용자

공간과 참여자

웹훅 상태

감사 로그

모든 변경은 감사 로그에 기록됩니다.

SERVICE ADMINISTRATION

빠른 이동ctrl K

↺

일반

서비스 기본 정보

재시작 없이 적용되며 로그인 화면과 서비스 전반에 반영됩니다.

서비스 이름

umm

공개 URL

OIDC Callback과 Origin 검증에 사용됩니다.

http://localhost:8080

세션 유지 시간

24 시간

서비스 시간대

Asia/Seoul

모든 변경사항이 저장되었습니다.

저장

일반 — 서비스 이름, 공개 URL, 세션 시간, 시간대

3. 설정

3-1. 환경 변수

`internal/config/config.go` 와 `internal/observability/observability.go` 가 읽는 전부입니다. 나머지 정책은 모두 관리자 화면에서 PostgreSQL 에 저장되며 재시작이 필요 없습니다.

이름	기본값	필수	설명
POSTGRES_DSN	—	필수	PostgreSQL 연결 문자열. <code>pool_max_conns</code> 를 여기 붙임 (예 <code>postgres://umm:change-me@db:5432/umm?sslmode=require&pool_max_conns=16</code>)
BOOTSTRAP_ADMIN	—	필수	최초 관리자 아이디
BOOTSTRAP_ADMIN_PASSWORD	—	필수	최초 관리자 비밀번호. 계정이 이미 있으면 무시
ENCRYPTION_KEY	—	필수	저장되는 비밀값 (OIDC·AI·Ptium·웹훅 secret, AI 프롬프트 로그)의 AES-256-GCM 키. 정확히 32바이트 — 원문 32자, 64자 hex, 또는 base64 32바이트
ENCRYPTION_KEY_PREVIOUS	비움	선택	숨표로 구분한 이전 키. 회전 기간에만
UMM_HTTP_ADDR	:8080	선택	바인드 주소 <code>host:port</code> (예 <code>127.0.0.1:18081</code>)
UMM_TRUSTED_PROXY_CIDRS	비움	선택	숨표로 구분한 신뢰 proxy IP/CIDR. 비우면 <code>X-Forwarded-For</code> · <code>X-Real-IP</code> · <code>X-Forwarded-Proto</code> 를 모두 버림. <code>0.0.0.0/0</code> · <code>:::/0</code> 금지
OTEL_EXPORTER_OTLP_ENDPOINT	비움	선택	있을 때만 OTLP HTTP trace exporter 활성화
OTEL_EXPORTER_OTLP_TRACES_ENDPOINT	비움	선택	위와 같음(traces 전용 주소)

필수 넷 중 하나라도 비면 기동하지 않고 로그에 `invalid startup configuration` 이 남습니다.

브라우저 변경 요청의 `origin` 은 공개 URL 또는 신뢰 proxy 로 확인한 현재 요청과 `scheme·host·port` 가 모두 같아야 합니다. HTTPS 서비스에 `http://` Origin 은 거부됩니다. Proxy 는 외부에서 온 forwarding header 를 지우고 자신이 확인한 값을 적도록 구성하세요.

3-2. 관리자 화면의 설정

왼쪽 메뉴 순서대로. 각 절의 자세한 값은 아래 부록에 있습니다.

메뉴	정하는 것	부록
일반	서비스 이름, 공개 URL, 세션 시간, 시간대	—
Keycloak SSO	Issuer, Client, 관리자·팀장 그룹 매핑, 연결 시험	부록 2
Dream Layer	자동 생성 시각·주기, 컨텍스트 범위, 최대 응답 토큰, 품질 기준선	부록 3
AI Gateway	채팅·임베딩 주소와 키, timeout·재시도, 비용, 임베딩 품질 측정, 자동 찾기	부록 4
Ptium 발표 자료	Ptium 주소·API 키· timeout_seconds , 연결 시험	부록 8-2
유사도 기준	연관 생각·군집·연결 추천의 상대 기준, 저장 전 재보기	부록 4-2
AI 품질 평가	Dream 회귀 케이스	부록 7
키 · 권한	허용 스코프, 기본 만료, 회전 중첩, 남용 방지, 암호화 키 상태·회전	부록 4-1 · 8
검토 프로세스	승인이 필요한 작업	—
사용자 / 공간과 참여자	역할 변경·비활성화, 공간 소유권 이전	4. 계정과 권한
웹훅 상태	실패 중인 웹훅, 멈추기	부록 9
감사 로그	모든 관리 행위	5. 운영

← 내 공간

um

Service Admin

umm control room

운영 현황

일반

Keycloak SSO

Dream Layer

AI Gateway

Ptium 발표 자료

유사도 기준

AI 품질 평가

키 · 권한

검토 프로세스

사용자

공간과 참여자

웹hook 상태

감사 로그

모든 변경은 감사 로그에 기록됩니다.

SERVICE ADMINISTRATION

Keycloak SSO

빠른 이동

ctrl K

↺

Keycloak SSO · OIDC

연결 시험

Issuer URL, Client ID, Client Secret만으로 Discovery를 통해 자동 연결합니다.

Keycloak SSO 활성화

Issuer URL

https://keycloak.internal/realms/umm

Client ID

Client Secret

관리자 그룹/역할

팀장 그룹/역할

umm-admin

umm-team-lead

Keycloak Confidential Client에서 Standard Flow를 켜고 Callback을 http://localhost:8080/api/v1/auth/oidc/callback으로 정확히 등록하세요.

모든 변경사항이 저장되었습니다.

저장

Keycloak SSO — Issuer URL, Client ID·Secret, 관리자·팀장 그룹, 연결 시험

← 내 공간

um

Service Admin

umm control room

운영 현황

일반

Keycloak SSO

Dream Layer

AI Gateway

Ptium 발표 자료

유사도 기준

AI 품질 평가

키 · 권한

검토 프로세스

사용자

공간과 참여자

웹hook 상태

감사 로그

모든 변경은 감사 로그에 기록됩니다.

SERVICE ADMINISTRATION

Dream Layer

빠른 이동

ctrl K

↺

Dream Settings

좋은 Dream이 없으면 생성하지 않는 것을 기본 원칙으로 합니다.

Dream 기능

자동 생성

생성 시간

02:00 AM

생성 주기

매일

사용자당 Dream

1 장

최소 메모

3 개

최근 분석 범위

7 일

최대 Context 메모

20 개

Model

최대 응답 Token

추론 토큰을 포함한 최대 출력 한도입니다.

4,096

4K 16K 64K 128K 256K

월 사용자 호출 제한

30

사용자 개별 OFF 허용

사용자 개별 OFF 허용

Dream 도착 알림 허용

하세요.

모든 변경사항이 저장되었습니다.

저장

Dream Layer — 자동 생성 시각과 주기, 분석 범위, 최대 응답 토큰

← 내 공간

um

Service Admin
umm control room

▼ 운영 현황

⊗ 일반

🔑 Keycloak SSO

⊞ Dream Layer

🔗 AI Gateway

📄 Ptium 발표 자료

📊 유사도 기준

📈 AI 품질 평가

🔑 키 · 권한

🔄 검토 프로세스

👤 사용자

👤 공간과 참여자

🔄 웹훅 상태

📄 감사 로그

모든 변경은 감사 로그에 기록됩니다.

SERVICE ADMINISTRATION

🔍 빠른 이동

ctrl K

🔄

AI Gateway

내부 AI Gateway

OpenAI 호환 Chat Completions 엔드포인트를 사용합니다. 외부 연결 없이 내부 모델 서버를 지정할 수 있습니다.

Base URL

서버 주소, /v1 주소 또는 전체 /chat/completions 주소를 사용할 수 있습니다.

http://llm-gateway.internal:8000/v1

API Key

👁

Timeout

긴 주론 모델은 충분한 시간을 지정하세요.

45 초

⬆

⬆

재시도

1 이상이면 추론만 생성된 응답을 비주론 모드로 복구합니다.

2

⬆

⬆

입력 \$ / 1M token

0

⬆

⬆

출력 \$ / 1M token

0

⬆

⬆

AI 로그 보존

90 일

⬆

⬆

Prompt Version

dream-v1

모든 변경사항이 저장되었습니다.

저장

임베딩 모델

비워 두면 외부 호출 없이 내장 로컬 임베딩을 사용합니다. 모델을 바꾸면 생각이 점진적으로 다시 임베딩됩니다.

AI Gateway — 채팅 Base URL·API Key, 임베딩 주소·모델, 임베딩 품질 측정

← 내 공간

um

Service Admin
umm control room

▼ 운영 현황

⊗ 일반

🔑 Keycloak SSO

⊞ Dream Layer

🔗 AI Gateway

📄 Ptium 발표 자료

📊 유사도 기준

📈 AI 품질 평가

🔑 키 · 권한

🔄 검토 프로세스

👤 사용자

👤 공간과 참여자

🔄 웹훅 상태

📄 감사 로그

모든 변경은 감사 로그에 기록됩니다.

SERVICE ADMINISTRATION

🔍 빠른 이동

ctrl K

🔄

Ptium 발표 자료

Ptium 연결

생각을 발표 자료로 만들 Ptium 서버입니다. 비워 두면 이 기능이 꺼집니다. umm은 Ptium에 생각을 그대로 보내며, 모델에게 다시 쓰게 하지 않습니다.

연결 테스트

Base URL

Ptium 서버 주소입니다. 예: http://ptium.internal:8080

http://ptium.internal:8080

API Key

Ptium에서 발급한 ptium_ 로 시작하는 키입니다. presentations 읽기·쓰기 권한이 필요합니다.

👁

템플릿 ID

비워 두면 Ptium의 기본 디자인을 씁니다. 연결 테스트를 하면 목록에서 고를 수 있습니다.

언어

Ptium이 텍스트를 만들 때 쓰는 언어 코드입니다.

ko

Timeout

텍스트 컴파일하는 데 걸리는 시간입니다.

