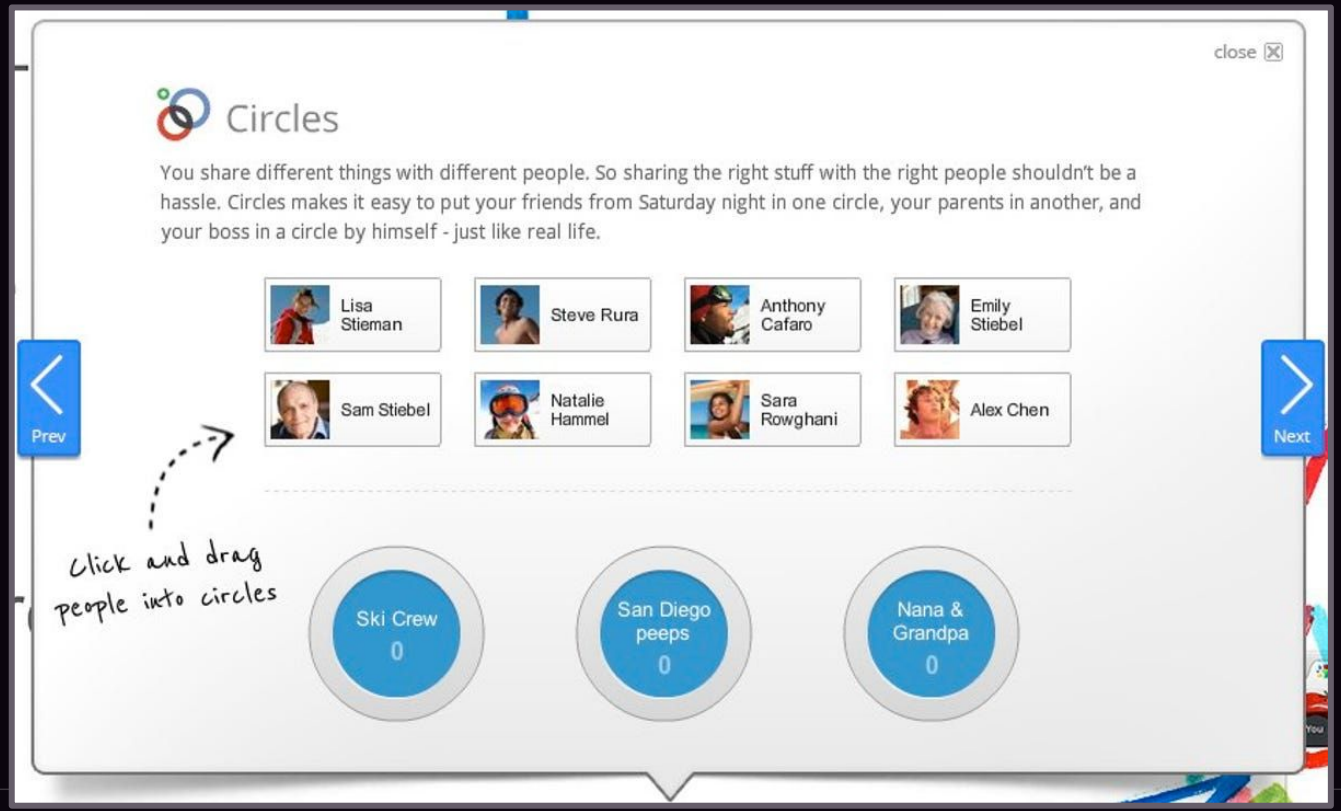
Delegated Authorization for AI Agents

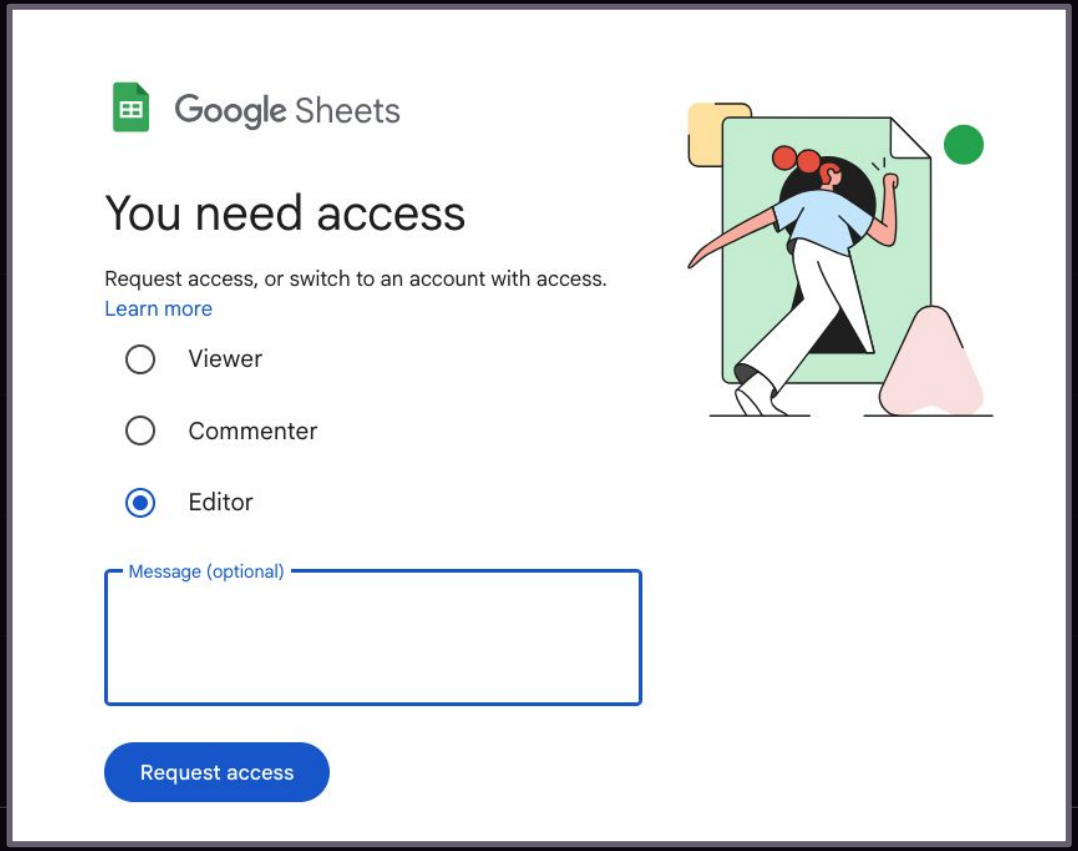
Build an Agent with Fine-Grained Permissions

