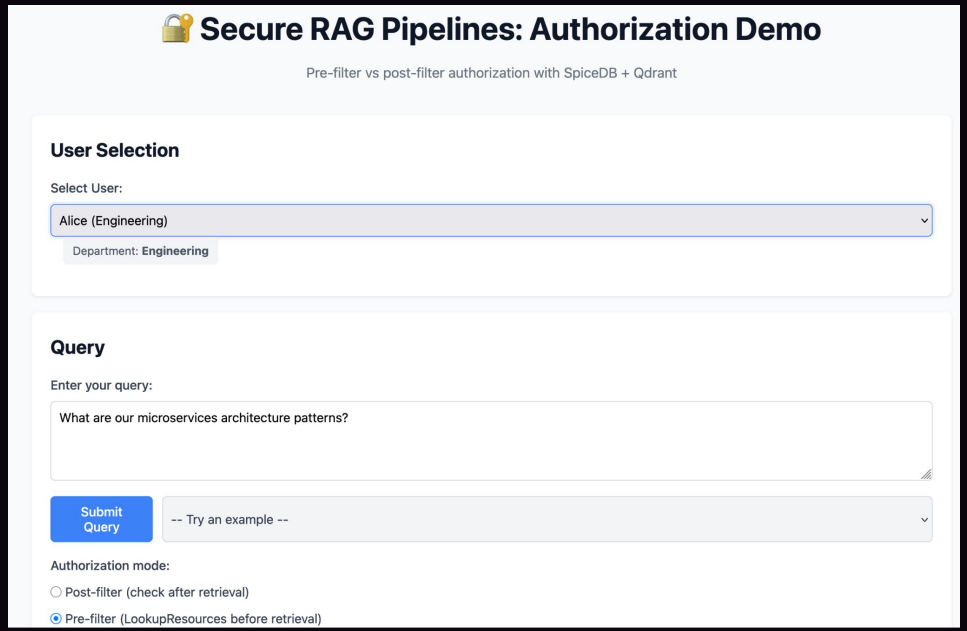
Workshop

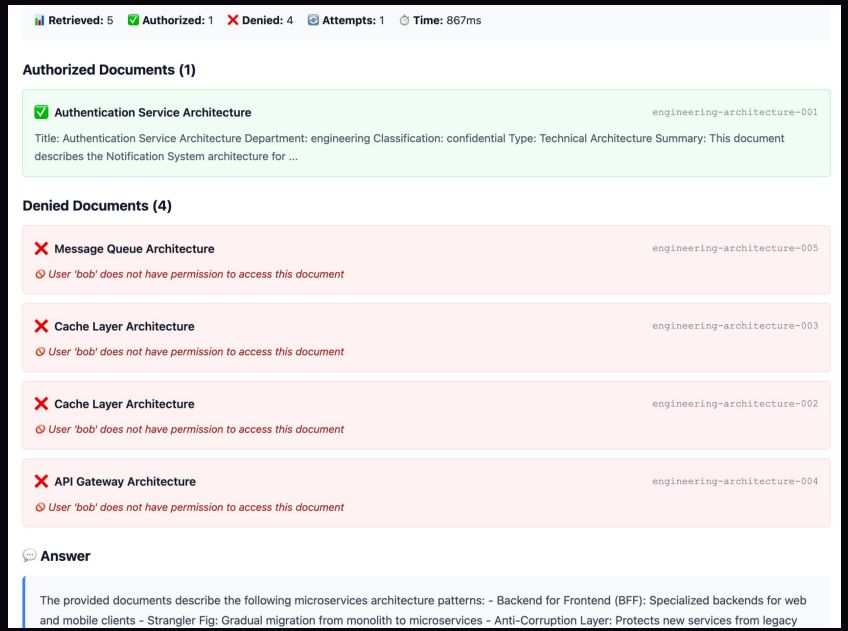
Securing RAG Pipelines with Fine Grained Authorization

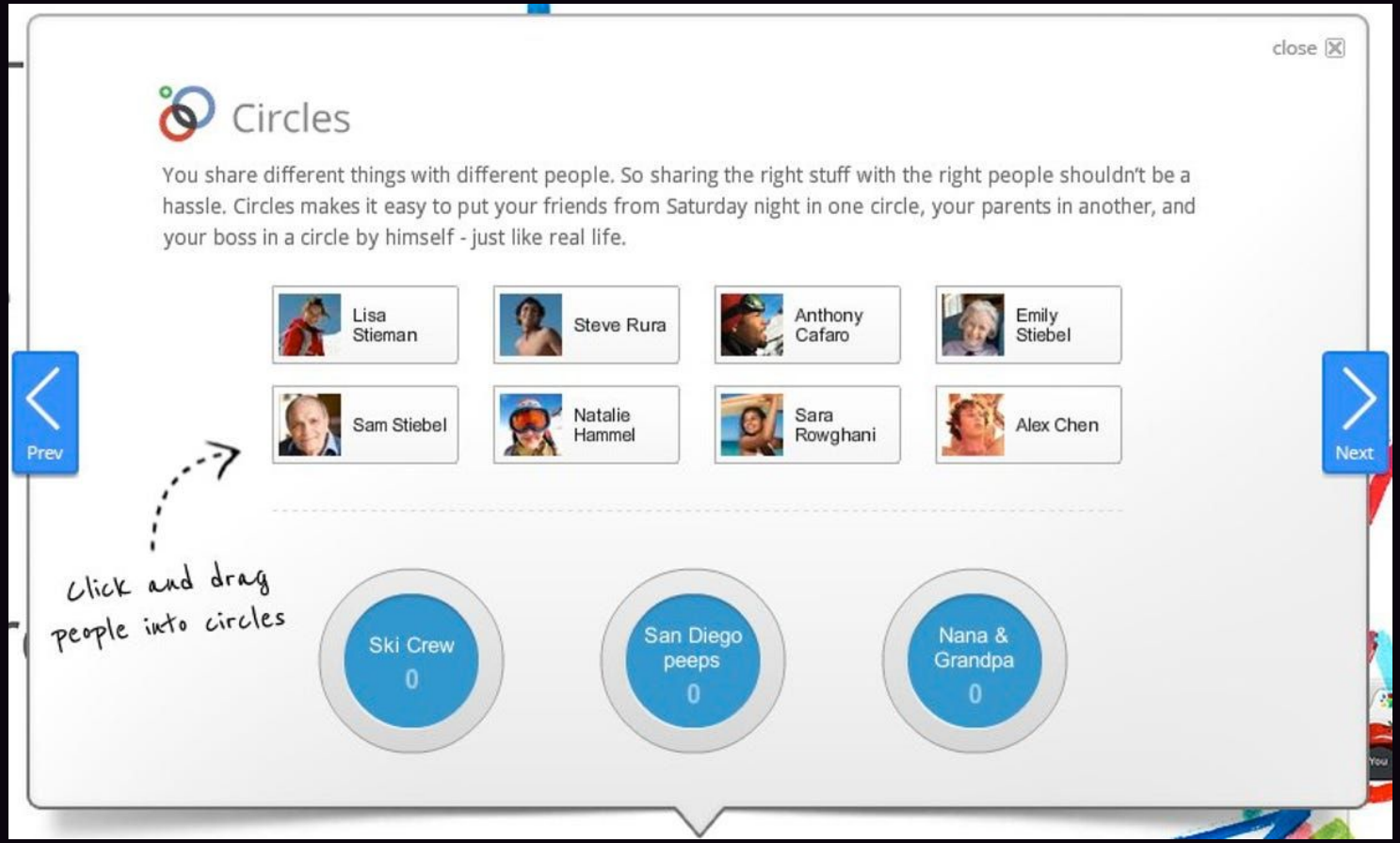
Pre-Requisites:

- Docker installed (or) GitHub Codespaces
- Python 3.11+
- A LLM API key

What we'll build






close
Circles
You share different things with different people. So sharing the right stuff with the right people shouldn't be a hassle. Circles makes it easy to put your friends from Saturday night in one circle, your parents in another, and your boss in a circle by himself - just like real life.
Lisa Stieman
Steve Rura
Anthony Cafaro
Emily Stiebel
Sam Stiebel
Natalie Hammel
Sara Rowghani
Alex Chen
Prev
Next
Click and drag people into circles
Ski Crew 0
San Diego peeps 0
Nana & Grandpa 0

Google Sheets

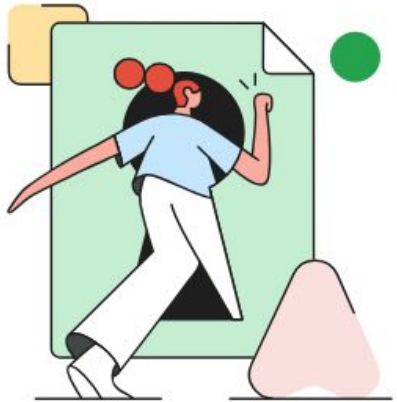
You need access

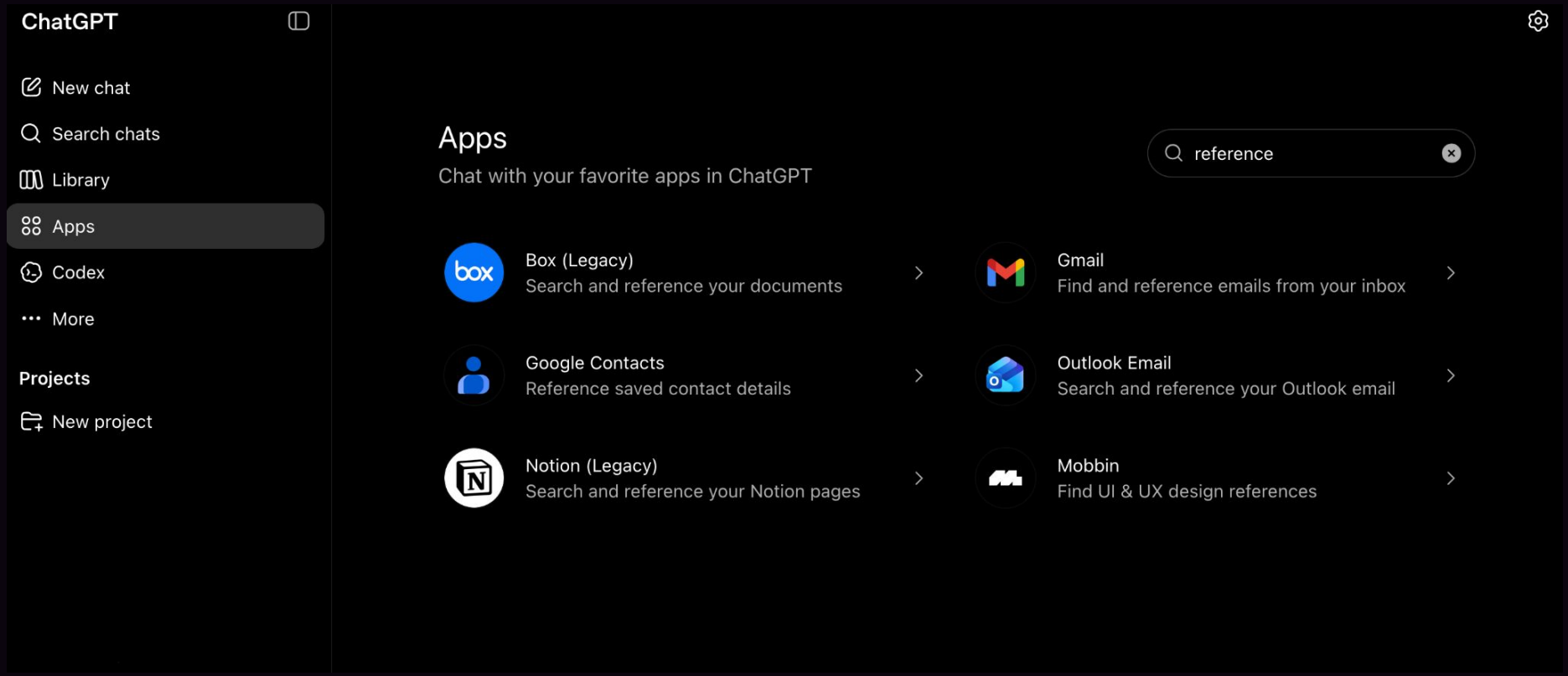
Request access, or switch to an account with access.
Learn more

- ○ Viewer
- ○ Commenter
- ◉ Editor

Message (optional)

