RAG · AUTHORIZATION

Authorization for **Retrieval-Augmented Generation**

Augment prompts with authorized data.

Evan Corkrean

Solutions Engineer · AuthZed

$ whoami

Evan Corkrean

ROLE	Solutions Engineer at AuthZed
FOCUS	Architecting authorization for RAG, agents & applications
WORKED WITH	OpenAI and large investment banks

QUICK REFRESHER

An AI framework that fetches **relevant, authoritative facts** from an external knowledge base **before** the model answers.

WHY IT MATTERS

Same question. **Two answers.**

WITHOUT RAG

What was AuthZed's revenue in Q1 2026?

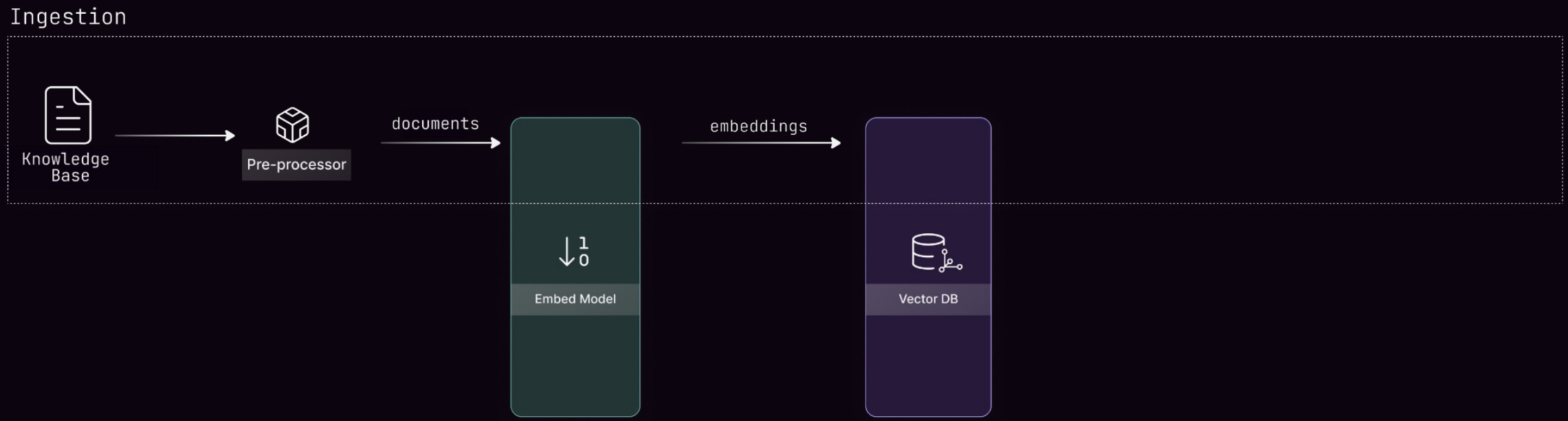
"I don't have that information."

WITH RAG

What was AuthZed's revenue in Q1 2026?