Sohan Maheshwar
@AuthZed
AAIF Ambassador

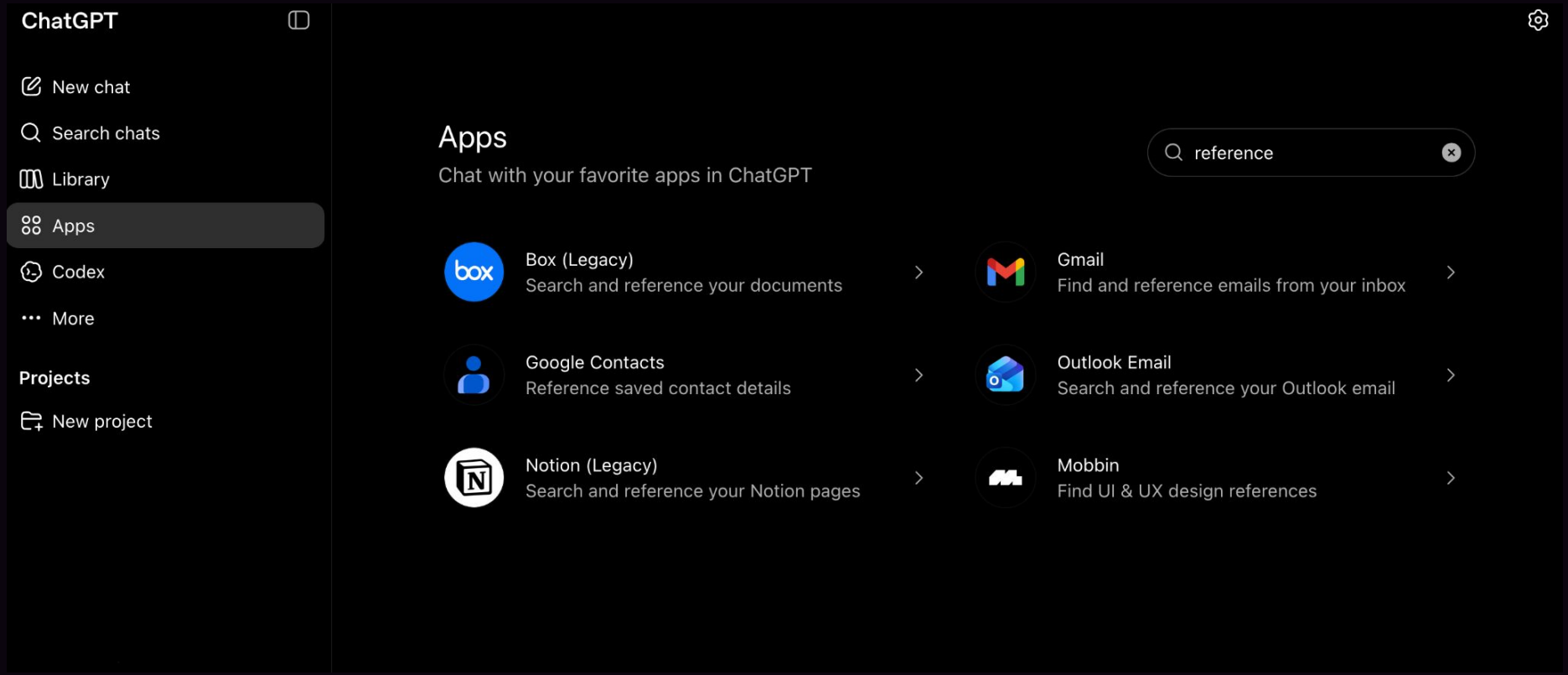
Throwback Thursday #tbt



Access Control in Google Products

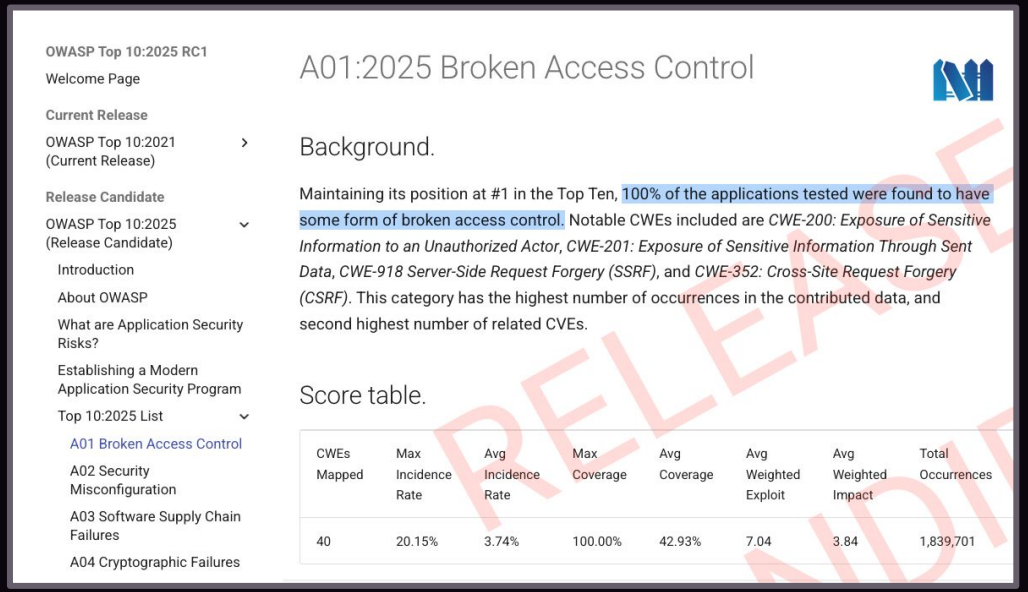


Access Control in ChatGPT Apps



OWASP Top Ten Web Apps Security Risks

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

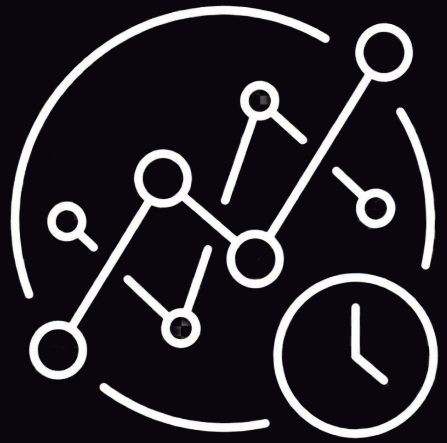


owasp.org/Top10/2025/

