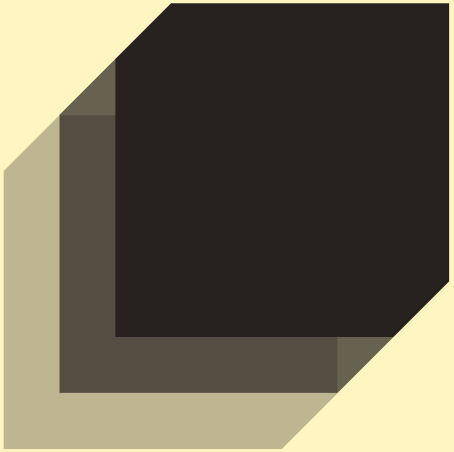


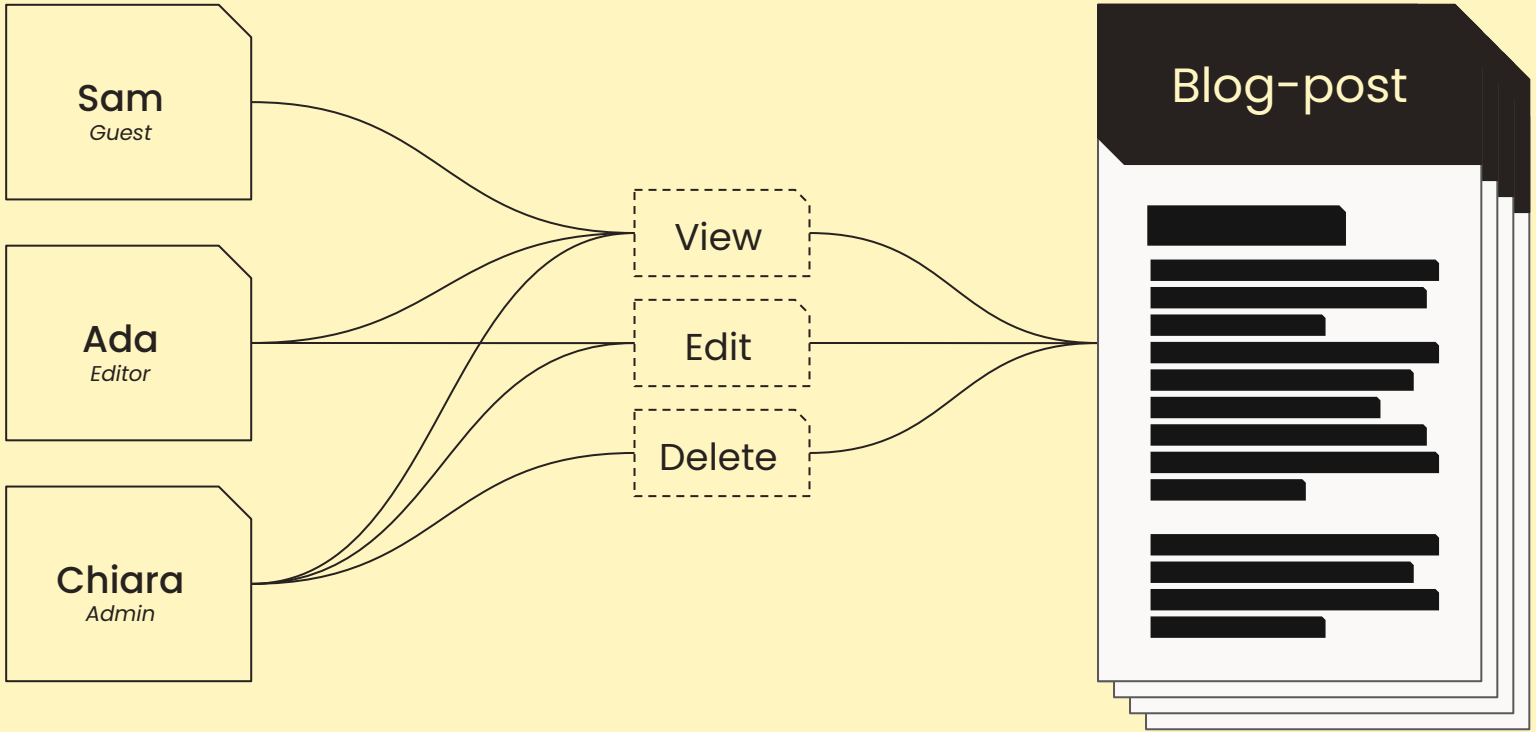
Paradigm *shift*

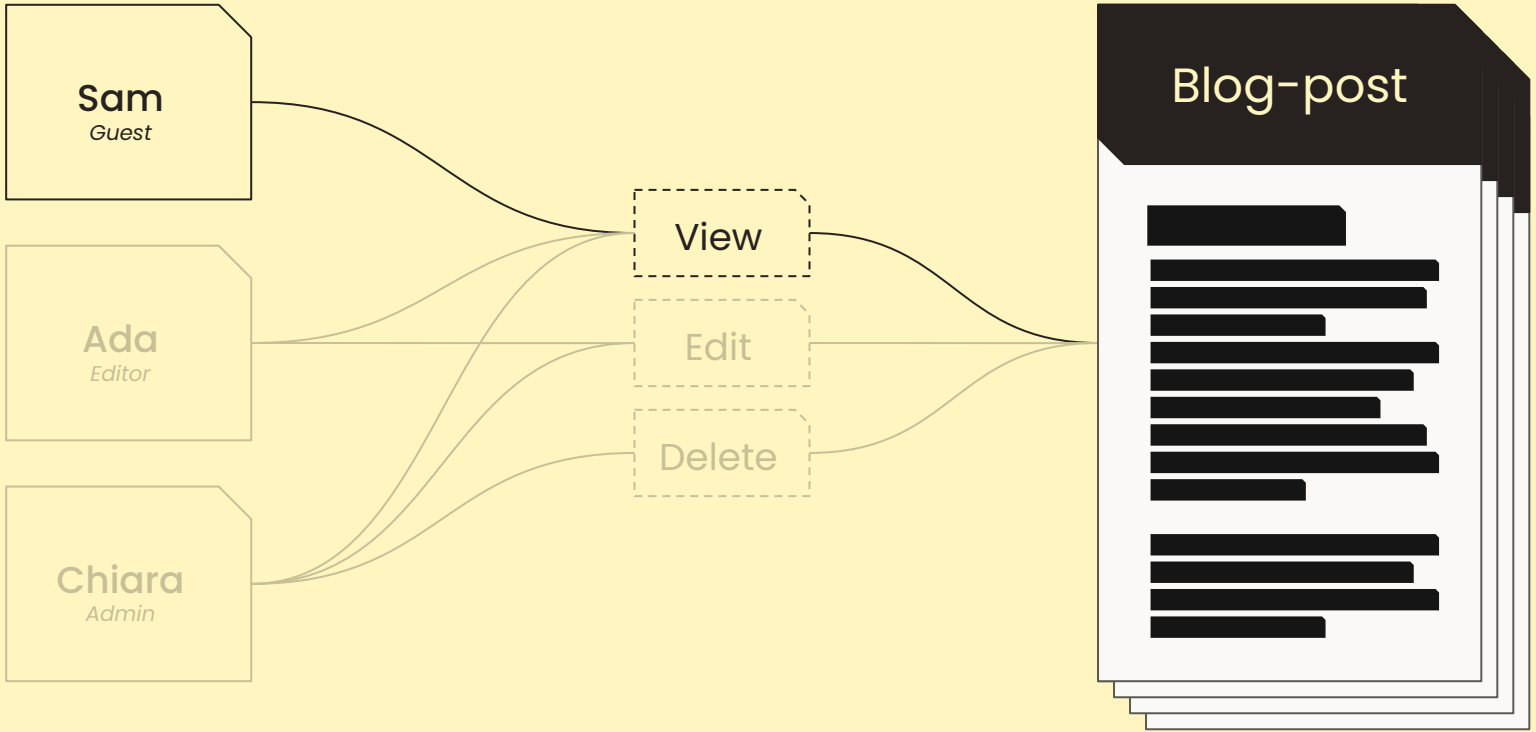
Moving beyond roles and permissions
to a ***fine-grained*** access control