30 초

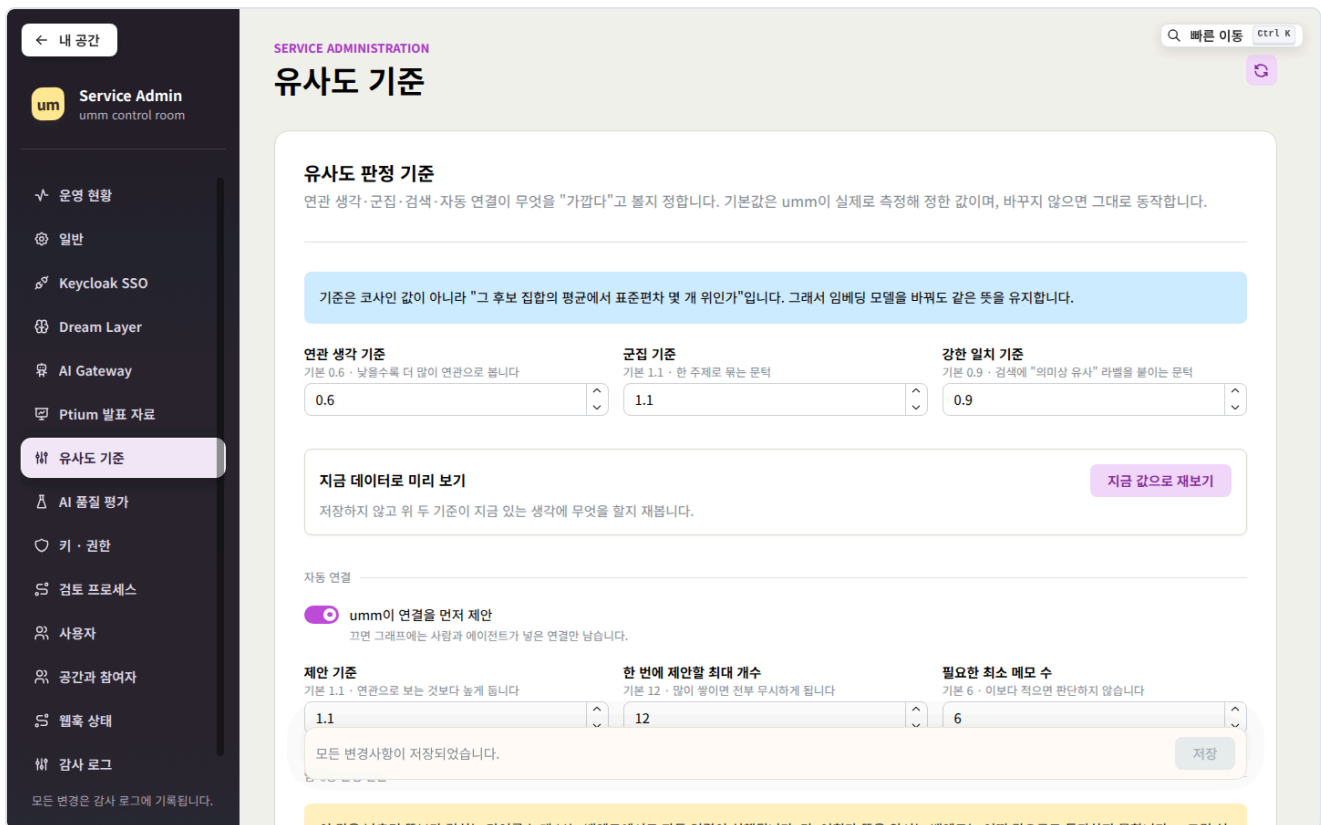
⬆

⬆

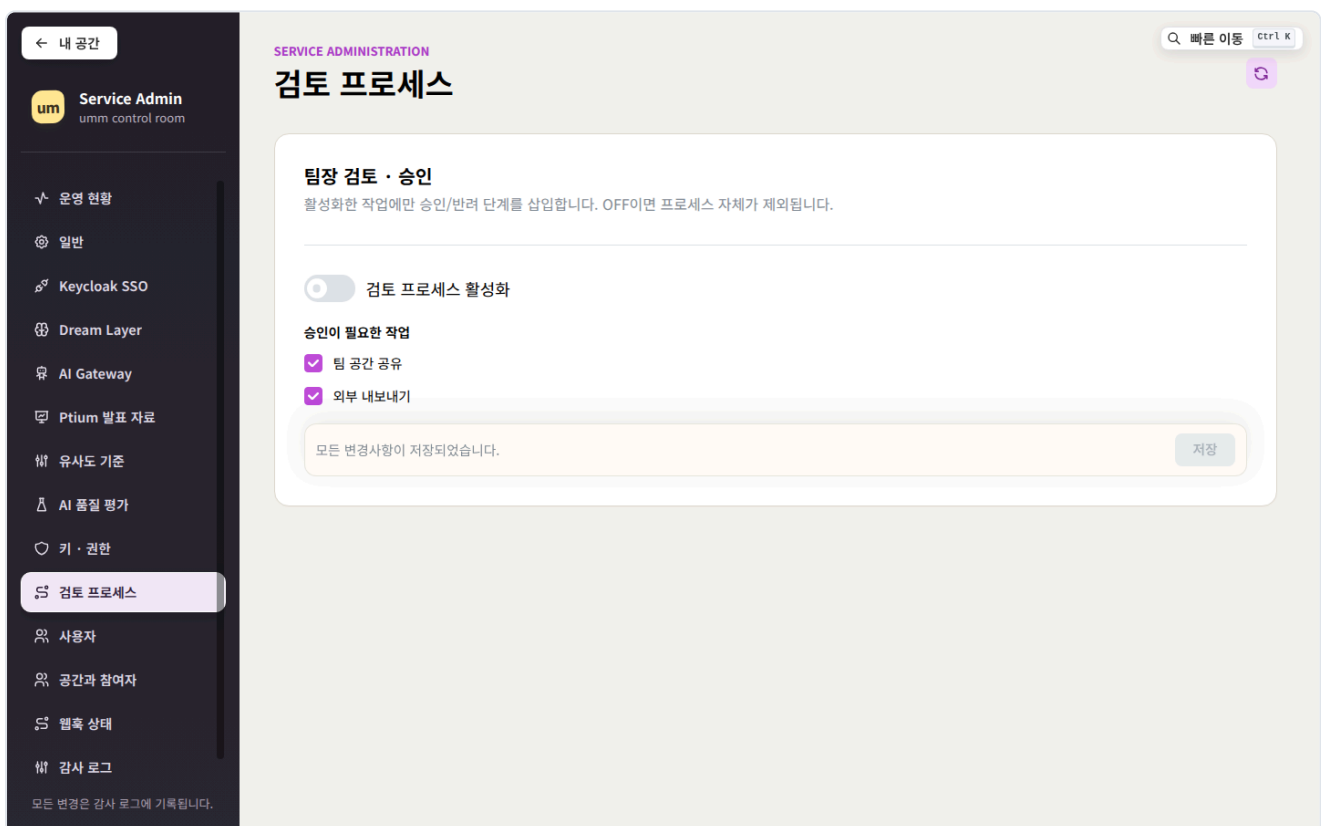
모든 변경사항이 저장되었습니다.

저장

Ptium 발표 자료 — Ptium 주소, API 키, 제한 시간, 연결 시험



유사도 기준 — 연관·강한 연관·군집·연결 추천 기준과 지금 값으로 재보기



검토 프로세스 — 승인이 필요한 작업을 고른다

4. 계정과 권한

역할은 셋입니다. 사용자는 bootstrap 관리자가 만들거나 Keycloak SSO 첫 로그인으로 생깁니다 — 사용자 생성 API 는 없습니다. 로컬 계정을 더 두려면 SSO 를 붙이거나 관리자가 DB 에서 만들어야 합니다.

역할	할 수 있는 일
admin (관리자)	서비스 설정 전부, 사용자 역할 변경·비활성화, 모든 공간과 참여자 열람·소유권 이전, 감사 로그, Dream 수동 실행, /api/v1/metrics
team_lead (팀장)	일반 기능 + 검토 · 승인 화면에서 자기 팀의 요청 결재(자기 요청·다른 팀 요청은 불가)
user (사용자)	자기 공간과 공유받은 공간, 개인 설정, API·MCP 키 발급

SSO 를 쓰면 Keycloak SSO 화면의 관리자 그룹·팀장 그룹으로 역할이 매핑됩니다.

← 내 공간

um

Service Admin

umm control room

▼ 운영 현황

⚙ 일반

🔑 Keycloak SSO

🏠 Dream Layer

🔗 AI Gateway

📄 Ptium 발표 자료

📋 유사도 기준

📊 AI 품질 평가

🔑 키 · 권한

🔄 검토 프로세스

👤 사용자

👤 공간과 참여자

📊 헬록 상태

📋 감사 로그

모든 변경은 감사 로그에 기록됩니다.