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



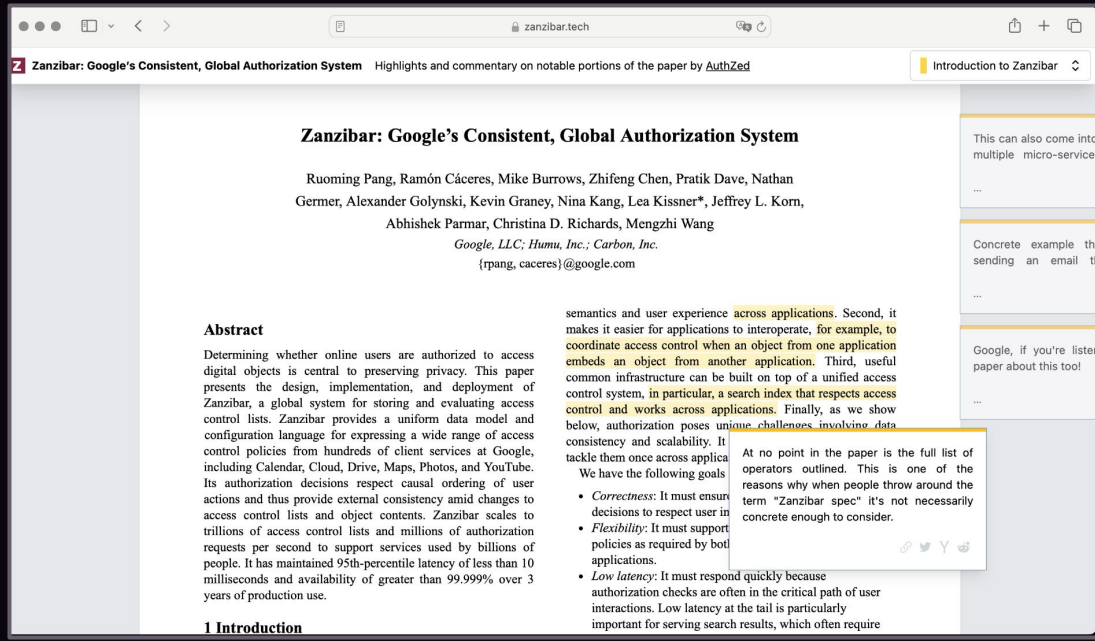
Primer on Authorization

Access Control Types

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

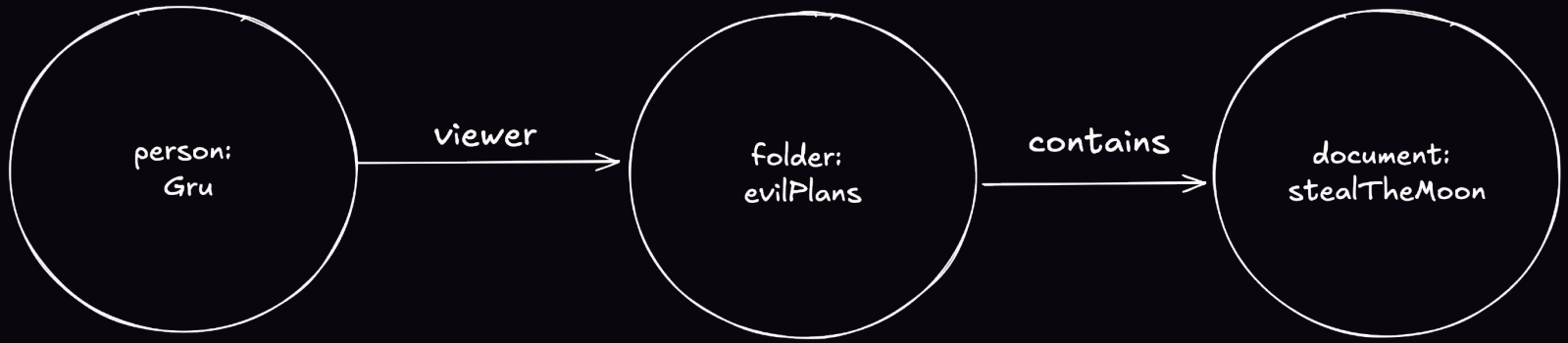
Google Zanzibar

- 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



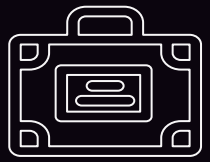
User

- Represents a natural person or service account