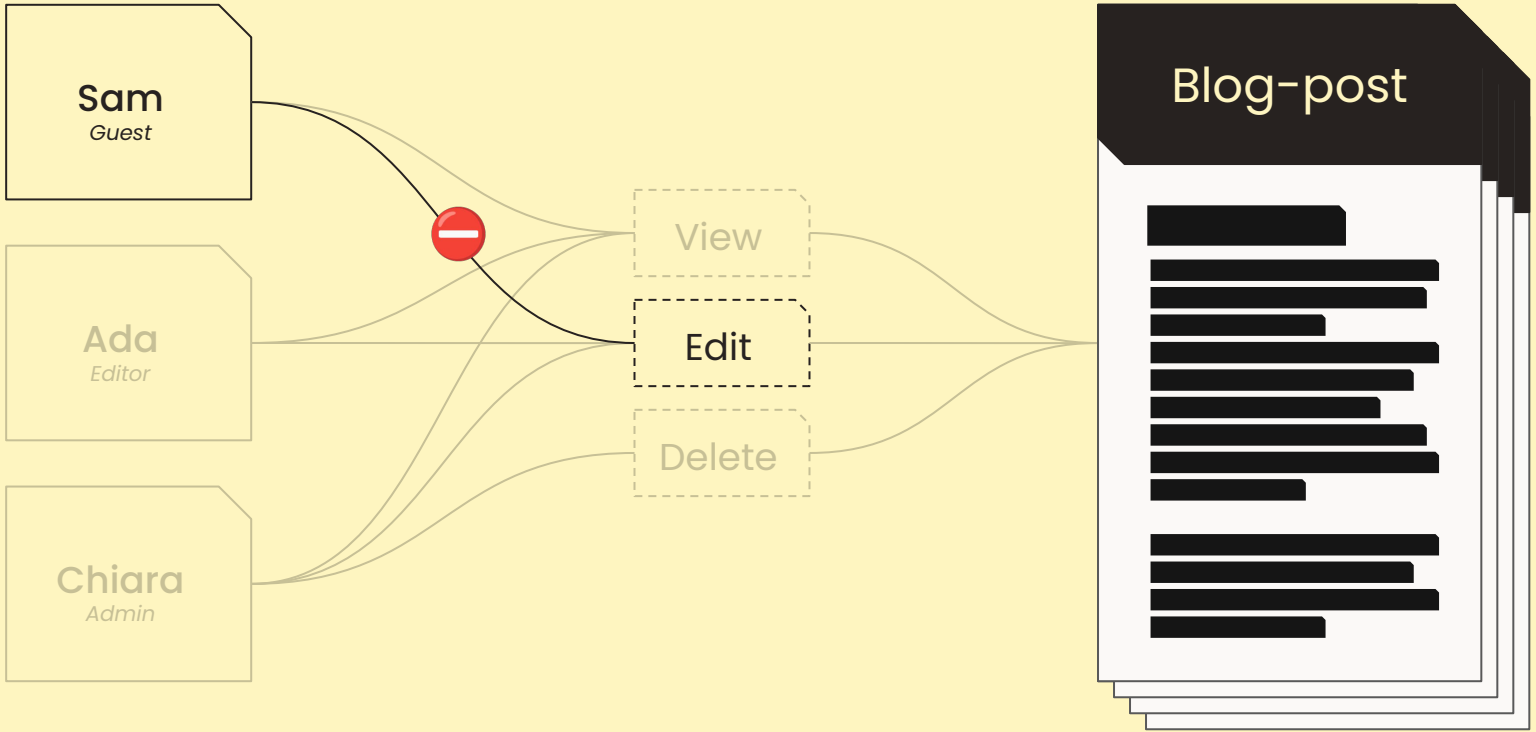
Access Control?

Can *this user* do *that action*?

An *access control method* decides whether an action is allowed or not


Sam
Guest
Ada
Editor
Chiara
Admin
View
Edit
Delete
Blog-post


Sam
Guest
Ada
Editor
Chiara
Admin
View
Edit
Delete
Blog-post

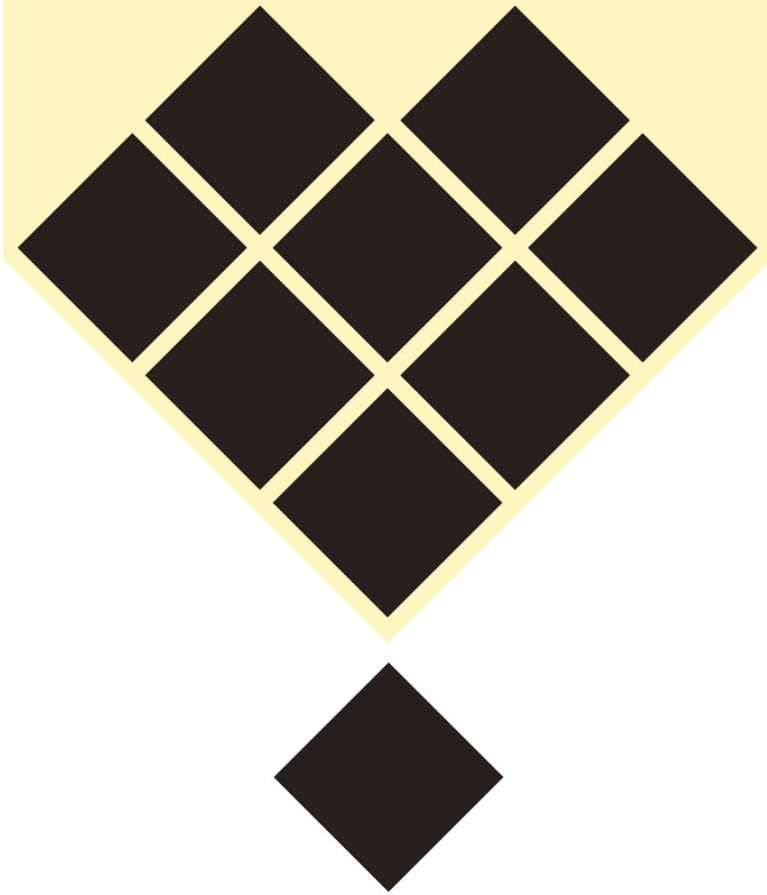

Sam
Guest
Ada
Editor
Chiara
Admin
View
Edit
Delete
Blog-post

Paradigm
shift

Fine-grained Access control

Take back **control**!