SERVICE ADMINISTRATION

사용자

🔍 빠른 이동

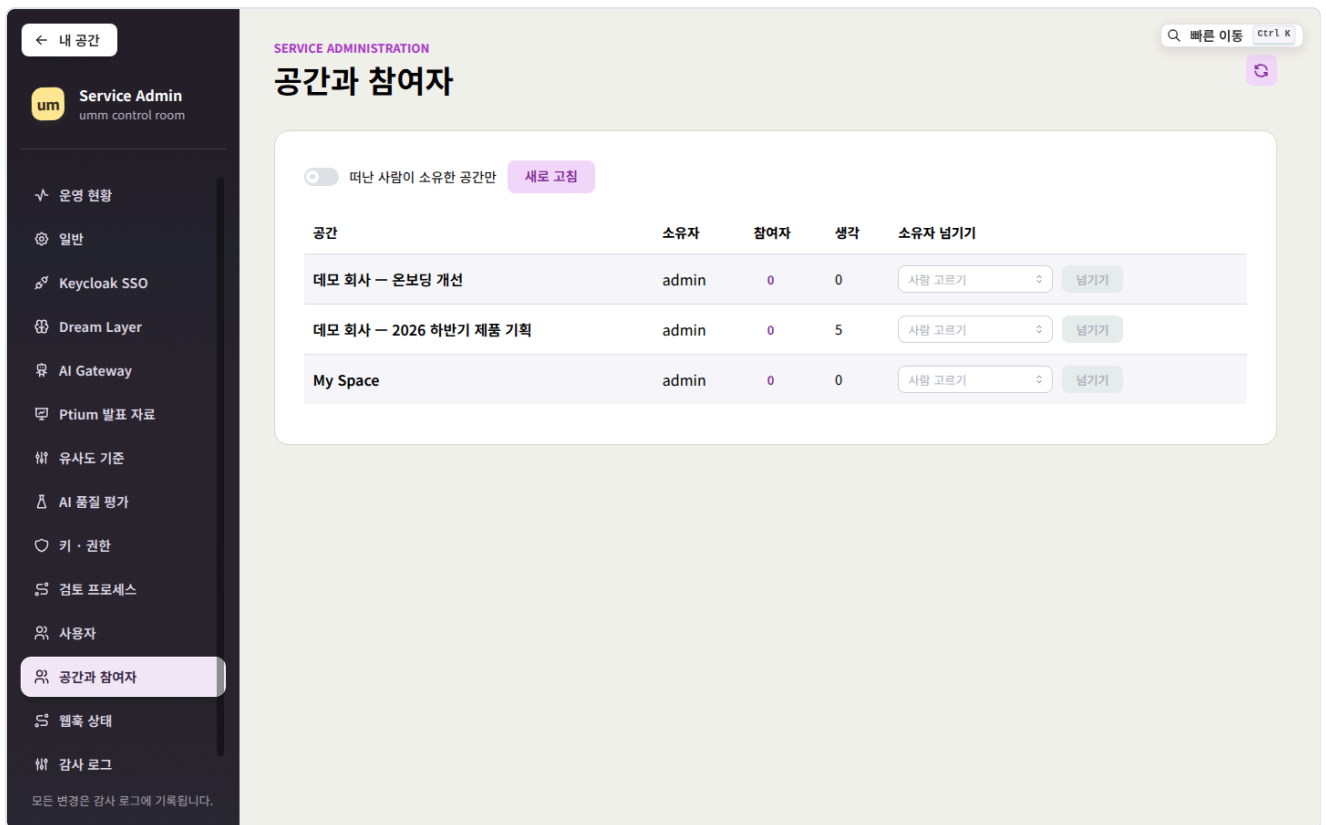
ctrl K

🔄

Keycloak 그룹 매핑 또는 여기에서 역할과 팀을 변경할 수 있습니다.

사용자	역할	팀	상태
admin admin ·	관리자	팀 없음	

사용자 — 역할 변경과 비활성화. 서비스 관리자·팀장·사용자 셋 중 하나



공간의 참여자 — 모든 공간의 소유자와 참여자를 보고 소유권을 옮긴다

공간 안의 권한(보기·편집·관리)은 공간 소유자가 **이 공간 함께 쓰기**에서 줍니다. 보기 도 댓글은 쓸 수 있습니다.

개인 API·MCP 키는 사용자가 스스로 만들되 **키 · 권한** 화면의 허용 스크프 안에서만 고를 수 있습니다. 현재 스크프: `notes:read` `notes:write` `spaces:read` `dreams:read` `approvals:write` `webhooks:write` `metrics:read` `ai:assist`. `ai:assist` 만이 외부 Gateway 호출과 AI 쿼터 소비를 허용합니다.

← 내 공간

um

Service Admin

umm control room

▼ 운영 현황

⊗ 일반

⚙ Keycloak SSO

⊞ Dream Layer

🔗 AI Gateway

📄 Ptium 발표 자료

📋 유사도 기준

📈 AI 품질 평가

🔑 키 · 권한

🔄 검토 프로세스

👤 사용자

👤 공간과 참여자

📊 헬록 상태

📋 감사 로그

모든 변경은 감사 로그에 기록됩니다.

SERVICE ADMINISTRATION

키 · 권한

🔍 빠른 이동

ctrl K

↺

개인 키 권한 체계

사용자가 자신의 키에 부여할 수 있는 권한과 회전 정책입니다.

허용 API/MCP Scopes

notes:read

notes:write

spaces:read

dreams:read

approvals:write

webhooks:write

metrics:read

ai:assist

기본 키 만료

회전 중첩 시간

90 일

24 시간

모든 변경사항이 저장되었습니다.

저장

남용 방지

로그인 실패와 요청 폭주로부터 서비스를 보호합니다. 값은 즉시 적용됩니다.

로그인 실패 허용 횟수

로그인 잠금 시간

8

15 분

모든 변경사항이 저장되었습니다.

저장

600

6

하루 AI 생성 한도

키 · 권한 — 허용 스코프, 기본 만료, 회전 중첩, 남용 방지 한도

5. 운영

5-1. 상태 점검

라우트 등록(`internal/httpapi/server.go`)에서 확인한 메서드입니다.

메서드	경로	용도	인증
GET	<code>/healthz</code>	프로세스 생존과 버전 <code>{"status":"ok","version":"0.71.6"}</code>	없음
GET	<code>/readyz</code>	PostgreSQL ping. 실패하면 503 database unavailable	없음
GET	<code>/api/v1/metrics</code>	Prometheus — route 별 요청 수·지연·in-flight·build 정보· umm_realtime_listener_up	관리자 세션 또는 <code>metrics:read</code> 키
GET	<code>/api/v1/meta</code>	버전·SSO 여부 등 공개 메타	없음
POST	<code>/mcp</code>	Model Context Protocol	API 키
POST	<code>/api/v1/admin/dreams/run</code>	Dream 을 지금 한 번 돌림	관리자

운영 현황의 실시간 협업 카드가 "폴백 폴링"이면 LISTEN 이 끊겨 1초 폴링 중이라는 뜻입니다. 협업은 되지만 DB 부하가 오르므로 umm_realtime_listener_up 에 알람을 걸어 두세요.

5-2. 로그

컨테이너 표준 출력에 JSON 한 줄씩(slog)입니다. docker compose logs -f umm . 요청 로그에는 requestId 가 있고 오류 응답에도 같은 값이 실리므로 사용자가 보낸 화면의 requestId 로 그 요청을 찾습니다.

5-3. 백업과 복구

상태는 전부 PostgreSQL 에 있습니다. 백업은 두 가지입니다 — DB 덤프와 ENCRYPTION_KEY . 키 없이는 덤프 속 OIDC·AI·Ptium·웹훅 secret 을 읽을 수 없습니다.

```
docker compose exec -T postgres pg_dump -U umm -Fc umm > umm-$(date +%F).dump
```

복구는 새 DB 에 pg_restore 한 뒤 같은 ENCRYPTION_KEY 로 umm 을 띄웁니다. scripts/restore-smoke.sh 가 이 순서를 카나리 데이터로 확인하는 절차입니다. 사용자 단위 백업은 캔버스의 내보내기 → Markdown 이며(그림은 담기지 않음), 같은 메뉴의 마크다운 가져오기로 돌아옵니다.

5-4. 업그레이드와 되돌리기

```
# 1. 백업
docker compose exec -T postgres pg_dump -U umm -Fc umm > before-upgrade.dump

# 2. 새 이미지 반입
gzip -dc umm-v0.71.6.tar.gz | docker load

# 3. 같은 환경 변수로 교체
docker compose up -d umm

curl -s http://127.0.0.1:8080/healthz
```

마이그레이션은 forward 로만 자동 적용됩니다. 되돌릴 때는 이전 이미지로 돌리기 전에 migrations/down/ 의 해당 스크립트를 새것부터 순서대로 직접 실행합니다 — 컬럼을 지우는 일이므로 자동으로는 절대 돌지 않습니다.

```
psql "$POSTGRES_DSN" -v ON_ERROR_STOP=1 -f migrations/down/027_note_attachments.down.sql
```

026 의 down 은 SELECT 1 뿐입니다 — 협업 로그에서 지운 본문 사본은 되돌려도 돌아오지 않습니다. 복수 인스턴스는 scripts/multi-instance-smoke.sh , 마이그레이션은 make migrate-dry-run 으로 미리 확인합니다.

5-5. 감사 로그

← 내 공간

um

Service Admin

ummm control room

▼ 운영 현황

⊗ 일반

🔑 Keycloak SSO

🔗 Dream Layer

🔗 AI Gateway

📄 Ptium 발표 자료

📋 유사도 기준

📈 AI 품질 평가

🔑 키 · 권한

🔄 검토 프로세스

👤 사용자

👤 공간과 참여자

🔄 웹훅 상태

📋 감사 로그

모든 변경은 감사 로그에 기록됩니다.

