



TOKENOME DOCUMENTATION

Team server setup

Standing up the server, enrolling a laptop, opting a project in, the share gate, retract and purge, and searching as a team.



tokenome.ai/docs/team

Rendered 5 October 2026

Team

Team adds a shared index your organization runs, so one search covers everyone's opted-in projects with attribution. Each laptop keeps its private local index exactly as before.

Team is in beta, and free while the beta runs. Nothing is enforced, there is no seat limit, and there is no key to install. No end date is set; you will get notice well before anything changes.

You run the server, on your own infrastructure. Tokenome does not host one for you.

The shape of it

Each person opts specific projects in. From then on, new and changed segments from those projects are mirrored to the server as the local indexer sees them, redacted on the laptop before they leave it. Searches then cover the team first, with each hit attributed to the person who shared it, and fall back to local results if the server cannot answer. Nothing is shared by default, and anyone can retract everything they have shared at any time.

Standing up the server

The server is one container image, `ghcr.io/tokenome/tokenome-server`, built and smoke-tested on x86_64 and aarch64 with every release. It needs a search engine of its own and a volume that survives redeploy. Do not point it at the search engine your laptop uses; it writes into the same collection.

With Docker

The compose bundle on the releases page pins the version and brings up the index, the server and automatic TLS, with only ports 80 and 443 exposed. Set your domain and a strong index key in its environment file, then bring it up.

On Railway

One project, two services, if you would rather not run a VM.

1. Create the Typesense service from the public image, with a volume mounted at `/data`, a data directory and API key set, the API address bound dual-stack, and a thread pool cap. Give it no public domain.
2. Create the server service from the container image, with its own volume at `/data`, the index host, port and key pointing at the first service, a JWT secret, and the bootstrap admin variable for the first boot only.
3. Generate a public domain for the server service on port 8080, and set the health check to `/v1/health`.

The volume on the server service is the step people miss. Without it the identity database holding users, devices, holds and the audit log lives on the container disk and is wiped by every deploy. The symptom is a laptop that enrolled minutes ago being told its device is unknown.

The first admin

Bootstrapping creates the identity database, the signing secret and the first admin, and prints a one-time enrollment token and eight recovery codes. Keep the codes offline and away from that laptop: one of them re-admits the admin if the laptop is lost. On a hosted deployment the bootstrap prints no recovery codes, because a deploy log is not the place for them; generate them in the console once you are in, and remove the bootstrap variable afterwards.

Check it from outside before you hand out any tokens:

```
curl -s https://<server>/v1/health
```

Enrolling a laptop

With the enrollment token and the server address in hand:

```
tokenome remote enroll --server https://<server> --token <token> --label "my-laptop"
```

Enrollment generates a keypair on the laptop and sends only the public half. There is no password. Tokens are single use and expire after 72 hours, so deliver them in person or over a channel you trust, one per device.

A second laptop never uses a second seat. While the old one still works, press **Add a device** on My footprint and enroll the new one with `--restore-shares`, which shares the same projects you shared before. Local history does not move between machines.

Opting a project in

The screenshot shows the 'Team' panel in the tokenome interface. The left sidebar contains navigation links: Dashboard, Search, Code context, Conversations, Journal, FAQ, Team (highlighted), Sources, and Settings. The main content area is titled 'Team' and displays the following information:

- Team Status:** `http://127.0.0.1:8612` enrolled as dana. Buttons: Check server, Push now.
- Backfill of acme-payments finished:** 8 segments shipped, 1 held back for review.
- Held back from sharing (1):** The share gate found something it could not make safe. Nothing here has left this machine. Release sends the redacted form; discard keeps it local for good. Below this, a message for 'acme-payments · reconciliation #0' states: 'matches your exclude pattern settlement file. Design a nightly reconciliation between the ledger and the processor's settlement file.' Buttons: Release, Discard.
- Projects:** Sharing a project mirrors its new conversations to the team server as they are indexed. *Backfill* sends what is already indexed locally. Unsharing stops future sharing; *retract* also deletes what was sent. A table shows project status:

Project	Local segments	Shared	Actions
acme-payments	9	☒	Backfill, Retract
acme-web	1	☐	

- What I've shared:** 8 segments on the server. Retention is set by your admin; a window means older conversations are removed automatically. A button 'All (8)' is shown, followed by 'acme-payments (8) · kept until you delete it'.
- Shared Data Table:**

When	Project	Conversation	Excerpt
2026-09-27	acme-payments	Two [REDACTED:name] postings landed on their old account after this morning's migration... #0	Two [REDACTED:name] postings landed on their old account after this morning's migration. Is that the cache?
2026-09-25	acme-payments	resolve_account is most of our p95 latency. Can we cache it? #0	resolve_account is most of our p95 latency. Can we cache it?
2026-09-21	acme-payments	Support found two ledger entries with the same idempotency key, about 400ms apar... #1	Do it, and keep the metric.
2026-09-21	acme-payments	Support found two ledger entries with the same idempotency key, about 400ms apar... #0	Support found two ledger entries with the same idempotency key, about 400ms apart. Why is the ledger double-posting?
2026-	acme-	The ledger.post metric spikes during incidents. Is that	The ledger.post metric spikes during incidents. Is that real traffic or

At the bottom left, a status bar shows 'ready' and '10 segments'.