Request access

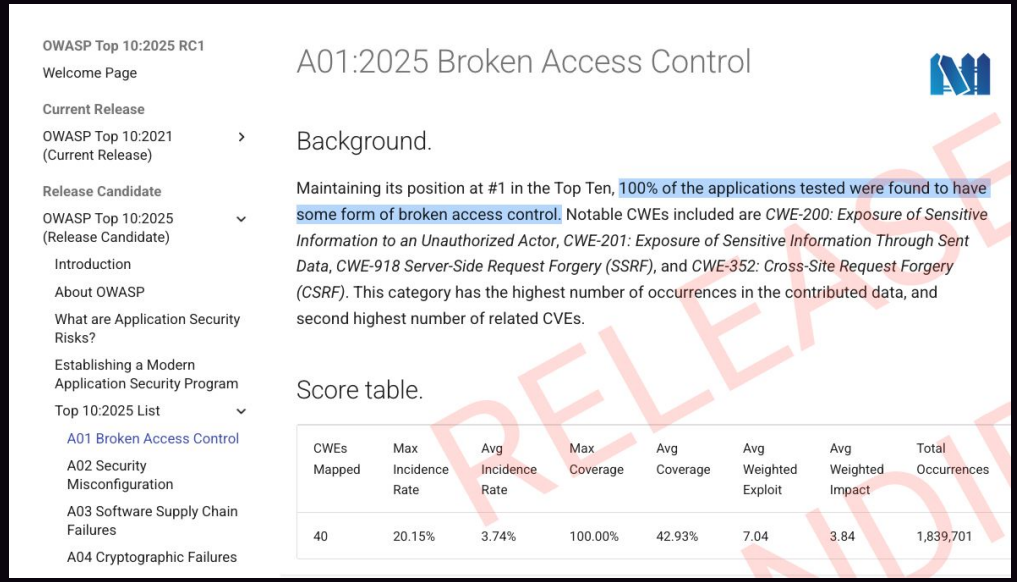



ChatGPT
New chat
Search chats
Library
Apps
Codex
More
Projects
New project
Apps
Chat with your favorite apps in ChatGPT
reference
Box (Legacy)
Search and reference your documents
Gmail
Find and reference emails from your inbox
Google Contacts
Reference saved contact details
Outlook Email
Search and reference your Outlook email
Notion (Legacy)
Search and reference your Notion pages
Mobbin
Find UI & UX design references

Risks facing LLM and GenAI applications

OWASP Top Ten
Web Apps Security Risks

- #1 Broken Access Control (2021 and 2025)
- 100% of apps tested had broken access control issues

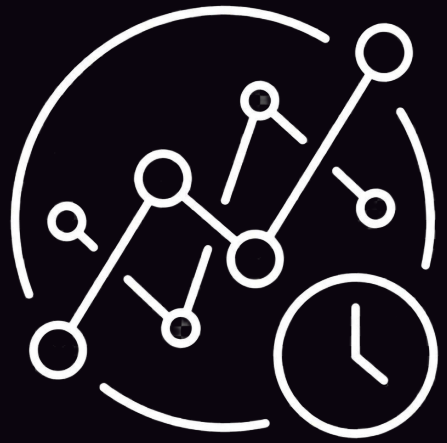


owasp.org/Top10/2025/

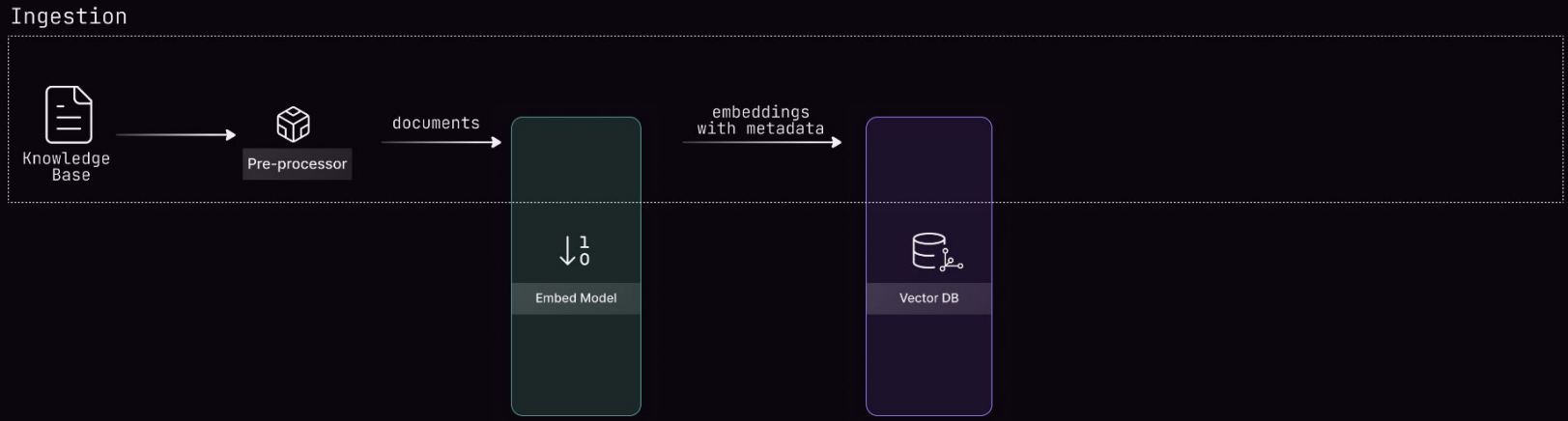
Ambient Context

- N users x M agents x O actions
- Evolving graph of relationships, states, and entities (users, agents, resources)

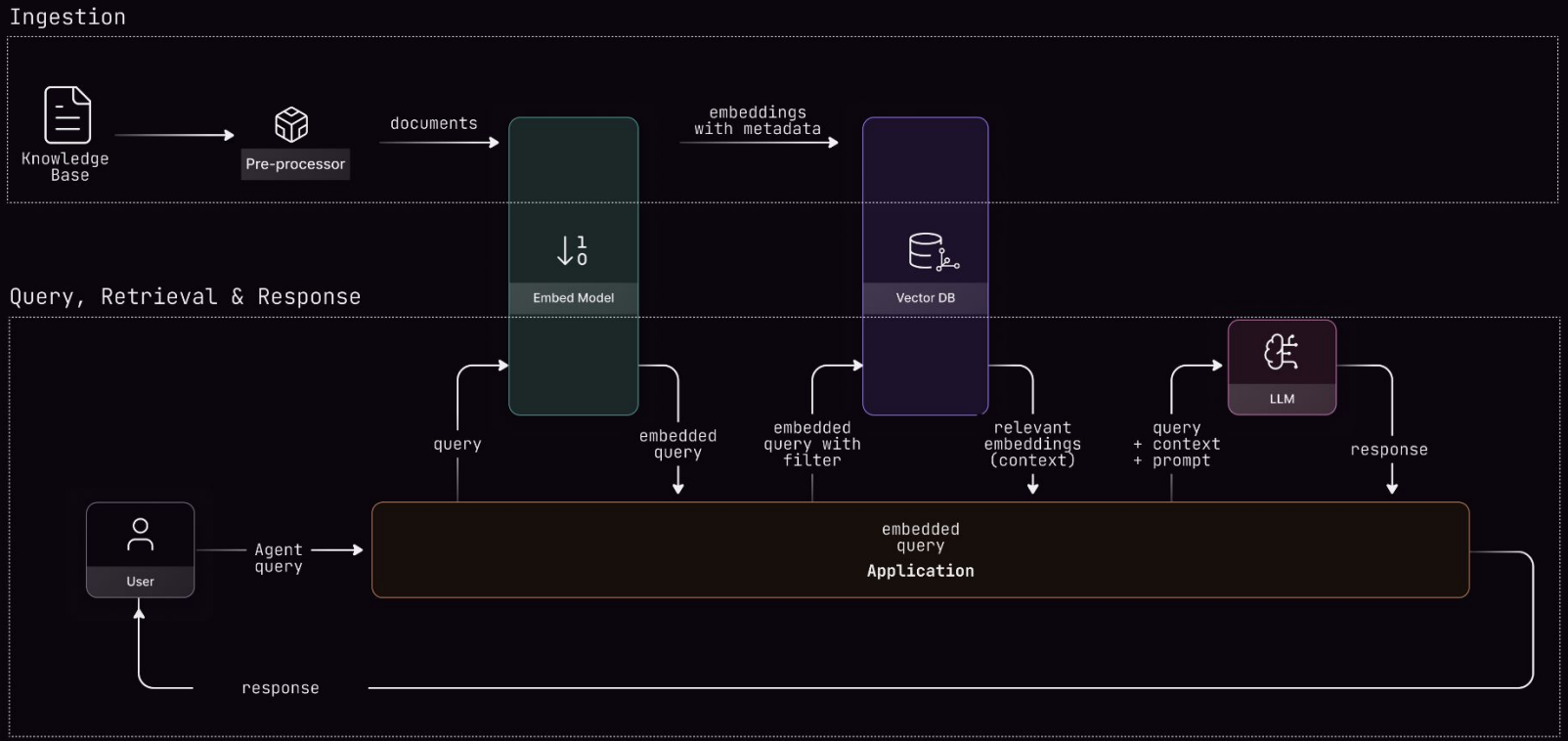
- Not fully contained in tokens (JWTs), ACLs, or request payloads.
- Too complex for coarse-grained techniques such as RBAC



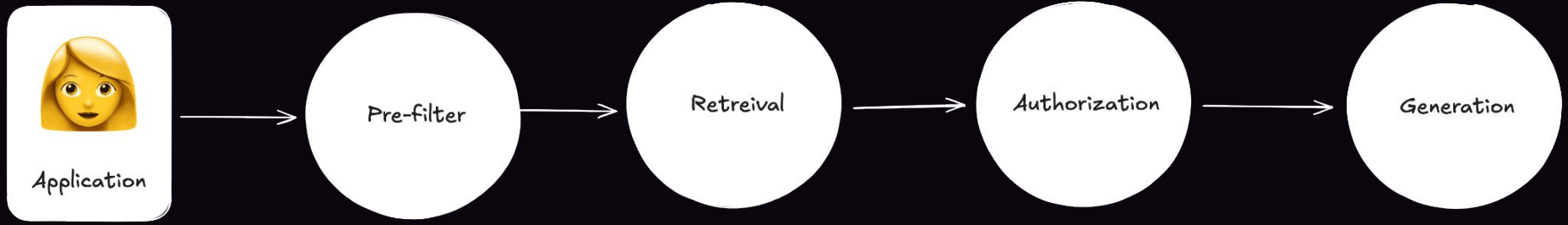
Components of a RAG Pipeline



Components of a RAG Pipeline

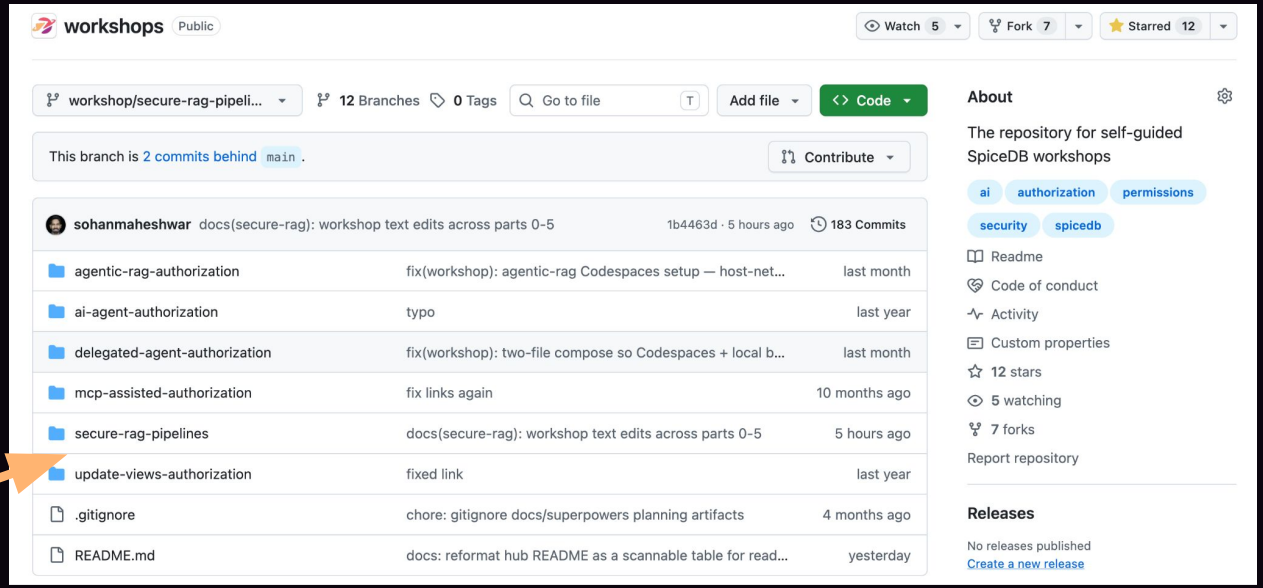


Part 1 - Build an Agentic RAG System



Part 0 & 1 - Secure RAG Pipelines

- github.com/authzed/workshops
  → secure-rag-pipelines
  → Part 0 - Setup, Part 1 - Run the RAG