Coarse grained



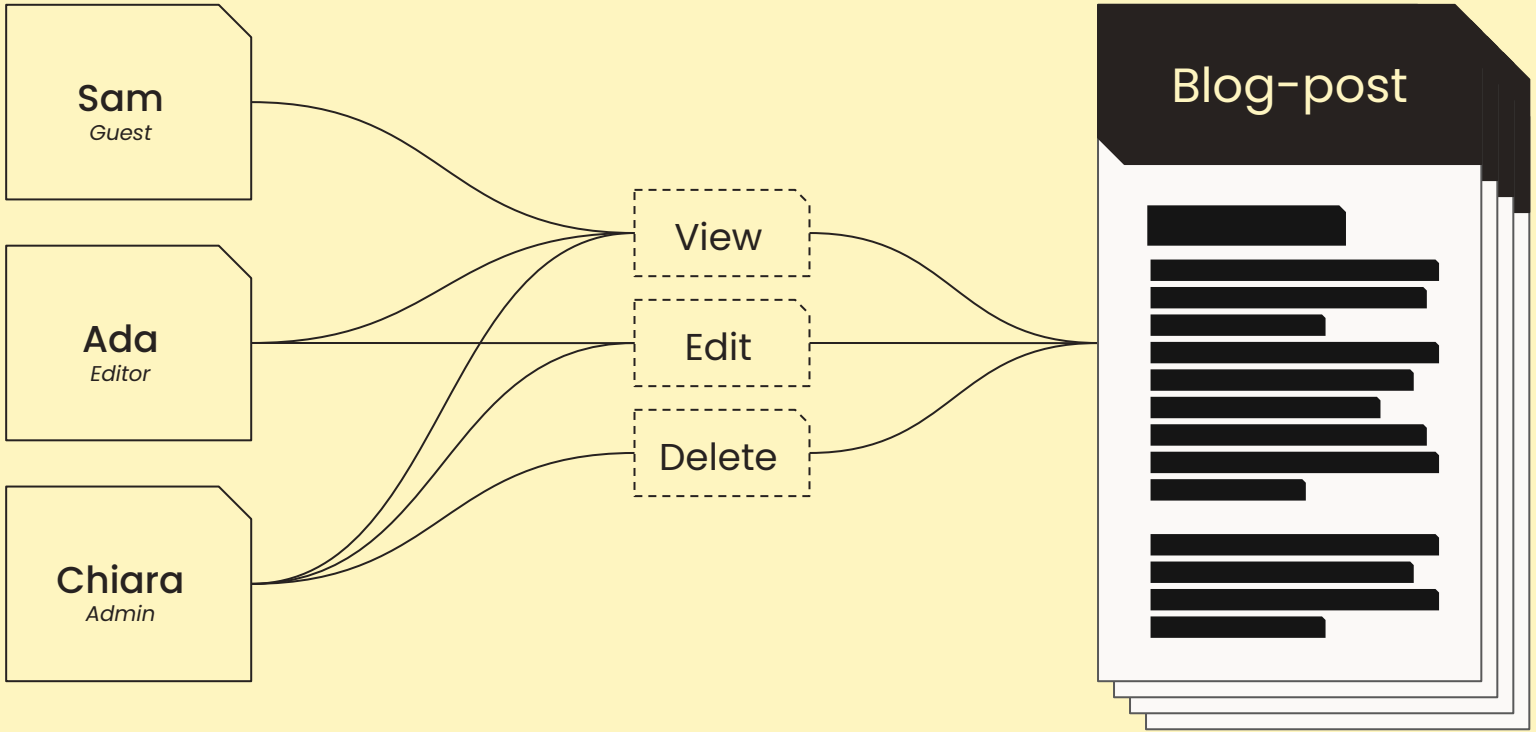
Access control decisions are made based on **resource type**