SERVICE ADMINISTRATION

🔍 빠른 이동

ctrl K

↺

감사 로그

행위자

아이디 또는 system

작업

전체

대상 ID

공간·메모·키의 ID

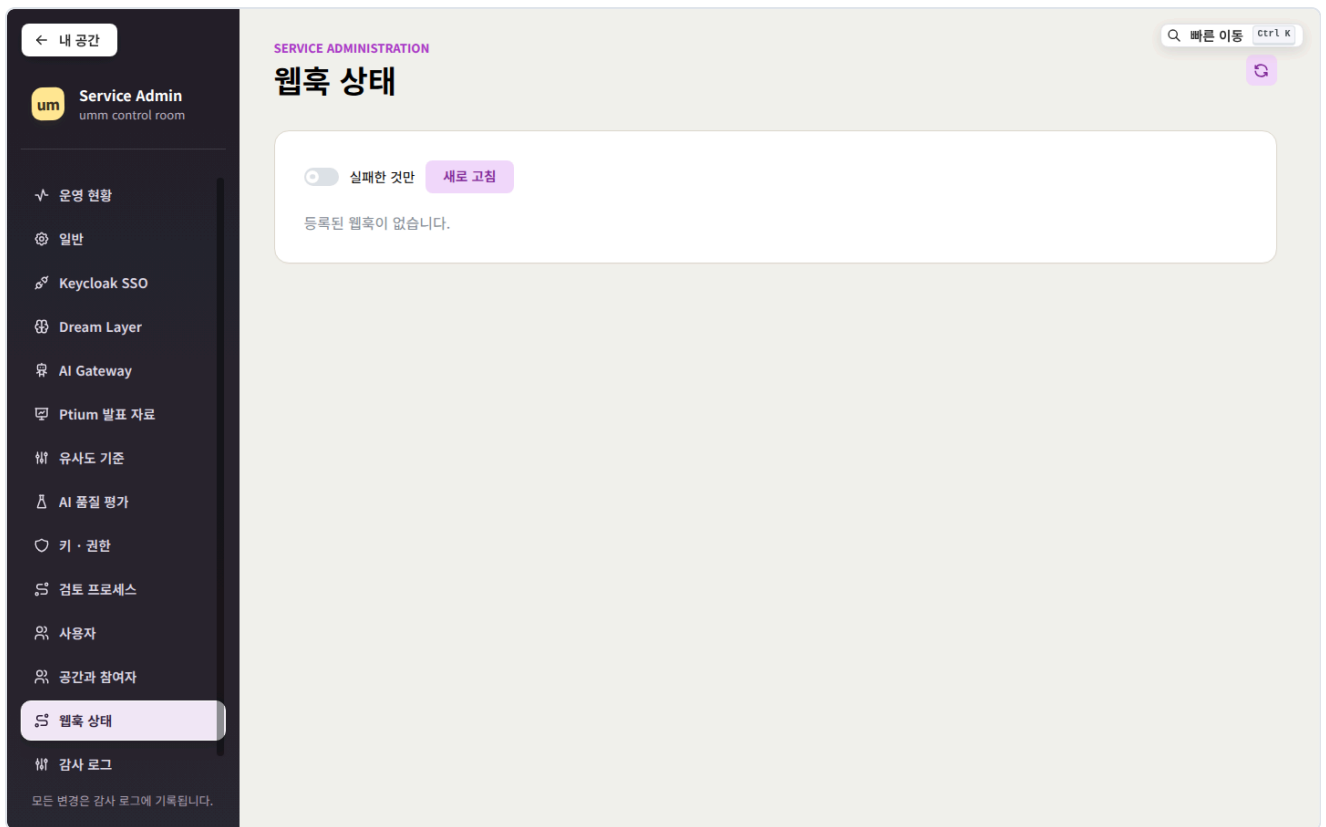
🔍

시각	행위자	작업	대상
2026. 9. 11. 오전 8:45:21	admin	NOTE.CREATE	note · 7682b019-7a7a-4c3d-b4c9-ed2ddf3b74e9
2026. 9. 11. 오전 8:45:21	admin	NOTE.CREATE	note · 19621a92-f195-4309-a64c-edff3f6fd9dc
2026. 9. 11. 오전 8:45:21	admin	NOTE.CREATE	note · f8ca81e4-5f64-4908-9ebd-62a0e39bd020
2026. 9. 11. 오전 8:45:21	admin	NOTE.CREATE	note · aa7d88b3-44c1-45d2-ade8-b15b2cd029dd
2026. 9. 11. 오전 8:45:21	admin	NOTE.CREATE	note · 95b709f5-8346-47bc-9253-ccc4d1b12fbf
2026. 9. 11. 오전 8:45:21	admin	SPACE.CREATE	space · d401f7d8-3cbe-4ee8-8098-20e7b8072999
2026. 9. 11. 오전 8:45:21	admin	SPACE.CREATE	space · 4f6ba8c2-cada-4faf-9c8b-966349452dc5
2026. 9. 11. 오전 8:45:20	admin	AUTH.LOCAL.LOGIN	user · a680b67f-ac58-45c6-aa6a-bedc20a85db5
2026. 9. 11. 오전 8:45:11	admin	SPACE.CREATE	space · 013000eb-d86f-48d2-9d2c-9435e0feb927
2026. 9. 11. 오전 8:45:11	admin	SPACE.CREATE	space · 871c07a3-818a-41cc-89a0-41c2edbb6894
2026. 9. 11. 오전 8:45:10	admin	AUTH.LOCAL.LOGIN	user · a680b67f-ac58-45c6-aa6a-bedc20a85db5

감사 로그 — 시각·행위자·작업·대상. 설정 변경, 역할 변경, 키 발급, 승인, 웹훅 멈춤이 남는다

audit_logs 테이블에 영구 보존됩니다. 관리자의 설정 저장, 역할 변경, 공간 소유권 이전, webhook.pause 등이 남고 지우는 API 는 없습니다.

5-6. 웹훅



웹훅 상태 — 설치된 모든 웹훅을 나쁜 것부터. 연속 실패, 마지막 오류, 소유자, 멈추기

사용자가 개인 설정에서 만든 웹훅이 실패하고 있으면 여기서 보이고 **멈추기**로 `active` 만 내립니다. 자세한 것은 부록 9.

6. 장애 대응

증상	확인할 곳	조치
컨테이너가 바로 죽음, 로그 invalid startup configuration	필수 환경 변수 넷. ENCRYPTION_KEY must be exactly 32 bytes... 면 키 길이	값을 고쳐 재시작
로그 database unavailable	POSTGRES_DSN, DB 기동, sslmode	/readyz 가 503 이면 같은 원인
로그 migration failed	DB 사용자의 extension 생성 권한 (pgcrypto · citext · pg_trgm), 디스크	권한 부여 후 재시작. 스키마는 트랜잭 션이라 반쯤 적용되지 않음
로그 bootstrap admin failed	BOOTSTRAP_ADMIN 이 빈 값이거나 DB 쓰기 실 패	—
로그 encryption initialization failed	키 회전 중 ENCRYPTION_KEY_PREVIOUS 형식	선포로 구분한 32바이트 키인지
로그 collaboration listener disconnected / ... disabled because the pool cannot reserve two request connections	pool_max_conns 가 1~2 이거나 DB 연결 한 도	운영에서는 4 이상. 풀백 폴링 중이라 협업은 계속됨
사용자에게 "짧은 시간에 너무 많 은 요청이 들어왔습니다"	키 · 권한 → 남용 방지 (분당 API 600, AI 분당 6·하루 80). 분당 API 는 인스턴스별이라 실효 상한은 설정값 × 인스턴스 수	한도 조정. 저장 즉시 적용
로그인 잠김	같은 주소 8회 실패 → 15분. 계정별은 그 3배	기다리거나 남용 방지 값 조정
Dream 이 안 옴, 로그 dream generation failed / dream eligibility failed	AI Gateway 주소·키·timeout, Dream Layer 의 자동 생성 스위치와 시각, 최소 메모 수 2·최 근 14일	AI Gateway → 연결 테스트. POST /api/v1/admin/dreams/run 으로 즉시 재현
연관 생각·군집이 전부 비어 있음	AI Gateway → 임베딩 품질 측정이 🟡/🔴	임베딩 모델 설정 또는 주소·모델명 확인. 내장 임베딩은 어휘만 켜니다
로그 AI prompt log encryption failed / AI call log write failed	키 · 권한 → 암호화 상태의 unreadable	unreadable=0 이 되도록 키 복구 후 현재 키로 회전
발표 만들기 실패 (unreachable · unauthorized · no-api · remote-error)	Ptium 발표 자료 → 연결 시험	부록 8-2 의 표
발표 만들기 timed-out	timeout_seconds 기본 30	큰 공간이면 상향

증상	확인할 곳	조치
웹훅 연속 실패	웹훅 상태 화면의 마지막 오류	수신 쪽 확인. 10회 연속 실패면 자동 중지
배포 뒤 옛 화면이 남음	proxy/CDN 이 /manifest.webmanifest · /umm-sw.js · /umm-icon.svg · /asset-manifest.json 을 immutable 로 캐시	그 넷은 no-cache 로 재검증되게

7. 보안

- **바뀌어야 하는 기본값** — `compose.yaml` 의 `local-development-only` · `change-this-before-production` · `0123456789abcdef...` 는 개발용입니다. 셋 다 바꾸고 나서 띄웁니다.
- **밖에 열면 안 되는 것** — PostgreSQL (5432), embeddings (11434). umm 의 8080 도 TLS proxy 뒤에만 둡니다. `/api/v1/metrics` 는 관리자 세션이나 `metrics:read` 키가 있어야 하지만 route 별 지연을 드러내므로 수집기에만 엽니다.
- **Proxy** — `UMM_TRUSTED_PROXY_CIDRS` 를 proxy 의 실제 대역으로만. 비우면 forwarding header 를 전부 버리므로 잠금·요청 제한이 proxy 주소 하나에 걸립니다. `0.0.0.0/0` 은 클라이언트가 주소를 위조하게 합니다.
- **인증 연동** — Keycloak OIDC(부록 2). Client 는 Confidential, redirect URI 는 `https://<umm>/api/v1/auth/oidc/callback` . 로컬 로그인은 남용 방지의 잠금 규칙을 따릅니다(부록 4-1).
- **키** — `ENCRYPTION_KEY` 는 백업과 함께 별도 비밀 저장소에. 회전은 부록 8. API 키는 사용자가 만들되 스코프는 관리자가 허용한 범위 안이고, `notes:read` 만으로는 외부 AI 호출이 일어나지 않습니다.
- **AI 로 나가는 것** — 생각 본문은 사용자가 Dream 분석에서 제외 하거나 공간에서 AI Dream 분석 을 끄면 어디로도 나가지 않습니다. 임베딩 API 키는 채팅 키와 따로이며 비우면 보내지 않습니다. 개인 설정의 내 AI 사용 내역이 사용자에게 같은 사실을 보여 줍니다.
- **감사** — 관리 행위는 전부 감사 로그에 남고 지울 수 없습니다.
- 자세한 위협 모델은 [SECURITY.md](#).

부록 — 화면별 상세

아래는 각 관리자 화면의 값과 동작을 자세히 적은 것입니다. 번호는 위 표의 "부록 n" 과 같습니다.

부록 2. Keycloak OIDC SSO 연동

umm 은 Keycloak OpenID Connect Discovery를 통해 복잡한 설정 없이 엔터프라이즈 SSO를 즉시 연동합니다.

1. Keycloak Client 생성:

- Client type: OpenID Connect
- Client authentication: ON (Confidential)

- Standard Flow: ON
- Valid redirect URIs: `https://<umm-domain>/api/v1/auth/oidc/callback`

2. 관리자 콘솔 설정 (/admin/oidc):

- Keycloak SSO 활성화 스위치 ON
- Issuer URL: `https://keycloak.company.internal/realms/enterprise`
- Client ID & Client Secret: Keycloak에서 발급받은 자격증명 입력
- 관리자 그룹/역할: `umm-admins`
- 팀장 그룹/역할: `umm-leads`

3. 연결 시험:

- 연결 시험 버튼을 클릭하여 OIDC Discovery 및 토큰 엔드포인트 도달 가능 여부를 실시간 검증합니다.