LangGraph Nodes (stateful)

- Agent orchestration
- Multi-step agentic workflows
- Authorization cannot be skipped

```python
from langgraph.graph import StateGraph, END
from langchain_spicedb import create_auth_node, RAGAuthState

graph = StateGraph(RAGAuthState)

# Add nodes
graph.add_node("retrieve", retrieve_node)
graph.add_node("authorize", create_auth_node(
    spicedb_endpoint="localhost:50051",
    spicedb_token="sometoken",
    resource_type="article",
    resource_id_key="article_id",
))
graph.add_node("generate", generate_node)

# Wire it up
graph.set_entry_point("retrieve")
graph.add_edge("retrieve", "authorize")
graph.add_edge("authorize", "generate")
graph.add_edge("generate", END)

# Run
app = graph.compile()
result = await app.ainvoke({
    "question": "What is SpiceDB?",
    "subject_id": "alice",
})
```

Primer on Authorization

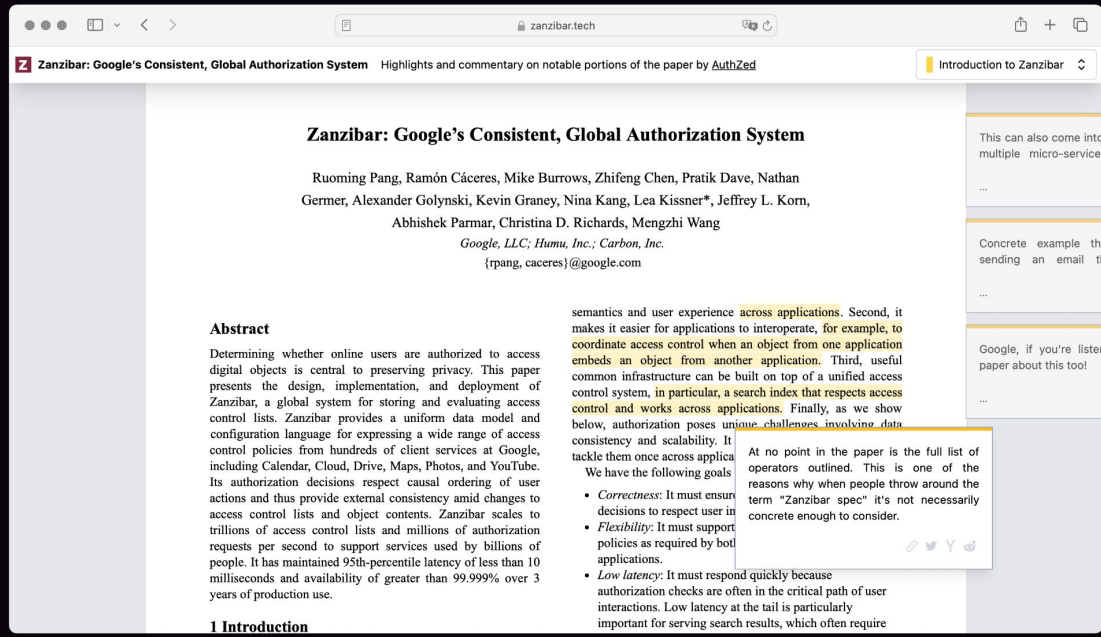
Access Control Types

- Access Control Lists
- Role-Based Access Control
- Attribute-Based Access Control



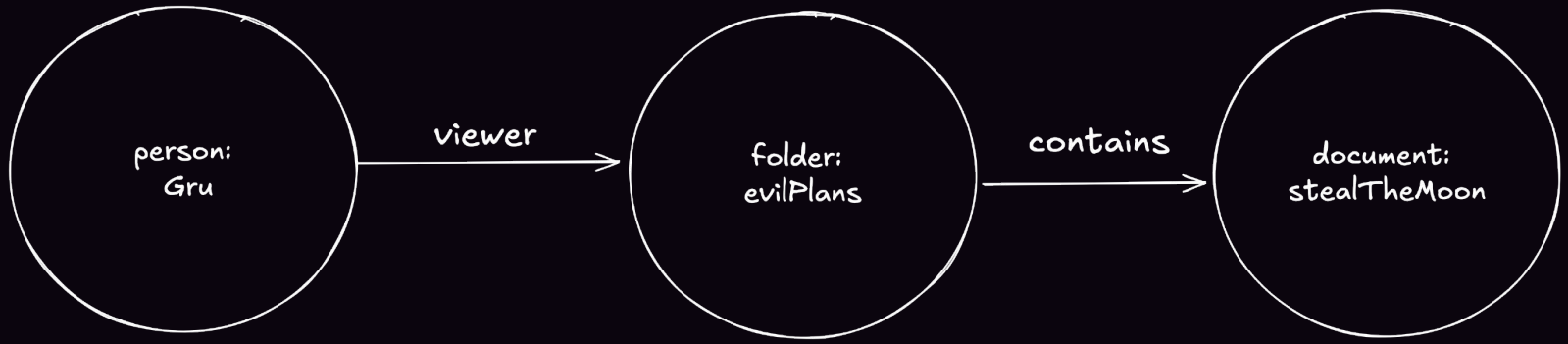
Google Zanzibar

- Globally distributed authorization system
- >10 million client queries per second
- From the developer's perspective - it's an API
- ReBAC



www.zanzibar.tech

Relationship Based Access Control



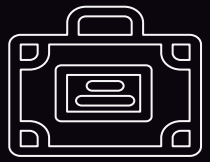
User

- Represents a natural person or service account

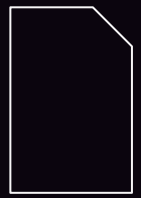
👩 **userid: 10957135**

Object

- Represents a non-user entity
- Unique identifier prefixed with an object type



`folder:`
`evilPlans`



`doc:`
`secret-doc-id`

Relation

- The way one object can be related to another object or user
- Examples
  - `member` of a group
  - `editor` of a document
  - `uploader` of a video

Relation Tuple

- Datum which expresses a single relationship

`document:123#owner@user:3`



User 3 is an **owner** of a **document** with identifier 123

Relation Tuple

- Datum which expresses a single relationship
- Objects can relate to other objects
- Objects can relate to other objects' relations

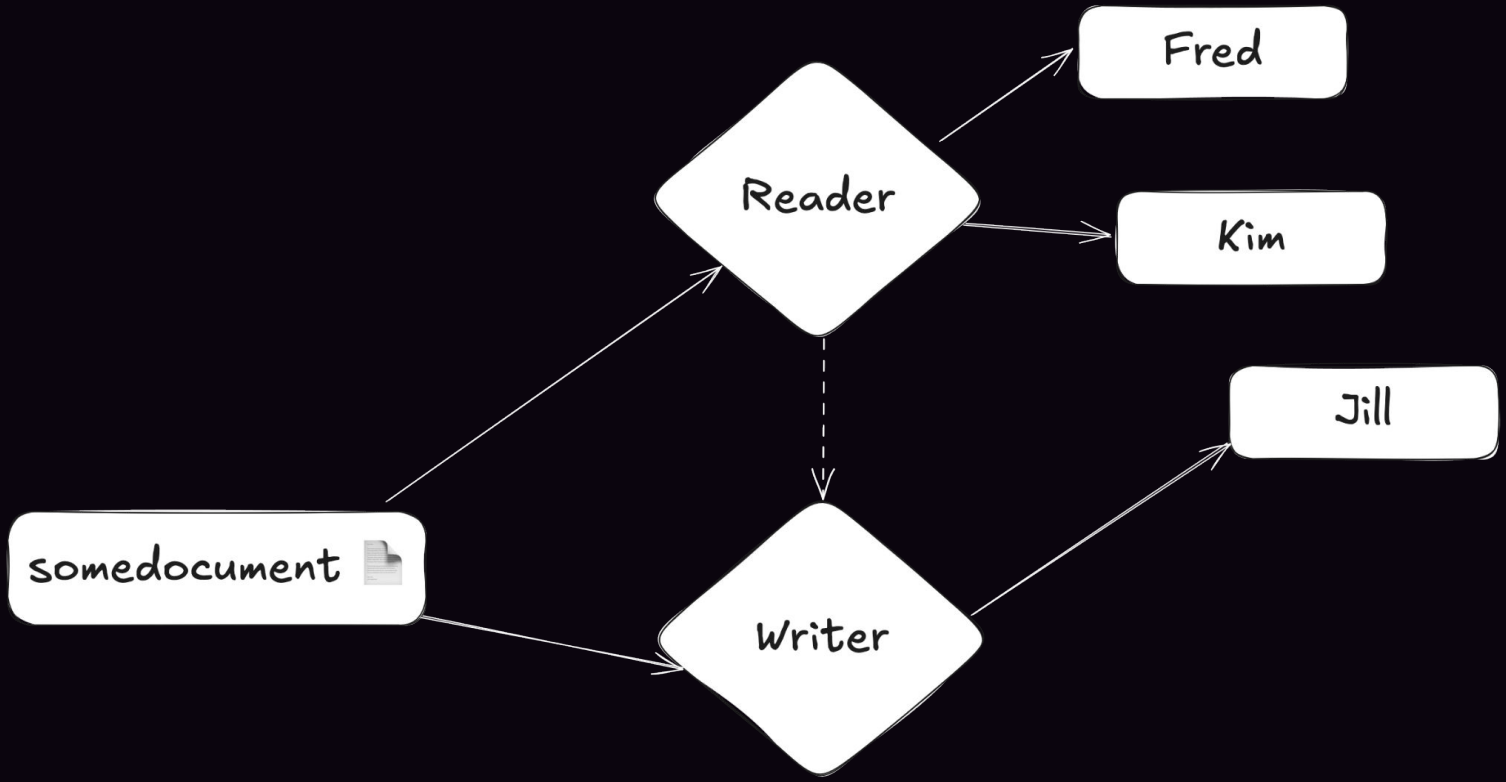
`group:engineering#member@group:security`



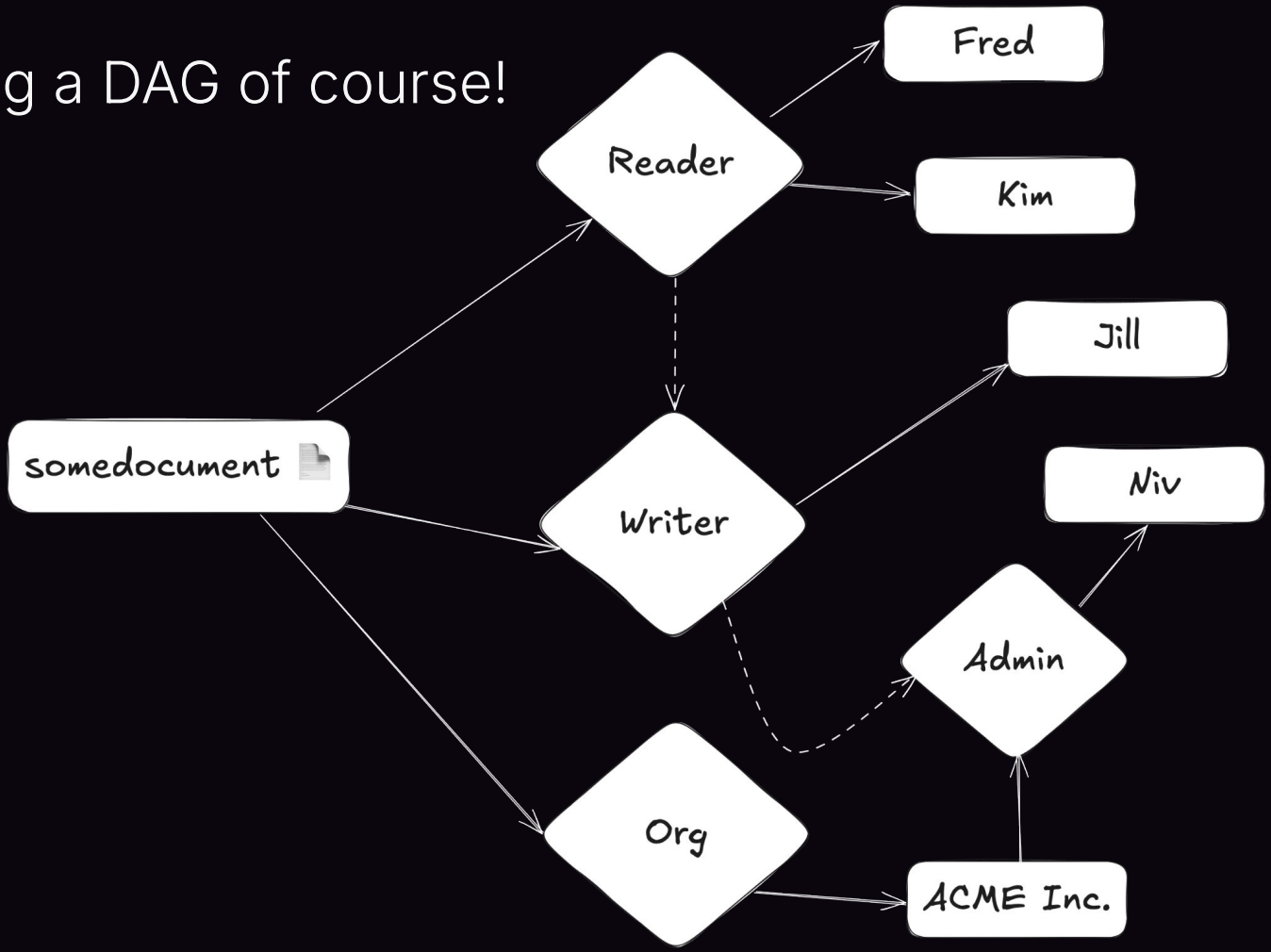
group security is a direct **member** of the **group engineering**