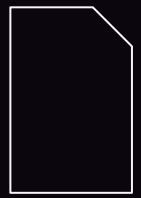
👩 **userid: 10957135**

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

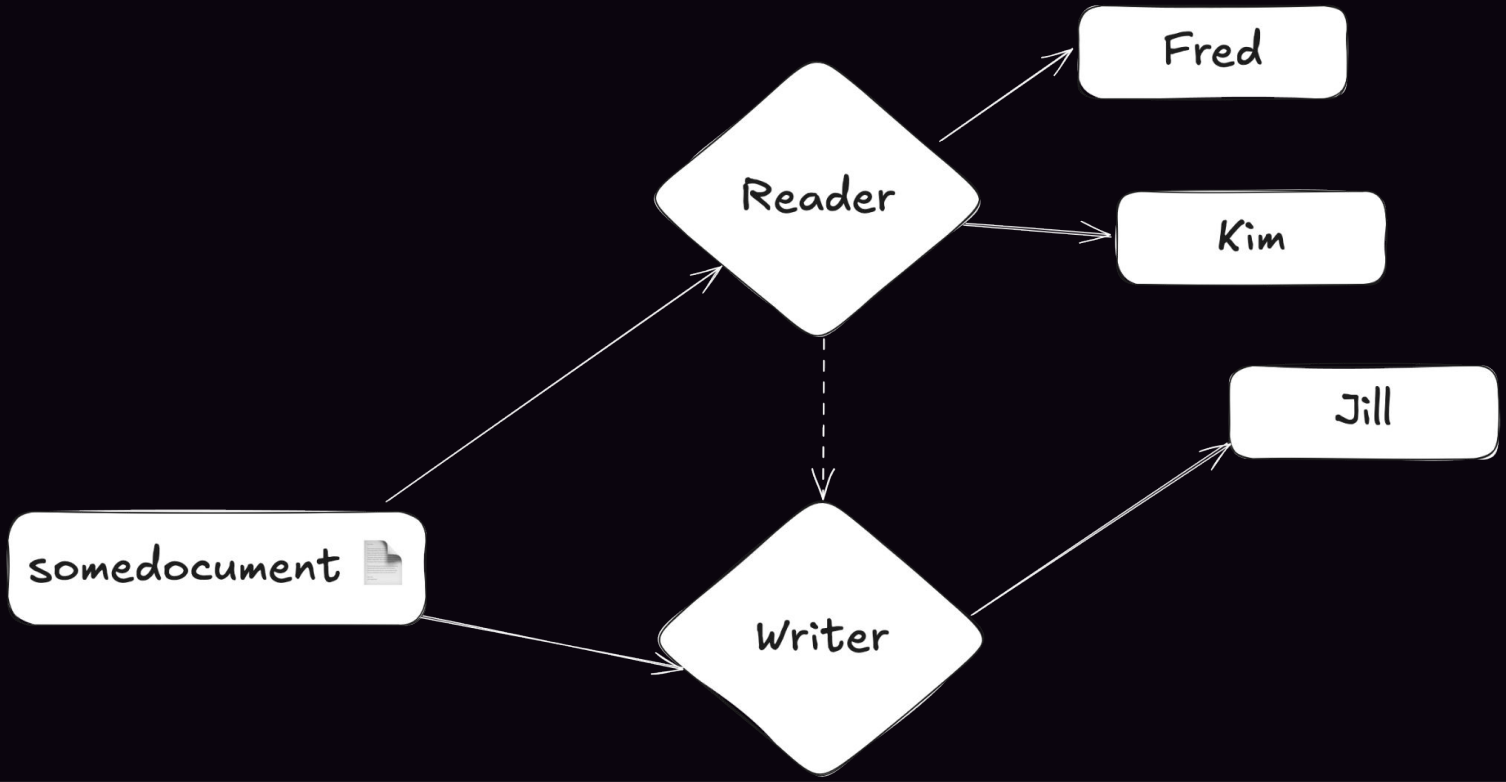
- Datum which expresses a single relationship

`document:123#owner@user:3`

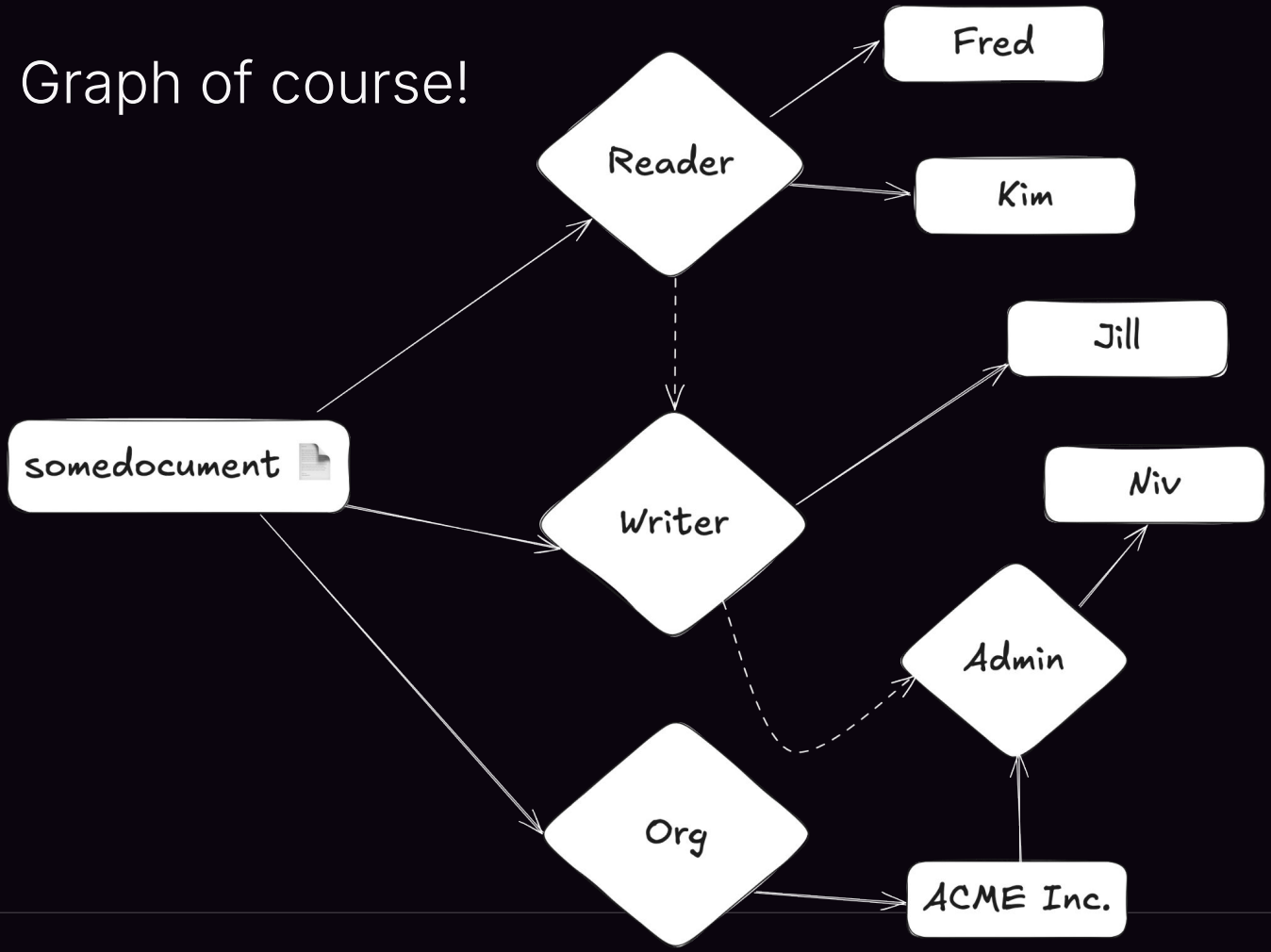
User 3 is an **owner** of a **document** with identifier 123

What are we really doing here?

