Vibe Coders

관리자 가이드

설치·설정·계정·운영·장애 대응

버전	v0.85.1
작성일	2026-09-12
문서	관리자 가이드

관리자 가이드

> 이 문서는 게이트웨이를 **띄우고 지키는 사람**을 위한 것입니다. 도구를 연결해 쓰는 쪽은 사용자 가이드를 보세요. PDF: ADMIN_GUIDE.pdf. 기동·종료·백업·런북의 긴 절차는 운영 가이드에 있고, 이 문서는 그리로 가리킵니다.

1. 구성 요소

구성 요소	무엇	주고받는 것
`gateway` 컨테이너 (`ai-coding-proxy-gateway:<버전>`)	Go 단일 바이너리. `/v1/*` 프록시, `/admin/*` API, `/app` 새 콘솔, `/admin` 기존 콘솔, `/metrics` 를 한 프로세스에서 냅니다	8080/TCP 로 들어오는 도구·콘솔 요청, 업스트림 LLM 으로 나가는 HTTPS
데이터 볼륨 `proxy-gateway-data` → `/data`	SQLite `gateway.db` + 폴백 로그 `fallback.ndjson`	백업 대상 전부
업스트림 LLM 공급자	OpenAI 호환 API(OpenAI, Anthropic, 사내 vLLM 등). 하나 이상	`UPSTREAM_BASE_URL` 로 나가는 호출; 키는 게이트웨이만 보관
(선택) PostgreSQL	SQLite 대신 쓰는 저장소 (`DB_DRIVER=postgres`)	POSTGRES_GUIDE.md
(선택) Keycloak	SSO 로그인 (`SSO_KEYCLOAK_*`)	OIDC 리다이렉트
(선택) ClickHouse	장기 분석 싱크 (`CLICKHOUSE_*`)	배치 적재
(선택) Slack/Mattermost 웹훅	알림 규칙·예산 임박 통지	콘솔 **보안 → 알림** 에서 등록

2. 설치

릴리즈 자산(GitHub Release `v0.85.1`)으로 처음부터 끝까지. 빌드 호스트에서 이미지를 만드는 절차와 오프라인망 적재 배경은 OPERATIONS.md 2.5에 있습니다.

항목	값
포트	`8080/TCP` (컨테이너 안 `LISTEN_ADDR=:8080`)
볼륨	`proxy-gateway-data:/data` — nonroot(65532) 소유
자원	CPU 1코어·RAM 512 MB 로 시작 가능. 요청 로그는 `/data` 디스크를 씁니다(보존 기간은 `RETENTION_*`)
실행 계정	distroless `nonroot` . root 로 띄우지 않습니다

```
# 1) 자산 검증과 적재
sha256sum -c ai-coding-proxy-gateway-v0.85.1.tar.gz.sha256
gunzip -c ai-coding-proxy-gateway-v0.85.1.tar.gz ¦ docker load

# 2) 비밀값 파일 (mode 0600). ADMIN_TOKEN · GATEWAY_SECRET 을 무작위로 만들고 UPSTREAM_API_KEY 자리를 비워 둡니다.
sudo mkdir -p /opt/proxy-gateway
sudo bash init-deployment-env-v0.85.1.sh /opt/proxy-gateway/gateway.env
sudo sed -i 's¦^UPSTREAM_API_KEY=.*¦UPSTREAM_API_KEY=<업스트림 키>¦' /opt/proxy-gateway/gateway.env
#    최초 관리자 계정과 새 콘솔을 켭니다 (값은 가짜 예시입니다)
sudo tee -a /opt/proxy-gateway/gateway.env >/dev/null <<'EOF'
AUTH_ENABLED=true
AUTH_JWT_SECRET=<openssl rand -hex 32 결과>
AUTH_ADMIN_BOOTSTRAP_EMAIL=admin@example.com
AUTH_ADMIN_BOOTSTRAP_PASSWORD=<처음 로그인 후 바꿀 임시 비밀번호>
UI_APP_ENABLED=true
EOF

# 3) 기동
export GATEWAY_VERSION=v0.85.1
docker compose --env-file /opt/proxy-gateway/gateway.env up -d
curl -fsS http://127.0.0.1:8080/ready

# 4) 최초 관리자 로그인 → http://<호스트>:8080/app
```

`AUTH_ADMIN_BOOTSTRAP_*` 는 계정이 없을 때 한 번만 `super_admin` 을 만듭니다. 로그인한 뒤 **사용자와 팀**에서 비밀번호를 바꾸고 env 파일에서 두 줄을 지우세요. 이전 배포의 볼륨을 이어받는다면 `up` 전에 `repair-data-dir` 로 소유권을 복구합니다(OPERATIONS.md 8.6).

3. 설정

환경 변수는 기동 시 한 번 읽습니다(`internal/config/config.go`). 콘솔 **시스템 설정**에서 바꾼 런타임 값이 같은 이름의 env 보다 우선하며 `SETTINGS_RELOAD_INTERVAL` 마다 다시 읽힙니다. 비밀값 예시는 전부 가짜입니다.

3.1 반드시 정하는 것

이름	기본값	필수	설명
`UPSTREAM_API_KEY`	(없음; `OPENAI_API_KEY` 도 읽음)	예	업스트림 공급자 키. 게이트웨이만 보관
`ADMIN_TOKEN`	(없음)	예 (compose)	기존 `/admin` 콘솔·API 관리자 토큰. 64 hex 권장
`GATEWAY_SECRET`	내장 개발용 값	예 (compose)	키 해시·서명용 비밀. **기본값 그대로 두면 안 됩니다**
`UPSTREAM_PROVIDER`	`openai`	아니오	기본 공급자 이름
`UPSTREAM_BASE_URL`	`https://api.openai.com`	아니오	기본 공급자 주소
`LISTEN_ADDR`	`:8080`	아니오	수신 주소
`DB_DRIVER` / `DB_DSN`	`sqlite` / `data/gateway.db`	아니오	저장소. `postgres` 면 `DB_DSN` (또는 `DATABASE_URL` / `POSTGRES_DSN`) 에 DSN
`LOG_FALLBACK_PATH`	`data/fallback.ndjson`	아니오	DB 기록 실패 시 폴백 로그
`UI_APP_ENABLED`	`false`	아니오	새 콘솔 `/app` 노출
`PROXY_API_KEYS`	(없음)	아니오	부팅 시 등록할 proxy key: `이름:키:소유자:팀,…` . 콘솔 발급이 일반적
`MODEL_PRICING_KRW_PER_1M`	`{}`	아니오	모델별 단가 JSON `{"모델":{"input_krw_per_1m":540,"output_krw_per_1m":2160}}` . 음수는 부팅 거부

3.2 인증·SSO

이름	기본값	필수	설명
`AUTH_ENABLED`	`false`	아니오	이메일·비밀번호 세션 로그인
`AUTH_JWT_SECRET`	(없음)	`AUTH_ENABLED` 일 때	액세스·리프레시 토큰 서명 키
`AUTH_ACCESS_TOKEN_TTL` / `AUTH_REFRESH_TOKEN_TTL`	`15m` / `168h`	아니오	토큰 수명
`AUTH_ADMIN_BOOTSTRAP_EMAIL` / `AUTH_ADMIN_BOOTSTRAP_PASSWORD`	(없음)	최초 1회	계정이 없을 때 만들 `super_admin`
`AUTH_API_KEY_PREFIX` / `AUTH_SERVICE_KEY_PREFIX`	`vc_sk_` / `vc_sa_`	아니오	발급 키 접두사
`ADMIN_READONLY_TOKEN`	(없음)	아니오	읽기 전용 관리자 토큰
`ATTRIBUTE_EXTERNAL_KEYS`	`true`	아니오	미등록 키를 지문으로 `ext_…` 사용자로 분리
`SELF_SERVICE_KEYS_ENABLED`	`false`	아니오	개발자가 자기 키를 직접 발급
`SSO_KEYCLOAK_ENABLED`	`false`	아니오	Keycloak OIDC 로그인
`SSO_KEYCLOAK_ISSUER_URL` / `_CLIENT_ID` / `_CLIENT_SECRET` / `_REDIRECT_URI` / `_SCOPES`	(없음)	SSO 켤 때	OIDC 클라이언트
`SSO_KEYCLOAK_ALLOW_LOCAL_LOGIN`	`true`	아니오	SSO 와 로컬 로그인 병행
`SSO_KEYCLOAK_DEFAULT_ROLE` / `_ROLE_CLAIM` / `_GROUP_CLAIM`	`developer` / `realm_access.roles` / `groups`	아니오	클레임 → 역할·팀 매핑

3.3 업스트림·라우팅

이름	기본값	설명
`UPSTREAM_MODEL_PATTERNS`	(없음)	이 공급자가 받는 모델 glob
`UPSTREAM_DEFAULT_MODEL`	(없음)	모델 미지정 요청의 기본 모델
`UPSTREAM_LOAD_BALANCE`	`first`	다중 공급자 분배 방식
`UPSTREAM_TIMEOUT` / `UPSTREAM_RESPONSE_HEADER_TIMEOUT`	`10m` / `60s`	업스트림 전체·첫 헤더 타임아웃
`UPSTREAM_FAILOVER_BUDGET`	`0`	폴백에 쓸 추가 시간(0=무제한)
`UPSTREAM_BREAKER_ENABLED` / `_THRESHOLD` / `_COOLDOWN`	`true` / `5` / `30s`	회로 차단기
`UPSTREAM_BREAKER_SHARED` / `_SYNC_INTERVAL`	`false` / `3s`	다중 인스턴스 차단기 공유
`UPSTREAM_STICKY_SESSIONS` / `UPSTREAM_STICKY_TTL`	`true` / `30m`	세션을 같은 공급자에 고정
`UPSTREAM_HEALTH_DEMOTE_THRESHOLD`	`50`	이 점수 아래면 공급자 강등
`LIMITS_MAX_REQUEST_BYTES` / `_MAX_MESSAGES` / `_MAX_OUTPUT_TOKENS`	`64 MiB` / `0` / `0`	요청 크기 상한(0=무제한)
`PRICING_FALLBACK_MODEL`	`qwen-plus`	단가를 모르는 모델의 비용 추정 기준

3.4 로깅·보존·캐시

이름	기본값	설명
`LOG_RAW_PROMPTS` / `LOG_RAW_BODIES` / `LOG_RESPONSE_TEXT`	`false`	원문 저장. 켜면 마스킹 전 본문이 DB 에 남습니다
`LOG_RESPONSE_MAX_BYTES` / `LOG_QUEUE_SIZE`	`1 MiB` / `4096`	응답 저장 상한·비동기 로그 큐
`RETENTION_REQUEST_DAYS` / `_PROMPT_DAYS` / `_RESPONSE_DAYS`	`90` / `30` / `30`	보존 일수
`RETENTION_TEXT2SQL_REPLAY_DAYS` / `_DOMAIN_EXAMPLE_DAYS` / `RETENTION_INTERVAL`	`30` / `365` / `1h`	보존 일수·정리 주기
`CACHE_EMBEDDING_ENABLED` / `_TTL` / `_MAX_BYTES` / `_SCOPE`	`true` / `24h` / `1 MiB` / `global`	임베딩 캐시
`CACHE_CHAT_ENABLED` / `_TTL` / `_SCOPE`	`false` / `1h` / `global`	채팅 응답 캐시(비결정적이라 기본 꺼짐)
`CACHE_CHAT_SEMANTIC_ENABLED` / `_MODEL` / `_MAX_CANDIDATES` / `_MULTITURN`	`false` / (없음) / `200` / `false`	의미 기반 캐시
`CACHE_EMBEDDING_PROVIDER` / `_BASE_URL` / `_API_KEY`	(없음)	캐시용 임베딩 공급자
`SESSION_INFERENCE_ENABLED` / `SESSION_INJECT_HEADER` / `SESSION_IDLE_TIMEOUT`	`true` / `true` / `30m`	세션 추론
`QUOTA_RESERVATIONS_ENABLED` / `QUOTA_RESERVATION_SWEEP_INTERVAL`	`true` / `5m`	진행 중 요청을 한도에 선반영
`SETTINGS_RELOAD_INTERVAL`	`10s`	런타임 설정 재적재 주기
`VCS_WEBHOOK_SECRET` / `VCS_INFER_FROM_CONTENT`	(없음) / `true`	Prompt→Commit→MR 상관(웹훅 비밀 필수)
`CARBON_MODEL_WH_PER_1K`	(없음)	모델별 전력량 맵

3.5 Text2SQL · MCP · 분석 싱크

이름	기본값	설명
`TEXT2SQL_ENABLED`	`false`	Text2SQL 기능
`TEXT2SQL_SCHEMA` / `TEXT2SQL_DIALECT`	(없음) / `PostgreSQL`	스키마 설명·방언
`TEXT2SQL_EXEC_DRIVER` / `TEXT2SQL_EXEC_DSN`	`postgres` / (없음)	실행 대상 DB
`TEXT2SQL_TWIN_DRIVER` / `TEXT2SQL_TWIN_DSN`	`postgres` / (없음)	검증용 트윈 DB
`TEXT2SQL_DEFAULT_LIMIT` / `TEXT2SQL_MAX_LIMIT`	`100` / `1000`	결과 행 상한
`TEXT2SQL_PREVIEW_MODEL` / `_EXECUTE_MODEL` / `_SUMMARY_MODEL` / `_ACCURATE_MODEL` / `_LOCAL_MODEL`	`gpt-4.1-mini` ×3 / `claude-sonnet-4` / `qwen-coder`	단계별 모델
`TEXT2SQL_MASK_RESULTS` / `_CACHE_ENABLED` / `_CACHE_TTL`	`true` / `true` / `1h`	결과 마스킹·캐시
`TEXT2SQL_CLARIFY_ENABLED` / `_REQUIRE_DATE_FILTER` / `_REPLAY_BUNDLES`	`false`	되묻기·날짜 필터 강제·재생 번들
`TEXT2SQL_STATEMENT_TIMEOUT` / `TEXT2SQL_WORK_MEM`	`15s` / (없음)	실행 제한
`TEXT2SQL_DAILY_RISK_LIMIT` / `_DAILY_RISK_WARN`	`20` / `0`	일일 위험 쿼리 한도
`MCP_AGENTIC_MODEL` / `MCP_MAX_AGENT_STEPS` / `MCP_MAX_TOKENS` / `MCP_MAX_TOOLS` / `MCP_FORCE_TOOL_FIRST`	(없음) / `8` / `2048` / `32` / `true`	MCP Gateway 에이전트 루프
`SKILLS_ENFORCEMENT`	`warn`	Skill 정책 모드 `off` / `warn` / `enforce`
`REDTEAM_POST_CHANGE_ENABLED` / `_COOLDOWN` / `_MAX_TARGETS`	`true` / `10m` / `20`	설정 변경 후 자동 Red Team 회귀
`CLICKHOUSE_URL` / `_USER` / `_PASSWORD` / `_DB` / `_TABLE`	(없음) / … / `default` / `analytics_daily`	분석 싱크(비우면 꺼짐)
`CLICKHOUSE_BATCH_SIZE` / `_FLUSH_INTERVAL` / `_MAX_QUEUE_SIZE` / `_SINK_DAYS` / `_SINK_INTERVAL`	`200` / `5s` / `10000` / `3` / `0`	적재 배치
`CLICKHOUSE_*_FACT_TABLE` (request, routing, policy, tool, skill, text2sql, eval, feedback, multimodel, redteam)	(없음)	팩트 테이블 이름
`UI_APP_DEFAULT_ENTRY` / `UI_APP_LEGACY_FALLBACK` / `UI_APP_FEEDBACK_ENABLED` / `UI_APP_TELEMETRY_ENABLED`	`/app/overview` / `true` / `false` / `false`	새 콘솔 진입·폴백·피드백·텔레메트리

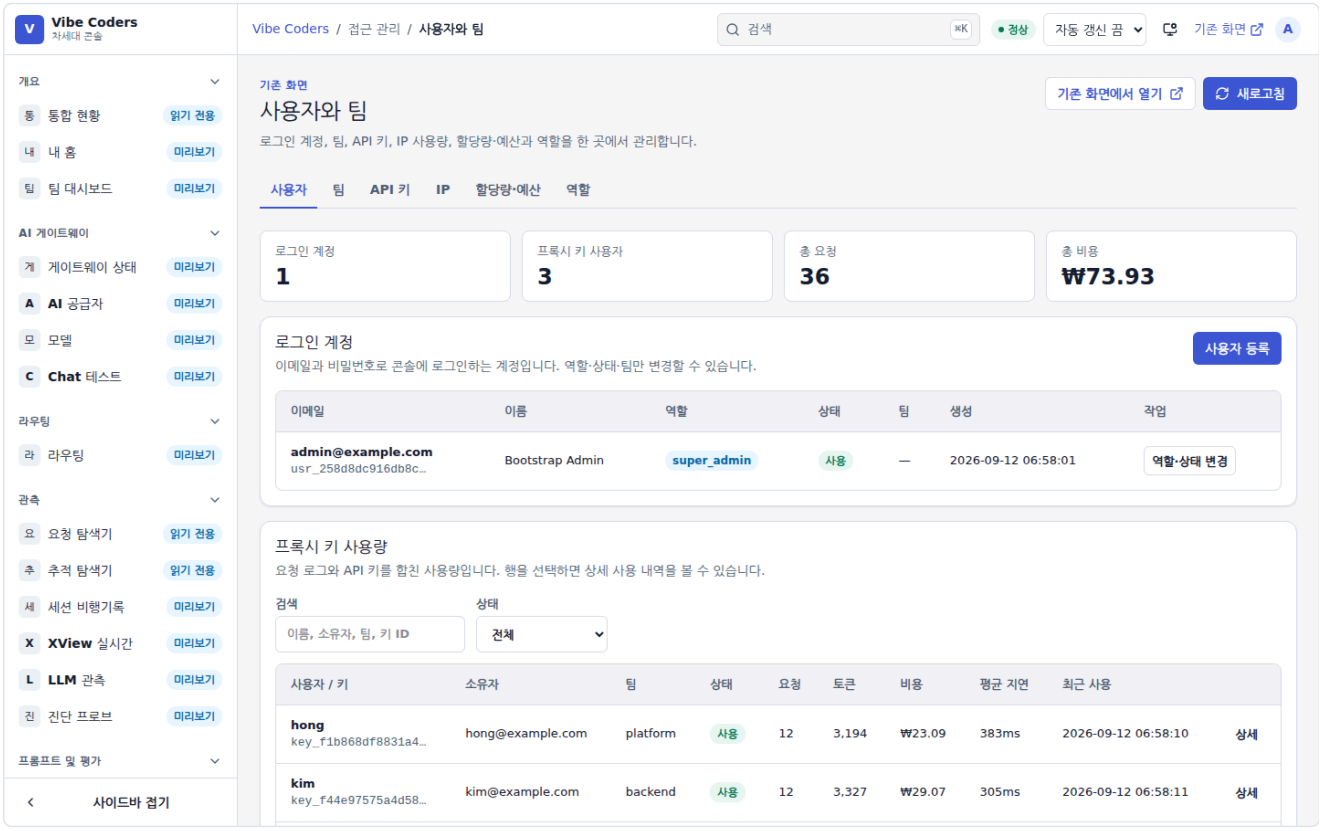
`TEST_*`, `CH_IT_*` 는 테스트 전용이라 운영에서 쓰지 않습니다.

4. 계정과 권한

두 가지 인증이 공존합니다. **관리자 토큰**(`ADMIN_TOKEN`)은 기존 `/admin` 콘솔과 `/admin/*` API 용이고, **세션 로그인**(`AUTH_ENABLED=true`)은 새 콘솔 `/app` 용입니다. 역할별 권한은 `internal/proxy/auth.go` 의 `roleScopes` 가 정본입니다.