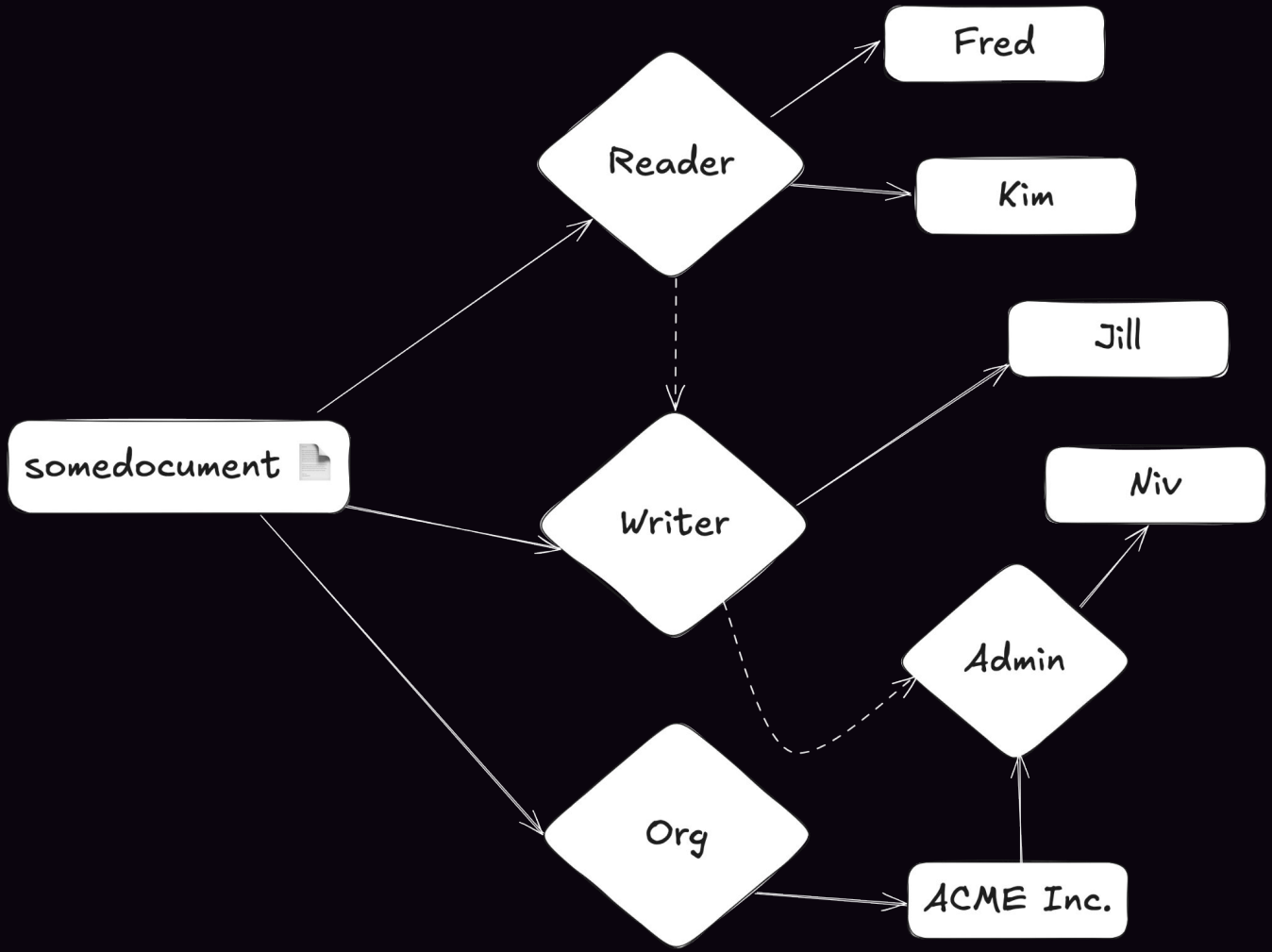
What are we really doing here?

Building a DAG of course!



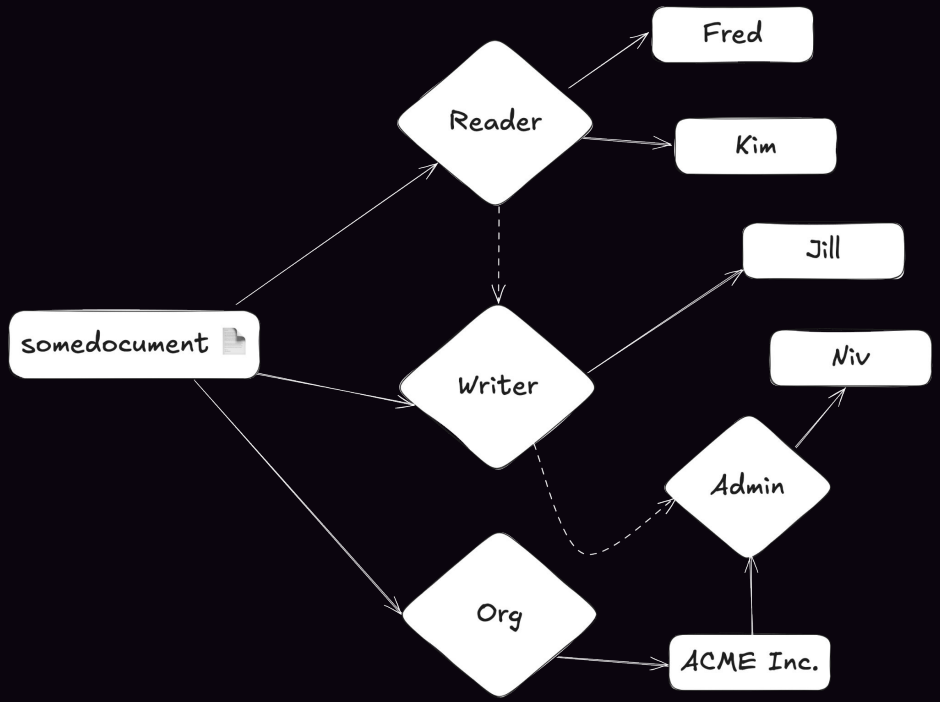
Building a DAG of course!




Fred
Reader
Kim
Jill
somedocument
Writer
Niv
Admin
Org
ACME Inc.

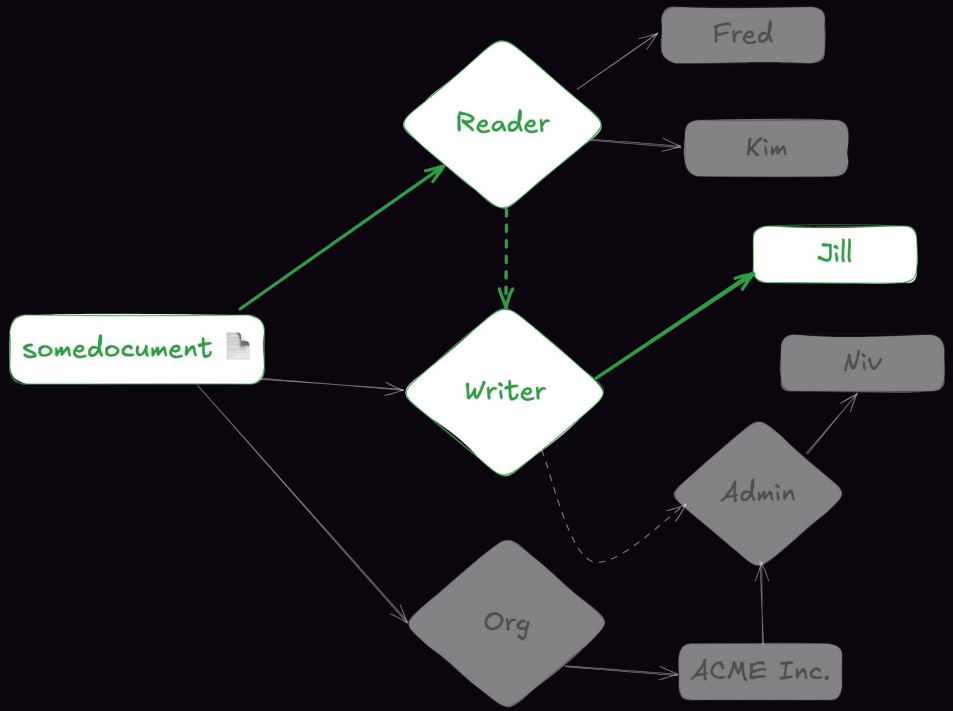
Check graph traversal example

```
check("document:somedocument", "reader", jill)
```



Check graph traversal example

```
check("document:somedocument", "reader", jill)
```



Performance strategies

- Break down into parallel subproblems
- Cache every problem and subproblem
- Reuse cache as much as possible
- De-duplicate requests
- Batch reads to datastore



Where do Zanzibar-like systems shine?

- Low-latency
- High-throughput authorization checks
- Hierarchical permission models
- Ambient Context



SpiceDB

- Open source implementation of Zanzibar
- Widespread project adoption
- Contributors from Fortune 500 companies

github.com/authzed/spicedb

7.1K 🌟s & counting!













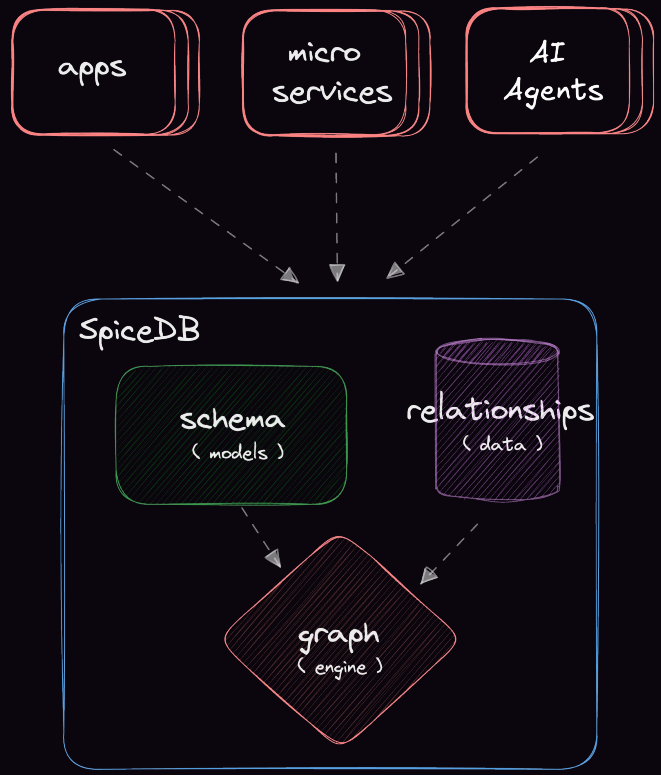




What *IS* SpiceDB

- Highly parallel OS graph database optimized for authorization queries

- gRPC & HTTP API service written in Go
  - Can <subject> take <action> on <resource>?
  - List all <subjects> that can take <action> on <resource>
  - List all <resources> <subject> can take <action> on



1. Schema

- Defines
  - Types of objects found
  - How those objects relate to one another
  - Permissions that can be computed off of those relations.