부록 3. Dream Layer & 야간 Scheduler 운영

Dream Layer는 사용자가 밤사이 휴식하는 동안 캔버스에 쌓인 생각들의 의미적 연관성을 분석하고 새로운 아이디어를 제안하는 핵심 백그라운드 엔진입니다.

생성 결과는 v0.6부터 캔버스에 즉시 삽입되지 않고 개인 Dream 검토함에 후보로 저장됩니다. 사용자가 채택할 때만 Dream 메모와 원본 연결선이 함께 생성되며, 기존 버전에서 이미 캔버스에 생성된 Dream은 업그레이드 시 채택 상태로 보존됩니다.

• 스케줄 설정:

- 자동 생성 스위치 ON
- 생성 시간: 기본 02:00 (사내 심야 시간대)
- 생성 주기: 매일(Daily), 평일(Weekdays), 주말(Weekends), 특정 요일 선택, N일 간격

• 컨텍스트 & 256K 토큰 한도:

- 최소 메모 개수: 2개 이상 메모가 있어야 분석 시작
- 분석 범위: 최근 14일
- 최대 응답 Token: 4K부터 **최대 256K (262,144 tokens)**까지 모델 사양에 맞춰 슬라이더로 조절

• 품질 기준선 (Quality Threshold):

- 원본 두 개 이상에 대한 근거성, 적정 새로움, 구체성, 출처 범위를 결합한 내부 최소 점수
- Quiet Mode: 가치가 확실하고 영감을 주는 의미 있는 Dream이 없을 경우 불필요한 노이즈 생성을 건너뛰니다.

• 선택 및 개인정보 보호:

- 최근 Dream이 적은 적격 공간을 순환하고, 연결되지 않았으나 의미적으로 이어질 수 있는 메모 조합을 우선합니다.
- 메모 또는 공간에서 AI 제외를 켜면 Scheduler 자격 계산, Dream 생성, AI Assist에서 모두 제외됩니다. 선택적 임베딩 모델을 설정했더라도 제외 콘텐츠는 Gateway로 전송하지 않고 서버 안의 로컬 vector만 사용합니다. 외부 임베딩 직전 note·space와 Gateway 설정을 다시 잠그고 응답 저장까지 유지하므로, 제외 설정이 먼저 적용되면 본문을 보내지 않고 임베딩이 먼저 시작되면 설정 변경이 호출 종료까지 기다립니다. 메모 하나

만 제외된 공간도 Canvas와 범위 검색 전체가 같은 로컬 vector 공간을 사용해 일반 메모의 의미 검색 결과가 사라지지 않습니다. 원격 범위 검색은 space·membership·활성 note를 검색 완료까지 잠그며 제외·접근 회수 뒤의 검색어는 Gateway에 보내지 않습니다.

- **운영 지표:**

- 단순 생성 수 외에 검토 완료, 채택률, 편집·연결·확장 기반 유의미 활용률, 채택 Dream당 비용을 함께 확인합니다. 노출 수는 검토함을 불러온 횟수가 아니라 카드가 화면에 50% 이상 실제 표시된 경우만 반영됩니다.

부록 4. 내부 AI Gateway 연동

사내망 내부의 LLM Gateway (vLLM, Ollama, TGI, SGLang 등 OpenAI 호환 서버)를 연결합니다.

- **Base URL:** `http://llm-gateway.internal:8000`, `.../v1`, 전체 `.../chat/completions` 주소를 모두 사용할 수 있습니다.
- **API Key:** 내부 보안 게이트웨이 인증 토큰
- **Timeout:** 긴 추론 모델을 위해 최대 1800초까지 설정 가능. 이 값은 재시도를 모두 포함한 한 AI 작업의 전체 시간 예산이며, umm HTTP write timeout은 최대값보다 60초 길게 설정됩니다. 앞단 reverse proxy도 설정값보다 최소 60초 길게 응답 timeout을 구성하세요.
- **재시도:** 0~5회. 전체 Timeout 안에서만 수행되므로 재시도마다 1800초가 다시 부여되지 않습니다.
- **vLLM 추론 모델:** 가능하면 서버에 모델별 `--reasoning-parser` 를 설정합니다. 최종 본문 없이 `reasoning / reasoning_content` 만 반환하거나 `<think>` 도중 출력 한도에 도달하면, 재시도가 1 이상일 때 umm이 비추론 모드로 다시 요청하고 최종 `content` 만 사용합니다.
- **비용 통계 관리:** 입력/출력 1M 토큰당 비용을 입력하면 관리자 대시보드에서 월간 예상 비용과 사용자당 비용을 실시간으로 추산합니다.
- **임베딩 모델:** 비워 두면 외부 호출 없이 내장 로컬 임베딩(문자 n-gram)을 사용합니다. 내장 알고리즘은 어휘 중복을 재며 동의어를 인식하지 못합니다(라벨 데이터셋 기준 쌍별 정확도 4.2%). 지금 붙어 있는 백엔드가 어느 쪽인지는 아래 **임베딩 품질 측정**에서 바로 확인할 수 있습니다. 연관 생각·군집·Dream 품질을 실제로 올리려면 모델 설정이 사실상 필수입니다. 유사도 판정 기준은 v0.9.0부터 백엔드 분포에 맞춰 자동 조정되므로, 모델을 바꿔도 임계값을 손볼 필요가 없습니다. 모델 이름을 넣으면 같은 Gateway의 `/v1/embeddings` 를 사용해 연관 생각·군집·검색의 의미 점수를 계산합니다. 모델이나 Gateway 주소를 바꾸면 endpoint fingerprint가 달라져 기존 생각이 열릴 때마다 점진적으로 다시 임베딩되며, 같은 모델명을 쓰는 서로 다른 제공자의 벡터도 섞여 비교되지 않습니다. 설정 변경 전에 시작한 이전 Gateway 응답은 저장 시점에 현재 설정과 다시 대조되어 폐기되므로 새 벡터를 뒤늦게 덮어쓰지 못합니다. 외부 호출과 vector 저장 동안에는 대상 note·space의 AI 제외 상태와 현재 Gateway 세대를 AI 전용 lease로 고정하므로 관리자가 제외를 저장한 뒤 캡처된 본문이 늦게 전송되는 경합도 없습니다. Gateway가 응답하지 않으면 전체 비교 집합을 로컬 임베딩으로 맞추고 5분간 로컬 회로 차단 상태를 유지해 Canvas를 열 때마다 원격 timeout이 반복되지 않습니다. 정책에 따른 로컬 전환은 장애로 세지 않아 회로 차단기를 열지 않습니다. 암호화된 API key를 현재 keyring으로 읽을 수 없는 경우에도 저장된 ciphertext를 Gateway 자격증명으로 시도하지 않고 즉시 로컬 provider로 답습니다. 이 전체 로컬 정규화 pass도 장애가 난 원래 Gateway 설정 세대에 묶여 있으므로, 두 pass 사이에 저장한 새 Gateway의 vector를 덮지 않습니다. 임베딩과 Dream 오류 응답 본문 일부는 다국어 rune 경계를 보존해 관리자 오류와 로그가 유효한 UTF-8을 유지합니다. 설정을 저장하면 회로 차단 상태가 즉시 해제되어 새 설정을 확인합니다.

임베딩 Gateway를 따로 두기

임베딩과 채팅 모델이 같은 서버에 있을 이유는 없습니다. 큰 모델은 어딘가 강한 장비에 두고, 임베딩은 umm 옆에 작게 띄우는 편이 흔합니다 — `compose.yaml` 의 `embeddings` 프로파일이 정확히 그 모양입니다.

항목	비우면	적으면
<code>embedding_base_url</code>	위의 Base URL 을 씁니다 (기존 동작 그대로)	임베딩만 이 주소로 보냅니다
<code>embedding_api_key</code>	인증 헤더를 보내지 않습니다	이 키로 인증합니다

채팅 API Key는 임베딩 주소로 따라가지 않습니다.
한 호스트에 발급된 키는 그 호스트의 자격증명입니다. 임베딩 키를 비워 뒀다는 이유로 채팅 키를 대신 보내면, 그 서버를 운영하는 사람에게 키를 건네주는 셈입니다. 인증이 필요 없는 임베딩 서버가 흔하므로 **비움 = 보내지 않음**입니다.

주소가 잘못되면 **저장을 거부**합니다. 조용히 무시하면 임베딩이 내장 알고리즘으로 돌아간 채, 설정 화면에는 저장된 값이 그대로 보이게 됩니다.

```
docker compose --profile embeddings up -d
docker compose exec embeddings ollama pull bge-m3
```

그 다음 임베딩 Gateway 주소에 `http://embeddings:11434` , 임베딩 모델에 `bge-m3` 을 넣으면 됩니다. 채팅 모델 주소는 그대로 두어도 됩니다.

임베딩 품질 측정

관리자 → **AI Gateway** → **임베딩 품질 측정**은 지금 설정된 백엔드로 라벨링된 한/영 문장쌍 22개를 직접 임베딩해 점수를 냅니다. 두 숫자만 보면 됩니다.

- **판별력** — 같은 뜻, 다른 표현 의 평균에서 단어만 겹침 의 평균을 뺀 값. 음수면 뜻보다 어휘를 높게 보고 있다는 뜻입니다.
- **쌍별 정확도** — 패러프레이즈와 어휘 함정을 짝지어 비교했을 때 올바른 순서를 매기는 비율. 임계값과 무관합니다.

세 가지 상태가 나옵니다.

표시	뜻	할 일
● 어휘가 겹치는 정도만 재고 있습니다	모델 미설정. 연관 생각·군집·검색의 "의미상 유사"는 실제로는 어휘 유사입니다	임베딩 모델을 설정
● 설정한 모델이 쓰이지 않고 있습니다	모델은 설정됐지만 벡터가 로컬에서 나왔습니다. 게이트웨이 무응답이거나 모델 이름 오타	주소·모델명 확인
● 의미 기반으로 동작합니다	표현이 달라도 같은 뜻을 알아봅니다	—

결과는 백엔드별로 10분 캐시되며, 설정을 바꾼 뒤에는 다시 측정을 누르세요.

