

EigenVertex Cloud

Connecting a Coding Agent to Your Data

A complete guide, with a real worked example

13 September 2026

Contents

(In Word: right-click the table above → Update Field)

1. Introduction

EigenVertex is built around three experiences: a UX for the people who use it, a DX for developers who integrate it, and — the subject of this guide — an AX, an Agent Experience, for the AI coding assistants that build on top of it directly.

This guide is for anyone with an EigenVertex Cloud account who wants their coding agent — Claude Code, Claude Desktop, Claude.ai, or any other MCP-compatible assistant — to work with their own data: exploring it, asking it questions, and building real integrations into their own applications. No self-hosting. No database credentials. No infrastructure access of any kind.

What makes this different

Everything in this guide has been tested against a live EigenVertex Cloud account, not just designed on paper. Section 5 walks through a real session, screenshot included.

Who this is for

- You have (or can create) an EigenVertex account with at least one Corpus — a knowledge base of your own content.
- You want an AI coding assistant to explore that data, or to build a feature (like a chat panel) into an app you're working on.
- You are not self-hosting EigenVertex — if you are, this guide isn't for you; ask your EigenVertex contact for the operator documentation instead.

2. Prerequisites

Two things, both created through the EigenVertex Studio web app (app.eigenvertex.com), no command line involved:

2.1 A Corpus

A Corpus is your knowledge base — documents, notes, and other content, grouped in one place. Creating one in Studio also creates the Workspace that owns it; you'll pick a name and, if relevant, invite collaborators. If you already have one (uploaded documents, a project space with content), you're set.

2.2 A Constellation

A Constellation is a queryable graph built from one or more Corpora — it's what you actually ask questions to. Most customers start with a Constellation over a single Corpus; you can group several together later if your content is spread across more than one.

Note

A Corpus can feed only one Constellation at a time today. If you're planning your content structure, keep that in mind.

3. Two Ways to Connect an Agent

Both are real, both are documented in this guide — which one you want depends on what you're trying to do.

	Option A — Chat live (MCP)	Option B — Build a feature (Relay SDK)
What it's for	Exploring your data, asking questions, having an agent act on it during a conversation	Adding a persistent chat feature to your own app, for your own users
Where it runs	Directly inside your conversation with Claude	Server-side, inside an app you (or your agent) build
What you need	Just your EigenVertex login (OAuth)	An API key from Studio
Setup time	About one minute	A few minutes, mostly agent-driven

Most people start with Option A, since it needs no setup beyond logging in. Section 5 gives you a real, complete example of it.

4. Option A — Chat Live With Your Data (MCP)

MCP (Model Context Protocol) is an open standard that lets an AI assistant discover and call real tools — not just generate text — in the middle of a normal conversation. It's the same mechanism behind connectors to Notion, Slack, or Google Drive. EigenVertex exposes one at a fixed address:

```
https://api.eigenvertex.com/mcp
```

4.1 Claude Desktop or Claude.ai (web)

1. Open Settings → Connectors.
2. Click "Add custom connector."
3. Paste the server address above.
4. Click "Connect." This opens EigenVertex's own login screen — Google, Apple, or email/password, whichever you normally use.

No Client ID or Secret to fill in — EigenVertex registers the connection automatically the first time.

4.2 Claude Code (terminal, inside your project)

If you haven't met Claude Code before: it's Claude running as a coding agent in your terminal or IDE, working directly inside a real project rather than a chat window. From inside your project folder:

```
claude mcp add --transport http eigenvertex
https://api.eigenvertex.com/mcp
```

Then, in a Claude Code session in that same project, run:

```
/mcp
```

This opens the same browser-based login. Section 5 shows exactly what this looks like in practice.

4.3 What you can ask for

No special syntax — plain language, and the agent picks the right tool:

- “Check EigenVertex's status.”
- “List my corpora.”
- “Ask my [Constellation name] what [a real question about your content].” — returns a grounded answer with citations back to your own documents.
- “Search only my [corpus name] corpus for X” — scoped to one corpus, even if it's part of a wider Constellation.
- “Ingest this into my [corpus name] corpus.” — the agent can write content it just produced directly into your knowledge base.