A user is a writer of a document
A writer can view and edit a doc

```
definition user {}

definition document {
    relation writer: user
    relation reader: user

    permission edit = writer
    permission view = reader + edit
}
```

Add an AI Agent

An AI Agent can only `view` a document



```
1  definition user {}
2
3  definition ai_agent {
4  }
5
6  definition document {
7
8      relation writer: user
9      relation reader: user | ai_agent
10
11     permission edit = writer
12     permission view = reader + edit
13 }
```

AI Agent acts on behalf of a user

An AI Agent can view a document if a user can view a document



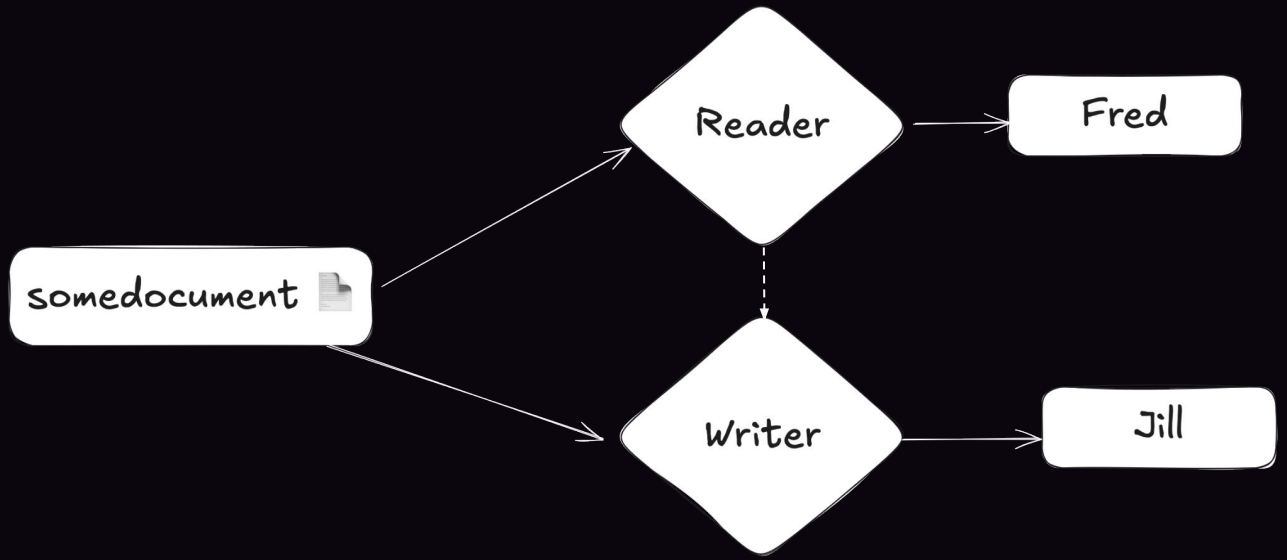
```
definition user {
    relation agent: ai_agent
    permission delegates = agent
}

definition ai_agent {}

definition document {
    relation writer: user
    relation reader: user
    permission edit = writer
    permission view = reader + edit +
                      reader->delegates + writer->delegates
}
```

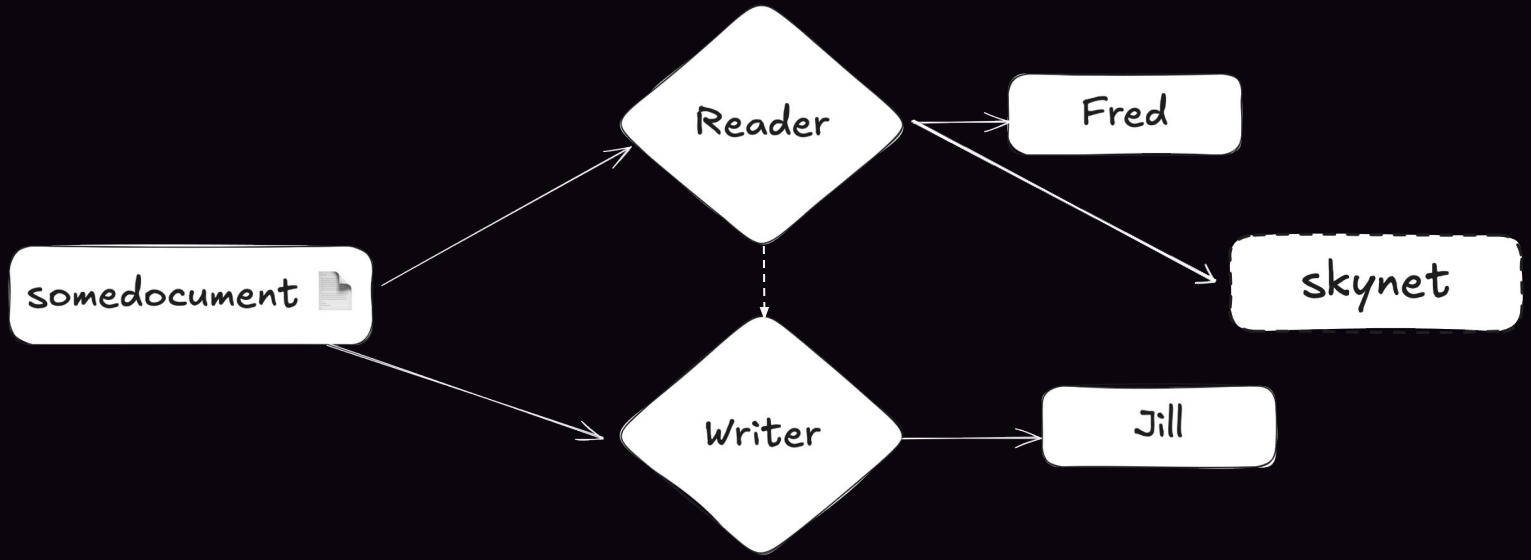
2. Relationships

- Relationships bind together a Subject and a Resource via a Relation.
- A functioning Permissions System is the combination of Schema and Relationships




```
document:somedocument#writer@user:jill
```

2. Relationships



`document:somedocument#reader@agent:skynet`

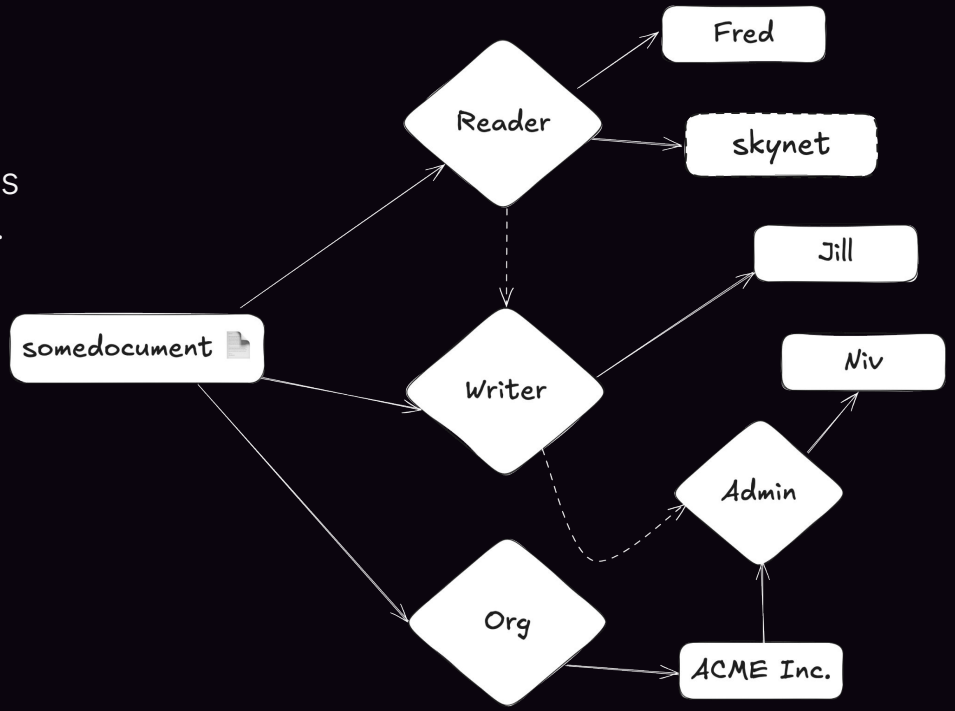
Add Relationships

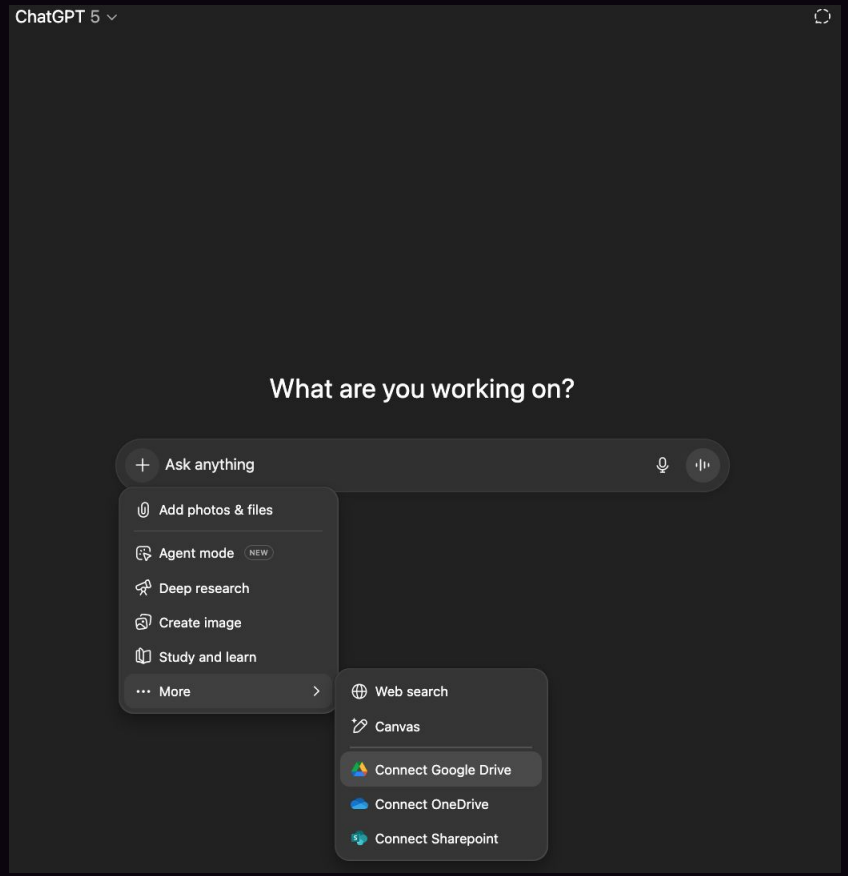
```
document:somedoc#writer@user:fred
user:fred#agent@ai_agent:skynet
```

The AI Agent gets Fred's view access automatically

3. Permissions

- CheckPermission
  Check whether a subject has a permission or is a member of a relation, on a specific resource.





OpenAI

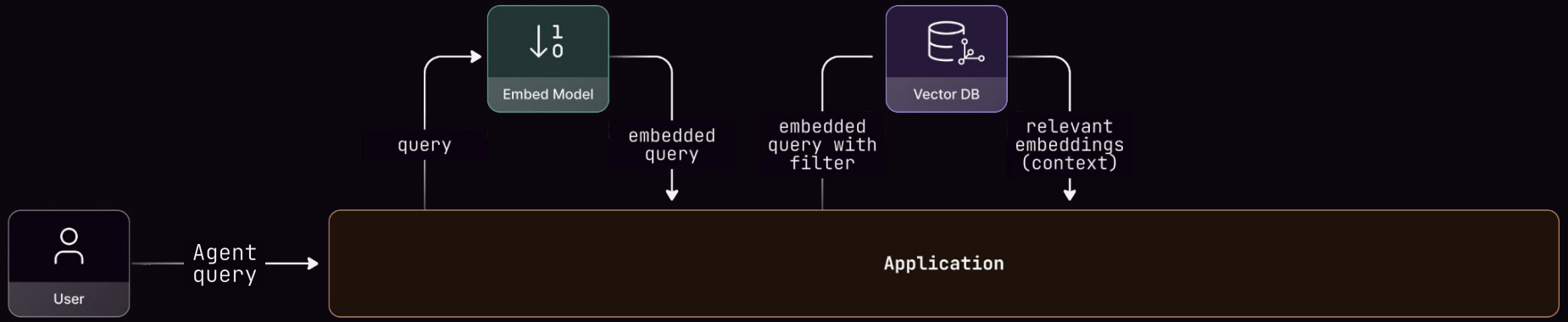
OpenAI securely connects enterprise knowledge with **ChatGPT** by using AuthZed

Learn how OpenAI built connectors to process and search billions of documents for over 5 million business users while respecting existing data permissions using AuthZed's authorization infrastructure.