The Team panel: what is shared, what was held back and why, and everything this laptop has on the server. The project and its conversations are sample data; the software is not.

```
tokenome remote share <project> --backfill
tokenome remote projects
tokenome remote status
tokenome remote shared
```

Nothing is shared until you run that first line. `--backfill` also sends the history you have already indexed for that project; without it, only turns indexed after you opted in are mirrored, which leaves a new server looking empty. The server reconciles by content, so running a backfill twice does not duplicate anything.

Three things are never shared: projects not on your list, segments with no project at all, and anything on a laptop that has not enrolled. Your local index stays complete either way.

The share gate and redaction

The screenshot shows the 'Team' section of the tokenome interface. On the left is a sidebar with navigation links: Dashboard, Search, Code context, Conversations, Journal, FAQ, Team (highlighted), Sources, and Settings. The main content area is titled 'Team' and shows a server status 'http://127.0.0.1:8612 enrolled as dana' with 'Check server' and 'Push now' buttons. Below this is a message: 'Backfill of acme-payments finished: 8 segments shipped, 1 held back for review.' A section titled 'Held back from sharing (1)' explains that the share gate found something it couldn't make safe. It shows a specific example: 'acme-payments · reconciliation #0' which matches an exclude pattern. It provides 'Release' and 'Discard' buttons. Below this is a 'Projects' section with a table showing local and shared segments for 'acme-payments' and 'acme-web'. The 'acme-payments' project has 9 local segments and 1 shared segment, with a 'Backfill' button. The 'acme-web' project has 1 local segment and 0 shared segments. A 'What I've shared' section shows a list of 8 segments, with a table of details including date, project, conversation, and excerpt. The bottom status bar shows 'ready' and '10 segments'.

tokenome

Dashboard

Search

Code context

Conversations

Journal

FAQ

Team

Sources

Settings

Team

http://127.0.0.1:8612
enrolled as dana

Check server Push now

Backfill of acme-payments finished: 8 segments shipped, 1 held back for review.

Held back from sharing (1)

The share gate found something it could not make safe. Nothing here has left this machine. Release sends the redacted form; discard keeps it local for good.

acme-payments · reconciliation #0
matches your exclude pattern settlement file
Design a nightly reconciliation between the ledger and the processor's settlement file.

Release Discard

Projects

Sharing a project mirrors its new conversations to the team server as they are indexed. *Backfill* sends what is already indexed locally. Unsharing stops future sharing; *retract* also deletes what was sent.

Project	Local segments	Shared
acme-payments	9	☒ Backfill Retract
acme-web	1	☐

What I've shared

8 segments on the server. Retention is set by your admin; a window means older conversations are removed automatically.

All (8) acme-payments (8) · kept until you delete it

When	Project	Conversation	Excerpt
2026-09-27	acme-payments	Two [REDACTED:name] postings landed on their old account after this morning's migratio... #0	Two [REDACTED:name] postings landed on their old account after this morning's migration. Is that the cache?
2026-09-25	acme-payments	resolve_account is most of our p95 latency. Can we cache it? #0	resolve_account is most of our p95 latency. Can we cache it?
2026-09-21	acme-payments	Support found two ledger entries with the same idempotency key, about 400ms apar... #1	Do it, and keep the metric.
2026-09-21	acme-payments	Support found two ledger entries with the same idempotency key, about 400ms apar... #0	Support found two ledger entries with the same idempotency key, about 400ms apart. Why is the ledger double-posting?
2026-	acme-	The ledger.post metric spikes during incidents. Is that	The ledger.post metric spikes during incidents. Is that real traffic or

ready
10 segments

The gate holds back what it cannot make safe, and redacts the rest before it leaves the laptop. Sample data.

Before a segment leaves the machine it passes the gate. Secrets, meaning keys, tokens, private keys, connection strings and assignments that look like credentials, are replaced with a marker. Personal data, meaning email addresses, phone numbers and card or identifier shapes, is replaced with typed markers unless you turn that off.