Build a Graph of course!

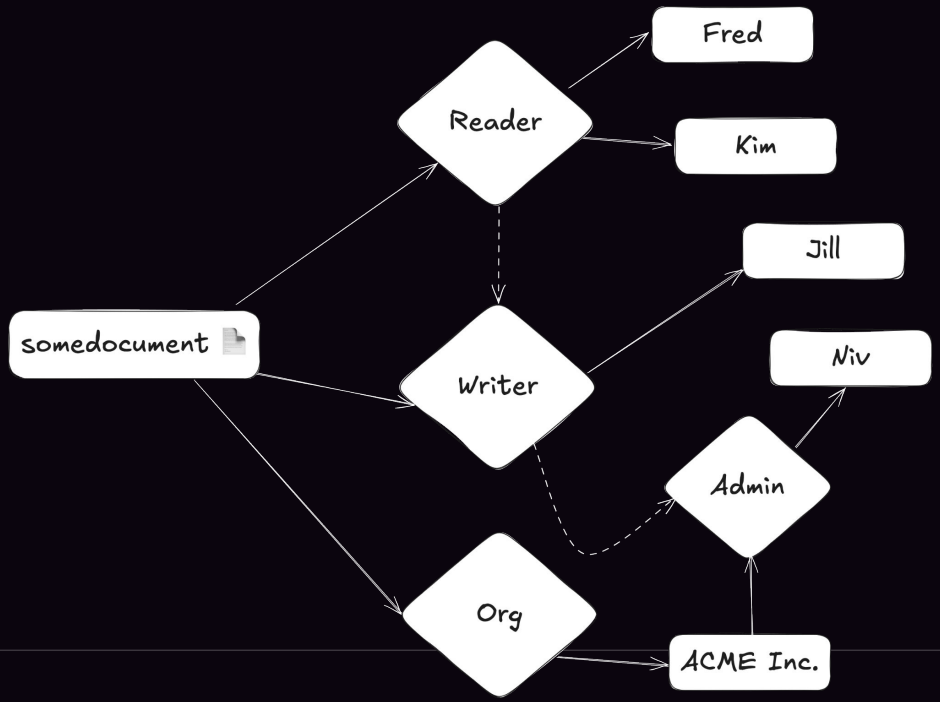


Build a Graph of course!



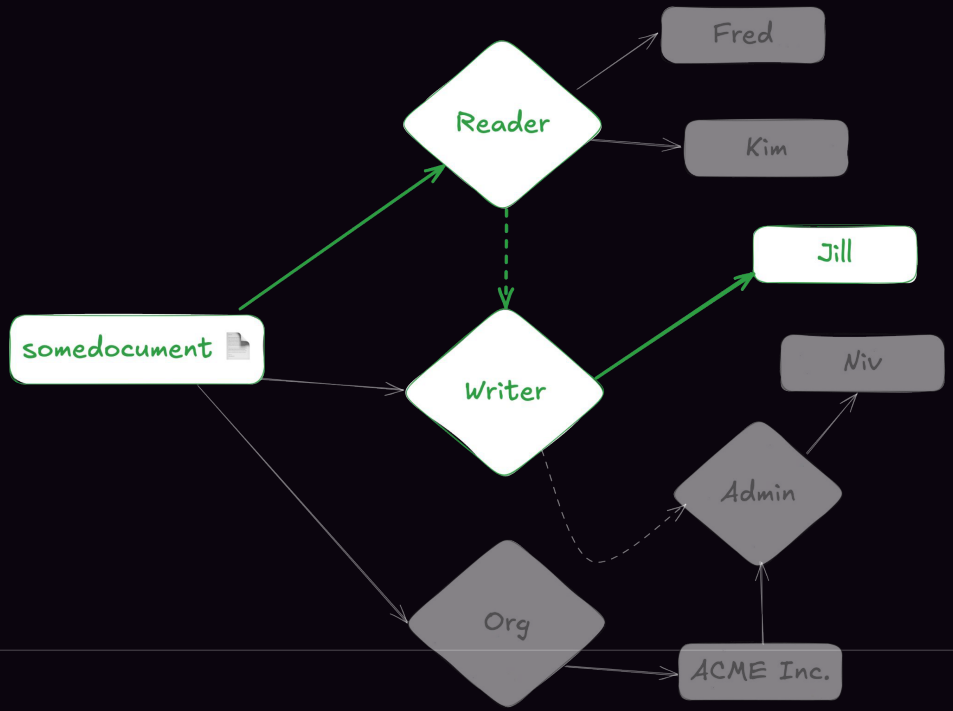
Check graph traversal example

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



Check graph traversal example

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



Where do Zanzibar-like systems shine?