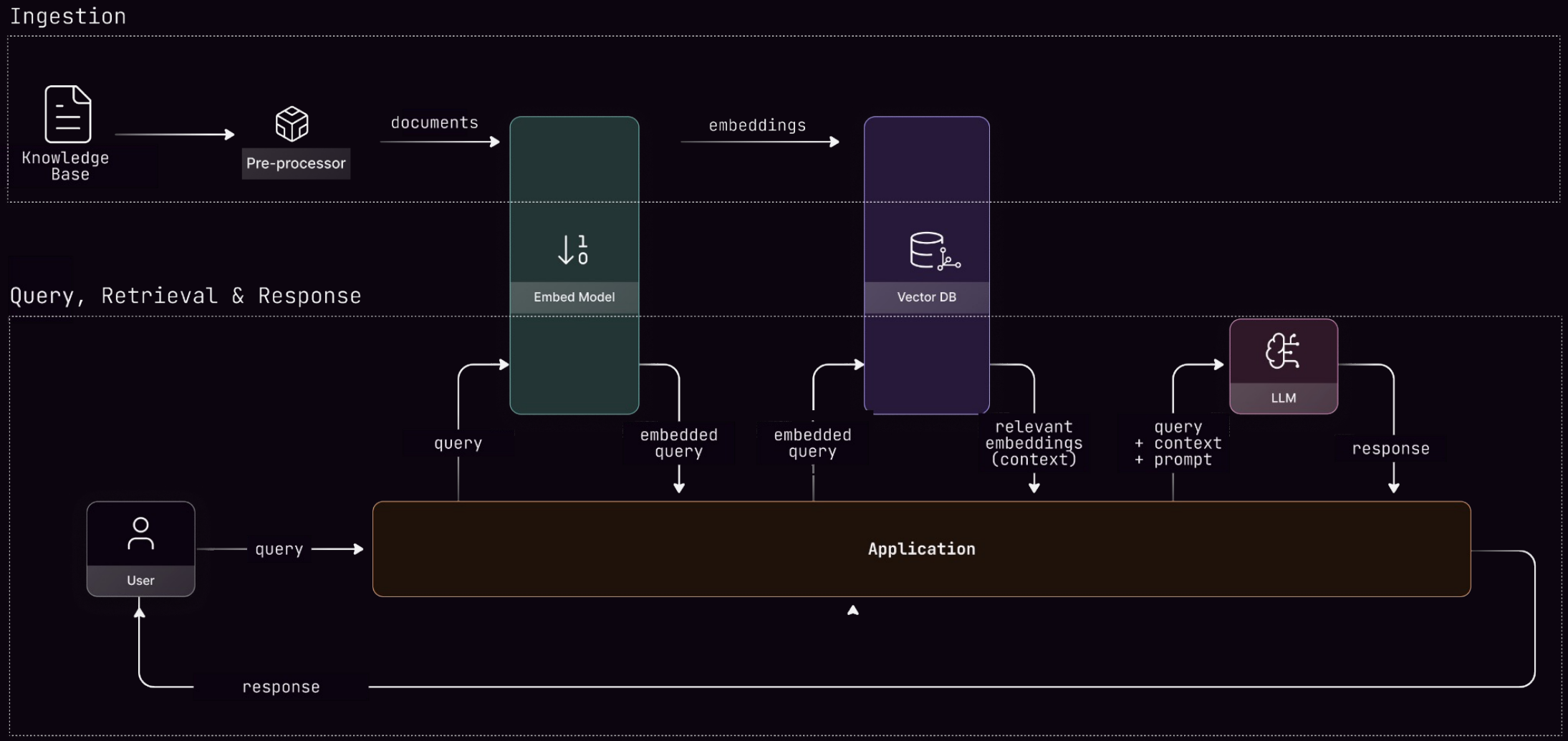
"It was $100 billion." — pulled from your data.

Ingestion



Documents are chunked, embedded, and stored as vectors — copied out of their source system.

Query, retrieval & **response**



THE PROBLEM

Bolting RAG on creates **a two-fold problem.**

01

A security vulnerability

02

Re-implementing authorization

PROBLEM 01

A **security vulnerability**

Data leaves its source. It's copied out and stored inside a vector DB.

Your API-layer authorization no longer applies. Retrieval queries never touch it.

So teams settle for coarse-grained compromises — and data leaks across the boundary.

PROBLEM 02

Re-implementing **authorization**

Fine-grained control means rebuilding your authz logic as filters on every vector-DB query.

TWO SOURCES OF TRUTH

Logic now lives in two places. Every change has to be mirrored — or the two drift apart.

TRAPPED IN METADATA

You're limited by embedding metadata. Inheritance like nested folders is hard or impossible.

THE SOLUTION

Teams today either compromise on RAG authorization — or reinvent it.

Use **one centralized authorization system** — for your app **and** your RAG pipeline.

OPEN-SOURCE OPTIONS

What's already out there?