역할	할 수 있는 일
`super_admin` / `admin`	전부. 역할 변경·비밀값·설정 쓰기 포함
`team_admin`	자기 팀 범위의 조회 + 호출. 관리자 화면 읽기
`team_manager`	자기 팀 대시보드(`/app/team`)만. 운영 대시보드 없음
`developer`	호출·모델 조회·자기 사용량(`/app/me`)
`viewer`	관리자 화면 읽기 전용
`service_account`	호출만(CI·봇)
`ops_admin` / `ai_admin` / `security_admin` / `billing_admin`	관리자 읽기 + 각자 영역의 런타임 설정 쓰기(운영·모델/라우팅·보안·비용)
`readonly_admin`	관리자 화면 읽기 전용(보안 포함)

콘솔 **접근 관리 → 사용자와 팀**에서 로그인 계정, 팀, API 키, IP, 할당량·예산, 역할을 한 곳에서 다룹니다. **사용자 등록**으로 계정을 만들고, 행의 **역할·상태 변경**으로 역할을 바꿉니다. proxy key 는 **API 키** 탭에서 발급하며 키 값은 발급 순간 한 번만 표시됩니다.



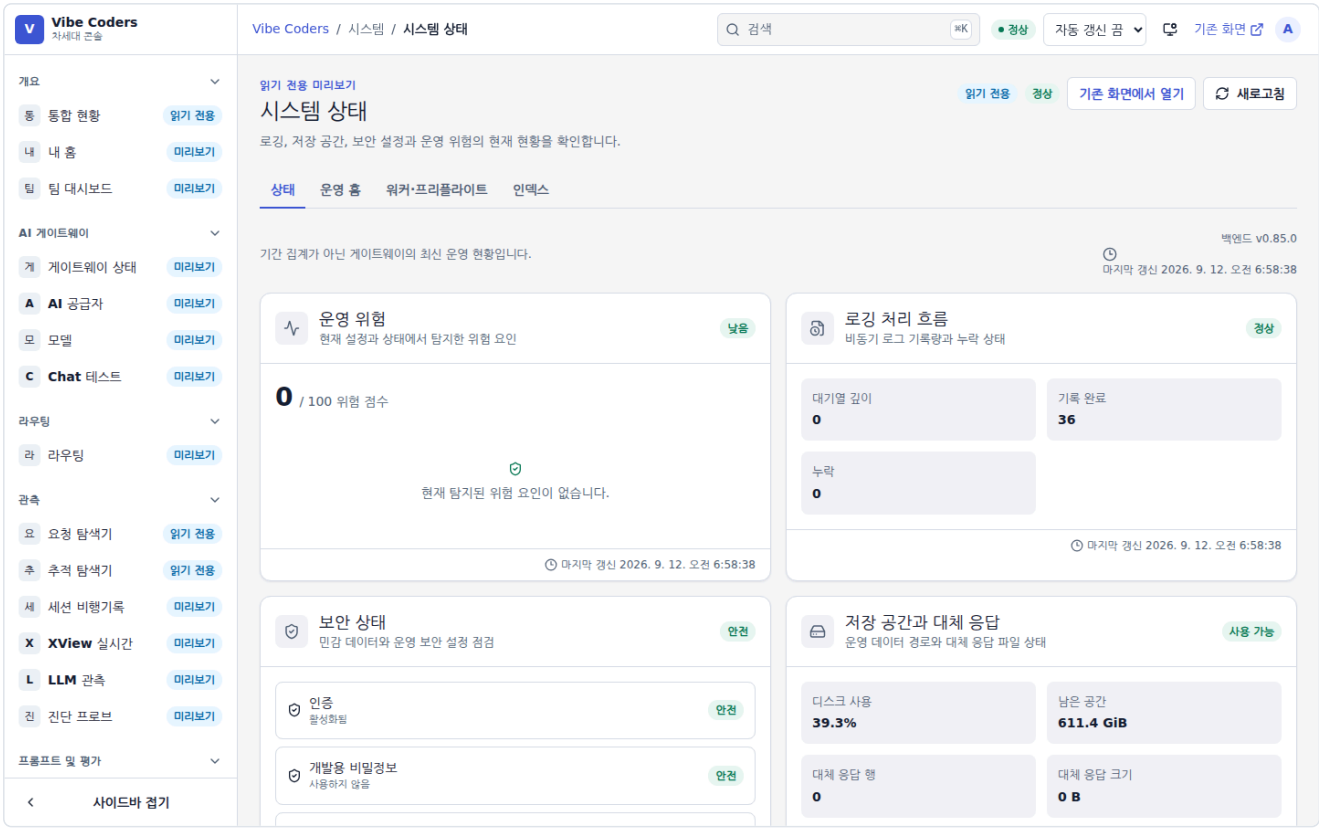
사용자와 팀 — 로그인 계정과 proxy key 사용량을 한 화면에서 관리한다

SSO 로 들어온 계정의 역할은 클레임 매핑(`SSO_KEYCLOAK_ROLE_CLAIM`)이 정하되, 콘솔에서 직접 올린 역할·팀은 다음 SSO 로그인이 내리지 않습니다(v0.83.2).

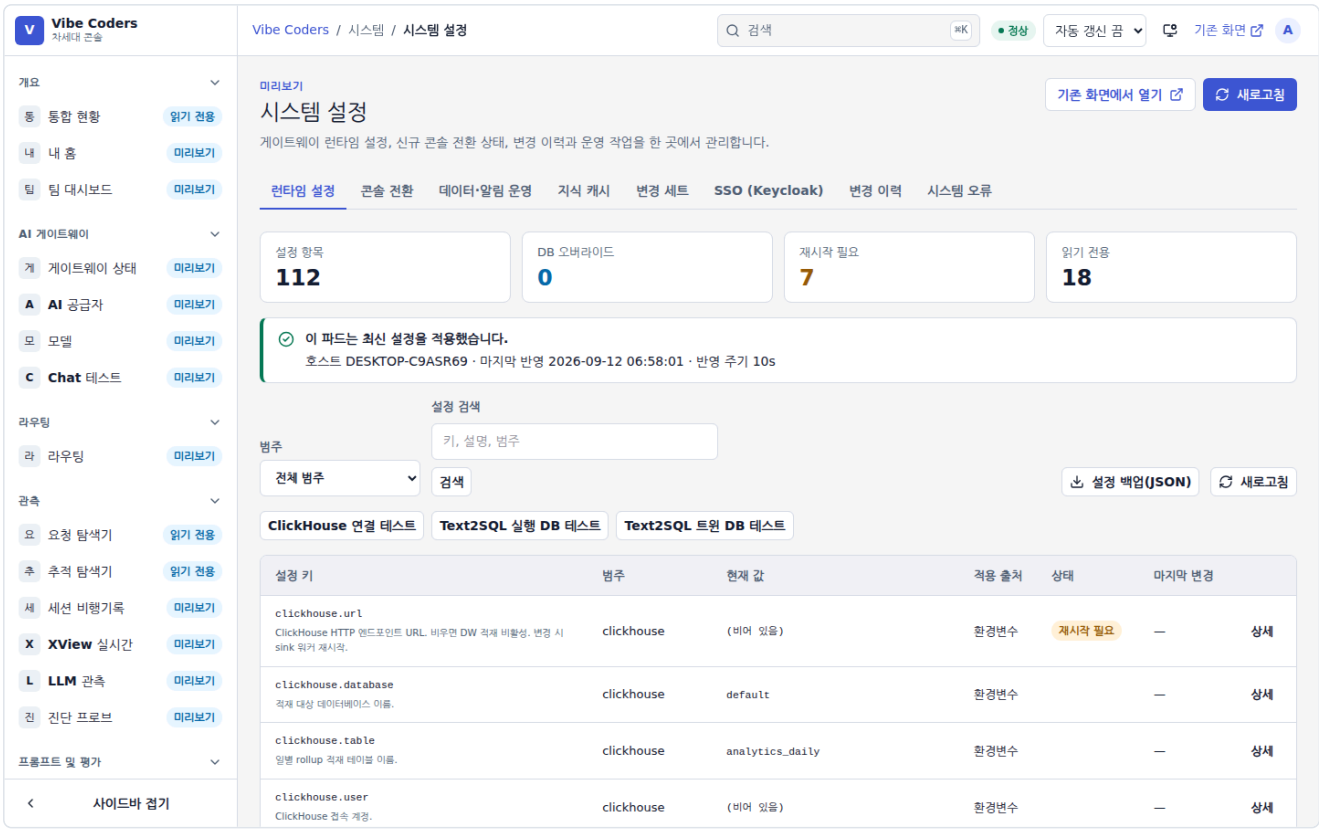
5. 운영

- **상태 점검**: `GET /health` (프로세스), `GET /ready` (DB 포함), `GET /metrics` (Prometheus). 콘솔 **시스템 → 시스템 상태**가 같은 신호를 사람 눈으로 보여 줍니다.

- **콘솔 홈: 개요 → 통합 현황**에서 게이트웨이 상태·보존 비용·P95 지연·라우팅·운영 위험을 봅니다. 상단 **자동 갱신**을 켜면 주기적으로 다시 읽습니다.
- **로그 위치**: 컨테이너 stdout(`docker compose logs -f gateway`), 폴백 로그 `/data/fallback.ndjson` (DB 기록 실패분; 콘솔 **시스템 설정 → Fallback 로그 재처리**로 되살립니다).
- **백업·복구**: `/data` 볼륨이 전부입니다. `backup-volume-v0.85.1.sh` 로 tar 백업, 복구는 볼륨을 유지한 채 컨테이너만 교체합니다 — OPERATIONS.md 6.
- **업그레이드**: 새 tar.gz 를 `docker load` → `GATEWAY_VERSION` 만 올려 `docker compose up -d` . 마이그레이션은 기동 시 자동입니다. **되돌리기**: 업그레이드 전 백업을 복구하고 `GATEWAY_VERSION` 을 이전 값으로 되돌려 `up -d` . 새 버전이 추가한 컬럼은 이전 바이너리가 무시합니다.
- **보존**: `RETENTION_*` 일수를 넘긴 요청·프롬프트·응답은 `RETENTION_INTERVAL` 마다 지워집니다. 콘솔 **시스템 설정 → 데이터 보존**에서 무엇이 함께 삭제되는지 볼 수 있습니다(9.3 절).



시스템 상태 — 프로세스·DB·업스트림 상태와 런타임 지표

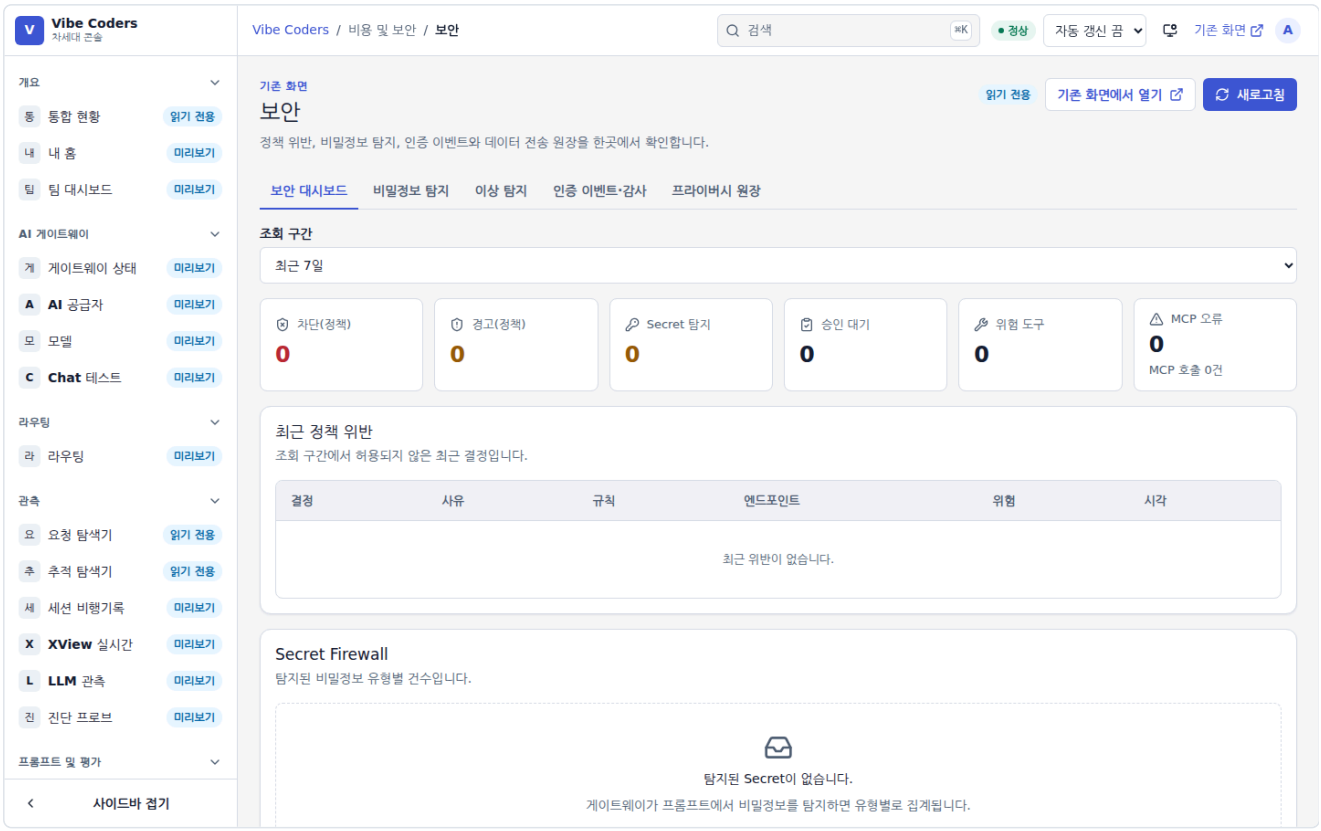


시스템 설정 — 런타임 설정을 카테고리별로 읽고 바꾼다

6. 장애 대응

증상	확인할 곳	조치
모든 호출이 `503 kill_switch_active`	콘솔 **보안 → Kill Switch**	의도한 정지가 아니면 끕니다. 켠 사람은 관리자 변경 이력에 남습니다
특정 사용자만 `429 quota_error`	**사용자와 팀 → 할당량·예산**	한도 조정 또는 다음 기간 대기. 헤더 `X-Quota-Scope` 가 어느 범위인지 알려 줍니다
`502 provider_unavailable` 가 잇따름	**AI 게이트웨이 → 게이트웨이 상태**의 공급자 점수·차단기	업스트림 장애. 폴백 공급자가 있으면 자동 전환. 없으면 ROUTING_GUIDE.md 대로 폴백 등록
컨테이너가 8080 에서 뜨지 않음, 로그에 `readonly database` 또는 `attempt to write a readonly database`	`docker compose logs gateway`	볼륨 소유권 문제. `repair-data-dir` 실행 — OPERATIONS.md 8.6
부팅 직후 종료, 로그에 `invalid configuration error="parse MODEL_PRICING_KRW_PER_1M: …"`	env 파일	단가 JSON 형식 오류. `{"모델":{"input_krw_per_1m":…,"output_krw_per_1m":…}}` 형태인지, 음수가 없는지 확인
`/ready` 는 200 인데 콘솔이 "데이터를 불러오지 못했습니다" + 요청 ID	서버 로그에서 그 요청 ID	대개 권한(403) 또는 만료된 세션. 역할과 `AUTH_JWT_SECRET` 변경 여부 확인
사용량이 전부 `anonymous` / `passthrough`	**사용자와 팀 → API 키**	등록되지 않은 키로 호출 중. 사용자별 proxy key 발급
SSO 로그인 뒤 메뉴가 거의 사라짐	감사 이력의 `role_changed`	OPERATIONS.md 8.7
디스크 가득 참 / DB 잠금	`df /data` , 폴백 로그 크기	보존 일수 단축, 폴백 로그 재처리 후 정리 — OPERATIONS.md 8.4

긴급 정지(모든 호출 즉시 차단)는 콘솔 **보안** 화면의 Kill Switch 또는 `POST /admin/kill-switch` 입니다. 절차와 되돌리기는 OPERATIONS.md 8.1.

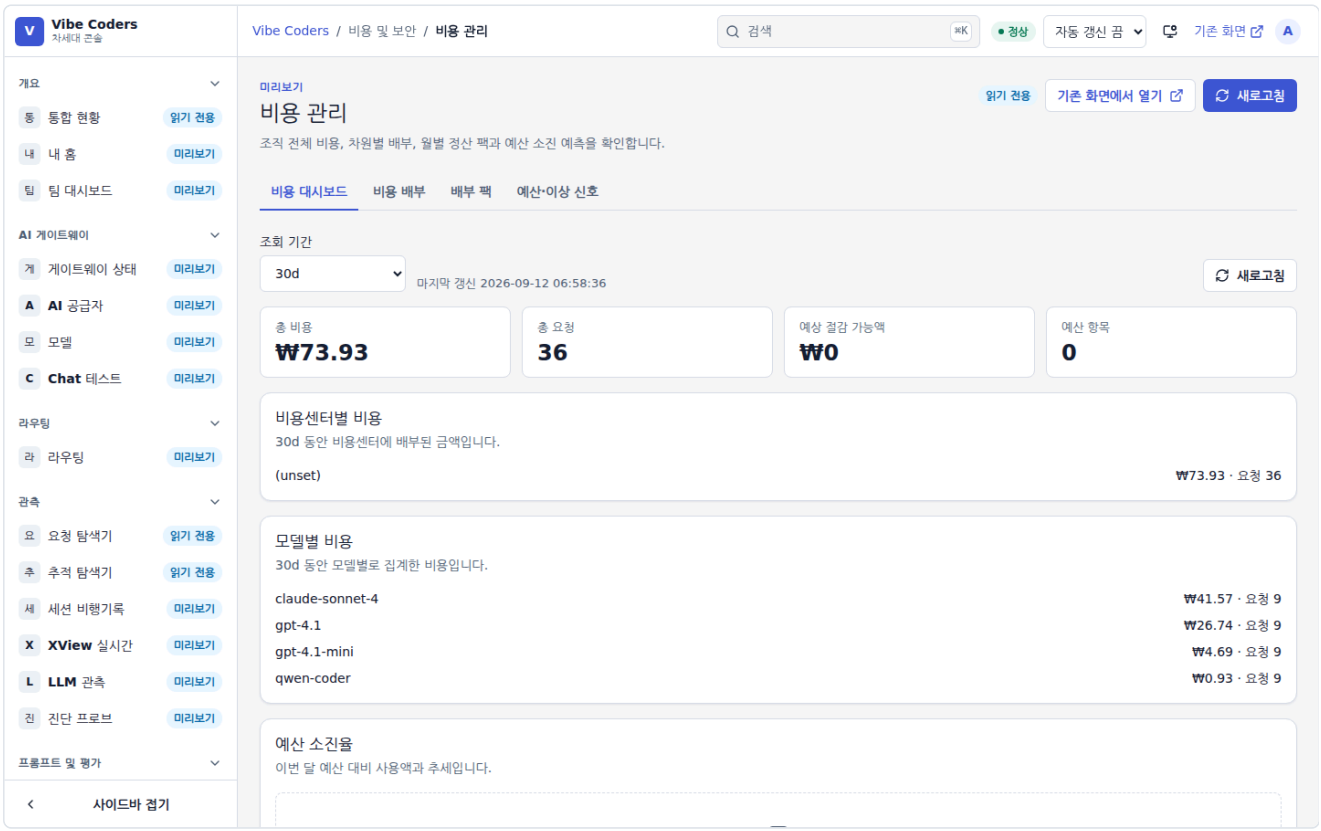


보안 — Kill Switch·비용 가드·알림 규칙을 한 화면에서 다룬다

7. 보안

- **바꿔야 하는 기본값**: `GATEWAY_SECRET` (내장 개발용 값), `ADMIN_TOKEN` (없으면 `/admin` 이 열립니다), `AUTH_JWT_SECRET` . 셋 다 `openssl rand -hex 32` 로 만들고 env 파일은 `0600` 으로 둡니다. `LOG_RAW_PROMPTS` · `LOG_RAW_BODIES` · `LOG_RESPONSE_TEXT` 는 기본 `false` 를 유지하세요.
- **외부에 열면 안 되는 것**: 8080 은 사내망·VPN 뒤에만. 콘솔· `/admin/*` · `/metrics` 를 인터넷에 노출하지 않습니다. TLS 는 앞단 리버스 프록시에서 종료합니다.
- **인증 연동**: 운영에서는 `AUTH_ENABLED=true` + SSO(`SSO_KEYCLOAK_*`)를 권장합니다. 관리자 토큰은 자동화·비상용으로만 쓰고, 읽기 작업에는 `ADMIN_READONLY_TOKEN` 을 씁니다.
- **키 유출 의심**: 콘솔 **사용자와 팀 → API 키**에서 해당 키를 비활성화하고 새 키를 발급합니다. 업스트림 키가 유출됐다면 공급자에서 회전한 뒤 `UPSTREAM_API_KEY` 를 바꿔 재기동 — OPERATIONS.md 8.5.
- **정책·마스킹**: 프롬프트의 비밀·개인정보는 기본으로 마스킹됩니다. 모델·공급자 허용 목록, DLP, 승인 워크플로는 콘솔 **거버넌스 → 정책 및 거버넌스**와 SAFETY_GUIDE.md.