오프라인 임베딩 서버 띄우기

제대로 된 임베딩 모델을 umm 이미지에 넣으려면 추론 런타임을 링크해야 하고, 그러면 umm을 단일 이식 바이너리로 만드는 CGO_ENABLED=0 정적 빌드가 깨집니다. 대신 옆에 세우면 정적 빌드를 유지하면서도 완전 오프라인으로 돌릴 수 있습니다. 모델은 한 번만 받으면 이후에는 볼륨만 있으면 됩니다.

```
docker compose --profile embeddings up -d
```

컨테이너가 뜨면서 모델을 스스로 받습니다(UMM_EMBEDDING_MODEL 로 변경 가능, 기본 bge-m3). 볼륨에 이미 있으면 다시 받지 않고, healthcheck는 서버 응답이 아니라 **모델이 실제로 있는지를** 확인합니다.

그다음 AI Gateway에서 **자동으로 찾기**를 누르면 주소와 모델 목록이 나옵니다. 모델을 고르고 **연결 테스트**로 확인한 뒤 저장하고, **임베딩 품질 측정**에서 ●가 뜨는지 보면 끝입니다.

자동 찾기가 조사하는 주소는 바이너리에 고정되어 있습니다(embeddings:11434 , embeddings:8080 , host.docker.internal:11434 , 127.0.0.1:11434). 요청에서 주소를 받지 않습니다. 목록의 * 표시는 모델 이름으로 짐작한 것이며, 실제로 임베딩하는지는 연결 테스트가 확인합니다.

폐쇄망이라면 모델을 받아 둔 볼륨이나 ollama/ollama 이미지를 함께 반입하면 됩니다.

후보 모델 측정 결과

같은 데이터로ぜん 값입니다. 임의의 OpenAI 호환 게이트웨이라도 쓸 수 있습니다.

모델	크기	차원	판별력	쌍별 정확도	주제 분리	최근접 동일 주제	umm 중단 테스트
내장 로컬 (문자 n-gram)	0	192	-0.250	4.2%	-0.023	18.8%	✗
bge-m3	1.2GB	1024	+0.079	83.3%	+0.105	75.0%	✓
bona/bge-m3-korean	1.2GB	1024	+0.146	81.2%	+0.101	68.8%	✓
paraphrase-multilingual	562MB	768	+0.115	85.4%	+0.223	87.5%	✗

앞의 네 지표와 마지막 열이 서로 다른 순서를 매깁니다. 이건 측정 오류가 아니라 실제로 알아 둘 값입니다.

paraphrase-multilingual 은 라벨 데이터 네 지표에서 모두 1위이고 크기도 절반 이하지만, umm의 중단 클러스터링 테스트(보안 3 + 자전거 3)에서 실패합니다. 인증 토큰 만료 시간을 24시간으로 정했다 의 최근접 이웃으로 라이딩이 끝나면 스트레칭을 꼭 한다 (0.32)를 고르기 때문입니다 — 같은 주제인 세션 쿠키... (0.23)보다 높게. 한 문장이 틀렸을 뿐인데 6개 메모 공간에서는 두 주제가 한 덩어리로 합쳐집니다.

즉 집계 점수가 높다고 특정 워크스페이스에서 더 나은 군집이 나오는 것은 아닙니다. 모델마다 틀리는 문장이 다르고, 메모 수가 적을수록 한 번의 실수가 결과를 지배합니다.

- 기본 권장은 bge-m3 입니다. 라벨 지표는 중간이지만 umm이 실제로 단언하는 종단 동작을 통과합니다.
- 한국어 비중이 높고 무관한 생각이 섞이는 쪽이 더 신경 쓰이면 판별력이 가장 높은 bona/bge-m3-korean .
- 디스크나 메모리가 빠듯하면 paraphrase-multilingual (562MB)도 실용적입니다. 다만 위 한계를 알고 쓰시고, 설정 후 실제 공간에서 연관 생각과 군집을 눈으로 확인하세요.

무엇을 고르든 임베딩 품질 측정에서 ●가 뜨는지, 그리고 실제 공간에서 군집이 납득되는지 두 가지를 함께 보는 것이 가장 확실합니다.

지표가 뜻하는 것:

- 판별력 / 쌍별 정확도 — 뜻이 같은 문장쌍과 단어만 겹치는 문장쌍을 구별하는 능력.
- 주제 분리 / 최근접 동일 주제 — 라벨된 4개 주제 16문장에서, 같은 주제끼리 더 가까운지와 각 문장의 가장 가까운 이웃이 같은 주제인지. 연관 생각과 군집이 실제로 하는 일이 이쪽입니다.

직접 후보를 재려면:

```
UMM_EMBEDDING_TEST_URL=http://127.0.0.1:11434 \  
UMM_EMBEDDING_TEST_MODEL=<모델명> \  
go test ./internal/intelligence -run Gateway -v
```

종단 동작까지 확인하려면 (PostgreSQL 필요):

```
POSTGRES_DSN=... UMM_EMBEDDING_TEST_URL=http://127.0.0.1:11434 \  
UMM_EMBEDDING_TEST_MODEL=<모델명> \  
go test ./internal/store -run ClustersAndRelated -v
```

모델을 바꾸면 차원과 fingerprint가 달라져 기존 생각이 열릴 때마다 점진적으로 다시 임베딩됩니다. 유사도 판정 기준은 v0.9.0부터 백엔드 분포에 맞춰 자동 조정되므로 임계값을 손볼 필요는 없습니다.

부록 4-2. 유사도 기준 (/admin/intelligence)

저장하기 전에 재보기

기준은 표준편차 개수이므로 같은 숫자가 코퍼스에 따라 전혀 다른 일을 합니다. 그런데 무슨 일을 하는지 알아내는 방법은 저장하고 가서 보는 것뿐이었습니다. 저장하면 설치된 모든 캔버스가 한꺼번에 바뀌고, 바꾼 사람이 가장 늦게 압니다.

지금 값으로 재보기 는 아무것도 쓰지 않고 지금 있는 생각으로 두 기준을 나란히 재봅니다. 가장 큰 공간 다섯 개, 공간마다 400개까지를 표본으로 씁니다 — 비교는 공간 안에서 제공으로 늘어나고 관리자가 기다리는 동안 돌기 때문입니다.

보이는 것	뜻
연관 생각이 하나도 없는 카드	기준을 너무 올리면 서서히 나빠지는 것이 아니라 패널이 비어 버립니다 . 가장 먼저 볼 숫자입니다.
카드당 연관 생각 (중앙값) · 가장 많은 카드	카드에 실제로 뜨는 관련 N 칩
묶음 수 · 묶임 생각 · 가장 큰 묶음	멀리서 본 캔버스가 그릴 모양
혼자 남는 생각	묶이지 못해 하나씩 그려질 생각

임베딩이 의미 비교에 부적합하다고 판정돼 있으면 캔버스는 위치로 묶으므로 **군집 기준은 아무 일도 하지 않습니다**. 그럴 때는 그 사실만 알려 주고 **묶음 관련 네 줄은 아예 표시하지 않습니다** — 아무 일도 하지 않는다고 써 놓고 바로 밑에 묶음 개수를 적으면 방금 한 말을 도로 무르는 셈입니다. 연관 생각 쪽은 어느 백엔드에서나 채점되므로 그대로 나옵니다.

연관 생각·군집·연결 추천이 얼마나 과감할지를 정합니다. **거의 전부 분포 상대적입니다** — 백엔드마다 "가깝다"의 뜻이 다르므로, 이 공간의 유사도 분포에서 평균보다 몇 표준편차 위인지로 판단합니다. 모델을 바꿔도 손볼 필요가 없는 이유입니다.

항목	기본값	의미
<code>related_band</code>	0.6 σ	연관 생각으로 보여 줄 기준
<code>strong_band</code>	0.9 σ	강한 연관
<code>cluster_band</code>	1.1 σ	군집으로 묶을 기준
<code>autolink_band</code>	1.1 σ	연결 추천 기준
<code>autolink_min_notes</code>	6	이보다 적으면 추천하지 않습니다
<code>autolink_max_per_run</code>	12	한 번에 제안할 최대 개수
<code>semantic_accuracy_bar</code>	0.65	이 아래면 "의미를 못 쥌다"고 판단해 의미 기능을 끕니다
<code>semantic_purity_bar</code>	0.60	같은 기준의 최근접 동일 주제
<code>duplicate_similarity</code>	0.92 (절대값)	근접 중복 판정
<code>quality_cache_minutes</code>	10	품질 측정 캐시

두 하한을 낮춰도 넘을 수 없는 바닥

`semantic_accuracy_bar` 와 `semantic_purity_bar` 는 **기본선 바로 아래에 있는 백엔드**를 받아 주기 위한 것입니다. 한 언어·한 좁은 도메인만 쓰는 설치는 그럴 이유가 있습니다.

뜻보다 어휘를 높게 보는 백엔드는 두 값을 아무리 낮춰도 통과하지 못합니다. 의미 판정에는 세 번째 조건이 있고 — 판별력이 양수일 것 — 그건 설정이 아닙니다. 내장 char-gram 임베딩은 -0.25 입니다.

두 칸의 설명에 "내장 임베딩은 0.042", "내장 임베딩은 0.188" 이라고 적혀 있어서 그 바로 아래로 낮추면 될 것처럼 보이지만, 그렇게 해도 달라지지 않습니다. 화면의 안내문이 같은 말을 하고 있습니다. **의미 기능을 실제로 켜는 방법은 임베딩 모델을 설정하는 것뿐입니다.**

`duplicate_similarity` 만 **절대값인 것은 의도적입니다**. 측정해 보면 실제 중복은 두 모델에서 0.943~0.986에 모이고 그다음 등급은 0.681에서 끊깁니다. 그리고 상대 기준은 여기서 실패합니다 — 중복이 많은 공간은 자기 분포가 통째로 올라가서, **중복이 가장 많을 때 중복이 특별해 보이지 않게 됩니다.**

규모에 따른 경계

중복 검사는 모든 쌍을 비교하므로 제공으로 자랍니다(1,024차원 실측: 1,000개 229ms · 2,000개 916ms · 10,000개 ≈ 23초). 한 공간에서 **최근 1,000개**까지만 비교하고, 그 위로는 **오늘의 리뷰가 ***"공간이 커서 최근 생각까지만 비교했습니다"*라고 **말해 줍니다**. 접어 둔 갈래의 생각은 별도로 최근 창과 교차 비교되므로, 오래전에 거절한 결정을 오늘 다시 써도 잡힙니다.