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

SpiceDB

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

github.com/authzed/spicedb

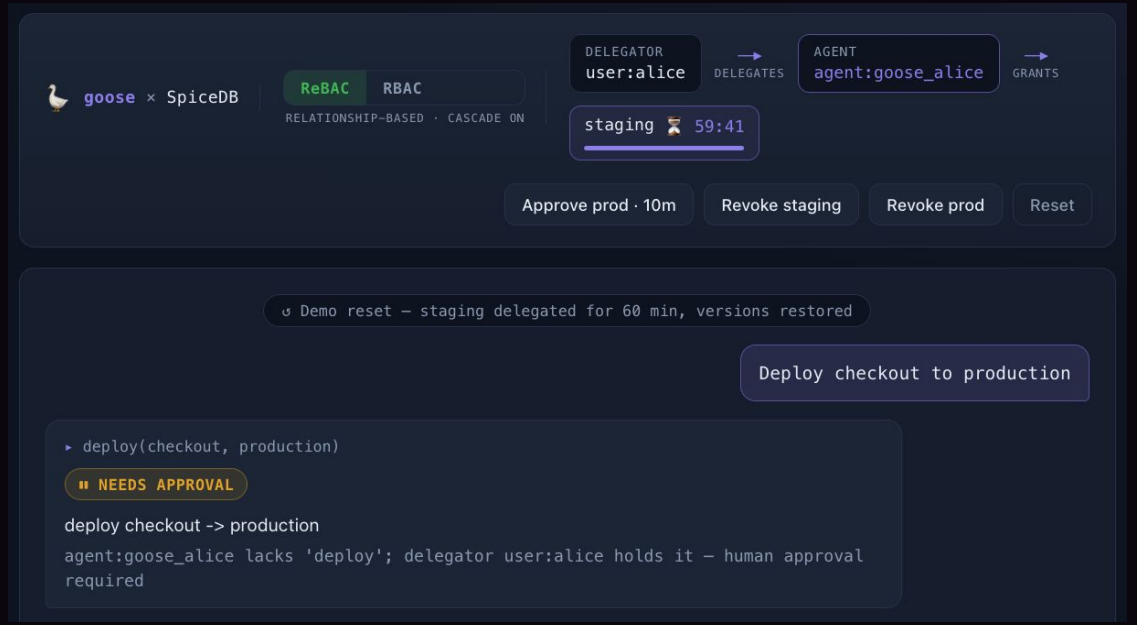
7K 🌟s & counting!

Goose by AAIF

- General-purpose AI agent that runs on your machine
- Connect to 70+ extensions
- Any LLM, including your subscriptions

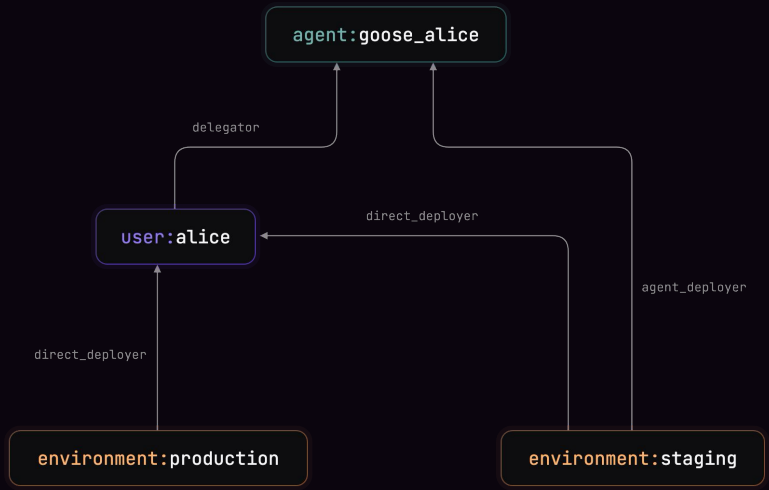
A DevOps Agent

- Delegated Authorization
- Has access to two environments
  - `staging`
  - `production`
- Time-bound grants
- Hierarchical contingencies



1. Delegated Authorization

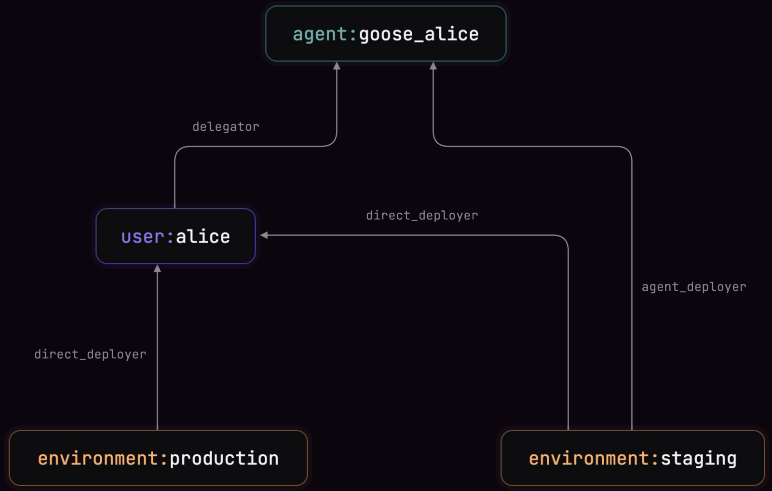
- Agent `goose_alice` acts on behalf of `user:alice`
- Permission is checked for every action



✕ With RBAC / tokens: copying a human's roles onto the agent drifts and over-grants

a. Schema

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

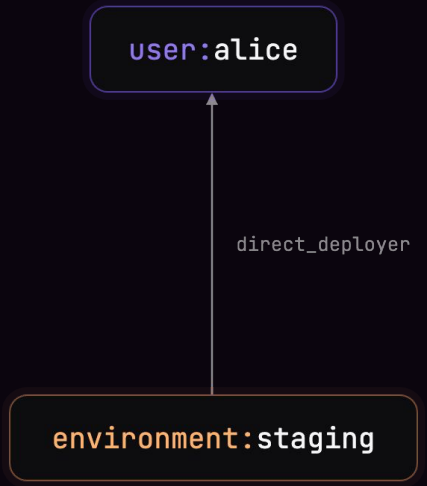



```
1 definition user {}
2
3 definition agent {
4     relation delegator: user
5 }
6
7 definition environment {
8     relation direct_deployer: user
9     relation agent_deployer: agent
10    permission deploy  = direct_deployer + agent_deployer
11 }
```