Some actions — generally the ones that change something — ask for a one-time approval in the interface before running. That's expected, not an error.

4.4 What this connection will never do

By design

This connection cannot create or reveal a raw API key. Key material never crosses it, in either direction. That boundary was specifically built and tested — see Section 8 for the security model in more detail.

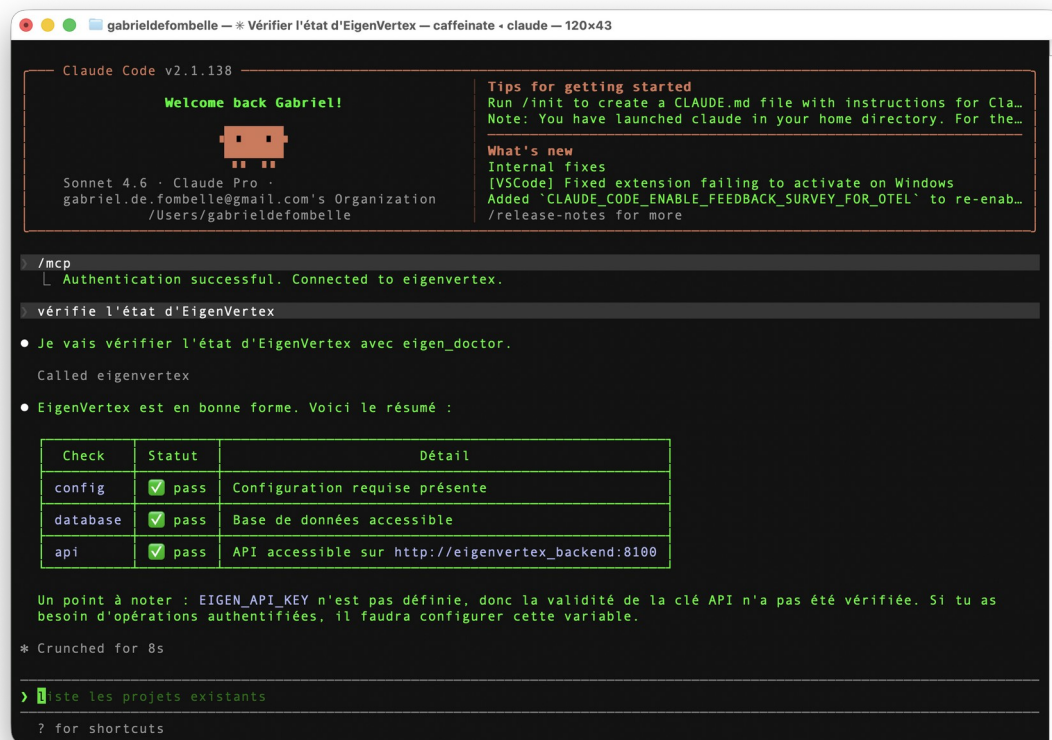
5. Worked Example — A Real Session

Everything below actually happened, in one real session, against a live EigenVertex Cloud account. Names and content are real too — nothing staged for this guide.

Step 1 — Connect Claude Code

```
claude mcp add --transport http eigenvertex
https://api.eigenvertex.com/mcp
```

Then, inside a Claude Code session, running `/mcp` opens a browser window for login. Once done, Claude Code confirms the connection and the tools become available — shown below, in the actual terminal:



The screenshot shows a Claude Code v2.1.138 terminal window. It displays a welcome message for 'Gabriel', a list of installed MCP servers, and the execution of the `/mcp` command which successfully connects to 'eigenvertex'. A subsequent natural language request in French, 'vérifie l'état d'EigenVertex', is processed by the `eigen_doctor` tool. The tool returns a status report for 'config', 'database', and 'api', all of which are 'pass'. A note mentions that the `EIGEN_API_KEY` is not defined. The session ends with a prompt to list existing projects.

```
Claude Code v2.1.138 — gabrieldefombelle — * Vérifier l'état d'EigenVertex — caffeinate - claude — 120x43

Welcome back Gabriel!

Sonnet 4.6 · Claude Pro ·
gabriel.de.fombelle@gmail.com's Organization
/Users/gabrieldefombelle

Tips for getting started
Run /init to create a CLAUDE.md file with instructions for Cla...
Note: You have launched claude in your home directory. For the...

What's new
Internal fixes
[VSCode] Fixed extension failing to activate on Windows
Added 'CLAUDE_CODE_ENABLE_FEEDBACK_SURVEY_FOR_OTEL' to re-enab...
/release-notes for more

/mcp
└─ Authentication successful. Connected to eigenvertex.

vérifie l'état d'EigenVertex

• Je vais vérifier l'état d'EigenVertex avec eigen_doctor.
  Called eigenvertex

• EigenVertex est en bonne forme. Voici le résumé :



| Check    | Statut | Détail                                             |
|----------|--------|----------------------------------------------------|
| config   | ✓ pass | Configuration requise présente                     |
| database | ✓ pass | Base de données accessible                         |
| api      | ✓ pass | API accessible sur http://eigenvertex_backend:8100 |



Un point à noter : EIGEN_API_KEY n'est pas définie, donc la validité de la clé API n'a pas été vérifiée. Si tu as
besoin d'opérations authentifiées, il faudra configurer cette variable.

* Crunched for 8s

> liste les projets existants

? for shortcuts
```

A real Claude Code session: OAuth authentication succeeding, then a natural-language request routed automatically to the `eigen_doctor` tool.

Notice what happened: the request “vérifie l'état d'EigenVertex” (“check EigenVertex's status”) was typed in plain French, in the middle of an otherwise English session — Claude Code picked the right tool (`eigen_doctor`) on its own and rendered the result as a clean table. No command syntax to remember.

Step 2 — Ask a real question

In the same connected session, a real question was asked against a Constellation built from piano-acoustics literature:

```
“What causes inharmonicity in piano strings, and how does it affect
tuning?”
```

The answer came back grounded in nine real citations — a 2013 academic paper on piano inharmonicity, literature on the “Railsback stretch,” and one of the account's own French-language notes on temperaments — not a generic answer assembled from general knowledge.

Step 3 — Prove it isn't guessing

To confirm the answers are genuinely grounded and not just plausible-sounding, a short note was written on the spot and ingested directly:

“The fictional village of Kelvinstead has a population of 4,217 according to the 2026 census, and its official motto is 'Ad Astra Per Aspera Lentum.'”

A fact invented for this test alone — impossible to know any other way. Asking the Constellation about Kelvinstead immediately afterward returned the population figure and the exact motto, correctly cited to the note just written.

Why this matters

This is the actual test worth running on your own account before trusting any integration: ingest something distinctive, then ask about it. If the exact answer comes back, the grounding is real.

6. Option B — Build a Chat Feature Into Your Own App

Use this when you want a persistent chat feature running inside something you're building — a website, a backend service, anything with its own server.

6.1 Get an API key

In Studio: Account → API Keys → Create key. Choose the Workspace it applies to, name it, and copy the value — it's shown once.

Current limitation

The Studio key form currently issues a combined read+write key; there is no separate read-only option from the interface yet. Treat the key as sensitive accordingly — store it only in your server's own secret store, never in frontend code or a public repository.