이 화면의 모든 항목은 의미를 재는 **백엔드에서만** 효과가 있습니다. 내장 임베딩에서는 연관 생각·군집·중복·연결 추천이 전부 꺼지고, **오늘의 리뷰가 backend-not-semantic** 으로 건너뛴 사실을 표시합니다.

부록 4-1. 남용 방지 (키 · 권한 → 남용 방지)

로그인 실패와 요청 폭주로부터 서비스를 보호하는 값들입니다. 저장 즉시 적용되며 재시작이 필요 없습니다.

항목	기본값	의미
로그인 실패 허용 횟수	8	같은 주소에서 이만큼 실패하면 잠깁니다. 계정별 임계값은 자동으로 이 값의 3배입니다
로그인 잠금 시간	15분	잠금이 유지되는 시간
분당 API 요청	600	호출자별 상한
분당 AI 요청	6	AI 생성(/ai/assist , Dream 재생성·발전, 관리자 평가 실행) 전용 상한
하루 AI 생성 한도	80	사용자별 24시간 상한. 0 이면 제한하지 않습니다

롤링 배포 중 구버전 관리자 화면이 새 남용 방지 필드를 모른 채 보안 섹션을 저장해도, 서버는 생략된 로그인·API·AI 한도를 PostgreSQL의 최신 확정값에서 transaction으로 병합합니다. 명시한 값은 0 을 포함해 그대로 적용하며 `null` 이나 범위 밖 값은 저장 전에 거부하므로, 업그레이드 도중 운영자가 조정한 한도가 기본값으로 조용히 돌아가지 않습니다.

계정별 임계값을 주소별보다 높게 둔 이유는, 아이디를 아는 사람이 반복 실패를 일으켜 남의 계정을 마음대로 잠글 수 있는 상태를 피하기 위해서입니다. 로그인 성공은 해당 계정의 실패 횟수만 초기화하며, 같은 주소가 다른 계정에 누적인 실패 횟수는 유지되어 정상 계정 로그인으로 주소 제한을 우회할 수 없습니다.

로그인 실패 횟수와 하루 AI 사용량은 데이터베이스에서 관리합니다. 같은 주소·계정의 로컬 로그인은 모든 인스턴스가 공유하는 PostgreSQL 잠금 아래 비밀번호 확인과 실패 기록을 함께 처리하므로 병렬 요청도 설정한 실패 횟수를 초과해 검증되지 않습니다. 대화형 AI, 자동 Dream, 관리자 AI 평가는 실제 Gateway 호출 직전에 PostgreSQL에서 원자적으로 한도 자리를 선점하고 24시간 사용량으로 영속화하므로, 요청 취소·로그 저장 실패·여러 인스턴스의 동시 실행에도 하나의 상한이 적용됩니다. 분당 API 요청 한도는 인스턴스별로 계산되므로, 실효 상한은 $\text{설정값} \times \text{인스턴스 수}$ 입니다.

허용 API/MCP scope에는 마이그레이션 009가 `ai:assist`를 자동 추가합니다. 이 scope는 선택한 생각을 외부 Gateway로 전송하고 AI 쿼터를 소비하는 `/ai/assist`에만 사용됩니다. 읽기 자동화에는 `notes:read`만, AI 발전 자동화에는 `ai:assist`만 또는 실제 작업에 필요한 두 scope를 함께 발급해 최소 권한을 유지하세요.

부록 7. Dream AI 평가 회귀

관리자 → AI 평가에서 최소 두 개의 입력 생각, 기대 단어와 금지 단어, Dream 유형을 저장합니다. 실행은 현재 AI Gateway와 prompt version을 그대로 사용하며 grounding, 기대/금지 단어, 구체성, 모델 응답 상태를 0~1 점수와 세부 항목으로 보존합니다. 모델·prompt·Gateway 설정을 바꾸기 전후에 같은 active case를 실행해 회귀를 확인하세요. Gateway 장애도 `error` run으로 남아 평가 이력이 사라지지 않습니다.

부록 8. Master-key 회전

새 키를 `ENCRYPTION_KEY`, 현재 키를 `ENCRYPTION_KEY_PREVIOUS`에 배치한 뒤 재시작합니다. 보안 화면에서 fallback 1개 이상, unreadable 0을 확인하고 **현재 키로 회전**을 실행합니다. 이 작업은 OIDC/AI secret, 웹훅 secret, 암호화 AI prompt를 한 트랜잭션으로 다시 암호화하며 `enc: wrapper` 도입 전 raw v1 AI prompt도 현재 `wrapper·v2` 형식으로 정규화합니다. 회전 전에 열어 둔 OIDC·AI Gateway 화면에서 마스킹된 값을 동시에 저장해도 서버가 같은 설정 lock 뒤 최신 암호문을 병합합니다. pending 0을 확인하고 새 백업을 만든 후에만 이전 키 환경변수를 제거합니다.

부록 8-2. Ptium 발표 연동이 실패할 때

사용자 화면에는 **무엇을 해야 하는지**가 적히고, 그중 절반은 "관리자에게 알려 주세요"입니다. 그 절반이 여기입니다.

화면에 뜨는 말	failure	실제 원인	관리자가 할 일
Ptium에 연결하지 못했습니다	unreachable	아무것도 응답하지 않음	Ptium 프로세스와 네트워크 확인
Ptium이 제때 답하지 않았습니다	timed-out	제한 시간 초과	생각이 많은 공간이면 timeout_seconds 상향
Ptium이 umm의 자격 증명을 거부했습니다	unauthorized	401 · 403	Ptium API 키 재발급 후 저장
그 주소에 Ptium API가 없습니다	no-api	404	주소 오타, 또는 앞단 프록시가 가로챈
Ptium이 이 발표 구성을 받아들이지 않았습니다	rejected	4xx	관리자 일이 아닙니다. Ptium이 어느 슬라이드인지 알려 주므로 작성자가 고칩니다
Ptium 쪽에서 오류가 났습니다	remote-error	5xx	Ptium 로그 확인
Ptium이 예상과 다른 응답을 보냈습니다	unexpected-response	문서와 다른 형식	두 서비스 버전 확인
Ptium에는 만들어졌지만 umm이 기록하지 못했습니다	not-recorded	덱 생성 후 DB 기록 실패	그냥 다시 시도하면 덱이 하나 더 생깁니다. Ptium에서 먼저 확인

묶음 제목을 AI가 짓기

발표 만들기에서 AI가 관여하는 곳은 여기 하나뿐이고, 그것도 제목뿐입니다.

- **켜야 동작합니다.** 기본은 꺼져 있고, dream 설정의 모델이 없으면 옵션 자체가 아무 일도 하지 않습니다
- **묶음으로 만들어진 슬라이드만 대상입니다.** 사람이 제목을 붙인 생각이나, 근거를 붙여 놓은 주장은 이미 자기 제목이 있습니다
- **한 번의 호출이고 최대 40묶음까지입니다.** 수백 묶음짜리 덱이 프롬프트 하나에 공간 전체를 담지 않도록
- **본문은 절대 안 바꿉니다.** 시험이 이걸 직접 확인합니다 — 제목만 바꾸고 lead·points·From 이 전부 그대로인지
- **실패하면 그냥 원래 제목입니다.** 게이트웨이가 죽었다고 발표를 못 만들게 하지 않습니다
- 나가는 본문은 `Complete` 안에서 **redaction** 을 거칩니다. 호출자가 빠뜨릴 수 없도록 공유 진입점 안에 넣었습니다

돌아온 답이 문장이거나, 여러 줄이거나, JSON 객체면 쓰지 않고 원래 제목을 유지합니다. 이해 못 한 답으로 제목을 만들면 청중 앞 화면에서야 발견됩니다.

긴 발표를 부로 나누기

묶음 제목 과 별개 옵션이고, 역시 기본은 꺼져 있습니다.

- **슬라이드를 더하고, 옮기지 않습니다.** 부 제목은 사람의 문장을 하나도 담지 않는 새 슬라이드이고, 원래 슬라이드는 자리·내용·출처 생각이 전부 그대로입니다
- **12장 미만이면 모델을 호출하지도 않습니다**

- 부는 2~8개, 각 부에 최소 2장. 첫 부는 반드시 0번에서 시작해야 합니다
- **답이 조금이라도 어긋나면 통째로 버립니다.** 제목과 달리 부는 서로 독립적이지 않아서, 하나를 버리면 앞 부의 범위가 조용히 늘어납니다 — 그러면 아무도 검토하지 않은 구성이 텍에 들어갑니다

슬라이드 위치는 앞에 끼워진 부 제목 수만큼 밀리고, 출처 지도는 그 뒤에 같은 목록으로 만들어지므로 **"이 슬라이드가 어느 생각을 인용했는가"**는 그대로 답할 수 있습니다.

협업 로그가 사본을 두지 않습니다 (v0.68.0)

`AppendSpaceEvent` 는 payload 하나를 만들어 세 곳에 씁니다. 셋의 요구가 다릅니다.

받는 쪽	본문이 필요한가	정리되는가
웹훅 전달	예 — 구독자가 그걸 구독했습니다	30일
먹등 응답	예 — 같은 HTTP 응답을 돌려줘야 합니다	24시간
<code>space_events</code>	아니오	안 됨

셋 중 본문이 필요 없는 유일한 곳이, 유일하게 정리도 안 되는 곳이었습니다. 결과적으로 모든 메모의 모든 판본이 무기한 쌓였고 작성자가 지운 메모의 본문까지 남았습니다(soft delete라 `space_events` 행은 그대로입니다). 시험 DB 실측으로 662행 중 629행이 본문을 담고 있었습니다.