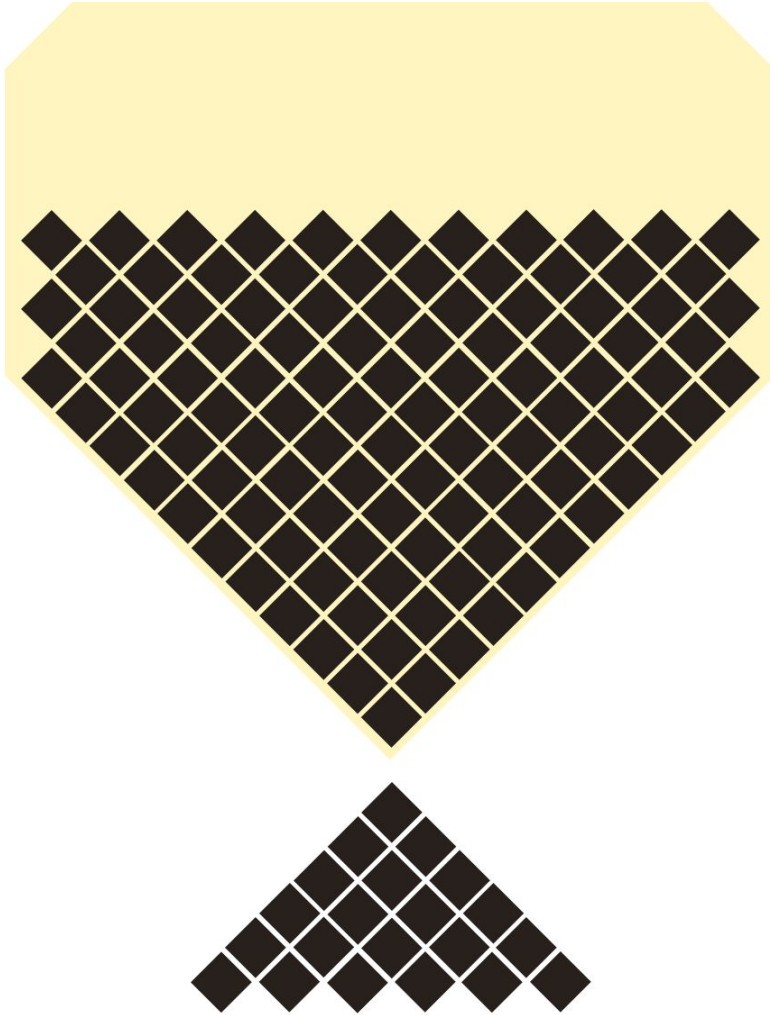
Example

An editor can edit all blog-posts

Example

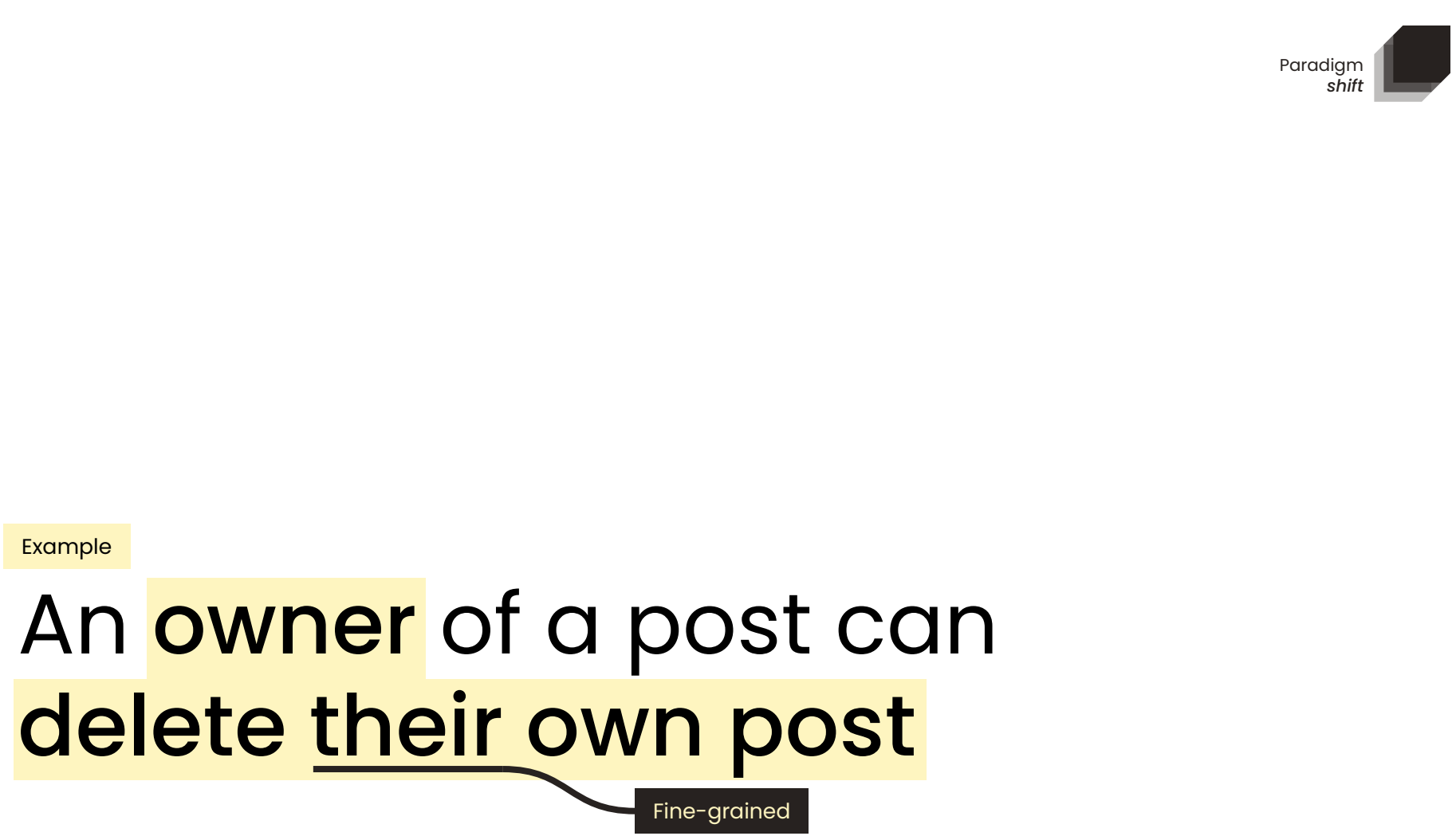
An admin can delete all blog-posts

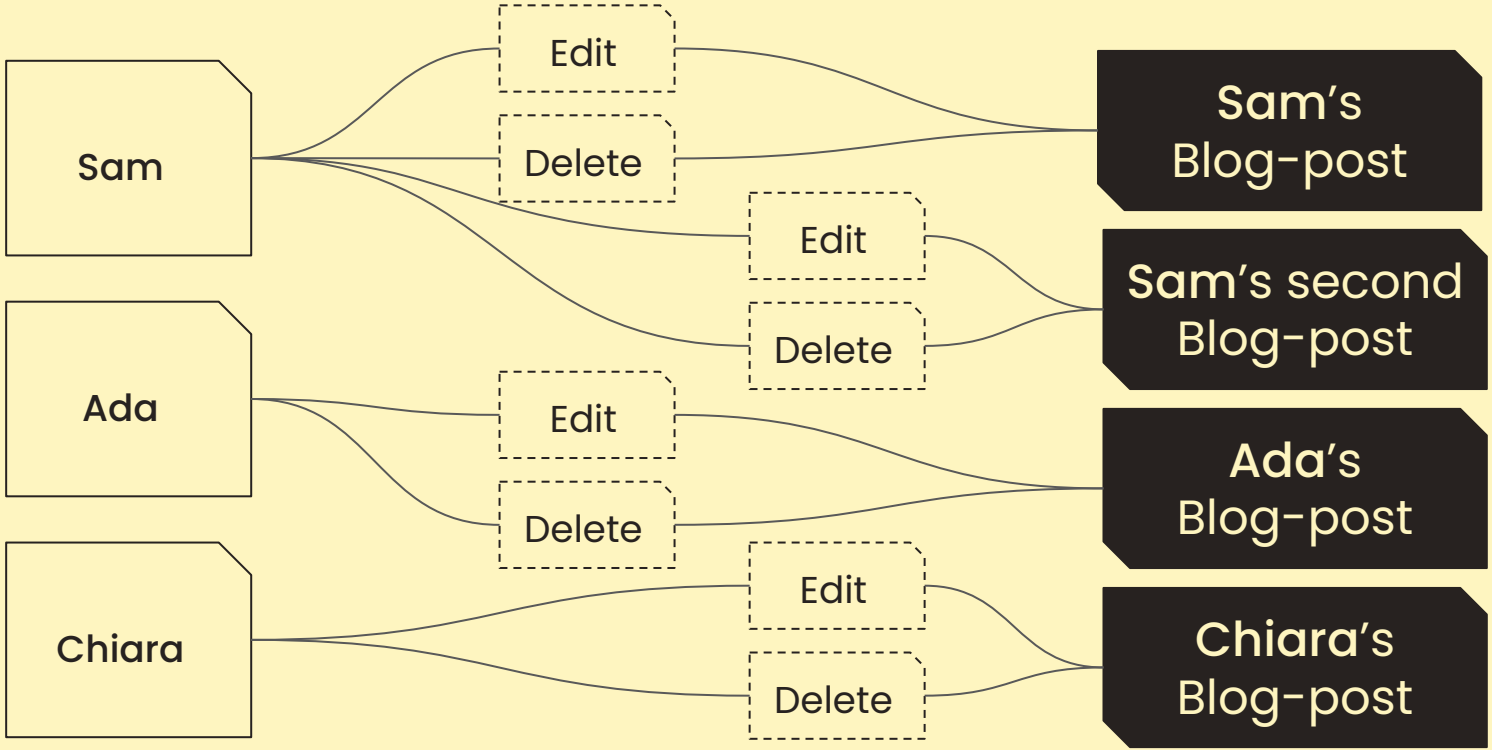

Sam
Guest
Ada
Editor
Chiara
Admin
View
Edit
Delete
Blog-post

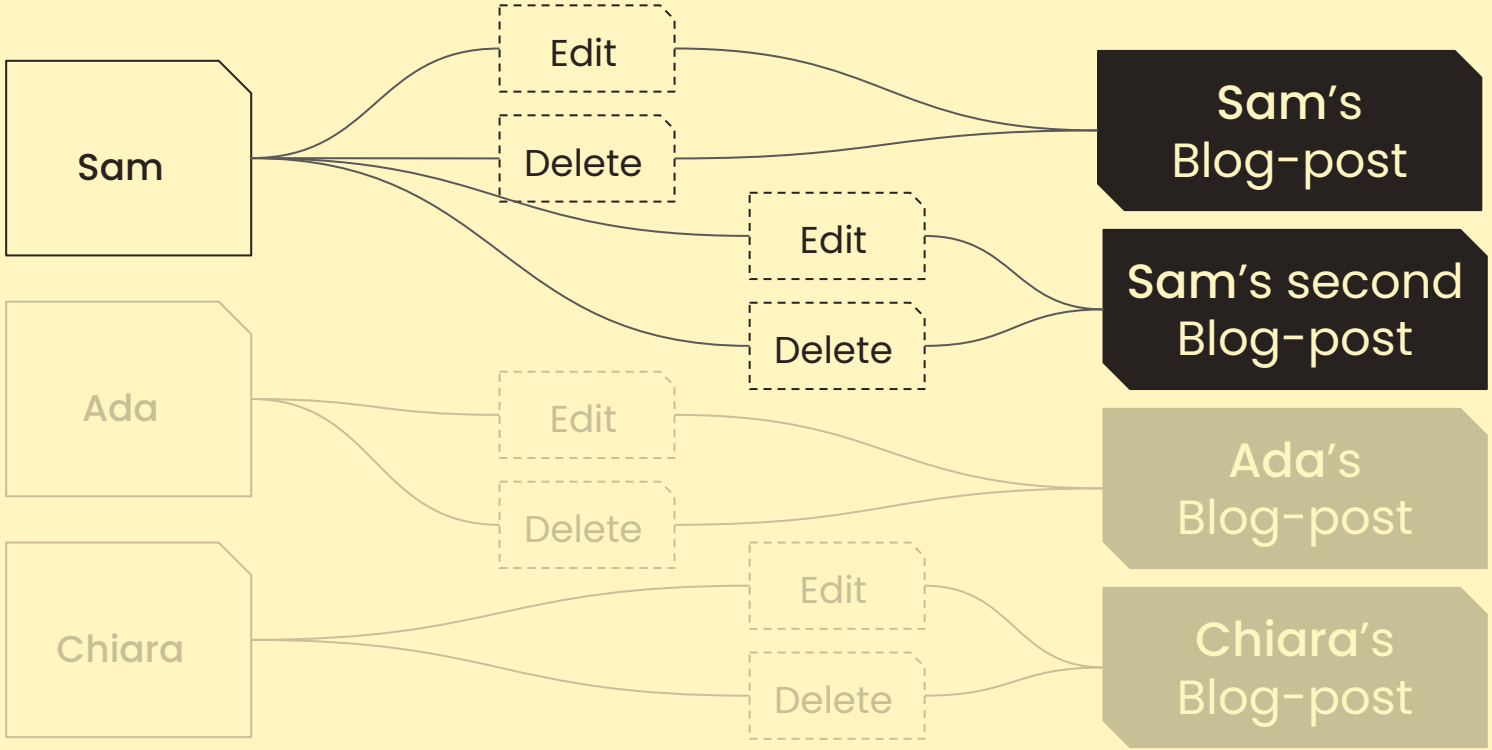


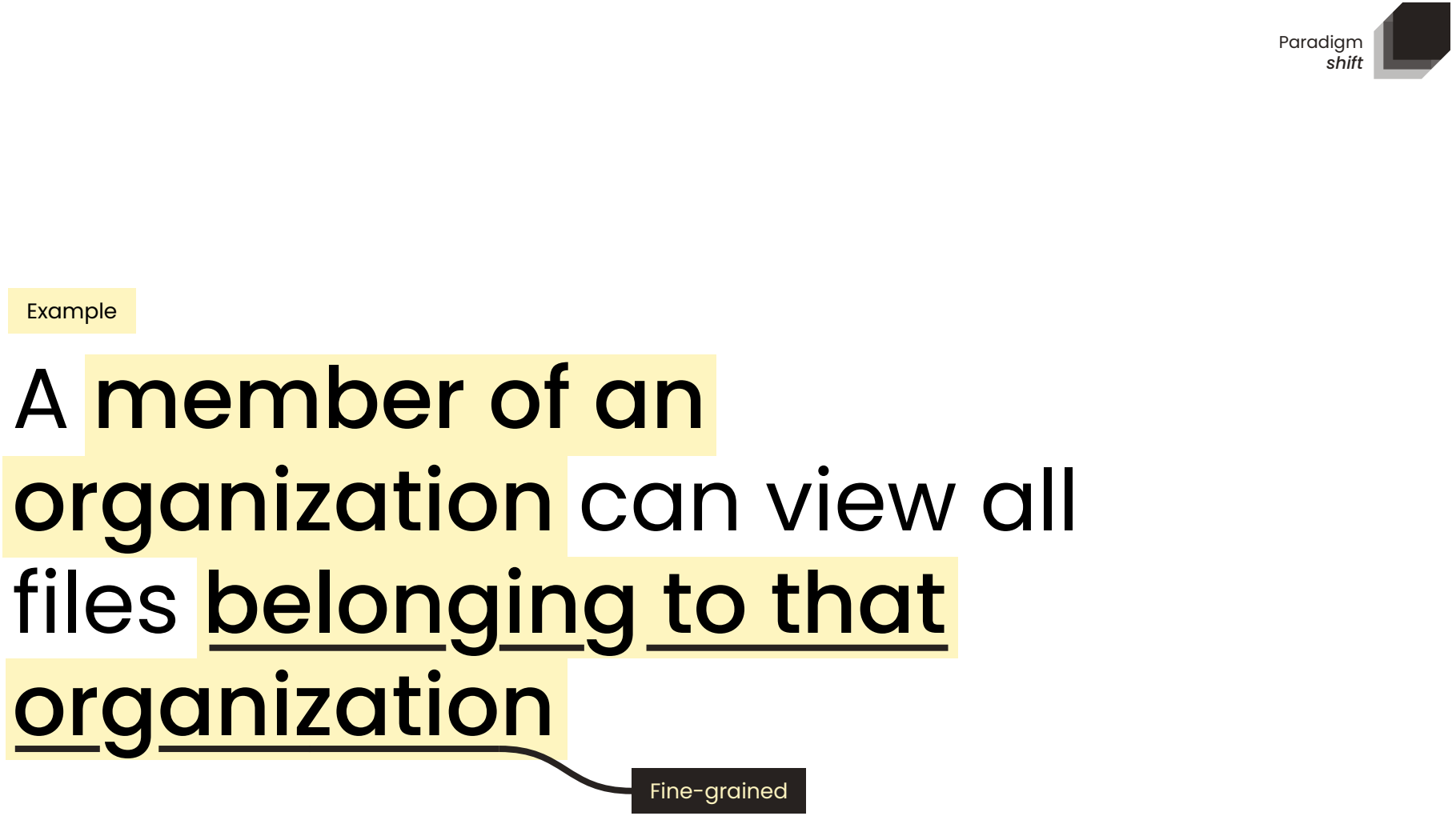
Fine grained