정책 및 거버넌스 — 모델 허용 목록·DLP·승인 규칙



비용 관리 — 모델·팀별 비용과 예산 소진 예측

화면 레퍼런스 (기존 `/admin` 콘솔)

아래는 기존 콘솔의 탭별 상세 설명입니다. 새 콘솔 `/app` 의 화면은 같은 API 를 쓰며, 각 화면의 **기존 화면에서 열기** 로 아래 탭에 닿습니다.

`http://<host>:8080/admin` 에 접속하는 운영 관리자를 위한 사용 설명서입니다. 한국어 UI 와 다중 탭으로 구성되어 있고, 모든 동작은 동일한 이름의 REST API 로도 자동화할 수 있습니다.

> [!NOTE] AI Proxy Gateway의 안전성 통제(긴급 정지, 비용 가드, 정책 엔진, 민감 정보 방화벽 등)에 관한 더욱 상세한 레벨의 스펙은 안전 및 보안 거버넌스 운영 가이드를 참고해 주십시오.

---

1. 첫 접속

접속 방법은 인증 모드에 따라 둘 중 하나입니다. `/admin` 에 들어가면 UI가 모드를 자동 감지해 알맞은 화면을 띄웁니다.

A. 계정 로그인 모드 (`AUTH_ENABLED=true` , 권장)

1. 브라우저로 `http://<host>:8080/admin` 접속 → **이메일/비밀번호 로그인 화면이 바로 표시**됩니다.
2. 최초 1회는 부트스트랩 계정(`AUTH_ADMIN_BOOTSTRAP_EMAIL` / `AUTH_ADMIN_BOOTSTRAP_PASSWORD`)으로 로그인하세요.
3. 로그인하면 헤더에 `이메일 · 역할` 칩과 "로그아웃" 버튼이 나타나고 대시보드로 이동합니다.
4. access token 만료 시 UI가 refresh token으로 **자동 갱신(rotation)** 하므로 끊김이 없고, refresh까지 만료되면 로그인 화면으로 돌아갑니다. 토큰은 sessionStorage에만 보관됩니다(탭 닫으면 소멸).

B. 레거시 토큰 모드 (`AUTH_ENABLED=false` , 기본)

1. 브라우저로 `http://<host>:8080/admin` 접속
2. 상단 우측 "관리자 토큰" 입력란에 `ADMIN_TOKEN` 값 붙여넣기
3. 데이터가 보이면 정상. 401 이 나오면 토큰이 잘못된 것입니다.

헤더의 운영 보조 도구

아이콘/버튼	기능
자동 새로고침 드롭다운	끔/5/10/30/60초 마다 현재 화면 다시 로드. 세션에 보관.
◐ / ☀	라이트/다크 테마 전환 (`t` 단축키 동일).
`?`	단축키 도움말 오버레이.
사용자 칩 · 로그아웃	(로그인 모드) 현재 로그인한 이메일·역할 표시, 로그아웃 시 세션/refresh token 폐기 후 로그인 화면으로.
관리자 토큰	(레거시 모드에만 표시) 입력값은 sessionStorage 에만 저장되고, 다른 탭/세션에는 공유되지 않습니다.

작업 결과 알림 (토스트)

저장·삭제·발급 같은 작업의 결과는 화면 우측 하단에 **토스트**로 표시됩니다. 브라우저 기본 경고창(`alert`)을 쓰지 않으므로 작업 흐름이 끊기지 않습니다.

종류	표시	동작
성공	✓ (강조색 테두리)	4초 후 자동 사라짐
정보	ℹ	4초 후 자동 사라짐
오류·검증 실패	⚠ (빨간 배경)	**자동으로 사라지지 않음** — 닫기(×)를 눌러야 사라집니다

오류가 자동으로 사라지지 않는 것은 의도된 동작입니다. 잠시 화면을 보지 않는 사이 저장 실패 알림이 사라지면 실패 사실을 놓치게 되기 때문입니다. 토스트는 최대 4개까지만 쌓이며, 오래된 것부터 밀려납니다. 스크린리더 사용자를 위해 `aria-live` 영역으로 읽힙니다.

이전에는 성공 시 화면만 새로고침되어 **작업이 실제로 됐는지 알 수 없었고**, 실패 시에만 경고창이 떴습니다. 이제 프로바이더·API 키·라우팅 규칙·예산·사용 한도·알림 규칙·AI 정책·계정/팀·Knowledge·SLO·템플릿·모델 일몰 저장과 키 폐기/회전·Kill Switch 전환·회로 차단기 해제·세션 고정 해제가 모두 완료 알림을 표시합니다.

콘솔 로딩 속도

관리자 콘솔은 단일 HTML 문서(약 1.2 MB)입니다. 다음이 적용되어 있습니다.

항목	효과
gzip 압축	전송량 **1218 KB → 284 KB** (약 77% 감소)
ETag 재검증	두 번째 로드부터는 **304 · 본문 0바이트**
1회 렌더링	요청마다 1.2 MB 문자열을 새로 만들지 않고 기동 시 한 번만 준비

`Cache-Control: no-cache` 이므로 브라우저는 **매번 재검증**합니다. 게이트웨이를 업그레이드하면 ETag가 바뀌어 곧바로 새 콘솔을 받습니다 — 캐시에 남은 이전 버전이 보이는 일은 없습니다.

VPN이나 폐쇄망처럼 대역폭이 제한된 환경에서 체감 차이가 큽니다.

비어 있는 설정 화면

팀·사용 한도·예산·알림 규칙·작업 템플릿 화면은 **아직 아무것도 없을 때** 무엇을 만들면 되는지, 만들지 않으면 어떤 상태인지 안내합니다. 예를 들어 사용 한도가 비어 있으면 *"한도를 만들기 전까지 API 키·팀·IP별 토큰과 비용에 제한이 없습니다"*라고 알려줍니다.

> 대시보드·분석 패널의 `데이터 없음` 은 여러 화면이 공유하는 공용 컴포넌트가 출력하는 것이라, 의도적으로 일반적인 문구를 유지합니다.

입력 오류 표시

필수 항목을 비운 채 저장하면 **비어 있는 입력칸이 빨갛게 표시되고 첫 번째 칸으로 포커스가 이동**하며, 어떤 항목이 빠졌는지 토스트로 한 번 안내합니다. 값을 입력하기 시작하면 표시가 사라집니다.

이전에는 `이름과 URL을 입력하세요` 같은 토스트만 떠서, 항목이 여러 개인 폼에서는 **어느 칸을 말하는지 직접 찾아야** 했습니다. 스크린리더에는 `aria-invalid` 로 전달됩니다.

시간 표시

목록·표의 시각은 **상대 시간**으로 표시됩니다 — `방금`, `5분 전`, `3시간 전`, `2일 전`. 한 달이 넘으면 날짜로 바뀝니다. 마우스를 올리면 **정확한 로컬 시각**이 툴팁으로 나옵니다(외부 로그와 대조할 때 필요).

기간·경계값(조회 구간, 만료 예정 등)은 상대 시간이 오히려 읽기 나쁘므로 `08-21 18:51` 형태의 절대 시각으로 표시하며, 올해가 아니면 연도가 붙습니다. 모든 시각은 **브라우저 로컬 시간대** 기준입니다.

이전에는 서버가 주는 UTC 원문(`2026-08-21T09:51:46Z`)이 29곳에서 그대로 노출돼, 표를 훑는 사람이 머릿속에서 시차를 계산해야 "이게 최근인가"를 판단할 수 있었습니다.

비어 있음 · 불러오는 중 · 오류 구분

세 상황이 모두 같은 회색 한 줄로 보여 구분이 되지 않던 문제를 정리했습니다.

상태	표시
데이터 없음	회색 안내 문구
불러오는 중	맥동하는 점 + 문구 (`prefers-reduced-motion` 존중)
오류	빨간 배경 + 좌측 강조선

특히 오류가 "데이터 없음"과 같아 보이던 것이 문제였습니다 — 실패한 화면을 빈 화면으로 오해하게 됩니다.

관리자 UI 회귀 테스트 (개발자용)

어드민 화면은 Go raw string 하나(`internal/proxy/admin_ui.go`) 안에 들어 있어 `go test` 가 그 JavaScript를 실행하지 않습니다. 두 종류의 테스트로 보완합니다.

파일	검사	의존성
`admin_ui_static_test.go`	raw string 무결성(내부 백틱), `alert()` 잔존, 읽기 전용 엔드포인트의 성공 알림, 토스트의 `innerHTML` 사용, 오버레이 dialog 시맨틱·포커스 헬퍼, 빈/로딩/오류 스타일 분리, 원시 타임스탬프 노출, **검증 대상 필드 id 실재 여부**	없음 (항상 실행)
`admin_ui_behavior_test.go` + `testdata/admin_ui_behavior.js`	토스트 동작, **입력 검증 표시**, 시간 포맷팅, 명령 팔레트 매칭·인덱스, 포커스 가둠·복원, 표 래핑 멱등성 — 스텁 DOM에서 실제 함수를 실행	node (없으면 skip)

동작 검사는 단독 실행도 됩니다: `cd internal/proxy && node testdata/admin_ui_behavior.js`

CSS 주석에 백틱을 넣어 raw string을 끊거나, 일괄 치환으로 `alert()` 가 되살아나거나, 표 래퍼가 재렌더마다 중첩되는 실수가 실제로 있었고 모두 이 테스트가 잡습니다.

화면 찾기 (명령 팔레트)

상단의 🔎 **화면 찾기** 버튼 또는 `Ctrl`/`⌘`+K(입력 중이 아닐 때는 k)로 엽니다. 화면 이름이나 경로 일부를 입력해 바로 이동합니다.

- 그룹 메뉴 9개에 흩어진 35개 화면을 **어느 그룹에 있는지 몰라도** 찾을 수 있습니다
- 한글 이름(라우팅)과 경로(`routing`) 양쪽으로 검색되며, 연속되지 않은 글자도 매칭됩니다(`rtn` → `#/routing`)
- 화면 이동 외에 `다시 불러오기` · `다크 모드 전환` · `단축키 도움말` · `XView 요청 탐색` 동작도 포함됩니다
- 목록은 **열 때 상단 메뉴에서 생성**하므로, 권한(`allowed_tabs`)으로 가려진 화면은 팔레트에도 나오지 않습니다
- ↑↓ 이동, `Enter` 열기, `Esc` 닫기

화면 로딩·오류 표시

화면 전환 시 상단에 얇은 진행 표시줄이 나타납니다. 이전에는 데이터를 받아오는 동안 **이전 화면이 그대로 보여** 클릭이 먹었는지 알 수 없었습니다.

화면을 불러오지 못하면 **어느 화면이 실패했는지**와 오류 내용, 그리고 `다시 시도` · `다른 화면 열기` 버튼이 표시됩니다. 이전에는 오류 메시지 한 줄만 나와 다음 행동을 알 수 없었습니다.

좁은 화면 · 노트북 대응

- **등록/편집 폼**이 좁은 화면에서 한 줄 배치로 바뀝니다. 각 폼은 컬럼 배치를 인라인 스타일로 지정하는데, 인라인 스타일은 미디어쿼리로 덮이지 않아 이전에는 노트북·분할 화면에서 입력칸이 뭉개졌습니다(31개 폼).
- **표는 각자 가로 스크롤 박스** 안에서 표시됩니다. 표가 페이지를 넘치지는 않았지만 컬럼이 많으면 모든 컬럼이 눌리면서 단어가 중간에서 잘려 읽기 어려웠습니다. 이제 좁은 화면에서는 표만 좌우로 스크롤됩니다.

키보드 접근성 (모달·팔레트)

모달과 명령 팔레트는 `role="dialog"` · `aria-modal` 로 선언되고, 그 약속대로 동작합니다.

- **포커스 가둠**: Tab/Shift+Tab 이 오버레이 안에서만 순환합니다. 이전에는 뒤쪽 페이지로 빠져나가 화면에 보이지도 않는 요소에 포커스가 갔습니다
- **포커스 복원**: 닫으면 열기 전에 있던 위치로 포커스가 돌아갑니다. 이전에는 문서 맨 위로 되돌아가 목록에서 여러 항목을 연속으로 확인할 때 매번 처음부터 탐색해야 했습니다
- **초기 포커스**: 모달은 닫기 버튼, 팔레트는 검색 입력에 자동으로 잡힙니다

키보드 단축키

- `?` 도움말, `/` 검색 입력 포커스, `t` 테마, `r` 새로고침, `Esc` 모달 닫기
- `g` 후 한 글자: `d` (대시보드), `x` (XView), `w` (Waterfall), `l` (LLM 관측), `c` (MCP), `e` (에이전트), `v` (VCS), `r` (호출 이력), `p` (프롬프트 검색), `u` (사용자), `m` (팀), `i` (IP), `q` (사용 한도), `a` (안전), `s` (설정)

---

2. 탭 한눈에 보기

탭	무엇을 할까
대시보드	총 요청·토큰·KRW·평균 지연·첫 청크 지연, 24h/7d/30d 시계열 차트, 상위 사용자, 상태 분포, 이상 징후, 시간대 히트맵, 최근 20건
XView	요청 1건=점 1개의 응답시간 분포(스캐터)로 이상치를 발견하고, 점을 클릭하면 그 요청이 **왜** 그렇게 처리됐는지(라우팅·폴백·캐시·안전·비용·세션) 설명하는 eXplainability View
Waterfall	한 세션의 트랜잭션(요청)들을 시간순 간트 막대로 펼쳐, 첫 응답 대기(TTFB)·스트리밍 수신 구간과 요청 사이의 대기(생각) 시간을 보고 LLM 처리시간(busy) vs 대기시간(idle)을 분해
LLM 관측	Datadog LLM Observability 대응 Trace/Span/Session/Prompt/Patterns/Insights/Feedback/Evaluation 화면
MCP	MCP/tool 서버·도구 리더보드, 호출/오류 집계, 오류 호출 drill-down
라우팅	Intelligent Routing 학습 루프, routing preview/decision 조회, Provider Health ranking/degradation/trend 운영 화면
에이전트	코딩 에이전트(Claude Code/Cursor/Roo/Qwen…)별 성공률·평균 비용·지연·도구 오류율 리더보드 (User-Agent 기반)
VCS	GitLab/Bitbucket 커밋·MR 을 세션·사용자에 연결한 목록(Prompt→Commit→MR 상관). 저장소/세션/키/유형 필터
호출 이력	IP/모델/언어 필터로 검색, 두 행 선택 후 비교, 행 클릭 시 상세 모달
프롬프트 검색	키워드/언어/IP/키/기간으로 마스킹 프롬프트 검색, CSV 다운로드, 저장된 필터
사용자	Proxy API 키별 사용량·비용 + **AI 활용지수**(요청·활동일·커밋·머지MR·성공률 기반), 키 클릭 시 상세 drill-down
팀	**팀 벤치마크**(월비용 × 생산성 점수) + 팀별 사용량, 팀 클릭 시 API 키/모델/IP/LLM trend drill-down
IP	호출 IP 별 사용량, IP 클릭 시 일별·모델별·키별 상세
사용 한도	API 키/팀/IP/전체 단위 일별·월별 토큰·KRW 한도
안전	Kill Switch + **AI Incident(프로바이더 장애 감지)** + 비용 가드 + AI 정책 엔진 + Secret Firewall 이벤트 + 승인 큐 + 정책 판단 이벤트 + 알림 규칙 + 발화 이력
설정	Proxy API 키 발급/비활성화, **로그인 계정·팀 관리(RBAC)**, 업스트림 provider, 보존 정책, 변경 이력 + 감사 CSV

2-1. 인증 / RBAC

기본 운영은 기존 호환 모드(`AUTH_ENABLED=false`)입니다. `AUTH_ENABLED=true` 로 켜면 Admin API는 `/auth/login` 으로 받은 JWT access token이 필요하고, refresh token은 `/auth/refresh` 호출마다 rotation 됩니다. `/auth/logout` 은 세션/refresh token을 폐기합니다. **어드민 UI는 이 모드에서 `/admin` 진입 즉시 이메일/비밀번호 로그인 화면을 띄우고**, 토큰 만료 시 자동 갱신, 만료 실패 시 재로그인을 유도합니다(1장 참고).

역할은 `super_admin` , `admin` , `team_admin` , `developer` , `viewer` , `service_account` 를 지원합니다. API key는 원문을 저장하지 않고 hash만 저장하며, `expires_at` , `revoked_at` , `allowed_ips` , `scopes` , `allowed_models` / `denied_models` , `allowed_providers` / `denied_providers` , `budget_limit_krw` 정책을 검사합니다. Scope는 `chat:completion` , `embeddings:create` , `models:read` , `admin:read` , `admin:write` , `routing:read` , `routing:write` , `observability:read` , `costs:read` , `security:read` , `mcp:use` , `mcp:admin` 입니다.

인증/정책 이벤트는 `GET /admin/audit/auth-events` 에 기록됩니다. 기록 대상은 `login_success` , `login_failed` , `api_key_created` , `api_key_revoked` , `api_key_denied` , `ip_denied` , `scope_denied` , `model_denied` , `budget_denied` , `role_changed` 입니다.