6.2 Mount the shared relay — don't hand-write one

The relay is the server-side code that forwards requests to EigenVertex's OpenAI-compatible endpoint, keeps your key off the browser, and fixes the target Constellation so it can't be swapped by a client request. It's a package EigenVertex ships — the right move is to mount it, not reimplement it.

```
from eigenvertex.integrations.fastapi import create_chat_relay_router

router = create_chat_relay_router(
    base_url="https://api.eigenvertex.com",
    api_key=EIGEN_API_KEY,           # from your server's own environment
    constellation_id=YOUR_CONSTELLATION_ID,
)
app.include_router(router)
```

6.3 Add a chat client

Point any chat interface at the route you just mounted (e.g. /chat). If an AI coding assistant is building this for you, describing the goal in plain language is enough — for example:

“Add a chat panel to this app, backed by our relay endpoint at /chat.”

The browser only ever talks to your own server — it never sees the EigenVertex key or the constellation ID directly.

7. Reference — Tools Available Over MCP

The exact set an agent can discover once connected (Option A). Availability may grow over time.

Category	Tools
Status	eigen_doctor
Corpus	eigen_corpus_list, eigen_corpus_create, eigen_corpus_query
Constellation	eigen_constellation_list, eigen_constellation_create, eigen_constellation_query

Content	eigen_document_ingest_text, eigen_document_ingest_url, eigen_session_source_document
Scanning (OCR/OMR)	eigen_scan_batch_create, eigen_scan_batch_status, eigen_scan_item_ingest, eigen_scan_item_omr
Projects, Sessions, Decisions	eigen_project_list/create, eigen_session_list/create, eigen_decision_list/create
Keys (operator use)	eigen_key_list, eigen_key_revoke — eigen_key_create is not available over this connection, by design

8. How This Stays Secure

A few design choices worth knowing, especially if you're deciding whether to connect a coding agent to real data:

- Every request over the MCP connection is tied to your own OAuth login — an agent connected to your account can only see and act on what you yourself have access to.
- Two different EigenVertex accounts, even both connected via MCP, cannot see or affect each other's data — this boundary has been specifically tested by attempting cross-account access and confirming it fails.
- Raw API key material never crosses the MCP connection in either direction. If you need a key for your own app (Option B), you get it once, directly from Studio — an agent connected over MCP cannot mint or retrieve one on your behalf.
- Every write action available over MCP still enforces the same permission checks as the rest of EigenVertex — connecting an agent doesn't grant it any capability you don't already have.

9. Known Limitations, Stated Plainly

Limitation	Detail
No read-only key from Studio yet	The self-service key form issues combined read+write access only — a narrower option is not yet available in the interface.
One Constellation per Corpus	A Corpus can feed only one Constellation at a time today.
MCP answers are complete, not streamed token-by-token	A question over MCP returns a full answer at once. The server-side relay (Option B) does stream to your frontend as it's generated.

10. Troubleshooting

Symptom	Likely cause	What to do
The connector shows “connection problem” in Claude	Usually a transient OAuth hiccup	Try reconnecting from Settings → Connectors. If it persists, contact support with the reference code shown.
A tool call is stuck waiting	It likely needs your one-time approval	Check for a pending approval prompt in the interface before assuming it's broken.
“EIGEN_API_KEY is unset” in a doctor-style check	Expected for the MCP connection — that check only applies to Option B's server-side key	Not an error for Option A; only relevant if you're setting up Option B.
An agent can't find your Corpus	It may not be part of any Constellation yet, or the name doesn't match	Ask the agent to list your corpora first, then reference the exact name or ID returned.

11. Frequently Asked Questions

Does this work with ChatGPT or Gemini?

Only Claude clients (Claude.ai, Claude Desktop, Claude Code) have been tested against this connection so far. Other MCP-compatible assistants may work, but haven't been verified — don't assume it until confirmed.

Do I need to install the eigen CLI?

No. That tool is for EigenVertex operators running their own self-hosted deployment. Nothing in this guide requires it.

Can my agent see other customers' data?

No — see Section 8. Every action is scoped to your own authenticated account.

What if I want to check whether an answer is really grounded, not guessed?

Do exactly what Section 5 demonstrates: ingest something distinctive only you would know, then ask about it. A correct, cited answer is the real proof.