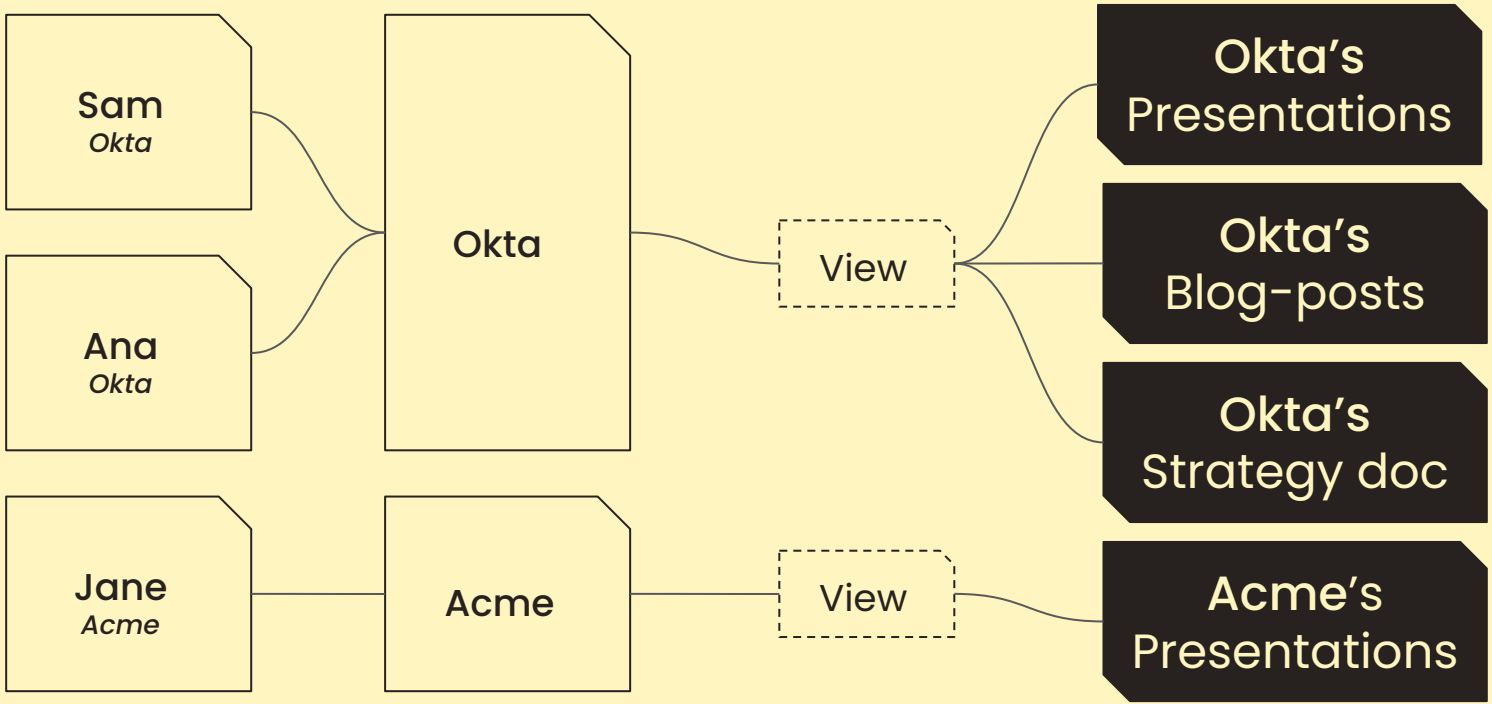
Access control decisions are made for a **specific resource or situation**

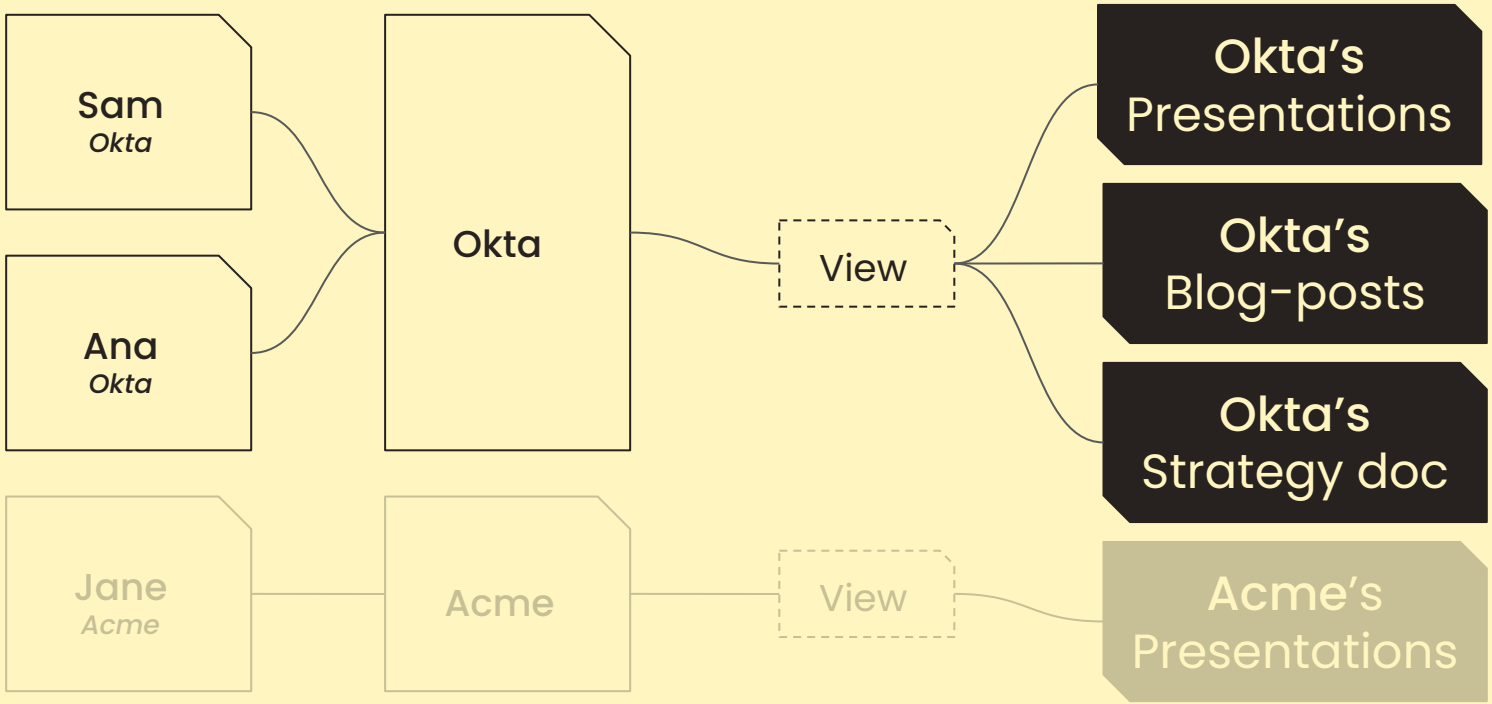

Example
An owner of a post can
delete their own post
Fine-grained


Sam
Ada
Chiara
Edit
Delete
Edit
Delete
Edit
Delete
Edit
Delete
Sam's Blog-post
Sam's second Blog-post
Ada's Blog-post
Chiara's Blog-post


Sam
Ada
Chiara
Edit
Delete
Edit
Delete
Edit
Delete
Edit
Delete
Sam's Blog-post
Sam's second Blog-post
Ada's Blog-post
Chiara's Blog-post


Example
A member of an organization can view all files belonging to that organization
Fine-grained


Sam
Okta
Ana
Okta
Jane
Acme
Okta
Acme
View
View
Okta's Presentations
Okta's Blog-posts
Okta's Strategy doc
Acme's Presentations