environment:staging#direct_deployer@user:alice

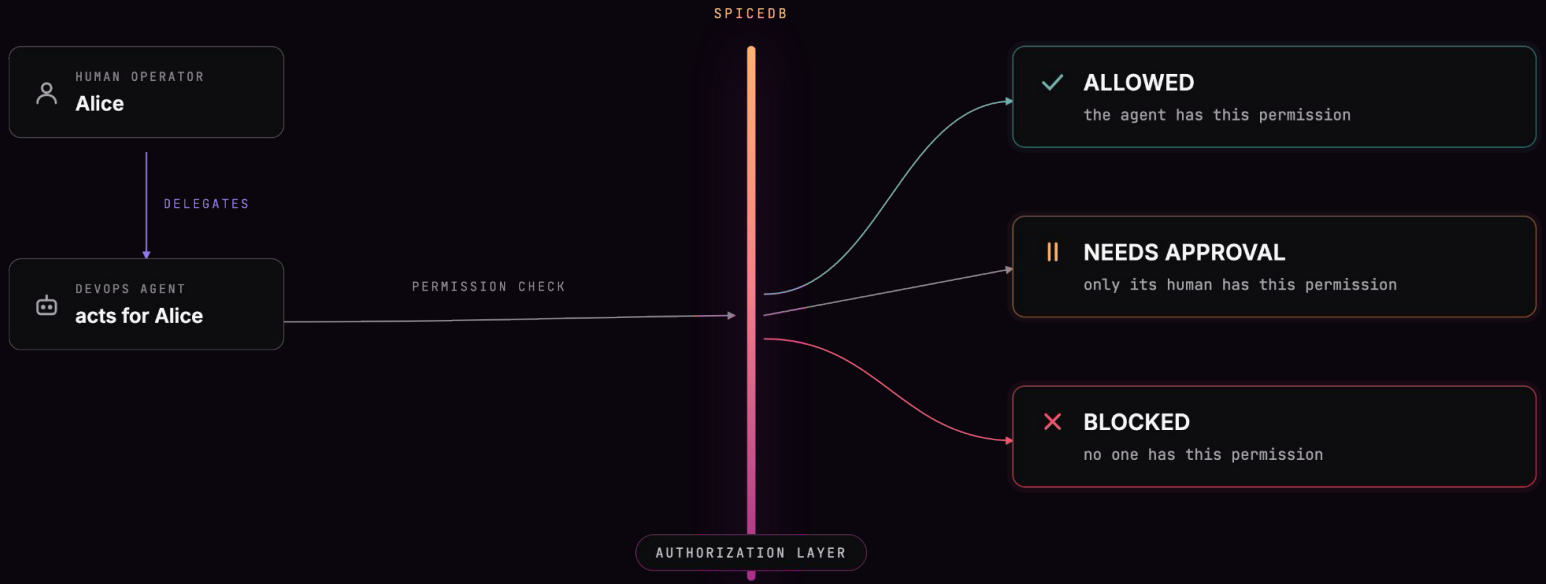
b. Relationships

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



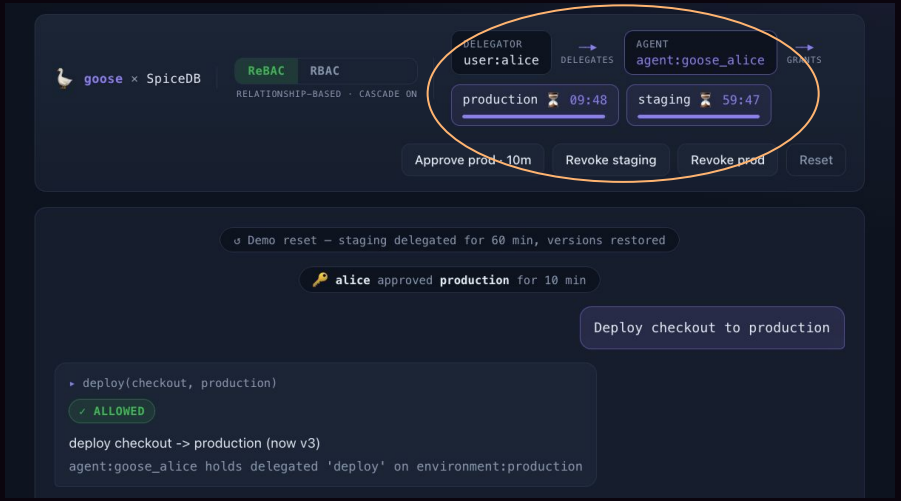
c. Permission Checks

- SpiceDB is called for all permission decisions
- Three-way decision



2. Time-Bound Access

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



✕ With RBAC/tokens: tokens lag; a cron leaves a window where a dead grant still works

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
    ...
 }
```

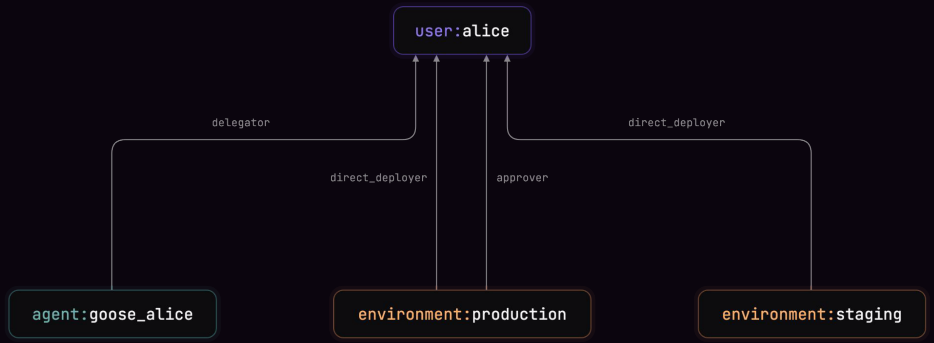
Add it to the relationship

```
from authz import expiry_from_now

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

3. Hierarchies

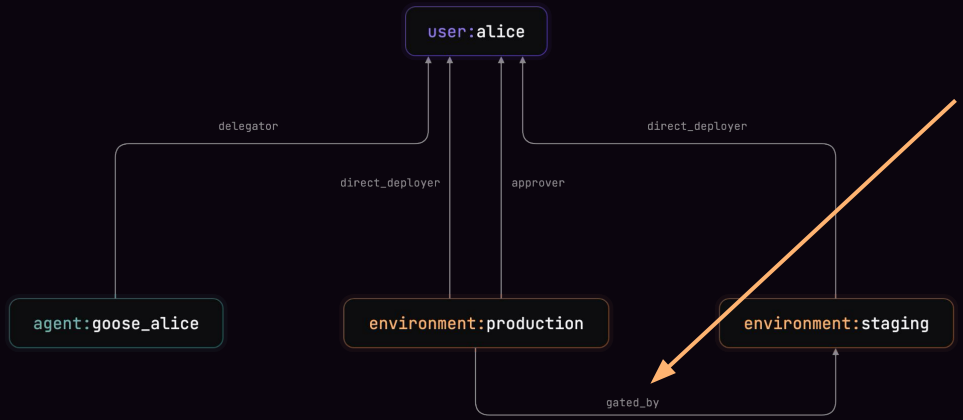
- “The agent can deploy to prod only while it can also deploy to staging”
- Model real-world hierarchies, nested groups and dependencies.
- Contingent evaluation ≠ cascading delete.