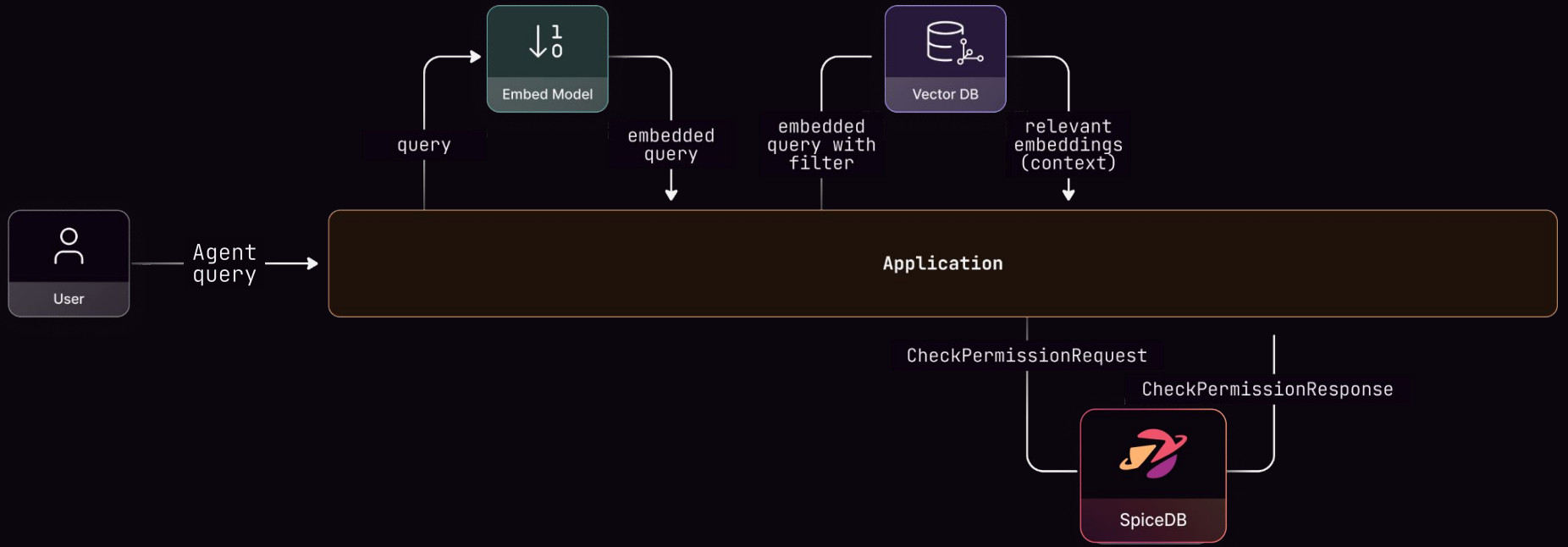
37 Billion+

Documents with fine-grained access controls

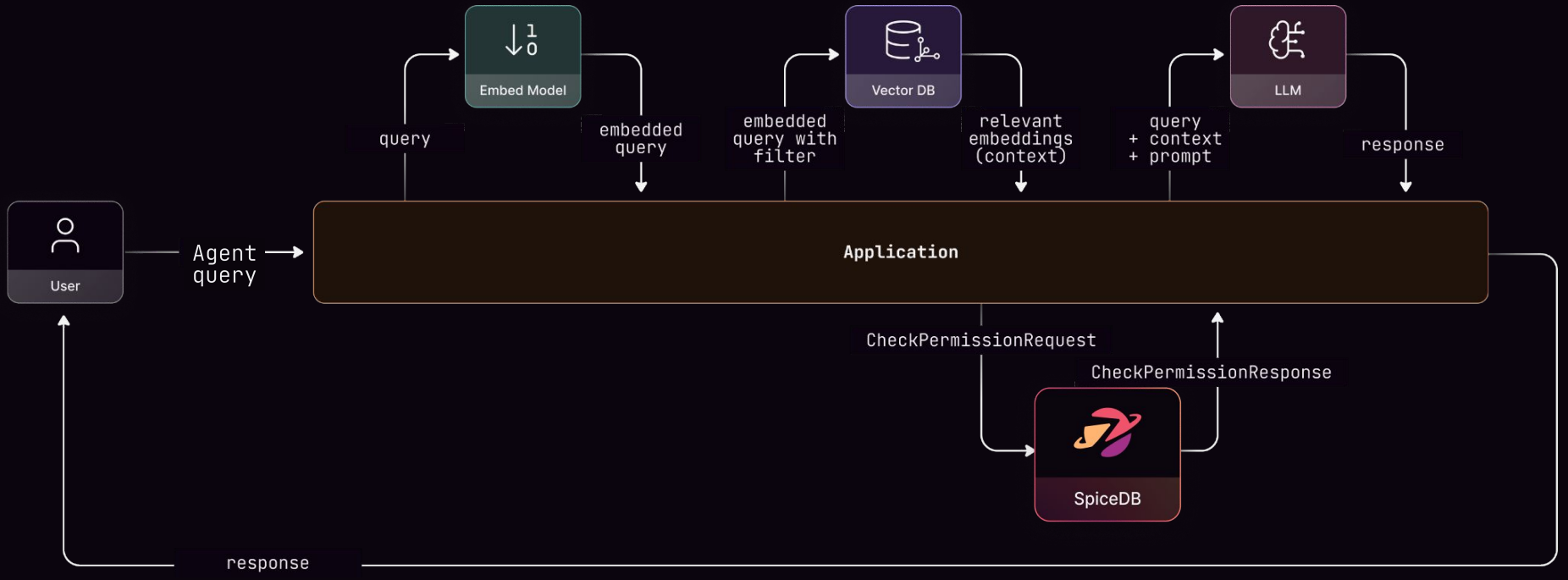
Post-Filter AuthZ



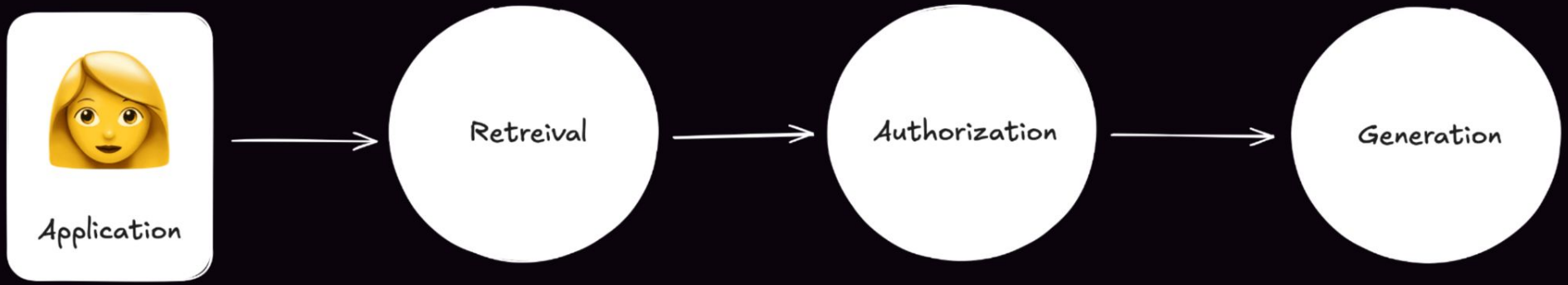
Post-Filter AuthZ



Post-Filter AuthZ

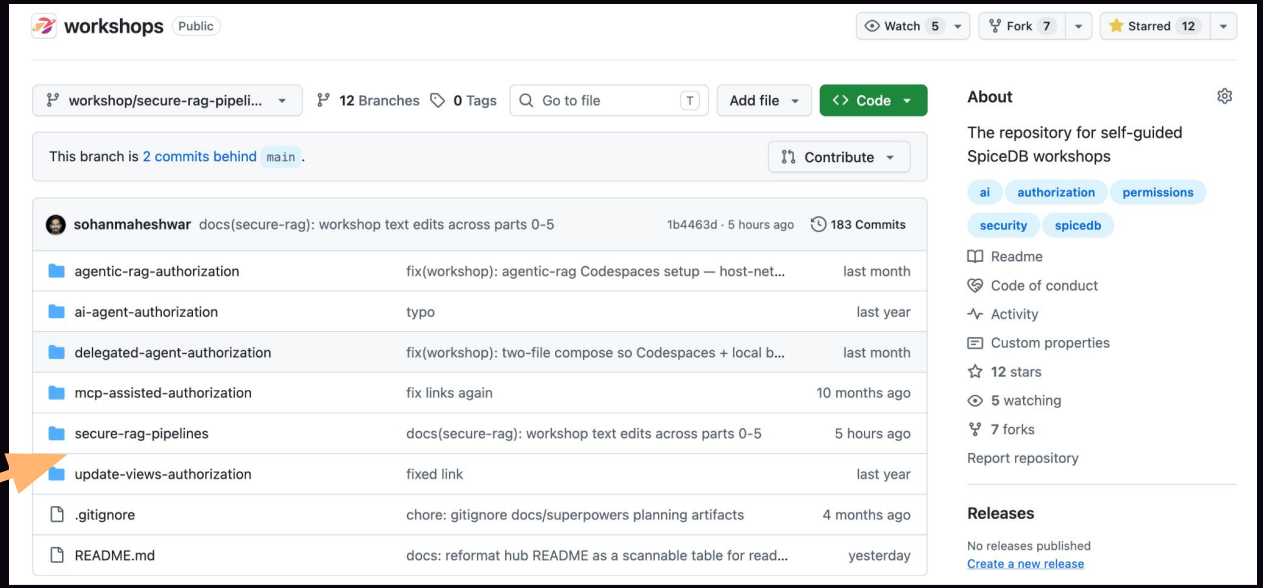


LangGraph Workflow



Part 2 - Secure it with post-filter

- github.com/authzed/workshops
  → secure-rag-pipelines
  → Part 2 - Secure it with post-filter

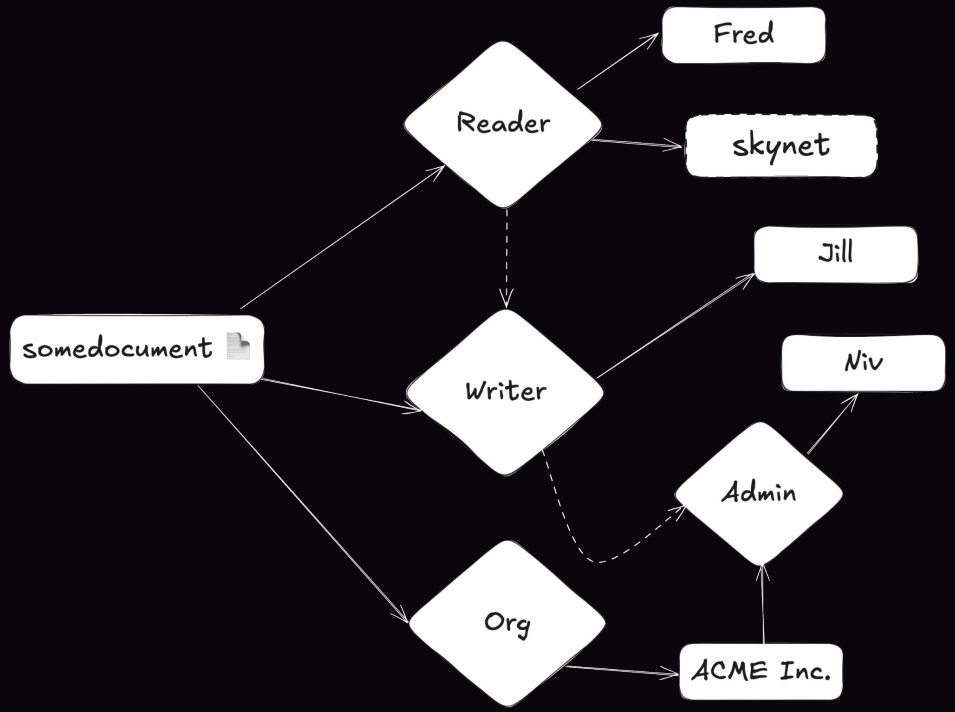


BREAK

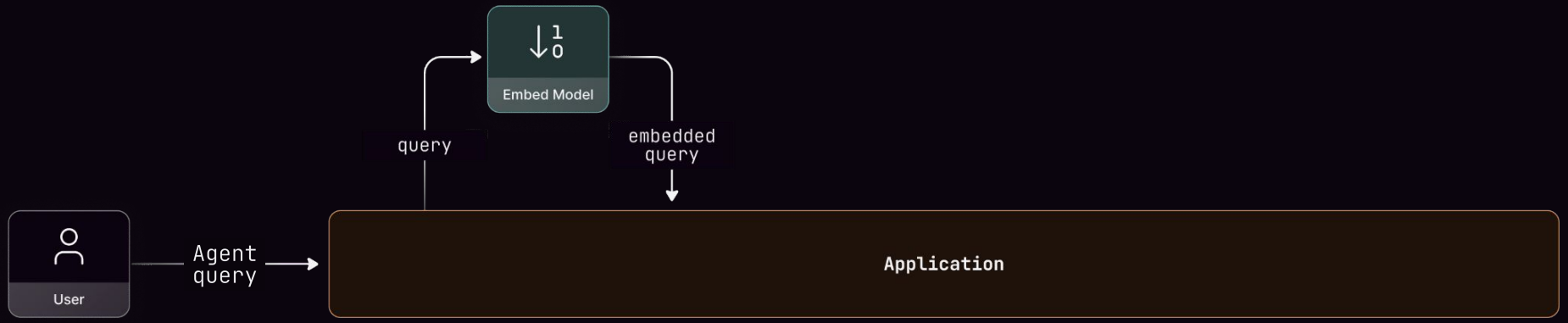
3. Permissions

- CheckPermission
  Check whether a subject has a permission or is a member of a relation, on a specific resource.