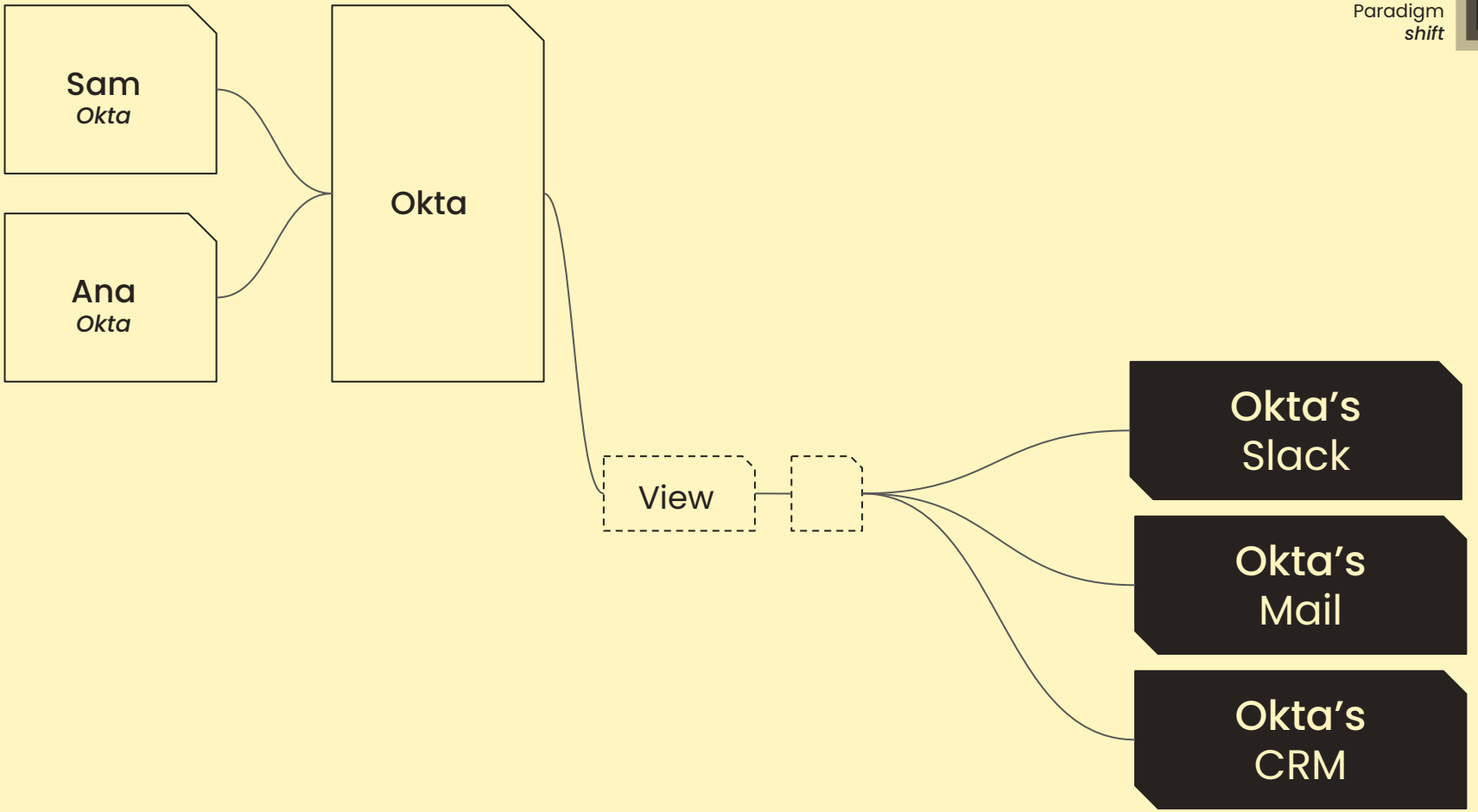

Sam
Okta
Ana
Okta
Okta
View
Okta's Presentations
Okta's Blog-posts
Okta's Strategy doc
Jane
Acme
Acme
View
Acme's Presentations

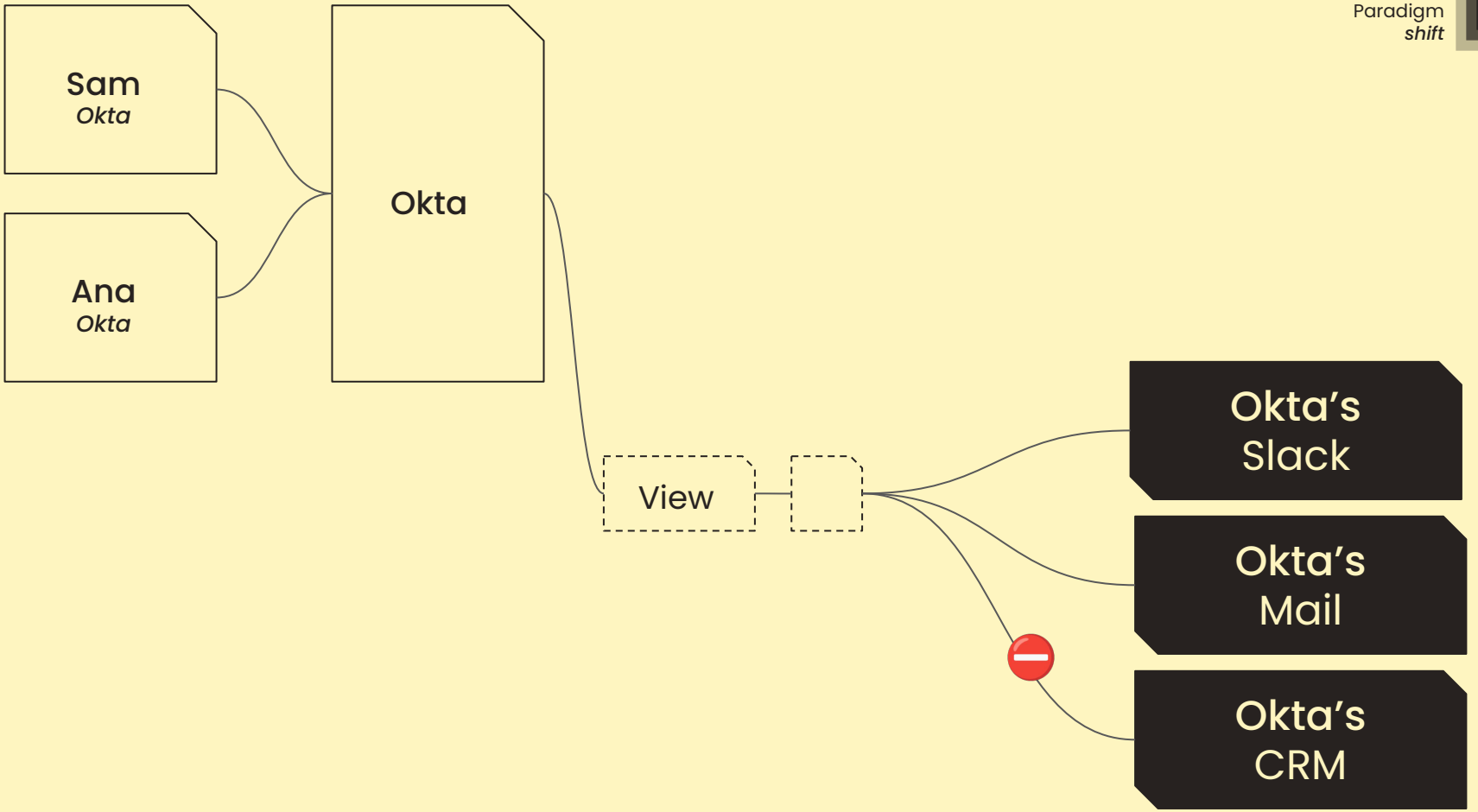
Example

An employee can access the CRM during office hours

Fine-grained

Paradigm
shift




Sam
Okta
Ana
Okta
Okta
View
Okta's
Slack
Okta's
Mail
Okta's
CRM

Coarse grained

Smaller applications
No user-generated content

Fine grained

User-generated content
Multi tenant applications
Sharing resources
Sensitive data
Contextual or environmental decisions

