remember

Architecture Whitepaper

A knowledge backend agents can call, and teams can own.

A markdown wiki with hybrid retrieval — BM25 + vector, RRF-fused — that scales from a personal vault on a laptop to a multi-tenant, access-controlled team brain deployed into your own cloud.

Version **0.1** · May 2026 · remember.dev
Authored by the remember team · hello@remember.dev

Contents

01 Executive summary 3

02 The problem with team knowledge 4

03 Architecture: one engine, two editions 5

04 Hybrid retrieval — BM25 + vector + RRF 6

05 Multi-tenant design 7

06 AI agent integration 8

07 Deployment model — self-owned cloud 9

08 Security & trust posture 10

09 Roadmap & design partner program 10

§ 01 — EXECUTIVE SUMMARY

What remember is, in one page.

remember is an open-core knowledge platform built around one technical conviction: retrieval is the product. A wiki that you cannot quickly search is a junk drawer with a nice CSS theme.

The platform ships in two editions that share a single retrieval engine:

- **Personal (local)** — an MIT-licensed CLI that indexes any folder of markdown into a hybrid BM25 + vector index, runs offline, and exposes a REST API plus a hosted viewer. Comfortable up to roughly 25–50,000 pages on commodity hardware.
- **Team (cloud)** — the same engine wrapped in a multi-tenant control plane. Schema-per-workspace isolation in Postgres, prefix-per-workspace storage in S3-compatible object stores, Clerk-backed SSO with IP allow-listing, audit logging, and an Anthropic-shaped `/tools` endpoint per workspace so AI agents can retrieve from the wiki as a typed tool call.

The cloud edition is **self-deployable**. Customers run it in their own Vercel / AWS / GCP. Their database. Their bucket. Their domain. We provide the platform code under a commercial license, plus the deployment and ongoing support — but never custody of the data.

> POSITION
>
> remember is not a competitor to a block-editor SaaS. It is a knowledge backend optimized for two consumers that traditional wikis serve poorly: a developer who wants fast, portable, inspectable retrieval; and an AI agent that needs a stable tool API to call.

§ 02 — THE PROBLEM

Why team knowledge fails.

Every team wiki we have used disappointed us on the same axis: search. You write a page. Six months later, when you need it, you cannot find it.

The diagnosis

Conventional wikis treat search as an afterthought bolted onto a CMS. They index titles and bodies with naive full-text matching, ignore semantic similarity, and surface results ranked by a

heuristic that no one on the team can explain. When the wiki grows past a few thousand pages, recall degrades silently — answers exist but cannot be surfaced.

The folk-remedy response is *tagging discipline*: ask humans to attach consistent metadata to every page. This fails reliably. Humans do not tag consistently, they do not maintain tag taxonomies, and they leave the company before they update the backfill.

The other emerging pressure

AI agents now want to call internal knowledge as a tool. They cannot. Most wikis expose no API, or expose only a search endpoint whose results are tuned for human eyes (HTML, no source attribution, no fusion logic). Teams that want their agents to ground answers in internal docs end up writing a bespoke RAG pipeline that duplicates the wiki's content and that someone has to babysit forever.

What "good" retrieval looks like

A retrieval system that actually serves a working team has four properties:

- **Hybrid**: both exact-keyword matching and semantic similarity run on every query, then fuse — neither alone is sufficient.
- **Inspectable**: every result carries which retriever surfaced it and at what score, so an engineer or an agent can explain why an answer was returned.
- **Portable**: content is plain markdown the team owns, not locked inside a proprietary block-editor schema that requires the vendor to read.
- **Callable**: there is a stable typed API that agents and humans both use, governed by the same auth and audit layer.

> DESIGN CONVICTION
>
> Search should be the first surface of the product, not a back-pocket feature.

§ 03 — ARCHITECTURE

One engine, two editions.

remember is open-core. The retrieval engine — chunker, parser, hybrid search, indexer, store interfaces — is MIT-licensed and lives in `@remember/core`. The cloud platform adds a commercial control plane around it.

LAYER	PERSONAL (LOCAL)	TEAM (CLOUD)
License	MIT	Commercial
Content store	Local filesystem	S3-compatible (R2 / S3 / MinIO)
Vector store	sqlite-vec	pgvector on Postgres + HNSW
Full-text store	sqlite-fts	Postgres FTS
Auth	None — local trust	Clerk + per-workspace API keys
Isolation	—	Schema-per-tenant
Surfaces	CLI, REST, Astro viewer	REST, /tools (agent), viewer, admin dashboard

The shared retrieval engine

Both editions run the same code path for a query:

1. Tokenize the query and the query embedding (the latter via a configurable embedder — OpenAI `text-embedding-3-small` by default, or any compatible endpoint).
2. Issue BM25 and vector retrievals in parallel against their respective indexes.
3. Fuse with Reciprocal Rank Fusion (k = 60) to produce a single ranked list with per-result retriever attribution.
4. Optionally apply a cross-encoder reranker (a planned feature on Team and above).