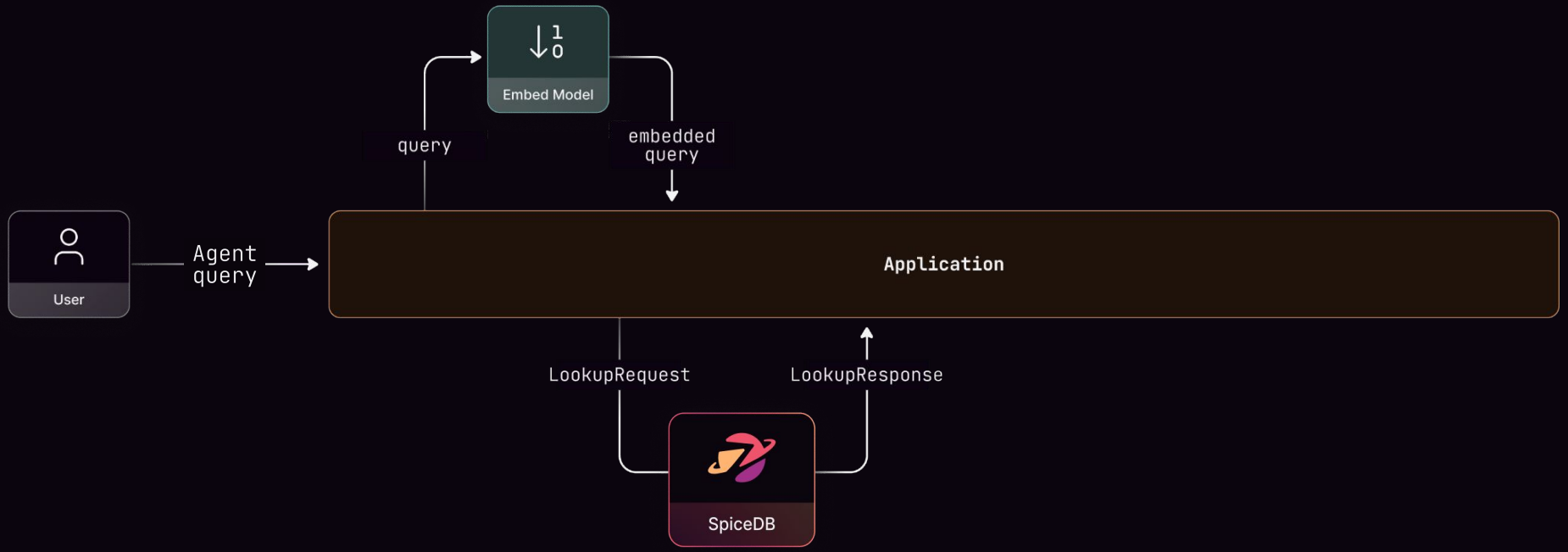
- LookupResources
  Returns all the resources of a given type that a subject can access



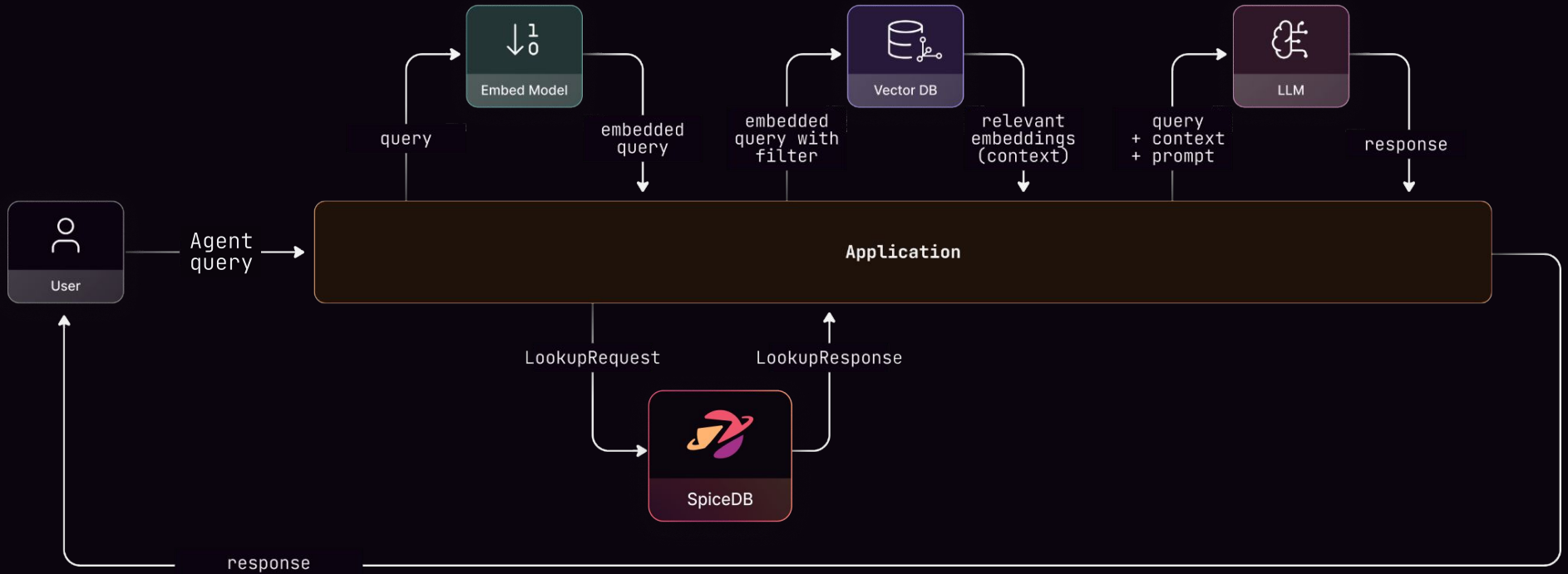
Pre-Filter AuthZ



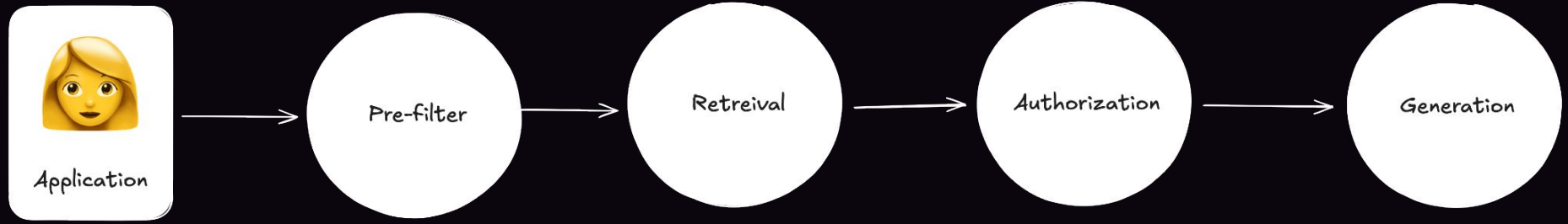
Pre-Filter AuthZ



Pre-Filter AuthZ

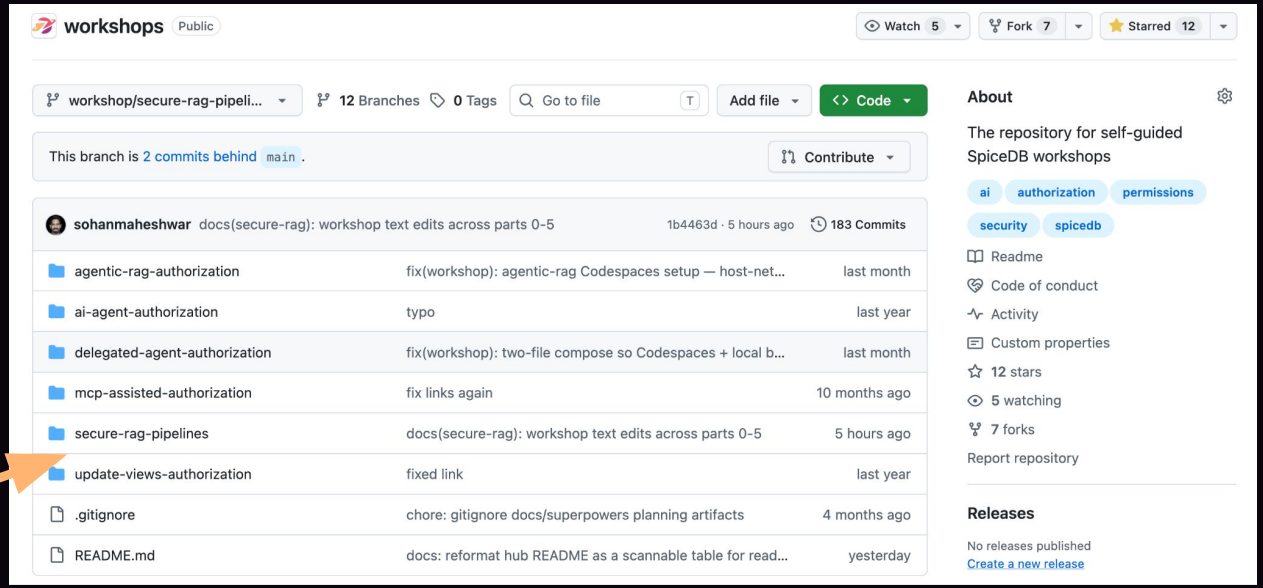


With the Pre-Filter node



Part 3 - Pre-filter

- github.com/authzed/workshops
  → secure-rag-pipelines
  → Part 3 - Pre-filter

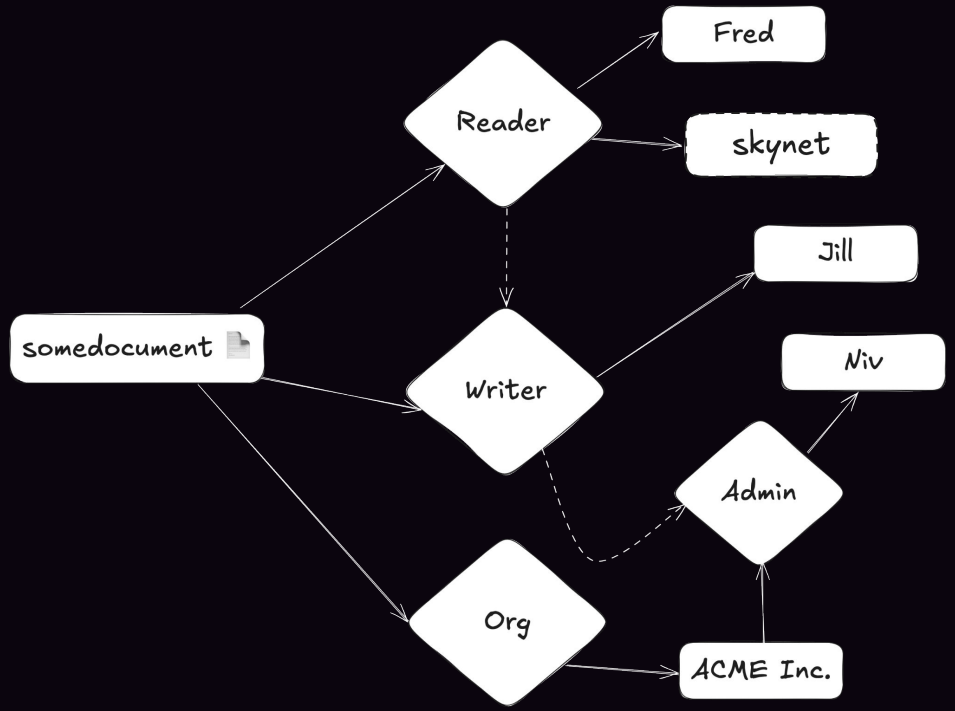


3. Permissions

- CheckPermission
  Check whether a subject has a permission or is a member of a relation, on a specific resource.