이 표를 읽는 소비자는 SSE로 연결된 캔버스 하나뿐이고, 클라이언트 코드는 `actorId` 만 보고 다시 불러옵니다 — payload는 한 번도 읽힌 적이 없습니다.

- 지금은 `space_events.payload` 가 항상 `{}` 입니다. 무엇이 바뀌었는지는 `event_type` 과 `resource_id` 로 충분합니다
- 마이그레이션 026이 기존 행의 payload도 비웁니다. 되돌리는 down 마이그레이션은 없습니다 — 지운 내용은 복구할 수 없고, 컬럼을 되돌린다고 돌아오지도 않습니다
- 웹훅 구독자가 받는 본문은 그대로입니다. 시험이 그쪽을 함께 확인합니다 — 아니면 이 수정은 기능 제거입니다

연결에 적힌 이유 (v0.65.0)

- 자체 컬럼 `note_edges.reason` , 200자 CHECK. 010에서 `relation` 이 자유 텍스트였던 것과는 다릅니다 — 그때 문제는 한 컬럼이 의미와 출처 둘을 담았던 것이지 이유를 적으면 안 된다는 게 아니었습니다
- 길이는 글자 수로 셉니다. 바이트로 세면 한국어 허용량이 3분의 1로 조용히 줄어듭니다. SQL `char_length` 와 Go `utf8.RuneCountInString` 이 같은 것을 셉니다
- **자르지 않고 거절합니다.** `400 + maxLength` . 남의 문장을 반만 저장하는 것이 안 하는 것보다 나쁩니다
- 편집 권한이 필요합니다 — 공간에 주석을 다는 것도 공간을 바꾸는 일입니다
- 마크다운 내보내기의 `## Connections` 줄 뒤에 — 이유 로 붙고, 가져오기가 같은 형태로 읽습니다. (`origin`) 은 읽고 지나갑니다 — 출처는 파일이 주장할 수 있는 것이 아니고, 이건 API가 request body에 적용하는 규칙과 같습니다

함께 막은 것: 연결 쓰기 권한 구멍

`notes:read` 만 가진 API 키가 연결을 지울 수 있었고, umm이 추측한 연결을 채택해서 그래프에 넣을 수도 있었습니다. `POST /spaces/{id}/edges` 에는 `notes:write` 검사가 있었지만 `DELETE /edges/{id}` 와 `POST /edges/{id}/accept` 에는 없었습니다. 세 번째 문(`PUT /edges/{id}/reason`)을 내면서 발견했고, 셋 다 막았습니다. 시험이 네 문을 전부 건습시다 — 한 문만 빠뜨리는 것이 이 결함이 계속 돌아오는 모양이기 때문입니다.

분량 맞추기 (v0.64.0)

AI가 아닙니다. 이 옵션만은 모델을 호출하지 않고, 그건 못 해서가 아니라 그게 맞아서입니다.

- 무엇을 남길지는 그래프가 정합니다: 몇 개의 생각이 그 슬라이드에 닿았는지(`From`), 부 제목인지, 표시해 둔 맞섬 인지. 전부 사람이 한 일이고, 글자는 하나도 읽지 않습니다
- 같은 공간 + 같은 분량 = 같은 덱. 미리보기가 읽을 만한 이유가 이것입니다. 모델이 정하면 만들 때마다 달라지고, 그러면 미리 본 것과 만들어진 것이 다릅니다
- AI 모델이 없어도 됩니다. 대부분의 설치가 그렇습니다
- 설명할 수 있습니다. "연결이 하나도 없어서 빠졌습니다"는 사람이 반박하고 고칠 수 있는 문장입니다. "모델이 이걸 골랐습니다"는 아닙니다
- 맞섬 슬라이드는 마지막까지 남습니다. 같은 메모를 요약해서는 절대 나오지 않는 슬라이드이기 때문입니다
- 순서는 건드리지 않습니다. 무엇이 들어갈지를 정하는 것이지 어떤 순서일지를 정하는 게 아닙니다
- 부 제목도 썩니다. 20장을 고르고 24장을 받으면 고른 분량이 아니므로, 부로 나눈 뒤 한 번 더 맞춥니다. 그래서 부로 나누면 내용 슬라이드가 그만큼 줄어듭니다 — 분량을 정한다는 게 그런 뜻입니다
- 10장 미만처럼 12장에 못 미치는 분량을 고르면 부 나누기는 아예 동작하지 않습니다. 짧은 발표는 나눌 필요가 없고, 분량이 먼저 적용되기 때문입니다
- 들어가지 못한 생각 수는 `trimmedCount` 로 저장됩니다. `excludedCount` 와 따로 두는 이유는 뜻이 달라서입니다 — 제외는 작성자가 분석에서 빼 둔 것이고, 이건 분량을 늘리면 그대로 돌아옵니다

모델이 주제와 건줘 무게를 다는 일은 그래프가 못 하는 일이고, 이건 그걸 하지 않습니다. 다만 손실은 생각보다 작습니다: 사람은 이미 어떤 생각으로 만들지 직접 고를 수 있고, 그게 모델의 추측보다 나은 답입니다.

큰 공간이 실패하는 경로

덱 만들기는 두 번의 호출입니다. Ptium이 덱을 열고, 그다음 umm이 소스를 컴파일해 넣습니다. 두 번째가 실패하면 덱은 이미 만들어져 있고 비어 있습니다. 예전에는 버튼을 다시 누르면 덱이 하나 더 생겼습니다.

v0.61.0부터는 실패 응답이 그 덱을 지목하고(`ptiumId` , `deckLeftBehind`), 화면이 "만들어진 발표 자료에 이어서 넣기" 를 제안합니다. Ptium의 소스 엔드포인트는 덮어쓰기라 두 번 넣어도 한 번 넣은 것과 같습니다.

큰 공간에서 가장 흔한 실패는 `timed-out` 입니다. 기본 `timeout_seconds` 는 30입니다. 슬라이드가 80장을 넘으면 화면이 보내기 전에 경고합니다 — 그 숫자는 umm이 자기 쪽에서 이미 알고 있기 때문입니다. 측정된 Ptium 한계가 아니라 "발표 하나로 읽을 만한가"에서 잡은 값입니다.

기술 정보는 관리자에게만 갑니다

바탕이 된 오류 메시지는 내부 호스트 주소, Go 타입 이름, SQL 제약 이름을 담고 있습니다. 슬라이드를 만들려던 사람에게는 쓸 데가 없고 보여 줄 이유도 없으므로, `admin` 역할일 때만 응답의 `technical` 에 실립니다. 화면에서는 "기술 정보 (관리자에게만 보입니다)" 아래에 접혀 나옵니다.

앞단 프록시가 Ptium 대신 HTML 오류 페이지로 답하는 경우, 그 페이지는 **Ptium의 설명이 아니므로 "Ptium이 보낸 설명"에 실지 않습니다**. 프록시는 Ptium이 아니고, 그 페이지는 상태 코드 이상을 말해 주지 않습니다.

지난 실패를 나중에 보기

발표 목록의 실패 배지에 마우스를 올리면 같은 분류가 뜹니다. 바탕이 된 오류는 `presentation_links.error` 에, 분류는 `failure_kind` 에 납니다. v0.60.0 이전에 기록된 행은 `failure_kind` 가 비어 있고, 그때는 저장된 원본 오류를 그대로 보여 줍니다.

부록 9. 서명 웹훅 운영

사용자는 개인 설정에서 허용된 `webhooks:write` scope로 subscription을 관리합니다. 대상은 공개 HTTPS 443만 허용합니다. 도메인 변경과 PostgreSQL delivery outbox는 원자적으로 커밋되고, 재시작 시 대기 또는 lease가 만료된 항목을 이어서 처리합니다. 전달 worker는 구독·소유 사용자·이벤트 공간·현재 membership과 정확한 delivery claim을 실제 HTTP 응답까지 잠급니다. 권한 회수·사용자 비활성화·구독 중지가 먼저 확정되면 payload를 보내지 않고, 전송이 먼저 시작되면 변경 transaction은 delivery terminal 상태와 payload 삭제가 확정될 때까지 기다립니다. 수신 시스템은 timestamp와 raw body의 HMAC-SHA256, 허용 시간 창을 검증하고 at-least-once 요청의 delivery UUID를 먹등 처리해야 합니다. 일시 실패는 세 번 재시도하고 연속 10회 실패 subscription은 자동 중지됩니다. 외부 오류는 잘못된 UTF-8을 제거하고 500 byte의 완전한 rune 경계 안에서 기록하므로 다국어 오류도 실패 횟수와 자동 중지를 막지 않습니다. terminal payload는 즉시 제거되고 metadata도 30일 후 정리되므로 운영 지표와 개인 설정의 마지막 오류를 함께 확인하세요.

어느 웹훅이 실패하고 있는지 (/admin/webhooks)

운영 현황에는 "최근 웹훅 실패" 숫자 하나만 있었습니다. 어느 웹훅인지, 누구 것인지, 수신 쪽이 뭐라고 답했는지는 모두 기록돼 있는데 아무것도 보이지 않았으니, 그 숫자는 뭔가 잘못됐다는 사실만 알려 주고 어디를 봐야 할지는 알려 주지 않았습니다.

관리자 → 웹훅 상태는 설치된 모든 웹훅을 나쁜 것부터 보여 줍니다. 연속 실패 횟수, 최근 24시간 실패, 마지막 전송 시각, 수신 쪽이 마지막으로 돌려준 오류, 소유자와 그 계정의 활성 여부, 아직 큐에 남아 있는 전달 수가 한 줄에 있습니다. 실패한 것만 을 켜면 실패가 기록된 것만 남습니다.

보내는 곳은 scheme과 host까지만 표시합니다. Slack이나 Discord의 incoming hook처럼 주소의 경로 자체가 자격 증명인 경우가 흔하기 때문에, 전달이 어디로 가는지는 보여 주되 그 주소로 직접 보낼 수 있게 하지는 않습니다.

멈추기는 `active` 만 내립니다. 주소, 서명 키, 이벤트 목록은 그대로 남아 소유자가 개인 설정에서 다시 켤 수 있고, 이 조치는 `webhook.pause` 로 감사 로그에 납니다. 이미 멈춘 웹훅을 다시 멈춰도 오류가 아닙니다.