The cloud-only control plane

The Team edition adds workspaces, members, roles, allow-listing, billing, custom domains, connector configurations, audit logging, and backup orchestration. None of these touch retrieval semantics — they govern *who can run a query, and against which data*.

```
// One engine. Same primitives across editions.
```

```
createIndexer({
  walker: createR2Walker(workspaceId),  // local: chokidar; cloud: R2
  parser: createRemarkParser(),
  chunker: createSmartSplitChunker(),
  embedder: pickEmbedder(),              // env-driven
  store: createPgvectorStore({ workspaceId }), // or sqlite-vec locally
}).indexAll()
```

§ 04 — HYBRID RETRIEVAL

BM25 and vector, fused with RRF.

Every query in remember runs through both retrievers in parallel. Reciprocal Rank Fusion combines them into one ranked list. You get the exact-keyword match *and* the semantically-related page in the same response.

Why hybrid

BM25 is excellent at exact-term and rare-token recall. Vector retrieval handles paraphrase and concept overlap that lexical matching misses. Neither alone is sufficient for production knowledge retrieval. The fusion step lets each retriever contribute candidates without one dominating.

The fusion math

Reciprocal Rank Fusion assigns each candidate a score based on its rank in each list, not the raw relevance score. For a result at rank `r` in one list and rank `r'` in the other, with a smoothing constant `k = 60`:

```
RRF_score(result) = 1 / (k + r_bm25) + 1 / (k + r_vector)
```

This avoids the score-calibration problem (BM25 scores and cosine similarities live on different scales) and surfaces results that perform well in *either* retriever. Results that appear in both retrievers naturally rise to the top.

Result attribution

Every fused hit carries a `retrievers` field — an array of which lanes produced it. This is small but important for agent use cases: a Claude agent asked "why this answer" can include the retriever attribution in its explanation. It is also a debugging tool — if recall is poor, the attribution tells you which lane is failing.

```
{
  "path": "guides/what-is-remember.md",
  "url": "https://acme.remember.dev/guides/what-is-remember",
  "snippet": "remember is a markdown wiki with hybrid retrieval ...",
  "score": 0.94,
  "retrievers": ["bm25", "vector"]
}
```

ROADMAP

Cross-encoder reranking lands on the Team edition next. A small model rescues the top-N fused candidates by re-scoring with full query-document attention — closing the last gap between retrieval and a true RAG answer surface.

§ 05 — MULTI-TENANT DESIGN

Schema-per-workspace, by construction.

The Team edition uses hard isolation: each workspace lives in its own Postgres schema (`tenant_<uuid_hex>`) and its own S3 prefix (`tenants/<uuid>/pages/`). Cross-workspace data access is impossible by construction — a query targeted at one schema cannot read another.

Why schema-per-tenant

- **Auditability**: a security review can verify isolation by inspecting schema permissions and prefix policies — not by reading every WHERE clause in the app.
- **Backup granularity**: per-workspace export and restore are atomic operations against a single schema. Useful for compliance, customer migrations, and enterprise carve-outs.
- **Performance shape**: indexes and pg statistics are per-tenant, so large workspaces do not degrade query plans for small ones.
- **Carveout path**: an enterprise tenant that needs dedicated hardware can move to a separate cluster without code changes — the workspace already looks like a self-contained database.

The access control layers

Three layers of allow-listing combine on every workspace-scoped request:

1. **SSO domain restriction**: only users with email addresses in the configured domain set can sign in to the workspace (e.g. `@yourco.com`).

2. **IP allow-list**: requests outside the configured CIDR range return `403 IP_NOT_ALLOWLISTED`. Useful for VPN-gated internal use.
3. **Per-workspace API keys**: keys are scoped to a single workspace, have their own role (owner / admin / editor / viewer), and are SHA-256 hashed at rest. The full key is shown once at creation, then never recoverable.

Audit log

Every write event — page upsert, member role change, settings edit, connector sync — writes one row to a per-workspace `audit_events` table. Schema: `{ id, ts, actor_id, actor_kind, action, target, meta }`. The Team plan exports the log via API for ingestion into the customer's SIEM.

§ 06 — AI AGENT INTEGRATION

The knowledge tool your agent calls.

The Team edition emits Anthropic-compatible tool definitions per workspace at `GET /v1/:workspace/tools`. A Claude agent ingests these definitions directly, calls them as typed tools, and receives ranked chunks with source URLs back — no glue code required.