계정 · 팀 관리 (설정 탭 → "로그인 계정 · 팀 (RBAC)")

동작	방법	비고
계정 생성	폼: 이메일·초기 비밀번호·이름·역할·팀	`POST /admin/users`. team_admin 은 자기 팀 계정만 생성 가능하며 자기 역할보다 높은 역할은 지정 불가
역할 변경	표의 역할 드롭다운 선택	`PATCH /admin/users/{usr_id} {"role":"…"}`. `role_changed` 감사 기록. 자기 역할 이상으로 승격하거나 자기보다 높은 역할 계정 수정은 차단. team_admin 불가
팀 변경	표의 팀 드롭다운 선택	`PATCH … {"team_id":"…"}` (빈 값 = 팀 해제). 존재하지 않는 팀은 400. 멤버십이 교체됩니다
비활성화/활성화	표의 버튼	`PATCH … {"status":"disabled"}`. **비활성화 즉시 그 계정의 모든 세션·refresh token 폐기** → 발급된 access token도 바로 거부됩니다
팀 생성/조회	팀 폼 또는 팀 목록	`POST /admin/teams` / `GET /admin/teams`. team_admin 은 팀 생성 불가, 자기 팀만 조회 가능
인증 이벤트 확인	같은 섹션 하단 표	실패 계열(login_failed/scope_denied/…)은 빨간 배지

API 키 스코프 편집 · 영구 삭제

- **스코프 편집**: 설정 탭 프록시 API 키 표의 "스코프" 버튼 → 12개 스코프 체크박스 모달 → 저장(`PATCH /admin/api-keys/{id} {"scopes":[…]}`). 스코프 밖 호출은 403 + `scope_denied` 감사 기록. 로그인 모드에서 team_admin 은 자기 팀 키만 볼 수 있고, 자기 역할에 없는 스코프나 높은 역할은 API 키에 부여할 수 없습니다.
- **영구 삭제**: 같은 표의 "삭제" 버튼(`DELETE /admin/api-keys/{id}?hard=1`). 비활성화(soft)와 달리 키 행을 제거하며 되돌릴 수 없습니다. **`AUTH_ENABLED=true` 에서는 super_admin 전용**(그 외 역할 403), 레거시 모드에서는 전권 관리자 토큰으로 가능. 과거 사용 이력 통계는 보존됩니다(이후 external 표시). `api_key.delete` 관리자 감사 + `api_key_revoked(hard_delete)` 인증 이벤트 기록.

`AUTH_ENABLED=false` 상태에서도 섹션은 보이지만(사전 준비용), 로그인 모드가 꺼져 있다는 경고 배너가 표시됩니다.

2-2. Governance Layer

거버넌스 레이어는 요청을 upstream으로 보내기 전에 정책, secret, 승인, MCP tool 위험도를 평가합니다. 정책은 안전 탭의 "AI 정책 엔진" 또는 `GET/POST /admin/policies` 로 관리하며 rule은 `conditions` 와 `actions` JSON을 사용합니다. 예: `{ "contains_secret": true, "block": true }`, `{ "risk_score": ">80", "require_approval": true }`, `{ "team": "security", "allow_models": ["gpt-5", "claude-sonnet"] }`. `team` 조건은 팀 ID와 팀명을 모두 매칭하며, 엄격히 구분하려면 `team_id` 또는 `team_name` 을 사용할 수 있습니다.

Secret Firewall은 API key, JWT, private key, password, AWS secret, DB connection string, access token을 탐지하고 정책에 따라 `detect`, `mask`, `block` 으로 처리합니다. 탐지 이벤트는 안전 탭의 "Secret Firewall 이벤트" 또는 `GET /admin/security/secrets` 에서 확인합니다. `/admin/security/secrets` 는 `request_id`, `action`, `secret_type`, `team_id`, `api_key_id`, `user_id`, `location`, `matched_hash`, `window`, `since`, `limit` 필터를 지원합니다. 정책 판단 이벤트는 안전 탭의 "정책 판단 이벤트", `GET /admin/policies/decisions`, 요청별 XView/요청 상세의 Governance 패널에서 확인합니다. 매칭 규칙이 없는 정상 허용 경로도 감사 추적을 위해 `decision=default` 이벤트를 남깁니다. 다만 운영 화면의 실질 판단 수(`policy_decision_count`)는 `default` 를 제외하고, 원시 감사 이벤트 수는 `policy_decision_total` 로 별도 노출합니다. `/admin/policies/decisions` 는 `request_id`, `decision`, `policy_id`, `rule_id`, `team_id`, `api_key_id`, `user_id`, `endpoint`, `phase`, `model`, `provider`, `window`, `since`, `limit` 필터를 지원합니다. 승인 필요 요청은 `pending` approval을 만들고 `X-Governance-Approval-ID` 헤더를 반환합니다. 운영자는 안전 탭의 "승인 큐" 또는 `GET /admin/approvals` 로 승인 항목을 조회하고, `POST /admin/approvals/{id}/approve` / `/reject` 로 결정합니다. `/admin/approvals` 는 `id`, `request_id`, `status`, `team_id`, `api_key_id`, `user_id`, `subject_type`, `subject_id`, `decided_by`, `reason`, `window`, `since`, `limit` 필터를 지원합니다. 클라이언트는 승인된 ID를 같은 헤더로 재전송합니다.

MCP Security Center는 MCP 탭 또는 `GET/POST /admin/mcp/tools` 에서 tool별 `low¦medium¦high¦critical` risk와 `allow¦require_approval¦block` action을 관리합니다. `/admin/mcp/tools` 는 `server`, `tool`, `api_key_id`, `mcp_only`, `risk_level`, `action`, `configured`, `window`, `limit` 필터를 지원하며, UI에서는 tool 행에서 risk/action/note를 바로 저장할 수 있습니다. `/admin/anomalies` 는 기존 모델 이상탐지에 더해 비용 anomaly event 뷰를 반환하며, replay/golden/context 운영 API는 `/admin/replay`, `/admin/golden-prompts`, `/admin/contexts` 입니다.

---

3. 대시보드

화면 위에서 아래로:

1. **요약 KPI**: 총 요청수 / 총 토큰 / 누적 KRW / 전체 지연 P50/P95/P99 / 첫 청크 지연 P50/P95/P99
2. **시계열 차트**: 24h(시간별) / 7d(일별) / 30d(일별) 토글. 실선은 요청 수, 점선은 KRW 비용. 점에 마우스를 올리면 토큰·비용까지 툴팁.
3. **상위 사용자**: 요청 수 기준 Top 5. 클릭 시 그 사용자의 상세 페이지로 이동.
4. **상태 분포**: 2xx/3xx/4xx/429/5xx 비율 막대 + 표.
5. **이상 징후**: 모델별 요청당 비용·지연을 최근 6시간 vs 7일 기준선으로 비교해 z-score ≥ 3 인 급변(급증/급감)을 표로 표시. 모델 가격 변동·성능 저하·폭주를 선제적으로 포착. `/admin/anomalies` API, `anomaly_zmax` 알림 지표.
6. **IP별 / 모델별 / 언어별** 표 (헤더 클릭 시 정렬).
7. **시간대 히트맵**: Asia/Seoul 기준 요일(가로) × 시간(세로). 색이 짙을수록 그 시간대 호출이 많음. 트래픽 패턴 + 비정상 시간대(새벽 폭주 등) 발견용.
8. **최근 호출 이력 20건**.

3-1. XView (트랜잭션 응답시간 분포)

평균 응답시간 차트는 9초짜리 장애가 100ms 요청들 사이에 묻혀도 "평균 130ms 정상"처럼 보입니다. XView는 **요청 1건을 점 1개**로 찍어 (가로=시간, 세로=응답시간) 이상치를 즉시 드러냅니다.

- **세로축 스케일**: 로그(기본) / 선형 토글. 로그 스케일이면 100ms 군집과 9초 이상치를 한 화면에서 봅니다.
- **지표 토글**: 전체 응답시간 / 첫 청크 지연(스트리밍 TTFB).
- **보조선**: P50(회색)·P95(노랑)·P99(빨강) 백분위 기준선.
- **색상 분류**:
  - 🟢 캐시 히트 (`provider=cache`)
  - 🔵 정상
  - 🟡 폴백 (upstream 장애로 대체 provider 사용)
  - 🔴 오류 (status ≥ 400, kill switch·정책 차단 포함)
  - 🟣 고비용/복잡 (토큰이 상위 10% 또는 4000 이상)
- **창**: 5m / 15m / 1h / 6h / 24h. **필터**: 모델, endpoint.
- **실시간 흐름**: 상대 구간에서는 기본으로 켜지며, 새 요청을 약 1.5초마다 현재 필터에 추가합니다. 승인 결과처럼 나중에 바뀌는 메타데이터도 5분마다 현재 창을 다시 투영해 새로고침 없이 반영합니다. 브라우저 탭을 벗어나면 일시정지하고 돌아오면 이어서 조회합니다. `to` 로 종료 시각을 고정한 과거 조회는 재현성을 위해 실시간 모드를 사용할 수 없습니다.
- **조사 보기**: 현재 기간·시간대·지표·스케일·색상 모드·모델·endpoint 조건을 이름으로 저장하고 다시 실행하거나 덮어쓰기·삭제할 수 있습니다. 공유 링크는 현재 조건을 직접 포함해 저장 항목 수명과 무관하게 재현됩니다.
- **드릴다운**: 점에 마우스를 올리면 모델·provider·지연·토큰·비용·상태 툴팁, 클릭하면 요청 상세 모달.
- 점이 6000건을 넘으면 최근 6000건으로 제한(범례 옆에 표시).

API: `GET /admin/scatter?window=1h&metric=latency&model=&endpoint=&limit=6000` — 점 배열(`request_id, created_at, ingested_at, latency_ms, first_chunk_ms, status_code, provider, model, total_tokens, cost_krw, stream, tool_count, failover`)과 `truncated`, 증분 조회용 `cursor` 를 반환합니다. 이후 `GET /admin/xview/delta?`

`after_ingested_at=...&after_request_id=...` 로 새 점과 다음 `cursor`, `has_more` 를 조회합니다. 호환 구간 갱신은 `reconcile=true`, 승인 상태 등 현재 창의 변경 가능한 메타데이터 재투영은 `refresh=true` 를 사용하며, 클라이언트는 `request_id` 로 중복을 제거해야 합니다. 새 버전의 로그 트랜잭션은 DB에서 커밋 직전 단조 커서를 배정받아 다중 인스턴스에서도 순서를 보존합니다.

저장 API는 `GET /admin/saved-filters?view=xview`, `POST /admin/saved-filters`, `GET¦PATCH¦DELETE /admin/saved-filters/{id}` 를 사용합니다. XView 저장 파라미터는 allowlist와 enum·기간·시간대 검증을 통과해야 하며 `window` 와 `from/to` 는 함께 저장할 수 없습니다.

모델 집계 API는 `GET /admin/xview/models?window=1h&top=5&models=...` (모델별 건수·오류율·P50/P95/P99·토큰·비용·폴백·거버넌스), `GET /admin/xview/model-series?window=24h&bucket=hour¦day` (모델별 시계열), `GET /admin/xview/model-outliers?window=1h` (P95 초과·오류·폴백·거버넌스 태그)입니다. 세 엔드포인트는 조회 창의 요청을 최신순으로 최대 20,000건까지만 읽어 집계하므로, 그보다 붐비는 창은 **최근 일부만으로 계산된 요약**이 나옵니다. 숫자만 봐서는 구분되지 않기 때문에 응답이 실제 집계 범위를 함께 알려줍니다.

필드	의미
`truncated`	창의 요청을 전부 읽지 못했으면 `true`
`sample_size`	실제로 집계에 사용한 요청 건수
`aggregate_limit`	이번 집계의 상한(기본 20,000)
`covered_since`	잘린 경우에만 존재. 이 시각 **이후** 요청만 집계에 포함됨

`truncated=true` 인데 `covered_since` 가 조회 시작 시각보다 한참 뒤라면, 그 사이 구간은 트래픽이 없었던 것이 아니라 읽지 않은 것입니다. 창을 좁히거나 `models=` 로 모델을 좁혀 다시 조회하세요. 특히 시계열은 잘린 구간의 버킷이 통째로 빠지므로 조용한 시간대처럼 보일 수 있습니다.

eXplainability View (점 클릭 → "왜 이렇게 처리됐나")

스캐터의 점(또는 호출 이력/상세 모달의 "🧭 XView 설명" 버튼)을 클릭하면 그 요청 1건의 처리 근거를 6개 패널로 설명합니다. 감사·보안·비용 통제 근거가 요청별로 남으므로 금융권 등 규제 환경에 적합합니다.

패널	내용
🧭 라우팅	선택된 provider·모델, 라우팅 근거 코드(`route_reason` : 헤더 지정/쿼리/모델 패턴 자동/기본/auto router 등), 매칭된 패턴, **복잡도 점수(0~100)와 티어**(simple/standard/complex/reasoning), risk score, provider health score, fallback chain, 사람이 읽는 decision reason
🔁 폴백	폴백 발생 여부, 최초→대체 provider, 사유(전송 실패 등)
🟢 캐시	캐시 히트 여부, cached 토큰, **절감액**(전체 캐시 / 프롬프트 캐시)
🛡️ 안전	차단 여부, 마스킹 적용, 실패한 안전·보안 평가(PII/인젝션/독성/도구 인자 시크릿)
💰 비용	실제 비용 vs 정가(캐시 미적용 시), **절감액**, 토큰 분해(prompt/completion/cached/reasoning)
🧵 세션	세션 타임라인 링크, 스트리밍 여부, 원문 상세 링크

복잡도 점수는 프롬프트 토큰·대화 깊이·도구 수 기반 휴리스틱 추정치이며(모델 산출값 아님) UI에 그 사실이 명시됩니다.

API: `GET /admin/requests/{id}/explain` → `{routing, fallback, cache, safety, cost, session}`. `routing` 에는 `chosen_model`, `chosen_provider`, `complexity`, `risk_score`, `health_score`, `fallback_path`, `route_reason`, `decision_reason` 이 포함됩니다. 이때 `fallback_path` 는 **실제로 일어난** 폴백 경로만 담으며, 폴백이 없었다면 비어 있습니다(사전 예측 경로는 `/admin/routing/preview` 의 `fallback_plan`). `GET /admin/requests/{id}/links` 는 요청 상세·XView·Waterfall·MCP Waterfall·Text2SQL Timeline·라우팅 결정 연결 정보와 카운트를 한 번에 반환합니다. 이때 `policy_decision_count` 는 `decision=default` 를 제외한 실질 거버넌스 판단 수이고, `policy_decision_total` 은 원시 감사 이벤트 수입니다.

> **provider 라우팅과 폴백의 전체 규칙은 ROUTING_GUIDE.md 를 참고하세요.** 선택 순서, 폴백 4조건, 폴백이 안 되는 흔한 이유, 구성 레시피를 한 곳에 모았습니다. 어드민에서는 설정 탭 → 업스트림 프로바이더 → 📖 라우팅 · 폴백 동작 설명 열기 버튼으로 같은 내용을 모달로 볼 수 있습니다.

Intelligent Routing Engine API:

- `POST /admin/routing/preview` — 실제 upstream 호출 없이 `auto` / `vibe/auto` / `vibe-coders/auto` 라우팅 결과 미리보기. 응답에는 자동화·필터링용 `route_reason` 과 사람이 읽는 `decision_reason` 이 함께 포함됩니다. body에 `api_key_id` 를 넣으면 해당 API 키의 allowed/denied model/provider 정책까지 반영합니다(team_admin은 자기 팀 키만 가능)
- `GET¦POST /admin/routing/pattern-conflicts` — 활성 provider의 `model_patterns` 교차·중복·catch-all 충돌 분석. GET은 현재 설정과 선택적 `model` 경로를 조회하고, POST는 `provider_name`, `model_patterns`, 선택적 `model` 을 받아 저장 없이 변경 영향을 미리 계산합니다. 응답에는 **폴백 커버리지**(`coverage`, `summary.failover_ready_provider_count` / `failover_uncovered_provider_count`)와 기본 provider 패턴 유무(`default_provider_has_patterns`)가 함께 포함되고, `model` 을 넘기면 시뮬레이션에 실제 폴백 후보 체인(`simulation.failover_candidates`)과 폴백 불가 사유(`failover_blocked_reason`)가 들어갑니다. 폴백 후보는 `model_patterns` 매칭으로만 만들어지므로, 패턴이 겹치지 않는 provider끼리는 서로 폴백되지 않습니다.
- `GET /admin/routing/decisions` / `GET /admin/routing/decisions/{id}` — 요청별 selected model/provider, complexity/risk/health, fallback path, decision reason 조회
- `GET /admin/routing/health` — 최근 latency/p95/timeout/429/5xx/fallback rate 기반 provider health score 조회. 응답에는 provider 원본 점수와 함께 `ranking`, `degraded`, `alerts`, `trend`, 그리고 **회로 차단기 상태**(`breakers.enabled` / `threshold` / `cooldown_seconds` / `states`)가 포함됩니다. 회로 차단기는 연속 실패한 provider를 폴백 후보에서 자동 제외하며(기본 5회 → 30초 차단 → 1건 탐침 복구), 관리자 화면의 `회로 차단기` 패널에서 상태 확인과 수동 해제가 가능합니다. 수동 해제 API는 `POST /admin/routing/breaker-reset` (`{"provider":"이름"}`, 비우면 전체)입니다. 같은 모델을 여러 provider가 서비스할 때는 `UPSTREAM_LOAD_BALANCE=round_robin` 으로 **세션 단위 라운드로빈**이 가능하며, 세션은 세션 헤더 → body 필드 → 대화 프리픽스 해시 → 추론 세션 순으로 식별합니다(qwen code 등 세션 식별자를 보내지 않는 에이전트도 대화별로 구분). 분산 검증은 `GET /admin/routing/balancer?model=&window=` (균형도 `balance_index`, provider 풀, 밸런서 intent vs 요청 로그 actual)과 라우팅 탭의 `로드밸런싱 · 세션 고정` 패널에서 하고, `POST` 로 세션 고정을 해제(노드 드레인)합니다. 자세한 내용은 ROUTING_GUIDE.md 3-2·3-3 절 참고. 폴백이 성공하면 호출자에게는 정상 응답이 가서 장애가 감춰지므로, 안전 탭 알림 규칙의 `failover_rate` 지표로 폴백률 알림을 함께 걸어두는 것을 권장합니다. 관리자 화면은 라우팅 탭의 `Provider Health` 하위 화면(`#/routing/health`)에서 같은 데이터를 표시합니다.

`auto` 계열 모델 별칭은 일반 라우팅 규칙보다 우선합니다. `X-Proxy-Provider` 또는 `?provider=` 로 provider 를 고정해도 auto 모델 rewrite 는 계속 수행되고, provider 선택만 클라이언트 지정값을 따릅니다. Provider `model_patterns` 가 `vibe/*` 처럼 alias 기준으로 등록되어 있으면, 선택된 실제 모델 패턴이 없을 때 요청 alias 기준 provider도 후보로 사용합니다. `GET /v1/models` 는 SDK 호환성을 위해 인증 모드에서도 공개 조회로 처리합니다.

3-2. Waterfall (트랜잭션 타임라인)

XView가 "요청 분포"를, 세션 비용 타임라인이 "누적 비용 곡선"을 본다면, **Waterfall은 한 세션 안에서 시간이 어디로 흘렀는지**를 봅니다. 분산 트레이싱 도구(Jaeger·크롬 네트워크 워터폴)와 같은 간트 막대 표현입니다.

세션은 어떻게 묶이나 (명시적 + 추론)

Waterfall·세션 비용 타임라인·LLM Session Explorer·에이전트 루프 탐지는 모두 `session_id` 로 요청을 묶습니다. 세션은 2단계로 정해집니다.

1. **명시적**: 클라이언트가 보낸 값(헤더 `X-Session-ID` / `X-Vibe-Session-ID` / `X-Conversation-ID` 또는 바디 `session_id` / `chat_id` / `conversation_id` / `thread_id` / `metadata.*`). Langflow·OpenWebUI 등이 해당.
2. **추론**: 명시적 세션이 없으면(Claude Code·Cursor·Roo·Qwen 등 대부분의 코딩 툴) `api_key + IP + User-Agent` (+ 옵션 `X-Vibe-Repo` / `X-Vibe-Branch`) 신원과 **슬라이딩 비활성 윈도우**로 자동 생성. ID는 `sess_<12hex>` . 같은 클라이언트의 연속 호출은 한 세션이 되고, `SESSION_IDLE_TIMEOUT` (기본 30분) 이상 비활성이면 새 세션이 시작됩니다.