OpenFGA

CNCF · Incubating

Both are inspired by **Google Zanzibar** — the service behind Google Drive. Their core check API answers one question:

```
does user X have permission Y on resource Z? → yes / no
```

REBAC

Relationship-Based **Access Control**

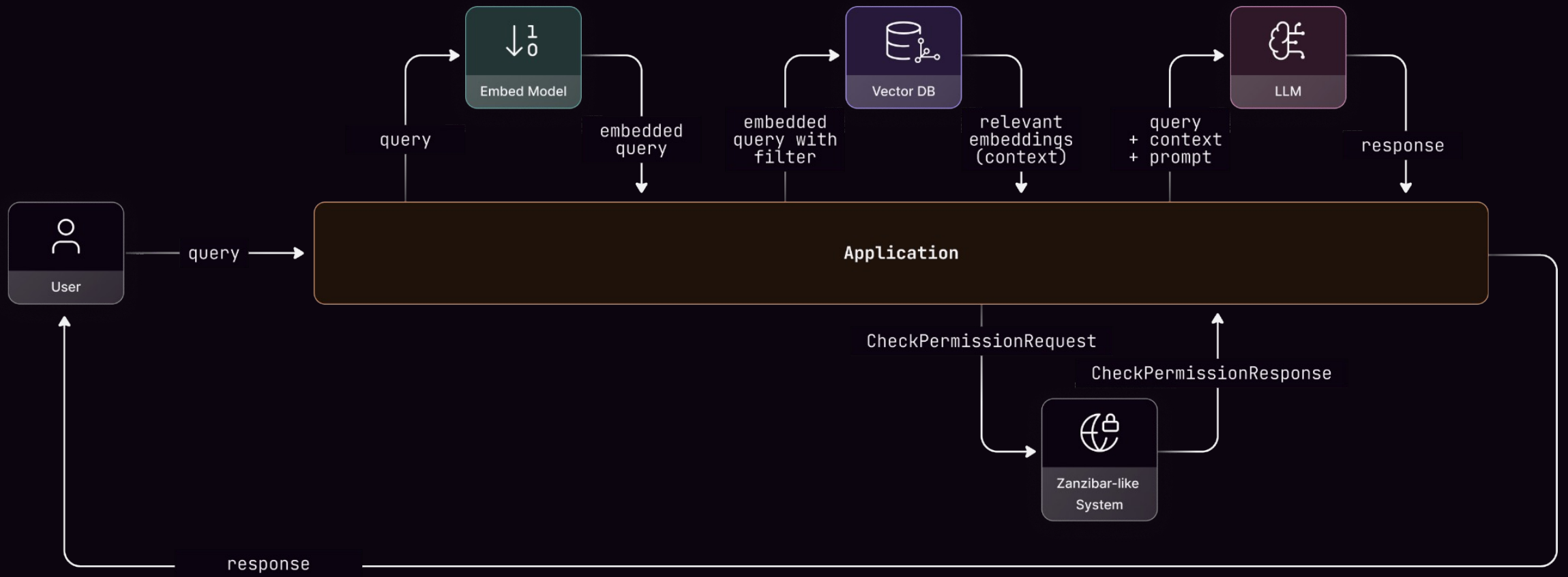
Decisions are made by traversing a graph of relationships — flexible enough to model RBAC, ABAC, and recursion like nested folders, all in one place.

```
// worth a whole talk of its own — go read up on it
```

Preserve provenance **from source to embedding**

Most vector DBs store metadata alongside each embedding. Tag every vector with the source it came from.

ReBAC as a **post-filter**



Each candidate is checked against a Zanzibar-like system before it ever reaches the LLM.

Filter results against **live permissions**

01

Use bulk permission-check APIs to validate many candidates at once.

02

Stop early — once you hit the desired count or a minimum similarity score.

03

Add cheap coarse pre-filters to shrink the candidate set first.

04

Pre-filter by org whenever it's trivial to do so.

Preserve provenance at ingestion. Post-filter against centralized ReBAC at query time.

Only authorized data reaches the model.

Thank you.

Evan Corkrean · linkedin.com/in/corkrean