The /tools surface

```
GET /v1/acme/tools

{
  "tools": [
    {
      "name": "search_wiki",
      "description": "Hybrid search across the workspace wiki. Returns ranked
chunks.",
      "input_schema": {
        "type": "object",
        "properties": {
          "query": { "type": "string" },
          "k":     { "type": "integer", "minimum": 1, "maximum": 50, "default": 10 }
        },
        "required": ["query"]
      }
    },
    { "name": "get_page",  "description": "Fetch one page by path.", ... },
    { "name": "list_pages","description": "List workspace pages, paginated.", ... }
  ]
}
```

Governance

Every agent tool call goes through the same auth and audit layer as a human request:

- The agent authenticates with a per-workspace API key (typically scoped to the `editor` role).
- The IP allow-list applies — agents calling from outside the configured network are rejected.
- Each tool call writes an audit row, with `actor_kind = 'api_key'` and the agent's identifying metadata in the `meta` field.
- Plan-aware rate limits prevent runaway agents from exhausting AI-call quota.

> WHY NOT JUST EXPOSE A SEARCH ENDPOINT?

> Because Anthropic-shaped tool definitions give the agent the schema it needs to call well. The agent does not need to read documentation; it reads the `input_schema` JSON and constructs valid invocations. The same is true for OpenAI's function-calling schema (the formats are similar enough that a small adapter suffices).

§ 07 — DEPLOYMENT MODEL

Self-owned cloud, not multi-tenant SaaS.

remember's cloud edition does not run on our infrastructure. It deploys into yours — your Vercel, AWS, or GCP — and uses databases you provision. We never have custody of your data, your encryption keys, or your domain.

The shape of a deployment

A standard install is five resources, all under the customer's account:

RESOURCE	RECOMMENDATION	WHY
Compute	Vercel (Fluid Compute)	Edge-close functions, zero ops, generous free tier
Postgres	Neon	Branching for backups, pgvector built-in, serverless pricing
Object storage	Cloudflare R2 or S3	Cheap, egress-free in the R2 case, presigned URL support
Auth	Clerk	SSO, SAML, JIT provisioning, drop-in middleware
Email	Resend	Transactional, deliverability, simple API

What we provide

- The platform code, under a commercial license that includes the right to deploy into your infrastructure for your team's use.
- A guided deployment — typically a short call plus a few hours of setup.
- Ongoing support: upgrades, security patches, incident response.
- The OSS retrieval engine, under MIT, with cloud features back-ported on a 90-day cadence.

What you keep

- Your data, in your database, behind your firewall.
- The encryption keys and the network policies.
- Your domain, your TLS certificates, your CDN configuration.
- The right to walk away at any time — content is plain markdown in your bucket.

> "We deploy the team build into your team's infrastructure as a guided onboarding. If that fits, the best next step is a short call."
>
> — remember handoff doc

§ 08 — SECURITY

Trust posture.

Because the cloud edition deploys into your infrastructure, your existing controls govern the data path. We do not hold customer data, do not have read access to content, and do not generate audit gaps your security team would have to question.

Compliance posture

The Team edition deploys into the customer's existing compliance envelope — if the customer is SOC 2 / HIPAA / FedRAMP, the data path inherits those controls. We share our architecture, threat model, and a sample DPA on request. Formal SOC 2 attestation on our own infrastructure is not currently in scope, because there is no such infrastructure holding customer data.

What we do not do

- We do not train AI models on customer content.
- We do not sell usage analytics to advertisers or third parties.
- We do not run marketing pixels, retargeting cookies, or behavioral analytics on the workspace surfaces.
- We do not embed proprietary editors that lock content into our format.

§ 09 — ROADMAP & DESIGN PARTNERS

Where this goes from here.

remember is at the design-partner stage. The personal edition ships under MIT today; the cloud edition is feature-complete and deployed with the first design partners as a guided onboarding.

What design partners get

- The cloud platform, deployed into their cloud, by us, with direct support.
- Custom pricing reflecting the design-partner discount.
- Roadmap influence — features land in the order partners need them.
- The OSS engine at parity (90-day back-port commitment).

Near-term roadmap (committed)

- Cross-encoder reranker for Team and above.
- Notion and WebDAV connectors at parity with the GitHub connector.
- Custom embedder support (bring-your-own model via Vercel AI Gateway).
- Admin dashboard for non-engineering workspace owners.

Longer-term (exploratory)

- Multi-step agent workflows over the workspace.
- In-browser markdown editor (read-write viewer mode).
- Self-serve cloud deployment with one-click provisioning.

To explore being a design partner, or to ask questions this whitepaper did not answer:

Email: hello@remember.dev
OSS engine: github.com/aakusch/remember
Web: remember.dev

© 2026 remember.dev · all rights reserved