`SESSION_INFERENCE_ENABLED=false` 로 두면 추론을 끄고 요청별(`trace:<id>`)로 분리됩니다(세션 묶음 없음). 추론 세션은 DB의 `inferred_sessions` 테이블에도 저장되므로 게이트웨이 재시작 후에도 idle window 안에 들어온 같은 클라이언트는 기존 `sess_<12hex>` 를 복구합니다. `SESSION_IDLE_TIMEOUT` 을 지난 추론 세션은 새 세션으로 분리되고, 오래된 복구 상태는 자동 정리됩니다. 세션 헤더가 전혀 없던 레거시 요청은 `no-session` 으로 묶입니다.

보는 법

1. `Waterfall` 탭 → 세션 목록에서 "보기" (또는 XView 설명 패널/세션 타임라인 모달의 "워터폴" 링크).
2. 각 요청이 가로 막대 한 줄. 가로축은 세션 시작 기준 경과 시간(벽시계).
   - **연한 부분** = 첫 응답까지의 대기(TTFB) — 모델이 첫 토큰을 내놓기까지.
   - **진한 부분** = 스트리밍 수신 구간.
   - **막대 사이 빈 공간** = 클라이언트(에이전트/사람)의 대기·생각 시간 — 서버는 놀고 있던 시간.
3. 막대 색은 XView와 동일: 파랑=정상, 초록=캐시 히트, 노랑=폴백, 보라=고복잡도(점수 ≥ 70), 빨강=오류. **빨간 테두리/**⚠ = 느린 요청.
4. 막대/표 행 클릭 → 해당 요청의 XView 설명(라우팅 근거)으로 이동.

병목 분석 (자동)

차트 위 **병목 분석** 카드가 눈으로 찾을 필요 없이 핵심을 짚어줍니다.

- **가장 느린 요청**: 최대 `total_ms` 요청 + 전체 대비 %. 클릭하면 그 요청의 라우팅 근거로 이동.
- **가장 긴 대기(생각)**: 최대 `gap_before_ms` + 전체 대비 %. 에이전트가 어디서 오래 멈췄는지.
- **판정 문구**: idle 〉 busy면 "클라이언트 대기 병목", 아니면 TTFB·스트리밍 중 큰 쪽을 지목.

세션 시간 구성 (스택 바)

요약 아래 가로 스택 바가 세션 전체 시간을 세 조각으로 분해합니다: **첫 응답 대기(Σ TTFB)** / **스트리밍 수신(Σ 본문)** / **클라이언트 대기(idle)**. "느리다"가 모델 큐(TTFB)인지, 긴 출력(스트리밍)인지, 에이전트 사고(idle)인지 한눈에 구분됩니다.

상단 요약 지표

지표	의미
총 소요(wall)	세션 첫 요청 시작 ~ 마지막 요청 종료까지 벽시계 시간
LLM 처리(busy)	실제 업스트림이 일한 시간(요청 구간들의 **합집합**, 동시 요청 중복 제거) + 처리율(busy/wall)
대기/생각(idle)	wall − busy. 이 값이 크면 병목은 모델이 아니라 클라이언트 쪽 사고/도구 루프입니다.
느린 요청	`slow_ms` 기준 초과 요청 수. 기준 미지정 시 `max(3000ms, p95)` 자동. 툴바의 "느림 기준(ms)" 입력으로 조정.
누적 비용·토큰·도구	세션 전체 합계

busy/idle 분해는 "느리다"의 원인이 **LLM인지 클라이언트 대기인지**를 가르는 핵심 단서입니다. 처리율이 낮은데 체감이 느리면 모델 증설이 아니라 에이전트 동작을 봐야 합니다.

필터 · 내보내기

- **분류 필터**: 범례 칩을 클릭하면 해당 분류(오류/캐시/폴백/고복잡도/정상)를 차트·표에서 숨기거나 다시 표시. 예: "오류만" 보기.
- **CSV 내보내기**: 현재 세션 스팬 전체를 CSV로 저장(엑셀 한글 대응 BOM 포함). 오프라인 분석·증빙용.

API

`GET /admin/waterfall?session_id=<id>&limit=<n>&slow_ms=<ms>` → `{session_id, requests, wall_ms, busy_ms, idle_ms, busy_ratio, wait_ms, stream_ms, slow_ms, slow_count, bottleneck, total_cost_krw, total_tokens, tool_calls, categories, spans[]}`. 각 span은 `start_offset_ms` (세션 시작 기준), `ttfb_ms`, `total_ms`, `gap_before_ms` (직전 대기), `category`, `slow` 를 서버에서 미리 계산해 내려줍니다. `bottleneck` 은 `{slowest_seq, slowest_ms, slowest_pct, longest_gap_seq, longest_gap_ms, longest_gap_pct}`. `slow_ms` 미지정 시 `max(3000, p95)` 자동. 세션 헤더가 없던 요청들은 `session_id=no-session` 으로 묶입니다.

---

4. LLM 관측

Datadog LLM Observability의 핵심 기능을 게이트웨이 내부 데이터로 재구성한 탭입니다.

영역	내용
Trace Explorer	요청 단위 trace 목록. session, prompt, 모델/provider, 첫 청크/전체 지연, 토큰·비용, tool count, 상태를 한눈에 확인. 상세 모달에는 파생 LLM/tool span 표시
Session Explorer	`session_id` 별 요청 수, 토큰, KRW, 오류, 평가 실패, 최초/최근 시각. 행의 `타임라인 > 보기` 로 세션 비용 타임라인 모달 표시
Session Timeline	한 세션의 턴을 시간순으로 펼쳐 누적 비용 곡선(SVG)과 턴별 모델·상태·첫청크·토큰·비용·누적비용·도구 호출 표를 보여줌. 점 색: 초록=정상, 노랑=평가실패, 빨강=오류. 어떤 턴에서 비용이 급증했는지 한눈에 파악. API: `GET /admin/llm/session?session_id=`
Prompt Tracking	prompt name/version 별 호출 수, 평균 지연, 토큰·비용, 오류, 평가 실패
Prompt Compare	Prompt Tracking 행에서 `비교` 를 눌러 버전 간 호출량, 토큰, KRW, 지연, 오류율, 평가 실패율 차이 확인. 상단 `API 키 ID` / `팀` 필터가 켜져 있으면 같은 스코프 안에서만 비교. baseline 미지정 시 가까운 이전 버전, 없으면 최근성 기준 대체 baseline 자동 선택하며, 선택 근거와 추천 후보 목록을 모달 상단에 표시. 추천 후보는 3/5/10개로 조절 가능하고, 버튼으로 바로 눌러 baseline을 교체 가능하며, 각 후보에 호출량/평균 지연/오류율/평가 실패율/최근 시각과 정렬 기준이 함께 보임
Patterns	최근 프롬프트를 debugging/testing/refactoring/security/prompt-injection-risk 등 운영 토픽으로 자동 묶음
Insights	평가 실패, 프롬프트 인젝션 위험, usage 누락, 느린 첫 청크, 오류 세션을 최근 윈도우 기준으로 자동 추출. 각 행의 `열기` 로 관련 trace/prompt/evaluation 위치로 즉시 drill-down 하고, prompt 계열 인사이트는 `비교`, session 계열 인사이트는 `세션 묶음` 으로 최근 trace bundle 모달을 바로 열 수 있음. 세션 묶음 모달에서는 JSON/CSV 다운로드 가능
Trend	최근 24시간/7일/30일 기준으로 요청량, 비용, 평가 실패, 부정 피드백, human/eval alignment 흐름을 시계열로 표시
Feedback	운영자가 trace 상세에서 `좋음/문제 있음/중립` 피드백과 라벨, 코멘트를 남기고 최근/라벨별/prompt별 피드백과 alignment를 집계
Evaluation	gateway-managed 평가(prompt PII, prompt injection, toxicity, completion, usage, first chunk latency)와 외부 평가 결과

LLM 관측 탭 상단 필터에서 `API 키 ID`, `팀` 값을 넣으면 trace/session/prompt/patterns/insights/evaluation/feedback/timeseries 패널을 해당 범위로 좁혀 볼 수 있습니다. drill-down 링크로 들어오면 `model`, `session_id`, `prompt_name`, `prompt_version`, `evaluation_name` 까지 함께 걸립니다. 사용자 상세와 팀 상세에는 같은 스코프를 채운 `필터된 LLM 보기` deep link가 있습니다.

정확한 세션·프롬프트 집계를 원하면 클라이언트가 다음 헤더를 보내도록 설정합니다.

```
X-LLM-Session-ID: sess-123
X-LLM-Prompt-Name: code-review
X-LLM-Prompt-Version: v7
X-LLM-Prompt-Variables-Hash: vars-sha256
```

요청 body의 `metadata.prompt`, `metadata.prompt_tracking`, `metadata._dd.ml_obs.prompt_tracking` 에 구조화 프롬프트 메타데이터가 들어와도 자동 수집합니다. 외부 평가기는 `POST /admin/llm/evaluations` 로 결과를 제출할 수 있고, 운영자는 trace 상세에서 사람 피드백을 남길 수 있습니다.

```
curl -X POST http://<host>:8080/admin/llm/evaluations \
  -H "Content-Type: application/json" \
  -d '{ "evaluations": [{
    "request_id": "req_xxxxxxxx",
    "name": "external.factuality",
    "category": "quality",
    "evaluator": "ci-check",
    "score": 0.82,
    "passed": true,
    "label": "pass"
  }] }'
```

```
curl -X POST http://<host>:8080/admin/llm/feedback \
  -H "Content-Type: application/json" \
  -d '{ "request_id": "req_xxxxxxxx", "rating": -1, "label": "hallucination", "comment": "근거 없는 답변" }'
```

4-1. MCP / Tool 관측

AI 코딩 도구(Roo Code·Cline·Cursor·Claude Desktop)가 MCP 서버나 함수 호출(tool/function calling)을 사용할 때, 게이트웨이는 어떤 서버·어떤 도구가 정의·호출·실패했는지 자동 집계합니다. 별도 설정 없이 OpenAI 호환 트래픽에서 추출됩니다.

수집 대상

종류	출처
정의(definition)	요청 `tools[]` / `functions[]` 카탈로그. Responses API `{type:"mcp", server_label}` 포함
호출(call)	응답 `tool_calls[]` (스트리밍·논스트리밍 모두), 요청 내 assistant `tool_calls`
결과(result)	요청의 `role:"tool"` 메시지. `{"isError":true}` 등은 오류로 분류

MCP 서버 분류 규칙

도구 이름에서 서버 라벨을 자동 추출합니다.

- `mcp__github__create_issue` → 서버 `github`, 도구 `create_issue` (MCP)
- `mcp__korean-law__search_law` → 서버 `korean-law` (MCP)
- Responses API `{type:"mcp", server_label:"filesystem"}` → 서버 `filesystem` (MCP)
- `github.create_issue`, `fs/read_file` → 서버 추출하되 MCP 플래그는 false (일반 함수)
- `web_search` 같은 built-in tool type → 서버 `builtin`

MCP 탭 구성

1. **요약 KPI**: tool 호출 수 / tool 오류 수(+오류율) / 고유 tool 수 / MCP 서버 수
2. **필터**: API 키 ID, 서버 라벨, "MCP만" 체크. 필터는 URL hash 에 보관되어 공유 가능
3. **MCP 서버별 표**: 서버마다 도구 종류·호출·오류·오류율·고유 키·**호출 IP(고유 IP 수 + 예시 IP)**·마지막 사용. 행 클릭 시 그 서버로 필터링. 서버 라벨이 `(none)` (일반 function tool, 서버 정보 없음)이어도 호출 IP·고유 키로 출처를 식별할 수 있습니다.
4. **Tool 리더보드**: (서버, 도구) 별 정의·호출·결과·오류·오류율·고유 키·**호출 IP**. `호출` / `오류` 버튼으로 해당 도구를 사용한 요청을 모달로 drill-down(요청별 IP·키 확인)
5. **에이전트 루프 의심**: 최근 24시간 동안 한 세션에서 같은 도구를 10회 이상 호출한 경우 표시. 폭주/무한루프 에이전트를 비용 사고 전에 발견. 30회 이상은 빨간색.
6. **도구 카탈로그 / 드리프트**: 서버별로 관측된 도구 목록과 최초/최근 관측 시각. 최근 24시간 내 처음 나타난 도구는 `신규` 배지로 강조(공급망 변조·권한 확대 탐지), 30일간 안 보이면 `미사용` 배지. 섹션 제목에 신규 도구 수 표시.
7. **MCP Gateway 업스트림**: 아래 게이트웨이 절 참고
8. **MCP 서버 정책**: 아래 보안 절 참고

MCP 업스트림 탐색 성능

등록된 업스트림의 도구·리소스·프롬프트 목록은 **동시에** 조회합니다. 따라서 전체 탐색 시간은 **가장 느린 업스트림 하나** 수준이며, 업스트림 수에 비례해 늘어나지 않습니다.

이전에는 순차 조회여서 업스트림당 최대 10초가 **합산**됐습니다. 캐시가 비어 있는 첫 `/mcp` 요청은 이 작업을 동기로 수행하므로 업스트림이 많으면 클라이언트가 그만큼 기다렸고, 백그라운드 갱신 예산이 45초라 **느린 업스트림이 5개를 넘으면 뒤쪽은 아예 탐색되지 못했습니다.**

- 응답이 돌아온 순서와 무관하게 **등록 순서대로 병합**하므로, 리소스 URI가 충돌할 때 "먼저 등록된 업스트림이 이긴다"는 규칙이 그대로 유지됩니다.
- 업스트림 하나가 실패해도 나머지 카탈로그는 정상 제공되며, 실패는 업스트림별로 기록되어 MCP 화면에서 확인할 수 있습니다.

MCP Gateway — 업스트림 서버 집약 (단일 /mcp)

vibe-coders 는 LLM 게이트웨이이자 **MCP 게이트웨이**입니다. 여러 업스트림 MCP 서버를 한 곳(`/mcp`, JSON-RPC 2.0 Streamable HTTP)에 모아, 클라이언트는 게이트웨이 하나만 연결하면 모든 서버의 도구를 씁니다.

- **등록 폼**: 이름(네임스페이스), Streamable HTTP MCP URL, Bearer 토큰(선택, 암호화 저장). ID는 이름에서 슬러그로 자동 생성되며 도구 접두사로 쓰입니다(`<id>__`).
- **표**: 이름/ID·URL·인증 여부·상태(사용/중지)·동작. 업스트림 도구 탐색에 실패하면 `탐색 오류` 배지(마우스 오버 시 사유).
- **등록 확인(중요)**: 각 행의 **"테스트/도구"** 버튼 → 그 업스트림에 **즉시 라이브 연결**(initialize+tools/resources/prompts list)하여 성공 여부와 노출되는 도구·리소스·프롬프트 목록(게이트웨이 네임스페이스 이름 포함)을 모달로 보여줍니다. 실패 시 사유 표시. API: `POST /admin/mcp/upstreams/{id}/probe` (라이브 연결과 발견 기록을 남기는 동작이므로 POST 입니다).
- **게이트웨이 호출 검증(curl)**:

```
# 집약된 도구 목록
curl -s http://<host>:8080/mcp -H "Authorization: Bearer <발급키>" \
  -H "Content-Type: application/json" -d '{"jsonrpc":"2.0","id":1,"method":"tools/list"}'
# 특정 도구 호출 (네임스페이스 이름 사용)
curl -s http://<host>:8080/mcp -H "Authorization: Bearer <발급키>" \
  -H "Content-Type: application/json" \
  -d '{"jsonrpc":"2.0","id":2,"method":"tools/call","params":{"name":"<업스트림ID>__<도구>","arguments":{}}}'
```

호출 후 MCP 탭의 서버/도구 표·관측에 해당 업스트림 이름으로 집계가 쌓입니다.

- **동작 원리**: 세 가지 MCP 1차 객체를 모두 집약합니다(30초 캐시).
  - **도구**: `tools/list` 를 합쳐 `<id>__<도구>` 로 노출, `tools/call` 은 네임스페이스로 라우팅.
  - **프롬프트**: `prompts/list` 를 `<id>__<프롬프트>` 로 노출, `prompts/get` 은 네임스페이스를 떼어 해당 업스트림으로 라우팅.
  - **리소스**: `resources/list`·`resources/templates/list` 를 집약(원본 URI 보존), `resources/read` 는 URI로 소유 업스트림에 라우팅. (URI 충돌 시 먼저 등록된 업스트림 우선) 모든 호출은 위의 MCP 서버 정책(allowlist/차단, 서버 라벨=업스트림 이름)과 사용자 귀속·MCP 관측에 통합 기록됩니다. `initialize` 응답은 tools+resources+prompts capability 를 광고합니다.
- **클라이언트 설정**: MCP 서버 URL 을 `http://<gateway>:8080/mcp` 하나로. 인증은 `/v1` 과 동일한 proxy key(`Authorization: Bearer …`).
- API: `GET¦POST /admin/mcp/upstreams` , `PATCH¦DELETE /admin/mcp/upstreams/{id}` . 메트릭은 기존 `proxy_mcp_tool_calls_total` / `proxy_mcp_tool_errors_total` 에 합산.

> 현재 Streamable HTTP 업스트림을 지원합니다(JSON 및 SSE 응답 모두 처리). 라우팅은 목록에 노출된 도구·프롬프트·리소스 대상입니다(템플릿으로 동적 생성된 미등록 URI 읽기는 미지원). stdio 서브프로세스 MCP 서버 연결은 향후 과제입니다.

Gateway MCP — 게이트웨이 자체 기능 (`/mcp/gateway`)

`/mcp` (업스트림 집약)과 **별개로**, 게이트웨이 자신의 기능을 MCP 도구로 노출하는 두 번째 엔드포인트입니다. 헷갈리기 쉬우니 운영자는 차이를 명확히 안내하세요.

	`/mcp`	`/mcp/gateway`
노출 대상	등록된 **외부 업스트림 MCP 서버**의 도구	**게이트웨이 자체 기능**(chat·라우팅·사용량·쿼터·Text2SQL·앱/워크플로 실행)
도구 이름	`<업스트림ID>__<이름>`	`gateway_*` (예: `gateway_chat` , `gateway_run_workflow`)
업스트림 등록	필요	**불필요**(내장 tool 집합)
실행 안전성	업스트림이 부수효과 수행	실행형 tool 은 `/v1` 파이프라인 재생(거버넌스·쿼터·정책·로깅 동일), 읽기형은 미실행 미리보기

- **카탈로그 확인**: 어드민 MCP 탭에 Gateway MCP 카탈로그(tools/resources/prompts)와 연결 설정이 표시됩니다. API: `GET /admin/gateway-mcp/info` .
- **무클라이언트 검증**: `POST /admin/mcp/gateway/test` 로 외부 MCP 클라이언트 없이 tool 을 name+arguments 로 직접 호출해 검증할 수 있습니다(읽기 진단).
- **인증·귀속**: `/v1` 과 동일한 proxy key. 모든 호출은 호출자 권한·쿼터·정책·MCP 관측에 통합됩니다.

MCP Tool Contract Registry (`/mcp/gateway` tool 계약·드리프트)

`/mcp/gateway` 가 노출하는 tool 의 입력/출력 스키마·위험등급(low/medium/high)·타임아웃·허용 역할·비용 정책·소유자를 **계약**으로 고정하고, 실제 노출 스키마와의 드리프트를 탐지합니다. MCP 탭 하단의 "MCP Tool Contract Registry" 섹션 또는 API로 관리합니다.