- LookupResources
  Returns all the resources of a given type that a subject can access

- LookupSubjects
  Returns all the subjects of a given type that have access



Time-Bound Access

- “Give staging access for the duration of the incident”

- Cron jobs don’t work
  - There's a window where a "gone" grant still works.
- RBAC & tokens can’t express this cleanly
  - Need another microservice to do this
  - Maintaining another system

Expiring Relationships

- Grant a user access to a resource for a limited time
- The relationship is not deleted immediately
  - It is subject to garbage collection

```
use expiration

definition user {
}

definition environment {
    relation direct_deployer: user
    relation agent_deployer: agent with expiration
    ...
 }
```

Add it to the relationship

```
from authz import expiry_from_now

rel("environment", "staging", "agent_deployer", "agent", AGENT_ID,
    expires_at=expiry_from_now(window_minutes)),
```

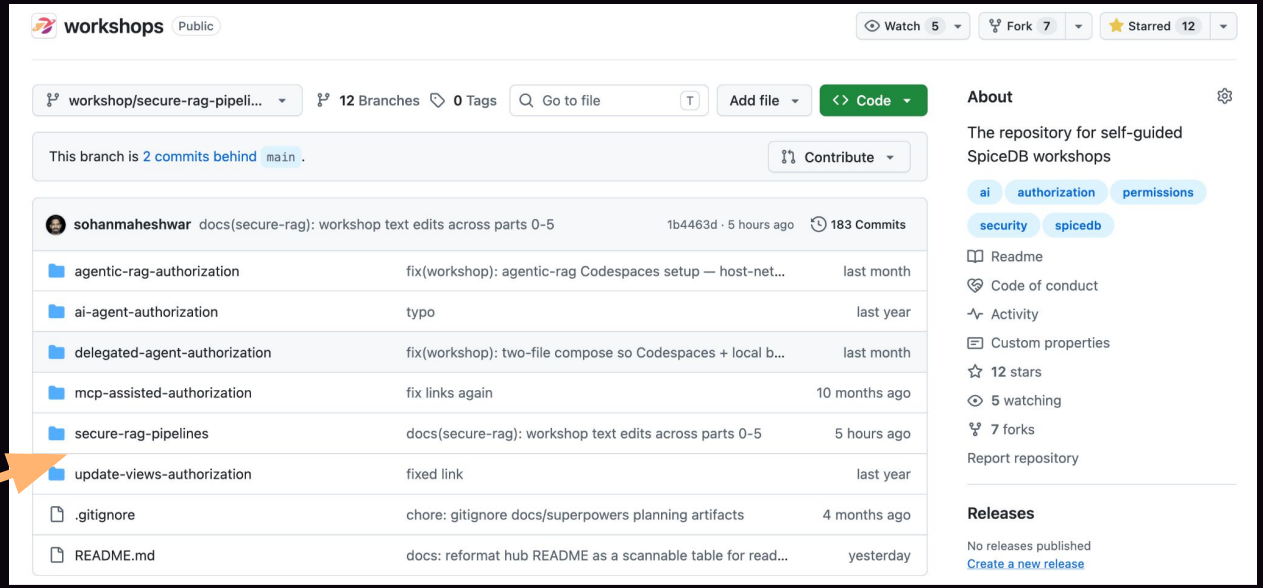
Other Features

- CheckBulkPermissions
  - Send all IDs in one request and get a result back per item
  - Saves round trip per resource

- Instant Revocation
  - Update or delete a relationship

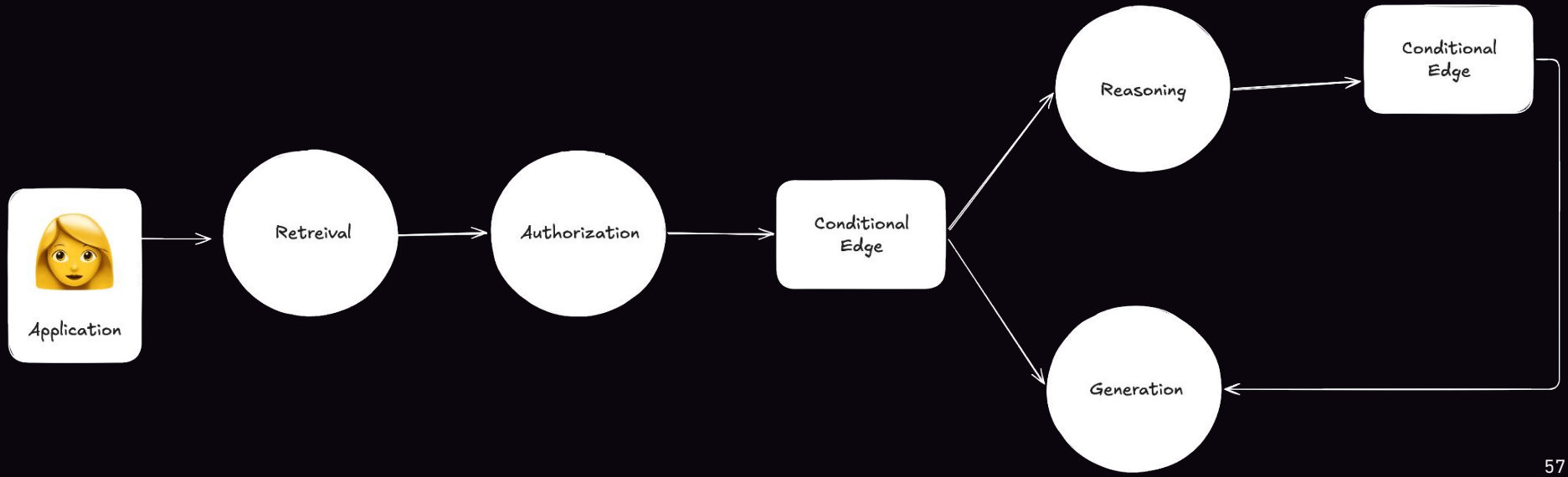
Part 4 - Smaller Features

- github.com/authzed/workshops
  → secure-rag-pipelines
  → Part 4 - Smaller Features



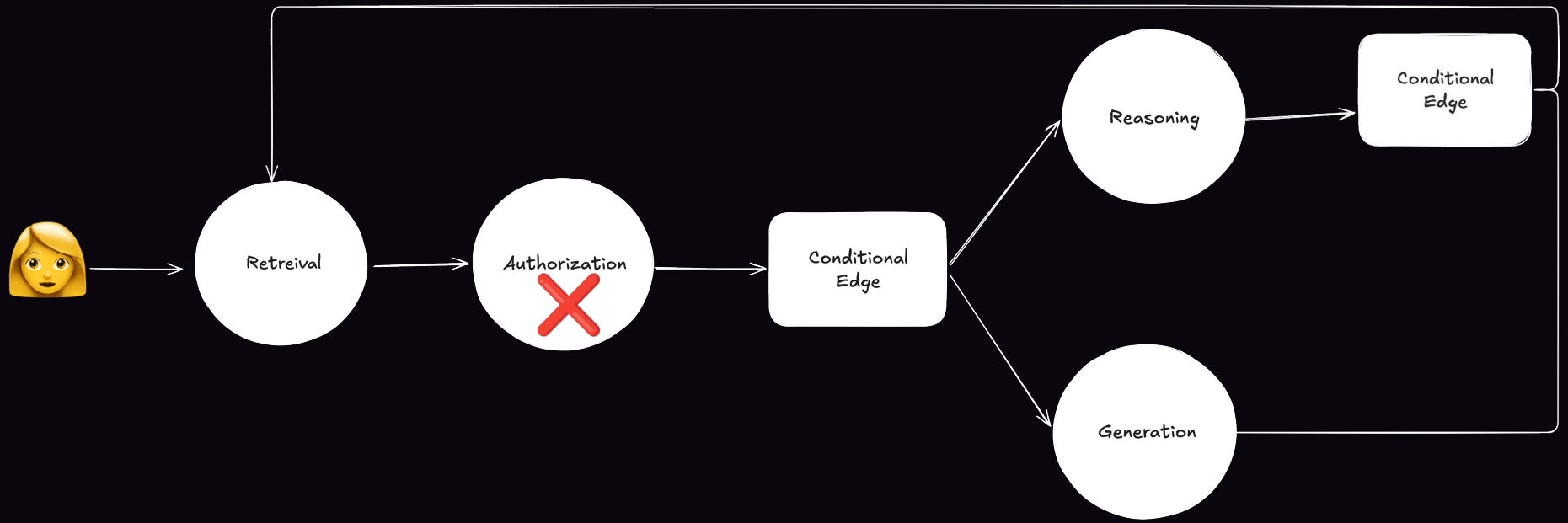
The Agentic Part

- Add reasoning to Agents with `max_attempts>1`



The workflow

- Add reasoning to Agents with `max_attempts>1`



Anti-Pattern ⚠

- Agent can reason about:
  - What to retrieve
  - What to generate after retrieval
  - Why authorization failed

- Agent cannot be allowed to reason about whether authorization applies.

RAG Patterns

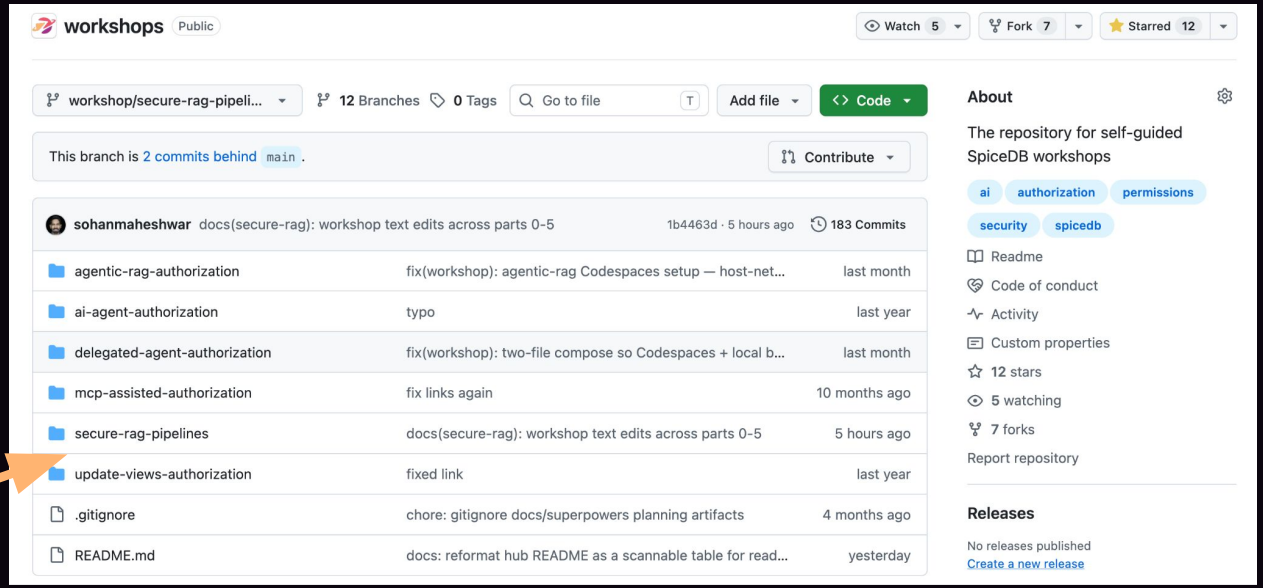
- Over-fetch to recover post-filter recall
- Compose the permission filter with metadata filters
- Authorize at the chunk

dev → prod

- Use a datastore such as Postgres or Cockroach to store relationships
- Use a hosted embedding model (Anthropic, Mistral etc)
- Vector Database
  - Self-hosted
  - Managed service (eg: Qdrant cloud)
- SpiceDB
  - Self-hosted
  - AuthZed Cloud / AuthZed Dedicated

Part 5 - Next steps

- github.com/authzed/workshops
  → secure-rag-pipelines
  → Part 5 - Next steps



Homework

Homework

- Add a new department and prove isolation
  - Create a `legal` department, for ex.
- Add an `editor` permission and check it
- Add access control for AI Agents into the mix

Links

- Read the Zanzibar paper
  - zanzibar.tech
- Demo Link
  - github.com/authzed/examples/tree/main/agentic-rag-authorization
- The LangChain - SpiceDB library
  - docs.langchain.com/oss/python/integrations/tools/spicedb

Thank You!

Sohan Maheshwar
linkedin.com/in/sohanmaheshwar/



Download today's presentation