✕ With RBAC/tokens: no role can depend on another - you drop into app code

3. Hierarchies

- “The agent can deploy to prod only while it can also deploy to staging”
- Model real-world hierarchies, nested groups and dependencies.
- Contingent evaluation ≠ cascading delete.



× With RBAC/tokens: no role can depend on another - you drop into app code

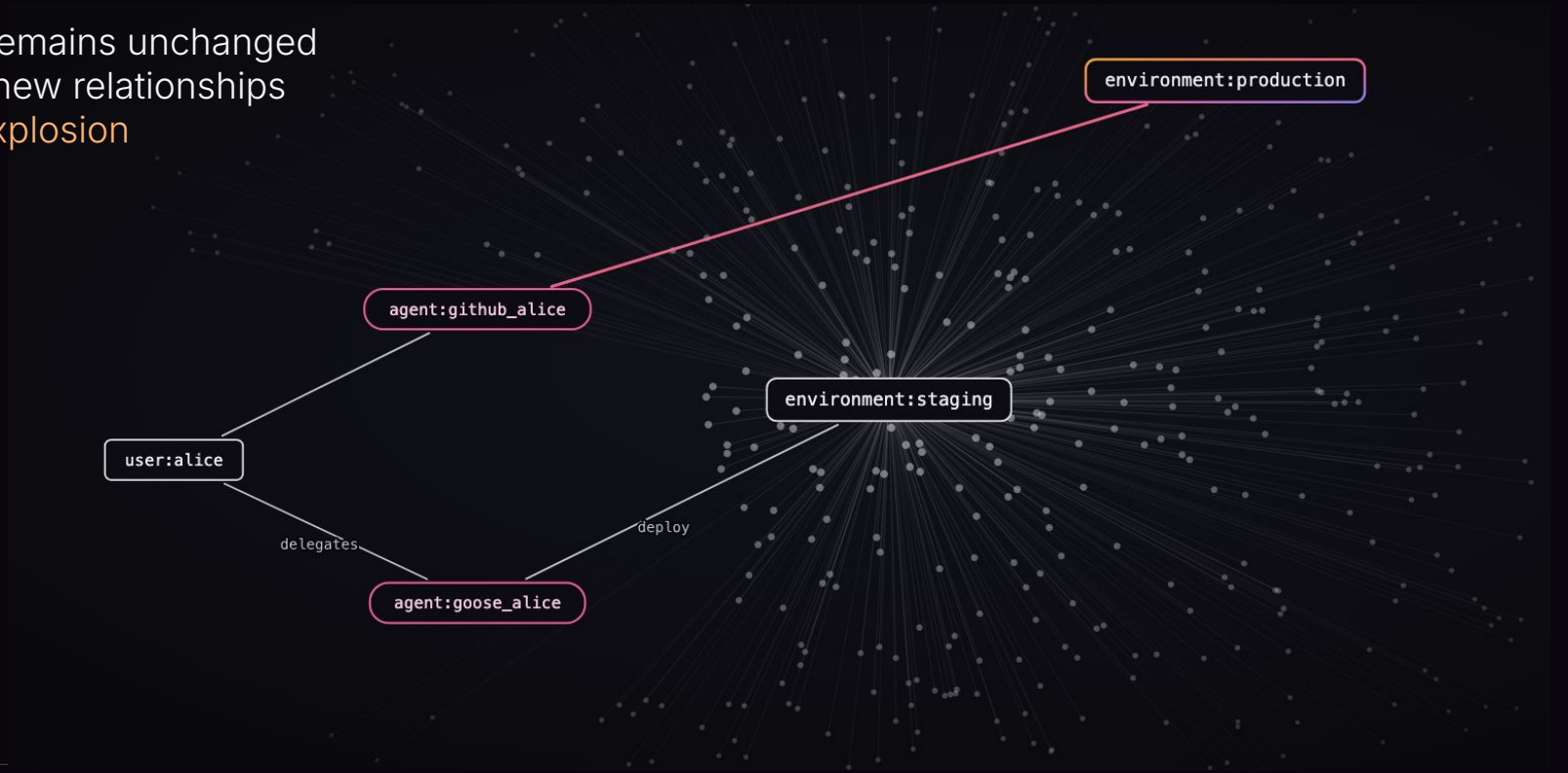
Schema Operators

- `+` Union
- `&` Intersection
- `-` Exclusion
- `->` Arrow

```
definition environment {
    relation direct_deployer: user
    relation agent_deployer: agent with expiration
...
    relation gated_by: environment
    permission agent_deploy = agent_deployer & gated_by->agent_deployer
    permission deploy = direct_deployer + agent_deploy
...
}
```

Many users, agents, environments

- Schema remains unchanged
- Just add new relationships
- No role explosion



Recap

Pattern	How ReBAC does it	Where Roles & Tokens break
Delegated Authorization	Agent holds its own grant, bounded to its human via `delegator`	Copy the human's roles onto the agent — a snapshot that drifts and over-grants
Expiring Grants	agent_deployer: agent with expiration — server-side, self-cleaning	Tokens live till they expire; a cron leaves a window where a dead grant still works
Instant revocation	Delete one relationship — the next check is a live lookup	Minted tokens stay valid; RBAC means cleanup in N places, hoping you got them all
Hierarchical contingencies	`agent_deploy = agent_deployer & gated_by→agent_deployer`	No role can depend on holding another role — you drop into app code

More Reading

- goose docs
  - goose-docs.ai
- Read the Zanzibar paper
  - zanzibar.tech
- Working Demo Link
  - github.com/sohanmaheshwar/goose-spicedb-delegation

Thank You!

Sohan Maheshwar
@AuthZed
AAIF Ambassador

Sohan Maheshwar
AAIF Ambassador | Experienced DevRel professional