- `GET¦POST¦DELETE /admin/mcp/contracts` — 계약 목록/등록/삭제(등록 시 스키마 JSON 유효성·risk_level 검증).
- `POST /admin/mcp/contracts/validate` — 등록 계약과 실제 게이트웨이 tool 집합을 비교해 `missing`(tool 소실)·`drift`(입력 스키마 속성 차이: `declared_only` / `live_only`)를 보고합니다. `gateway` 외 namespace는 자동 비교 대상이 아닙니다(`not_checkable`). 속성 키 집합 비교이며 타입 심층 비교는 아닙니다.
- 활용: 게이트웨이 버전업으로 tool 시그니처가 바뀌면 드리프트로 조기에 드러나므로, 계약을 갱신하거나 클라이언트 영향도를 점검하는 운영 신호로 사용하세요.

MCP Discovery 가상 모델

`/v1/chat/completions`에서 `vibe/grounded`, `vibe/research`, `vibe/all-mcp`(별칭 `vibe/all_mcp`)를 모델명으로 호출하면 MCP Discovery 경로가 동작합니다. 후보 MCP는 설명·도메인·도구명·health를 기반으로 정렬되지만, agentic 경로에서는 selector 점수가 hard gate가 아니라 **백킹 LLM에게 넘길 후보 순위 가중치**로만 사용됩니다. 백킹 LLM이 직접 tool call 여부를 결정하고, 백킹 모델을 사용할 수 없을 때만 정적 fallback이 관련성 gate를 적용합니다.

백킹 Chat 모델은 `MCP_AGENTIC_MODEL` 환경변수 또는 어드민 `설정 > 런타임 설정`의 `mcp.agentic_model`에서 지정합니다. 비워두면 기존 auto-router가 정책 기반으로 선택합니다. 예를 들어 `qwen-plus`, `claude-sonnet-4`, `gpt-4.1`처럼 provider model pattern에 등록된 실제 모델명을 넣으면 다음 MCP Discovery 요청부터 즉시 반영됩니다.

MCP 서버 정책 (allowlist / 차단)

미승인 MCP 서버 사용을 차단해 섀도우 MCP·신뢰 경계 밖 서버 연결을 막습니다. MCP 탭의 "MCP 서버 정책" 섹션 또는 API로 관리합니다.

- **모드**: `block`(차단), `warn`(허용하되 경고 헤더·기록), `allow`(명시적 허용)
- **Allowlist 모드** 토글: 켜면 `allow`로 등록된 서버만 통과하고 나머지 MCP 서버는 모두 차단(화이트리스트). 끄면 `block`으로 지정한 서버만 차단(블랙리스트).
- 차단된 요청은 upstream에 도달하기 전 `HTTP 403 + X-MCP-Blocked-Server: <서버>`로 거부되고 호출 이력에 `blocked` provider로 기록됩니다. `warn` 서버는 `X-MCP-Warn-Servers` 헤더를 붙여 통과시킵니다.
- 정책 변경은 5초 캐시로 모든 인스턴스에 전파됩니다.

```
# github MCP 서버 차단
curl -X POST http://<host>:8080/admin/mcp/policies \
  -H "Content-Type: application/json" \
  -d '{ "server_label": "github", "mode": "block", "note": "외부 PR 자동화 금지" }'

# allowlist 모드 켜기 (등록된 allow 서버만 통과)
curl -X POST http://<host>:8080/admin/mcp/policies \
  -H "Content-Type: application/json" -d '{ "allowlist_enabled": true }'

# 정책 삭제
curl -X DELETE http://<host>:8080/admin/mcp/policies/github
```

MCP 관련 알림 / 평가

- 알림 지표(안전 탭): `tool_errors`(윈도우 내 tool 오류 수), `tool_error_rate`(오류/호출 비율), `tool_loop`(한 세션에서 한 도구의 최대 호출수 — 루프 임계), `mcp_new_tools`(윈도우 내 새로 관측된 도구 수 — 드리프트).
- 요청 상세의 LLM 평가에 `tools.no_error`(tool 결과 오류 여부), `tools.mcp_servers`(사용된 MCP 서버 수), `tools.args_no_secret`(도구 호출 인자·결과에 시크릿/PII 포함 여부)가 추가됩니다.
- 도구 호출 인자와 결과는 기존 마스킹 규칙(시크릿/PII)으로 스캔되어, 민감정보가 도구 입출력으로 새는지 `tools.args_no_secret` 평가로 감지합니다.
- 요청 상세의 trace span에 도구마다 개별 span(`mcp` / `tool` kind)이 표시됩니다.

- Prometheus: `proxy_mcp_tool_calls_total`, `proxy_mcp_tool_errors_total`, `proxy_mcp_blocked_total`.

API

```
curl -H "Authorization: Bearer $ADMIN_TOKEN" http://localhost:8080/admin/mcp/servers
curl -H "Authorization: Bearer $ADMIN_TOKEN" "http://localhost:8080/admin/mcp/tools?mcp_only=1"
curl -H "Authorization: Bearer $ADMIN_TOKEN" "http://localhost:8080/admin/mcp/requests?server=github&tool=create_issue&errors=
curl -H "Authorization: Bearer $ADMIN_TOKEN" "http://localhost:8080/admin/mcp/loops?window=24h&threshold=10"
curl -H "Authorization: Bearer $ADMIN_TOKEN" "http://localhost:8080/admin/mcp/catalog?server=github"
curl -H "Authorization: Bearer $ADMIN_TOKEN" http://localhost:8080/admin/mcp/policies
```

> 보안 참고: MCP 도구 결과(`role:tool`)도 프롬프트로 캡처되어 기존 `prompt.injection` 평가가 도구 응답을 통한 프롬프트 인젝션까지 스캔합니다. MCP 서버가 신뢰 경계 밖이라면 이 평가 실패를 모니터링하고, 위험 서버는 위 정책으로 차단하세요.

4-2. 에이전트 성능 분석 (Agent Performance)

어떤 코딩 에이전트가 가장 잘 동작하고, 가장 비싸고, 도구 오류가 많은지 비교하는 리더보드입니다. 요청 **User-Agent**로 에이전트(Claude Code/Cursor/Roo Code/Cline/Qwen Code/Continue/…)를 분류하고 chat 호출을 집계합니다.

열	의미
에이전트	User-Agent 키워드로 분류(미상은 UA 앞 토큰). 행에 마우스를 올리면 예시 원본 UA
요청 / 토큰	윈도우 내 chat 호출 수와 누적 토큰
성공률	2xx · 오류 없음 · 폴백 없음 비율. ≥90% 녹색 · ≥75% 노랑 · 그 외 빨강
폴백률	업스트림 폴백 발생 비율
평균/누적 비용	요청당 평균 KRW와 합계
평균 지연 / 첫 청크	전체 지연과 TTFB 평균
도구 오류율	tool 오류 / tool 호출 (오류율 ≥10% 빨강)

상단 KPI: 에이전트 수, 총 요청, **가중 성공률**(요청수 가중), 누적 비용. 윈도우(24h/7d/30d) 토글. API: `GET /admin/agents?window=7d` → `{agents[]}`.

> 에이전트가 정확히 분리되려면 클라이언트가 식별 가능한 User-Agent를 보내야 합니다. 같은 UA를 쓰는 도구는 한 버킷으로 합쳐집니다.

4-3. VCS 상관 (Prompt → Commit → MR → Merge)

프롬프트가 실제 코드/MR 로 이어졌는지 추적합니다. **GitLab·Bitbucket(Server/Cloud)·범용** 수집을 지원하며 오프라인망에서 동작합니다(외부 의존성 없음).

활성화 (필수: `VCS_WEBHOOK_SECRET`)

이 환경변수를 설정해야 `/vcs/*` 수집 엔드포인트가 켜집니다(미설정 시 403).

소스	설정
GitLab	프로젝트 → 설정 → 웹훅. URL `http://<gateway>:8080/vcs/webhook/gitlab`, **Secret Token** = `VCS_WEBHOOK_SECRET`, 트리거: Push events, Merge request events
Bitbucket	웹훅 URL `http://<gateway>:8080/vcs/webhook/bitbucket?token=<VCS_WEBHOOK_SECRET>`. Server `pr:*` / `repo:refs_changed`, Cloud `pullrequest:*` / `repo:push`
범용 / CI · git 훅	`POST http://<gateway>:8080/vcs/events`, 헤더 `X-Vibe-VCS-Secret: <secret>`, 바디 `{provider,kind,repo,branch,sha,title,session_id?}` 또는 `{events:[...]}`

세션·사용자 연결

- 커밋 메시지 · MR 제목 · 브랜치에 **`Vibe-Session: <세션ID>`** (또는 `[vibe:<세션ID>]`) 마커가 있으면 그 세션에 연결됩니다. (개발자 commit template / `commit-msg` 훅으로 자동 삽입 권장)
- 연결된 세션의 **주 사용자(api_key)** 가 자동으로 함께 연결되어, 어느 개발자의 프롬프트가 이 커밋/MR 로 이어졌는지 추적됩니다.
- 범용 수집은 `session_id` 를 직접 지정할 수 있어, 마커 없이 CI 가 빌드 컨텍스트로 연결할 수 있습니다.

보기

두 곳에서 봅니다: **(1) `VCS` 탭** — 전체 커밋·MR 목록(저장소/세션/키/유형 필터, 세션·사용자 링크로 드릴다운). **(2) 세션 타임라인 모달** — 그 세션에 연결된 커밋/MR 표(유형 + MR 상태 배지, 제목 링크, 저장소·브랜치, 작성자, 시각). API: `GET /admin/vcs/events?session_id=&repo=&api_key_id=&kind=`.

> 현재 라우팅/표시는 마커로 연결된 이벤트 중심입니다. Bitbucket Server push 웹훅은 커밋 메시지를 포함하지 않으므로(레퍼런스 변경만), 그 경우 마커 연결은 MR 제목 또는 범용 수집(git 훅)으로 보완하세요.

---

5. 호출 이력 / 프롬프트 검색

호출 이력 탭

- IP / 모델 / 언어 입력은 datalist 자동완성이 켜져 있어 운영 중인 값 중 골라 선택 가능.
- 행 좌측 체크박스로 두 행을 선택하면 상단 `[두 요청 비교]` 가 활성화 → 모달에 좌우로 펼쳐 프롬프트·토큰·비용·상태를 한눈에 비교.
- 행 본문 클릭 시 단건 상세 모달. 호출 이력 표와 상세 모달에는 첫 청크 지연과 전체 지연이 함께 표시됩니다.

단건 상세 모달

영역	내용
메타	request_id, trace_id, 생성 시각(상대 + 절대), 상태, 첫 청크/전체 지연, 모델, provider, stream, IP, X-Forwarded-For, User-Agent, API 키
언어 추론	코드블록·파일명·키워드 기반 추정. 신뢰도 % 함께
토큰 분해	prompt / completion / cached / reasoning / total
비용	KRW. 가격표가 설정된 모델에만
프롬프트	마스킹 처리된 본문. 원문이 저장된 경우(`LOG_RAW_PROMPTS=true`) 안내 메시지
응답 메타	finish_reason, 응답 hash, (옵션) 캡처된 응답 일부
태그 · 메모 · 재실행	태그 콤마 구분 + 메모 + 동일 요청 재실행 버튼

프롬프트 검색 탭

- 키워드는 마스킹 텍스트 / 원문 모두 검색.
- 키워드를 `#태그명` 으로 시작하면 태그 검색 모드.
- 결과 행은 호출 이력과 동일 포맷. CSV 다운로드(BOM, Excel 한국어 호환), 저장된 필터 드롭다운.
- "현재 필터 저장" — 이름을 입력하면 현재 검색 조건을 `saved_filters` 에 보관, 이후 드롭다운에서 즉시 다시 불러올 수 있습니다.

프롬프트 지문 (Prompt Fingerprint)

검색 폼 아래의 별도 카드입니다. 의미적으로 유사한 작업 프롬프트를 **어휘 지문**(`fp_…`)으로 묶어 "반복되는 작업 유형"을 드러냅니다. 지문은 ① 붙여넣은 코드(``` 블록·인라인 코드) 제거 → ② 핵심 키워드 추출(필러/일반 동사 stopword 제거, 한국어 조사·어미 정규화) → ③ 작업유형 + 상위 키워드를 해시. **의미 임베딩이 아니라 결정적 어휘 휴리스틱**이므로(문서에 명시), 같은 템플릿·반복 작업은 잘 묶지만 표현이 크게 다른 동일 의도는 못 묶을 수 있습니다.

열	의미
예시 프롬프트	그 클러스터의 대표 요청에서 가져온 마스킹 프롬프트(축약) + 지문 ID
유형 / 건수	작업유형, 윈도우 내 호출 수
성공률	2xx · 오류 없음 · 폴백 없음
평균/누적 비용 · 평균 토큰	클러스터 비용 프로파일
모델 수	이 작업에 쓰인 distinct 모델 수
최다/최저가 모델	최다 사용 모델, 그리고 **최고 성공률 대비 5%p 이내에서 가장 저렴한** 모델(비용 최적 후보)

윈도우(24h/7d/30d) 토글. API: `GET /admin/prompts/fingerprints?window=7d&limit=100` . 활용: "이 반복 작업은 비싼 모델을 쓰는데 최저가 모델도 성공률이 비슷하다 → 다운그레이드", 또는 Knowledge Cache 후보(반복 정형 프롬프트) 식별.

---

6. 사용자 / IP

사용자(Proxy API 키) 목록

- 키 이름·소유자·팀·상태·총 요청·총 토큰·누적 KRW·평균 지연·마지막 호출 (상대 시간).

- 헤더 클릭 정렬, 키 행 클릭 시 상세.

사용자 식별 상태 (active / external / anonymous)

게이트웨이는 키의 **해시만** 저장하므로, 들어온 Bearer 키는 등록 여부에 따라 다음으로 귀속됩니다.

상태	의미
`active`	등록된 proxy key(`PROXY_API_KEYS` 또는 "API 키 발급"). 정확한 사용자/팀/쿼터 통제 대상.
`external`	등록 안 된 키. **키 지문으로 사용자별 자동 분리**됩니다(ID는 발급 키와 동일하게 `key_<해시16>` , 차이는 **상태**뿐). 같은 키=같은 사용자. 클라이언트가 `X-Vibe-User` / `X-Vibe-Team` 헤더를 보내면 이름·팀이 채워집니다.
`anonymous`	키가 없고 등록 키도 없는 호출.

"사용자별 이력이 전부 `passthrough` / `anonymous` " 로 보인다면, 그 사용자들이 **등록되지 않은 동일 키 또는 키 없음**으로 호출하고 있다는 뜻입니다. 해결: ① 사용자별로 키를 **발급**(권장 — 쿼터·팀 강제 가능), 또는 ② 사용자별로 **다른 키**를 보내게 하면 상태 `external` 로 자동 분리됩니다(`ATTRIBUTE_EXTERNAL_KEYS=true` , 기본). `ATTRIBUTE_EXTERNAL_KEYS=false` 면 구버전처럼 모두 `passthrough` 단일 버킷으로 묶입니다.

> 식별자 prefix 안내: 모든 사용자 식별자는 이제 `key_<해시16>` 로 통일되고, **등록 여부는 prefix 가 아니라 상태(active/external)** 로 구분합니다. (구버전의 `ext_…` 식별자는 게이트웨이 시작 시 자동으로 `key_…` 로 이관되며 과거 이력도 함께 이동합니다.)

"등록한 키인데 ext_/passthrough 로 로깅되거나 사용자 상세가 0건이에요"

핵심 원칙: **클라이언트가 실제로 보내는 키 문자열이 곧 사용자 식별자**이고, 게이트웨이는 그 키의 **해시**로만 매칭합니다. 발급한 `key_xxxx` 의 지표가 0이라면, 거의 항상 **클라이언트가 그 키를 정확히 보내고 있지 않은 것**입니다(오타·공백·옛 키· `Bearer` 값 오기재, 또는 발급 화면에서 한 번 표시된 `pcg_…` 시크릿을 클라이언트에 넣지 않음).

바로 확인하는 법: `/v1` 응답 헤더 `X-Api-Key-Id` 에 게이트웨이가 인식한 식별자가 담깁니다.

```
curl -i http://<host>:8080/v1/chat/completions -H "Authorization: Bearer <발급키>" \
  -H "Content-Type: application/json" -d '{"model":"gpt-4.1-mini","messages":[{"role":"user","content":"hi"}]}' | grep -i x-api
```

여기에 발급한 `key_xxxx` 가 보이면 정상, `passthrough` /다른 `key_` 가 보이면 클라이언트가 다른 키를 보내는 것입니다.

확인·복구 절차:

1. **사용자 목록**에서 어디에 트래픽이 쌓였는지 확인 — 같은 클라이언트가 미등록 키를 쓰면 상태 `external` 항목으로 잡혀 있습니다.
2. 그 `external` 행의 **"관리 등록"** 버튼(또는 사용자 상세의 동일 버튼)으로 이름·팀을 부여하고 active 로 **승격**합니다. 게이트웨이가 이미 그 키의 해시를 저장해 두었으므로 **plaintext 없이** 승격되며, **클라이언트 재설정도 필요 없습니다** — 그 클라이언트가 계속 보내는 키가 이제 정식 사용자로 집계되고 과거 이력도 그대로 그 식별자에 남습니다.
3. 또는 클라이언트가 보내는 키를 **발급한 키와 글자 단위로 일치**시키세요(발급 시 표시된 `pcg_…` 시크릿을 그대로 사용).

> 승격은 `PATCH /admin/api-keys/{id}` 에 `{"status":"active","name":"…","team":"…"}` 를 보내는 것과 동일합니다. 이름/팀만 바꾸려면 status 를 생략하세요. 키 해시는 항상 보존됩니다.

> 주의: `passthrough` · `anonymous` 는 키 해시가 없는 합산 버킷이라 승격할 수 없습니다(과거 트래픽은 소급 분리 불가). 분리가 필요하면 지금부터 사용자별로 다른 키를 쓰게 하세요.

사용자 상세

- 키 메타 (id/소유자/팀/상태)
- 고급 지표: 최근 24h 요청·토큰·KRW, 오류율, 전체/첫 청크 P95 지연, 평균 첫 청크, 토큰 분해 (prompt/completion/cached/reasoning), 고유 모델/IP 수
- 일별 사용량 표 (최근 60일)
- 모델별 / IP별 / 언어별 표
- 상태 분포와 Asia/Seoul 기준 최근 30일 시간대 히트맵
- 최근 호출 100건 (필터/상세 모달 사용 가능)

IP 목록 / 상세

같은 구조이며, IP 별 상세에는 "API 키별" 표가 함께 표시됩니다 — 한 공용 IP 에서 어떤 키들이 호출했는지 확인할 때 사용.

팀 벤치마크 / AI 활용지수 / AI Incident

화면	내용	API
팀 탭 상단 "팀 벤치마크"	팀별 활성 인원·요청·**월비용(30d)**·성공률·커밋·머지 MR·**생산성 점수** (멤버 점수의 요청 가중 평균)	`GET /admin/benchmark/teams?window=30d`
사용자 탭 하단 "AI 활용지수"	사용자별 Prompt 수·세션·활동일·커밋·머지 MR·도구 호출·성공률·비용·**활용지수(0~100)**	`GET /admin/benchmark/users?window=30d&limit=100`
안전 탭 "AI Incident"	프로바이더별 **폴백/5xx 급증(시간당 ≥ 5건)** 을 장애로 추정, 연속 시간대 병합, 폴백·5xx·**영향 사용자 수**·진행 중 여부	`GET /admin/incidents?window=7d&min_events=5`

12. 거버넌스 · 운영 신규 도구 (v0.64~v0.74)

아래는 코드 검증·세션 관측·정책 롤아웃·자산 거버넌스 관련으로 추가된 관리자 화면/엔드포인트입니다. 모두 RBAC(`admin:read` 또는 `security:read`)이며 원문 prompt/코드는 저장·노출하지 않고 메타데이터만 다룹니다.