Names cannot be recognised by shape, so the gate works from a dictionary you keep: people, customers, codenames. A listed name is replaced wherever it appears, in any case, and never holds a segment back.

A segment the gate cannot make safe, a private key block, a credential dump, or a match on one of your exclude patterns, is held back. It stays on your machine, the app tells you, and the Team panel lists it with Release, which sends the redacted form, and Discard, which keeps it local for good.

```
tokenome remote gate # the rules in force
tokenome remote gate --exclude '^customers/' # hold back anything touching that path
tokenome remote gate --redact-name 'Acme Corp' # everywhere
tokenome remote gate --project acme --exclude '^invoices/'
tokenome remote gate --pii off # keep email addresses in shared text
tokenome remote quarantine # what is held back, and why
tokenome remote quarantine --release <id>
tokenome remote quarantine --discard <id>
```

Exclude patterns and names can be global or apply to a single project, on top of the global lists. The same controls are in the Team panel's share gate section, with a block per shared project.

When the gate rewrites any text, the segment ships without its vector and the server computes a new one from the redacted text, so the vector cannot carry what the redaction removed.

Retract and purge

```
tokenome remote unshare <project>          # stop sharing future turns
tokenome remote unshare <project> --purge    # and retract what is already there
tokenome remote push                          # flush the queue now
```

Deleting a conversation in the app removes it here and queues its removal on the server, whether or not the project is still shared. Removals ride the same queue as pushes and in the same order, so being offline only delays them. Documents under a legal hold stay on the server, and the app tells you how many were kept.

Searching as a team

The screenshot shows the tokenome Search interface. On the left is a sidebar with navigation links: Dashboard, Search (highlighted), Code context, Conversations, Journal, FAQ, Team, Sources, and Settings. At the bottom of the sidebar, there's a status bar showing 'ready' and '10 segments'. The main content area is titled 'Search' and contains a search bar with the query 'how do refunds stay idempotent'. Below the search bar are filters for 'platform' and 'project', and a date range selector with 'after' and 'before' fields. A 'Search' button is on the right. Below the search bar, there's a 'Team (10)' section showing a list of search results. Each result includes the team name, a user icon, the team name, the project name, the date and time, the score, and an 'open' link. The results are as follows:

- Team (10)**
 - Refunds processed twice on webhook retry**
Q: Refunds are processed twice when the processor retries its webhook. How should the refund path handle idempotency?
Refunds are processed twice when the processor retries its webhook. How should the refund path handle idempotency?
 - Support found two ledger entries with the same idempotency key, about 400ms apart...**
Q: Support found two ledger entries with the same idempotency key, about 400ms apart. Why is the ledger double-posting?
Because the retry wrapper sits outside the ledger write. When the commit is slow the wrapper times out and retries, but the first write has already landed, and nothing checks the key before writing again. Idempotency lives in the ledger now, because retries are the client's job a...
 - For a payments ledger, should duplicate detection live in the client or the serv...**
Q: For a payments ledger, should duplicate detection live in the client or the server?
On the server. Clients retry blindly after timeouts, so the service that writes the record should reject a key it has already stored, rather than trusting every caller to check first.
 - Two [REDACTED:name] postings landed on their old account after this morning's migratio...**
Q: Two [REDACTED:name] postings landed on their old account after this morning's migration. Is that the cache?
Yes. The sixty second cache served the pre-migration account. Account lookups went back to direct reads, because a stale account misroutes money and the latency win is not worth that. Invalidating on migration is the option to try if p95 climbs again.
 - The ledger.post metric spikes during incidents. Is that real traffic or retries?**
Q: The ledger.post metric spikes during incidents. Is that real traffic or retries?
Retries. The decorator emits once per attempt, which is why the dashboard over-counts during incidents. If backoff moves to the client, emit once per posting inside post_entry, with the tenant attached.

One query, yours and the team's, with attribution on every team hit. Sample data.

Once enrolled, search covers the team by default and every team hit names the person it came from. The Search page gains a scope selector and the CLI takes `--scope team`, `mine` or `all`; see [Finding things](#).

A teammate's conversation opens in a read-only reader that names who shared it.

If the server does not answer quickly you get your local results with a visible warning, rather than a search that hangs. Two filters, label and conversation, are always answered locally.

What the server holds

The server holds shared text and its vectors in the clear. That is what makes keyword and semantic search possible, and we would rather say so than imply otherwise. The protection is organizational: you run the server, on your infrastructure, behind TLS and encrypted at rest, and redaction is a safety net rather than a licence to share what the organization should not hold.

Next: [the admin console](#), which is where people, devices, retention and the audit log are managed.