AI 코드 출력 검증 게이트

- 모델 응답의 코드블록을 언어별 정적 점검(위험 API·파괴적 명령·하드코딩 시크릿·구문 균형)해 위험도/테스트 가능성을 산출. 폐쇄망·외부 컴파일러 미사용.
- `POST /admin/code-verify {text}` — 임의 텍스트 즉시 검증(무상태). `GET /admin/code-verify/stats?days=` — 영속된 verdict의 모델별 위험 리더보드.
- Chat 테스트 멀티런에 "코드 검증" 버튼, 단일 Chat 응답 "실행 요약"에 코드 검증 섹션, 자동 평가(rubric) safety 점수에 반영. /v1 응답 verdict는 응답 텍스트 캡처(`LOG_RESPONSE_TEXT` /캐시) 시 영속되어 요청 상세·트레이스에 노출.

Agent Session Flight Recorder (세션 비행기록)

- "세션 비행기록" 탭: 최근 코딩 세션(클라이언트 `session_id`) 목록 → 세션별 시간순 타임라인(요청·종류·모델·지연·토큰·비용·도구) + 위험 오버레이(시크릿/정책 차단/위험 코드) + 규칙 기반 RCA 요약(정상/주의/위험).
- `GET /admin/sessions?days=` , `GET /admin/sessions/{session_id}/flight-recorder` . 요청 상세 모달의 Session 행 "비행기록" 링크에서도 진입.

Policy Canary & Shadow Enforce

- 정책 어드바이저의 추천 카드 "섀도우 영향" 버튼: 최근 트래픽에 시뮬레이션해 차단 예상·영향 사용자/팀·**오탐 후보(과거 2xx였으나 차단될 요청)**·차단 비용(절감 추정)을 표시(`POST /admin/policies/simulate` 의 `shadow` 블록).

- 정책에 `rollout_percent` (canary): 1~99면 결정적 트래픽 슬라이스에만 enforce, 슬라이스 밖은 `canary_shadow` 결정으로 기록만. `GET /admin/policies/canary-status` + 어드바이저 "Canary 롤아웃 현황" 카드에서 실집행 vs 섀도우 비교 후 "N%로 상향".

AI 자산 SBOM

- "AI 자산 SBOM" 탭 / `GET /admin/sbom?type=` : 스킬·워크플로·앱·모델계약·프롬프트 자산의 소유권·의존성·상태를 통합 명세로 보여주고 거버넌스 공백("owner 없음", high 위험 production, 적합성 검증 미흡)을 표시. JSON 내보내기.

Journey Probe (개발도구 연결 합성 점검)

- "Journey Probe" 탭 / `POST /admin/journey-probe {proxy_key, clients?}` : Cursor·Roo·Cline·OpenAI SDK 등 도구별 실제 연결 journey(모델 목록·MCP initialize/tools-list)를 supplied Proxy API Key로 합성 점검(비용 발생 chat 호출 없음). "서버는 살아있는데 Cursor만 안 됨"을 분리.

변경 후 자동 Red Team 회귀 점검

- Provider, 라우팅 규칙, MCP upstream·도구 정책, Governance 정책, Text2SQL 스키마·권한, AI App·Workflow가 변경되면 관련 등록 대상만 골라 Red Team 캠페인을 자동 생성합니다.
- 자동 캠페인은 항상 `dry-run` 시뮬레이션으로 실행되며 provider나 MCP upstream을 실제 호출하지 않습니다.
- Red Team 화면의 **변경 후 자동 점검** 상태에서 활성 여부·중복 억제 시간·변경당 최대 대상 수를 확인하고, 캠페인 **생성 경로**에서 원인이 된 audit action과 대상 참조를 확인합니다.
- 동일한 변경 상태는 기본 10분 동안 한 번만 실행됩니다. 전체 범위 변경은 대상 유형을 순환 선택해 기본 20개 대상까지만 검사합니다.
- Red Team Kill Switch가 켜져 있으면 자동 캠페인도 생성·실행하지 않습니다. `REDTEAM_POST_CHANGE_ENABLED` , `REDTEAM_POST_CHANGE_COOLDOWN` , `REDTEAM_POST_CHANGE_MAX_TARGETS` 로 운영값을 조정합니다.

파드 운영 맵 / 프라이버시 원장 / AI 업무성과 / 온보딩 점검

- "파드 운영 맵" 탭 / `GET /admin/pods` : 멀티 파드 하트비트·빌드·런타임 설정 수렴(applied vs current token) 상태. live/stale·설정 최신 여부.
- "프라이버시 원장" 탭(security) / `GET /admin/privacy-ledger?dimension=team|model|provider&days=` : 민감정보 탐지/마스킹/차단량 + 외부 provider 전송 요청·토큰을 차원별 감사 원장으로 집계. `?format=csv` 내보내기.
- "AI 업무성과" 탭 / `GET /admin/productivity?days=` : `X-Vibe-Repo` 귀속 AI 사용량과 VCS commit/merge_request를 repo별로 상관, 머지당 비용 산출.
- 온보딩 준비 점검: `POST /admin/apps/onboarding-check` (AI 앱), `POST /admin/mcp/onboarding-check` (MCP 업스트림) — 등록 전 owner·접근범위·문서·(고위험 MCP는)승인 게이트 등 required/recommended 체크리스트로 SBOM 공백을 생성 시점에 예방.

개발자 오버레이

- `GET /me/overlay` (사용자): 현재 모델·예산 사용/한도/남음·액션 항목·추천을 한 번에 폴링하는 컴팩트 상태(IDE 확장/패널용). CLI `vibe status [--table]` 로도 확인.

활용지수 공식(관측 기반 휴리스틱, **인사평가 지표 아님** — 도입 현황 파악용): `요청량 30% + 활동일수 20% + 커밋 20% + 머지 MR 15% + 성공률 15%` (포화 상한: 요청 300, 활동일 20, 커밋 30, MR 10 / 30일 기준). 커밋·MR 은 VCS 상관(웹훅 또는 추론)으로 사용자에 연결된 것만 집계됩니다 — VCS 연동이 없으면 해당 컬럼은 0으로 나오고 나머지 요소만으로 점수가 계산됩니다.

인덱스 상태 (드리프트 · 추가/삭제 후보)

운영 홈 하단의 **인덱스 상태** 카드 / `GET /admin/index-health` . 읽기 전용이며, 어떤 DDL도 실행하지 않습니다 — 각 항목은 운영자가 검토할 SQL만 보여줍니다.

두 가지를 나눠서 답합니다.

1) 선언한 스키마와 실제 DB의 차이. 마이그레이션이 만드는 인덱스 목록을 그대로 읽어 실제 DB와 대조합니다. 별도 목록을 관리하지 않으므로 어긋날 곳이 없습니다.

구분	뜻	왜 위험한가
정의 불일치	이름은 같은데 컬럼·유니크 여부가 다름	`CREATE INDEX IF NOT EXISTS` 는 **이름만** 봅니다. 예전 정의가 남아 있으면 마이그레이션은 성공했다고 보고하면서 인덱스는 잘못된 채로 남습니다. 이것만은 다른 어떤 것도 잡아주지 않습니다
DB에 없음	선언했는데 실제로는 없음	아무것도 실패하지 않고 쿼리만 느려집니다
선언에 없음	손으로 추가한 인덱스	지금 DB에서는 동작하지만, 새로 설치하는 환경과 다른 환경에는 없습니다

2) 접근 통계가 말해주는 추가·삭제 후보. Postgres는 테이블별 순차 스캔 수와 인덱스 스캔 수, 인덱스별 사용 횟수를 셉니다. 스캔으로 읽히는 큰 테이블은 인덱스가 없다는 뜻이고, 한 번도 읽히지 않은 인덱스는 쓰기 비용만 내고 있다는 뜻입니다. 각 항목에는 판단 근거(행 수·스캔 횟수·디스크 크기)가 함께 표시되므로 그대로 받아들이지 말고 따져볼 수 있습니다.

SQLite에는 이런 카운터가 없습니다. 그래서 숫자를 지어내는 대신 "이 드라이버로는 볼 수 없다"고 표시하고, 스키마만으로 알 수 있는 것(기본키 외에 인덱스가 하나도 없는 테이블)만 보고합니다. **목록이 비어 있는 것과 볼 수 없는 것은 다릅니다.**

빌드 쪽에도 같은 질문이 걸려 있습니다. `internal/store` 의 테스트가 저장소의 SQL을 직접 읽어, 트래픽에 따라 무한히 커지는 테이블(`request_id` 를 가진 테이블과 만료로 청소되는 테이블)에서 필터·정렬에 쓰이는 컬럼에 인덱스가 있는지 확인합니다. 없으면 `migrationStatements()` 에 인덱스를 추가하거나 `indexCoverageDecisions` 에 "왜 스캔해도 괜찮은지"를 적어야 빌드가 통과합니다.

이 검사가 처음 돌았을 때 찾은 것: 보존 정책이 `WHERE request_id IN (...)` 으로 지우는 요청 단위 자식 테이블 11개 중 9개에는 `request_id` 인덱스가 있고 3개(`response_logs`, `language_stats`, `domain_routing_decisions`)에는 없었습니다. 같은 쿼리, 같은 증가 속도인데 인덱스만 빠져 있어 청소할 때마다 전체 스캔이 일어났습니다. `auth_sessions` 는 기본키 외에 인덱스가 아예 없었습니다. 지금은 모두 추가돼 있습니다.

마이그레이션 SQL 전문 보기

인덱스 상태 카드의 📜 **마이그레이션 SQL 전체 보기** 버튼 / `GET /admin/migration-sql` .

드리프트 표는 **어긋난 것만** 보여줍니다. 그런데 손으로 인덱스를 여러 개 추가한 뒤에 실제로 하게 되는 질문은 "이건 내가 만든 건가, 빌드가 만든 건가"이고, 그 답이 필요한 인덱스는 대부분 **어긋나지 않은 쪽**입니다. 드리프트 표에는 나오지 않으니 소스를 열어보는 수밖에 없었습니다.

이 모달은 `Migrate` 가 적용하는 문장 **전부**를 적용 순서대로 보여주고, 선언된 인덱스마다 이 DB 의 상태를 함께 표시합니다.

표시	뜻
`DB에 있음`	빌드가 선언했고 이 DB 에도 그대로 있습니다
`DB에 없음`	빌드가 선언했는데 이 DB 에는 없습니다 (게이트웨이를 재시작하면 다시 만들어집니다)
`정의 불일치`	이름은 같은데 정의가 다릅니다 — 재시작해도 고쳐지지 않습니다
상단 노란 블록	**이 DB 에만 있는 인덱스.** 목록에 없으므로 손으로 추가한 것이고, 새로 설치한 DB 에는 생기지 않습니다

- 번호는 적용 순서입니다. `ALTER TABLE` 은 해당 `CREATE TABLE` 뒤에 옵니다.
- 검색창과 구분 필터(테이블 / 인덱스 / 컬럼 추가 / 기타), **불일치만** 체크박스로 좁혀 볼 수 있습니다.
- **PostgreSQL 에서는 실제로 실행된 문장**을 보여줍니다. 저장소에는 `BLOB` · `REAL` 로 적혀 있지만 Postgres 에는 `BYTEA` · `DOUBLE PRECISION` 으로 나갑니다. 바뀐 문장은 "선언 원문"을 펼쳐 원래 형태를 확인할 수 있습니다. 이걸 구분하지 않으면 드라이버 차이를 드리프트로 오해하게 됩니다.

- 목록 실행 후 Postgres 에서 추가로 일어나는 컬럼 타입 확장(`REAL` → `DOUBLE PRECISION` , 카운터→ `BIGINT`)은 목록에 없으므로 하단에 따로 적어둡니다.
- **전체 복사** 버튼은 전체 문장을 세미콜론으로 구분해 클립보드에 넣습니다.

읽기 전용입니다. 이 화면은 SQL 을 보여줄 뿐 실행하지 않습니다.

7. 사용 한도 (쿼터)

진행 중 요청도 한도에 포함됩니다

사용량은 요청이 **끝난 뒤에** 기록됩니다. 따라서 완료된 사용량만 본다면 한도 검사의 사각지대는 로깅 지연 정도가 아니라 **진행 중인 모든 요청의 전체 지속시간**입니다 — LLM 호출이면 수십 초에서 수 분입니다. 그 사이 들어온 요청들은 서로를 전혀 보지 못하므로, 바쁜 키는 첫 요청이 끝나기 전에 시작할 수 있는 만큼 한도를 **초과**할 수 있었습니다.

이를 막기 위해 요청이 시작될 때 **예상 사용량을 예약**해 두고, 끝나면 해제합니다. 한도 검사는 `완료된 사용량 + 진행 중 예약` 을 합산하므로 동시 요청이 서로를 셉니다.

환경변수	기본값	설명
`QUOTA_RESERVATIONS_ENABLED`	`true`	진행 중 요청을 한도에 포함. 요청당 INSERT·DELETE 1회씩 추가되므로 쿼터를 안 쓰면 끄면 됩니다
`QUOTA_RESERVATION_SWEEP_INTERVAL`	`5m`	만료된 예약 정리 주기

- `/v1/chat/completions` 와 `/v1/embeddings` **모두 예약합니다.** 임베딩은 배치 작업이 수천 건을 동시에 던지는 워크로드라 한도를 넘기기 가장 쉬운 형태입니다. 다만 임베딩은 완성(completion)이 없으므로 **입력 토큰만** 예약합니다 — 채팅과 같은 출력 예측을 적용하면 매 호출을 부풀려 정상 트래픽을 막게 됩니다.
- **예약은 자동 만료됩니다.** 게이트웨이가 요청 도중 죽어도 한도를 영구히 점유하지 못합니다. 조회 자체가 만료 행을 제외하므로 정리 작업이 밀려도 과다 집계되지 않습니다.
- **예약은 다중 인스턴스에서 공유됩니다**(DB 기반). 인스턴스가 여러 대여도 서로의 진행 중 요청을 봅니다.
- **예약 조회가 실패하면 완료된 사용량만으로 판정**합니다. 예약은 정확도를 높이는 장치일 뿐 통과·차단의 권한이 아니므로, 읽지 못한다고 요청을 막지 않습니다.
- 팀 범위 판정은 완료된 사용량과 **동일한 식**을 사용합니다. 두 값을 더하는 이상 "팀"의 정의가 갈리면 한쪽이 조용히 잘못 세어집니다.

> 이 기능을 켜면 이전보다 **더 일찍 429가 발생할 수 있습니다.** 그것이 의도입니다 — 한도가 실제로 한도로 동작합니다.

왜 막혔는지 확인하기

한도 판정은 `완료된 사용량 + 진행 중 예약` 으로 하므로, **완료된 사용량만 보면 "한도 아래인데 왜 429?"** 라는 상황이 생깁니다. 그래서 양쪽 모두 내역을 함께 보여줍니다.

429 응답 헤더

헤더	의미
`X-Quota-Scope`	어떤 한도에 걸렸는지 (`범위:값:주기`)
`X-Quota-Tokens` · `X-Quota-Cost-KRW`	판정에 사용된 **합계**(완료 + 진행 중)
`X-Quota-Reserved-Tokens` · `X-Quota-Reserved-Cost-KRW`	그중 **진행 중 요청** 몫
`X-Quota-Token-Limit` · `X-Quota-Cost-Limit-KRW`	걸린 한도값
`Retry-After`	다음 주기까지 남은 초

합계에서 예약분을 빼면 완료된 사용량이 나오므로, 호출자가 자기 대시보드 숫자와 대조해 차이를 설명할 수 있습니다.

관리자 화면 — 사용 한도 탭의 토큰·비용 열은 **판정과 동일한 합계**를 표시하고, 진행 중 요청이 있으면 그 몫을 아래에 따로 적습니다(`진행 중 N 토큰 · ₩M 포함`). 화면이 여유 있다고 보이는데 실제로는 429가 나가는 불일치가 생기지 않습니다.

폭주를 방지하고 부서별 예산을 강제하는 핵심 도구.

추가 폼 (왼쪽부터)

필드	값
대상	API 키 / 팀 / IP / 전체
대상 값	"전체" 는 자동, 그 외에는 키 ID / 팀 이름 / IP
주기	일별(매일 KST 00:00 리셋) / 월별(매월 1일 KST 00:00 리셋)
토큰 한도	0 이면 미적용
KRW 한도	0 이면 미적용
메모	운영자가 참고할 자유 텍스트

토큰·KRW 둘 다 채우면 둘 중 먼저 도달한 쪽에서 차단됩니다. 둘 다 0 이면 저장이 거절됩니다.

사용 한도 표

각 행마다 토큰 진행률 / KRW 진행률 막대가 함께 표시되고 80% 이상은 노란색, 100% 이상은 빨간색입니다. 같은 행에서 "중지"(잠깐 끄기), "삭제"(완전 제거) 가능합니다.

평가 흐름

요청이 들어올 때 게이트웨이는 다음 순서로 매칭되는 쿼터를 검사합니다.

1. global / `*`
2. api_key / 현재 키 ID
3. ip / 현재 클라이언트 IP
4. team / 키 소유 팀 (있을 때)

하나라도 초과되면 HTTP 429 + `Retry-After` + `X-Quota-*` 헤더 + 본문에 어떤 한도가 초과되었는지 표기됩니다.

월 예산 소진 예측 (Budget Burn-down)

사용 한도 탭 하단의 별도 섹션입니다. 쿼터가 "도달하면 차단"하는 **경성(hard) 한도**라면, 예산은 "이 추세면 월말에 얼마 쓸지"를 **예측·경고**하는 연성(soft) 관측 도구입니다. 차단은 하지 않습니다.

추가 폼:

필드	값
대상	전체 / 팀 / API 키
대상 값	"전체"는 자동, 그 외에는 팀 이름 / 키 ID
월 예산(KRW)	이번 달 목표 상한 (양수)
메모	자유 텍스트

표의 각 열:

- **이번 달 누적 / 월 예산** — 월초(KST 1일 00:00)부터 현재까지 실제 지출과 진행률 막대. 경과 일수(예: `경과 15/30일`)도 함께 표시.
- **월말 예상 지출** — 현재 일평균 소진율(누적 ÷ 경과일)을 월말까지 연장한 예측값과 예산 대비 %. 120% 이상이면 빨간색, 100% 초과면 노란색.
- **소진 예측** — `정상 추세` (예측이 예산 이하) 또는 `예산 초과 추세` 배지. 추세대로면 예산을 다 쓰는 **소진 예상일**(KST 날짜)을 함께 표시(이번 달 안에 소진될 때만).

기준 시간대는 KST(매월 1일~말일)이며 쿼터의 월별 리셋과 동일합니다.

예산 임박 알림 연동

안전 탭의 알림 규칙에서 지표 **"예산 소진 예측 비율(최대)"**(`budget_burn_ratio`)을 선택하면, 등록된 모든 예산 중 가장 높은 *월말 예상 / 월 예산* 비율이 임계치를 넘을 때 Webhook으로 통지됩니다. 예: 임계치 `1.0` → 어떤 예산이든 현재 추세로 월말에 예산을 초과할 것으로 예측되면 발화. `1.2`로 두면 20% 초과 추세부터 알립니다.

---

8. 안전 (Kill Switch + 알림)

Kill Switch

⚠️ "모든 /v1 호출 즉시 차단" 버튼. 누른 즉시 모든 /v1/* 호출이 HTTP 503 + `Retry-After: 60` + `X-Kill-Switch: global` + `X-Kill-Reason: <사유>` 헤더로 응답합니다. 5초 캐시를 사용하므로 멀티 인스턴스 운영에서도 약 5초 안에 모든 인스턴스에 전파됩니다.

복귀는 같은 화면의 "정상 운영 재개" 버튼.

언제 사용하나요?

- 한 도구가 비용을 폭주시키는 게 확실하지만 어느 키인지 모를 때
- 릴리즈 롤백 / 보안 사고 / vendor 측 대량 장애
- 짧은 시간 안에 다시 켤 예정일 때 (몇 시간 차단은 쿼터/키 비활성화로 대체)

비용 가드 / 예측 (Cost Guard)

호출을 업스트림에 보내기 **전에** 입력/출력 토큰·KRW 비용·지연을 예측하고, 예상 비용이 임계값을 넘으면 차단합니다(쿼터가 *누적 사용량*을 막는다면, 비용 가드는 *단일 호출*의 예상 비용을 막습니다).

- **가드 사용 + 임계값(KRW)**: 켜면 예상 비용 〉 임계값인 chat 호출을 `HTTP 402` + `X-Cost-Guard: blocked` 로 차단. 클라이언트가 `X-Cost-Approve: 1` 헤더를 보내면 승인되어 통과합니다(대형 작업 의도적 실행). 메트릭 `proxy_cost_guard_blocked_total`.
- **응답 헤더**: 모든 chat 응답에 `X-Estimated-Input-Tokens / X-Estimated-Output-Tokens / X-Estimated-Cost-KRW / X-Estimated-Latency-MS`.
- **예측 근거**: 출력 토큰은 모델별 최근 7일 평균(표본 ≥5), 없으면 요청 `max_tokens`, 그것도 없으면 기본 600. 비용은 모델 가격표 기준(가격 미설정 모델은 차단하지 않음).

- **비용 예측기(dry-run)**: 같은 카드에서 모델·입력 토큰·max_tokens 를 넣어 즉시 예상 비용을 확인. API `POST /admin/cost/predict` . 가드 설정 API `GET|POST /admin/cost {enabled, threshold_krw}` .

알림 규칙

지표	의미	예시 임계값
`requests`	윈도우 안 요청 수	5분에 500건
`errors`	윈도우 안 4xx/5xx 비율 (0~1)	0.10 (10%)
`krw`	윈도우 안 KRW 비용 합	100000 (10만원)
`tokens`	윈도우 안 토큰 합	1000000
`latency_p95_ms`	윈도우 안 전체 응답 지연 P95(ms)	3000
`first_chunk_p95_ms`	윈도우 안 upstream 첫 응답 청크 지연 P95(ms)	1500
`llm_eval_failures`	윈도우 안 실패한 LLM evaluation 수	10
`llm_eval_failure_rate`	윈도우 안 LLM evaluation 실패율 (0~1)	0.2
`failovers`	윈도우 안 폴백으로 처리된 요청 수	10
`failover_rate`	윈도우 안 폴백 발생률 (0~1)	0.05

- **윈도우(초)**: 평가 기간. 알림 평가는 1분 주기로 돌고, 발화 후에는 같은 윈도우 동안 디바운스 됩니다.
- **대상**: 전체 / API 키 / 팀 / IP / 모델 중 선택.
- **webhook URL**: Slack 호환(`text` 필드 + 컨텍스트 JSON). 비워두면 발화 이력에만 기록.

발화 이력

같은 탭 하단에서 최근 50개. 시각 / 규칙 / 지표 / 값 / 임계값 / 전송 성공 여부를 표시합니다.

---

9. 설정

9.1 프록시 API 키 발급

폼: 이름 / 소유자(이메일·이름) / 팀 / 시크릿(선택). 시크릿을 비우면 게이트웨이가 `pcg_...` 형식으로 자동 생성합니다.

발급 직후 한 번만 표시되는 시크릿을 사용자에게 안전한 채널(사내 메신저 1:1, 1Password 등)로 전달하세요. 다시 볼 수 없습니다.

이름 클릭 시 사용자 상세로, "비활성화" 버튼으로 즉시 차단할 수 있습니다.

9.2 업스트림 프로바이더

vendor API 본인의 키를 게이트웨이에 저장하는 화면입니다. 평문이 아닌 AES-GCM 으로 암호화되어 보관됩니다(키는 `GATEWAY_SECRET`).

`모델 패턴` 컬럼에 콤마 구분 글롭(`claude-*` , `anthropic/*` 등)을 넣으면, 클라이언트가 `X-Proxy-Provider` 를 지정하지 않아도 모델명만으로 라우팅됩니다.

9.2.1 복잡도 기반 비용 최적 라우팅 규칙

요청 복잡도(0~100)에 따라 모델을 자동 교체합니다. 예: 저복잡도(0~34)는 저가 모델로 다운그레이드, 고복잡도(70~100)는 프리미엄으로.

필드	의미
우선순위	낮을수록 먼저, 첫 매칭 적용
모델 패턴	들어온 모델에 매칭할 glob (`*` =전체)
복잡도 범위	min~~max (0~~100)
대상 모델	교체할 모델 (body의 `model` 재작성)
대상 provider	선택. 비우면 자동 결정

- 클라이언트가 provider를 지정했거나 `X-Proxy-No-Route: 1` 헤더면 규칙 미적용.
- 교체 시 `X-Routed-Model` 헤더 + XView 설명에 "원본 → 대상" 표기. 메트릭 `proxy_routing_overrides_total` .

```
curl -X POST http://<host>:8080/admin/routing-rules \
  -H "Content-Type: application/json" \
  -d '{ "match_pattern":"gpt-*", "min_complexity":0, "max_complexity":34, "target_model":"gpt-4.1-mini", "priority":10 }'
```

9.2.2 라우팅 학습 추천 (Routing Learning Engine)

위 규칙이 **사람이 정한 고정 규칙**이라면, 학습 추천은 **실측 결과로 최적 모델을 제안**하는 학습 계층입니다. 게이트웨이는 모든 chat 호출에 **작업유형**(프롬프트 키워드 추정: 리팩토링/생성/디버그/설명/테스트/변환/문서/리뷰)과 **복잡도 버킷**(낮음 0 - 33 / 중간 34 - 66 / 높음 67 - 100)을 기록하고, 모델별 성공률·비용·지연·피드백을 누적합니다.

설정 탭의 **"라우팅 학습 추천"** 표:

열	의미
작업유형 / 복잡도	학습 셀 키
추천 모델	표본이 충분한(기본 `min_samples=20`) 모델 중 **성공률 최고(동률 시 저비용)**. `저신뢰` 배지 = 비교 대상 중 표본 부족 모델 존재
성공률 / 평균 비용 / 표본	추천 모델의 실측치. 성공 = 2xx · 오류 없음 · 폴백 없음
현재 최다 사용	그 셀에서 실제로 가장 많이 쓰인 모델과 성공률(추천과 다르면 노란 배지)
동작	**"규칙으로 적용"** → 해당 복잡도 구간을 추천 모델로 바꾸는 라우팅 규칙 생성

"상세 매트릭스"를 펼치면 (작업유형 × 복잡도 × 모델) 셀별 요청 수·성공률·폴백률·비용·지연·피드백을 모두 볼 수 있습니다.

- API: `GET /admin/routing/learning?window=7d&min_samples=20` → `{cells[], recommendations[]}` .
- **human-in-the-loop**: 추천은 자동 적용되지 않습니다. 운영자가 "규칙으로 적용"을 눌러야 9.2.1 규칙으로 반영됩니다. 적용 규칙은 **복잡도 구간 단위**로 동작하므로(작업유형은 참고용), 같은 구간에서 작업유형별 추천이 갈리면 운영자가 판단해 적용하세요.
- 작업유형·복잡도는 프롬프트 기반 휴리스틱 추정치입니다(모델 산출값 아님).

9.2.3 Knowledge Cache (반복 규칙 중앙 등록)

매 호출에 반복 전송되는 사내 코딩 규칙·시스템 프롬프트를 한 번 등록해 두고, 클라이언트가 짧은 참조만 보내면 게이트웨이가 업스트림 전송 시 전체 텍스트로 확장합니다.

- 등록 폼: 이름, ID(slug, 비우면 이름에서 자동 생성), 본문. 토큰 추정치는 자동 계산. 표에서 사용 횟수·최근 사용·참조 문자열 (`{{kb:ID}}`)·사용/중지·삭제.

- 클라이언트 참조 방법(둘 중 하나):
  - 메시지 본문 플레이스홀더 `{{kb:ID}}`
  - 헤더 `X-Vibe-Knowledge: ID1,ID2` → 시스템 메시지로 맨 앞에 주입
- 확장된 응답에는 `X-Knowledge-Expanded: <id,...>` 헤더. 메트릭 `proxy_knowledge_expansions_total`, `proxy_knowledge_tokens_total`. 5초 캐시로 변경이 약 5초 내 전파.
- **감사 로그에는 확장 전 짧은 참조가 그대로 보존**되고(저장 절감), 모델에는 전체 본문이 전달됩니다. 프롬프트 지문에서 발견한 반복 정형 프롬프트를 여기에 등록하는 워크플로가 자연스럽습니다.

효과	설명
거버넌스	규칙을 한 곳에서 고치면 모든 호출에 즉시 반영 (클라이언트 수정 불필요)
페이로드·저장	클라이언트→게이트웨이 본문과 프롬프트 로그가 짧아짐
업스트림 비용	provider 프리픽스 캐싱(cached 토큰)과 결합될 때 절감 — 게이트웨이가 안정적 프리픽스로 주입

API: `GET /admin/knowledge`, `POST /admin/knowledge` (`{name, id?, content, enabled?}`), `PATCH|DELETE /admin/knowledge/{id}`.

9.3 데이터 보존 정책

현재 적용 중인 보존 일수와 누적 삭제 행 수를 표시합니다. "지금 정리 실행" 으로 워커를 즉시 1회 트리거할 수 있습니다(디스크가 가득 찼을 때 임시 조치).

무엇이 함께 삭제되는가

`RETENTION_REQUEST_DAYS` 로 요청 로그를 삭제할 때, **그 요청에 딸린 기록도 함께** 삭제됩니다.

함께 삭제	이유
`prompt_logs` · `response_logs` · `token_usage` · `language_stats`	요청 본문·응답·사용량
`llm_evaluations` · `llm_feedback` · `tool_invocations`	요청 단위 평가·도구 호출
`routing_decisions` · `mcp_route_decisions` · `domain_routing_decisions` · `code_verify_results`	요청 단위 라우팅·검증 텔레메트리

이 표의 **세 번째 줄은 이전까지 삭제되지 않았습니다.** 각 행은 `request_id` 옆에 API 키·사용자·팀을 함께 담고 있어, 요청이 보존 기간 만료로 지워진 뒤에도 **"어떤 키가 그 요청을 했다"는 기록이 남았습니다.** 특히 `routing_decisions` 는 그 프롬프트에서 탐지된 **PII·시크릿 분류 (`risk_categories`)** 까지 보관하고 있었습니다. 부수적으로, 요청당 1행씩 영구히 쌓이는 동안 `request_logs` 만 줄어들어 오래 운영한 배포에서는 이 테이블들이 가장 커집니다.

API 명세는 어떻게 유지되는가 (개발자용)

`/openapi.json` 과 `/swagger` 가 노출하는 명세는 **손으로 관리하는 표**이고, 라우트는 별도로 등록됩니다. 둘을 연결하는 장치가 없어서 **서비스되지만 명세에 없는 엔드포인트가 생길 수 있었고**, 실제로 두 개가 그 상태였습니다.

`internal/proxy/openapi_completeness_test.go` 가 양방향으로 검사합니다.

- **누락**: `/admin` 라우트를 등록하고 명세 항목을 빼면 실패합니다. 공개 API가 아니라면 사유와 함께 예외 목록에 등록해야 합니다.
- **잔존**: 명세에는 있는데 아무도 서비스하지 않으면 실패합니다 — **문서화된 엔드포인트가 404를 내는 것**이 문서에 없는 것보다 나쁘기 때문입니다.

보존 정책은 어떻게 유지되는가 (개발자용)

요청 로그와 함께 삭제할 테이블 목록은 코드 안의 고정 목록입니다. 그래서 나중에 추가된 테이블은 **추가한 사람이 기억해야** 했고, 실제로 요청 단위 테이블 4개와 만료 컬럼을 가진 테이블 3개가 오랫동안 누락됐습니다.

`internal/store/retention_completeness_test.go` 가 이를 **빌드 단계에서 강제**합니다.

- `request_id` 나 `expires_at` 을 가진 테이블은 **반드시 판단이 등록**돼야 합니다 — 요청과 함께 삭제 / 만료로 정리 / 의도적 보존(사유 포함) 중 하나.
- 판단 없이 그런 테이블을 추가하면 테스트가 **어느 테이블인지와 무엇을 결정해야 하는지** 알려주며 실패합니다.
- "삭제한다"고 등록해놓고 실제 삭제문이 없으면 그것도 실패합니다 — 등록은 의도이고, 테스트가 현실과 대조합니다.

즉 보존 판단이 코드 리뷰에서 눈에 띄지 않고 지나가는 대신, 테스트가 결정을 요구합니다.

만료된 행 정리

만료 시각(`expires_at`)을 가진 일부 테이블은 **읽을 때는 만료로 처리되지만 행이 삭제되지 않아** 계속 쌓이고 있었습니다. 이제 보존 워커가 함께 정리합니다.

테이블	내용
`refresh_tokens`	만료된 토큰 + 폐기 후 24시간 지난 토큰
`auth_sessions`	만료된 세션 + 폐기 후 24시간 지난 세션
`text2sql_cache`	만료된 캐시 항목

`refresh_tokens` 가 특히 빨리 쌓입니다 — 토큰은 **로그인당 1행이 아니라 갱신(rotation)마다 1행**이 생기고, 직전 토큰은 즉시 폐기되지만 삭제되지는 않았습니다.

폐기 직후 24시간은 남깁니다. 재시도가 폐기된 토큰을 다시 제시했을 때 "이미 사용됨"으로 인식돼야지, 행이 사라져 "알 수 없는 토큰"으로 보이면 안 되기 때문입니다.

> `login_attempts` 는 앱이 읽지 않는 **보안 감사 로그**라 `secret_events` 와 같은 원칙으로 남깁니다.

삭제하지 않는 것

다음은 요청에 연결돼 있어도 **의도적으로 남깁니다** — 요청 텔레메트리가 아니라 운영자가 만든 기록이거나, 자체 수명 주기를 갖습니다.

남기는 것	이유
`request_notes` · `approvals`	운영자가 직접 남긴 메모·승인
`secret_events`	보안 감사 추적 — 보통 더 긴 보존이 요구됩니다
`policy_decision_events`	거버넌스 판단 이력. 감사 목적이 있을 수 있어 임의로 지우지 않습니다
`redteam_case_results`	캠페인 결과
`text2sql_replay_bundles` · `text2sql_query_logs`	Text2SQL 자체 보존 설정·분석 대상
`quota_reservations`	자체 만료·정리 주기 보유

> `policy_decision_events` 를 함께 지울지는 판단이 갈립니다. 거버넌스 판단이 감사 자료로 요구되는 환경이 있어 **자동 삭제 대상에 넣지 않았습니다.** 조직 정책상 함께 정리해야 한다면 알려주세요.

9.4 Fallback 로그 재처리

DB 장애 중 fallback NDJSON 로 빠진 감사 로그를 DB 에 다시 적재합니다. 성공한 라인은 파일에서 제거되고, 깨진 JSON 이나 아직 삽입할 수 없는 라인은 남습니다. 재처리 실행은 `fallback.replay` 감사 로그로 기록됩니다.

9.5 관리자 변경 이력

API 키 발급/상태 변경, provider 변경, quota CRUD, kill switch, 알림 규칙, 요청 태그, 저장된 필터 등 모든 admin 동작이 append-only 로 기록됩니다.

"감사 로그 CSV 다운로드" 로 관리자 변경 + 알림 발화 이력을 한 파일에 한국어 CSV(UTF-8 BOM) 로 받을 수 있습니다 — 분기 감사 보고 / 회계 첨부용.

---

10. 일상 운영 체크리스트

매일

- ☐ 대시보드의 KPI / 상태 분포 카드에서 4xx·5xx 비율이 평소와 다른지 확인
- ☐ 시계열 차트 비용 곡선이 평소 곡선과 다른지 확인
- ☐ 안전 탭에서 발화한 알림이 있는지 확인
- ☐ (자동화 안 되어 있다면) `scripts/backup.sh` 실행

매주

- ☐ 사용자 탭 정렬 → 누적 비용 상위 5명 확인 → 평소와 다른 폭주가 있는지
- ☐ IP 탭에서 "고유 키 수" 가 비정상적으로 많은 IP (한 IP 에서 키 여러 개로 호출) 가 있는지
- ☐ 사용 한도가 80% 진행률을 넘은 항목이 있다면 다음 달 한도 조정 검토

매월

- ☐ 감사 로그 CSV 받아 보관
- ☐ 비활성화된 키 / 미사용 provider 정리
- ☐ `RETENTION_*` 값이 현재 데이터 크기와 운영 정책에 맞는지 재검토
- ☐ backup 디렉토리 용량 확인 (자동 보존 정책이 잘 작동 중인지)

---

11. 권한 분리

`ADMIN_READONLY_TOKEN` 을 운영자가 별도 발급하면, 회계/감사/리더는 GET/HEAD 만 가능한 읽기전용 어드민 접근권을 받을 수 있습니다.

- 대시보드 / 통계 / 검색 / CSV 다운로드 → ✅ 가능
- 키 발급 / provider 변경 / quota CRUD / kill switch / 알림 / 저장된 필터 변경 / 태그 수정 → ❌ 401

읽기전용 토큰을 분실하거나 인사 이동이 있는 경우 운영자가 `ADMIN_READONLY_TOKEN` 환경변수를 새 값으로 바꾸고 재기동하세요.

---

12. 자주 묻는 관리자 질문

Q. 키 이름을 바꿀 수 있나요? A. 현재 PATCH 는 status 만 받습니다. 이름을 바꾸려면 같은 키 ID 로 POST 를 다시 보내거나(같은 시크릿이면 동일 ID), 비활성화 + 새 키 발급을 권장합니다.

Q. 프롬프트 원문도 보고 싶어요. A. 운영 정책상 기본 OFF 입니다. 게이트웨이를 재기동할 때 `LOG_RAW_PROMPTS=true` 를 설정하면 이후 호출의 원문이 `prompt_logs.content_text` 컬럼에 저장됩니다(이전 호출은 hash 만). 동시에 `LOG_RAW_BODIES=true` 면 요청 재실행도 가능해집니다. 원문 저장은 PII / 보안 리스크가 있으니 디스크 암호화와 접근 제한을 먼저 확보하세요.

Q. quota 가 너무 빨리 트리거되어 사용자가 불편해해요. A. 안전 탭의 알림 규칙으로 80% 도달 시 미리 통보하도록 해두면 한도를 미리 조정할 수 있습니다.

Q. 새 vendor 를 추가하고 싶어요. A. 설정 → 프로바이더 → 폼에 이름/Base URL/key 입력 후 저장. 모델 패턴을 함께 등록하면 클라이언트 코드 변경 없이 라우팅됩니다.

Q. 비용이 anonymous 로 잡혀요. A. 키 발급 전의 호출이거나, 발급된 키가 한 개도 없을 때입니다. 키를 최소 1개 발급하면 그 이후의 미인증 호출은 401 로 차단됩니다.

Q. 운영 중 게이트웨이를 옮기려면? A. (1) 새 인스턴스를 같은 `GATEWAY_SECRET` 으로 띄움 (2) `data/gateway.db` 복사 (3) DNS 변경. 사용자 측 코드 변경 불필요.