Paradigm
shift

RBAC

Role-based Access control

Roles and *permissions* define access

Decisions are based on **permissions** assigned to a user via **roles**

Roles

Guest

Admin

Editor

Paying customer

Enterprise

Paradigm
shift

Permissions

View

Create

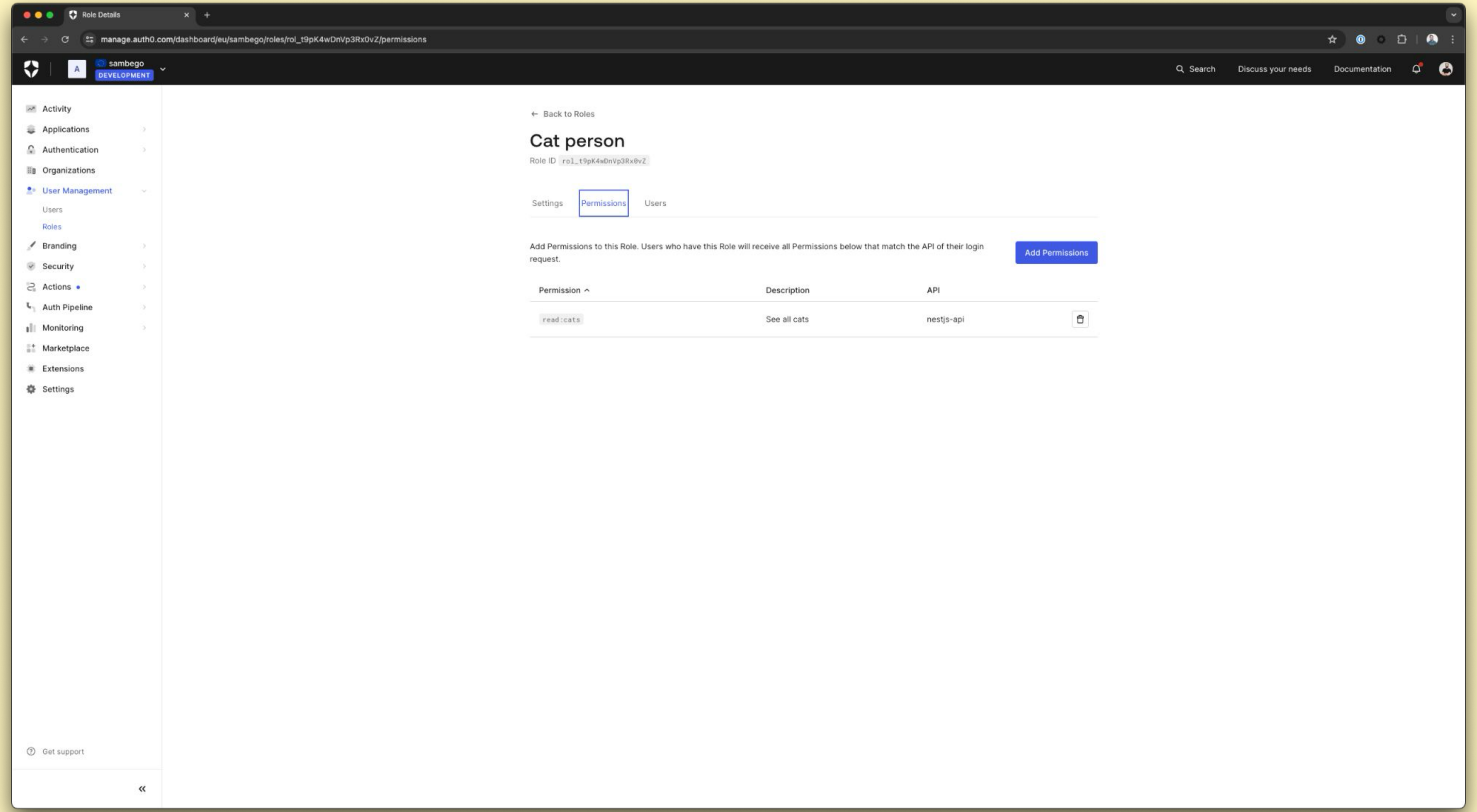
Edit

Delete

Paradigm
shift

Auth0
Roles and permissions

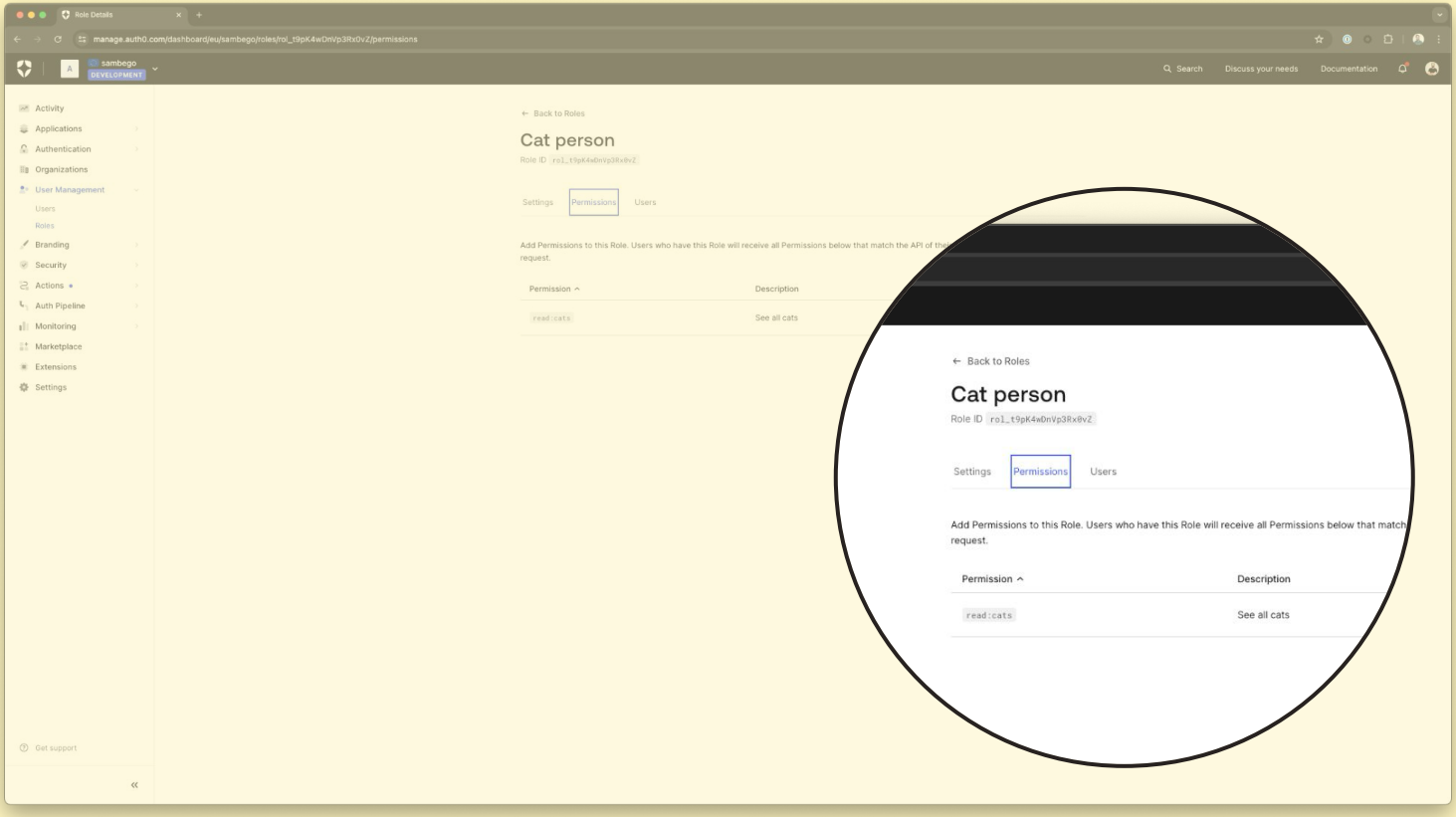
RBAC straight out of the box



Paradigm *shift*

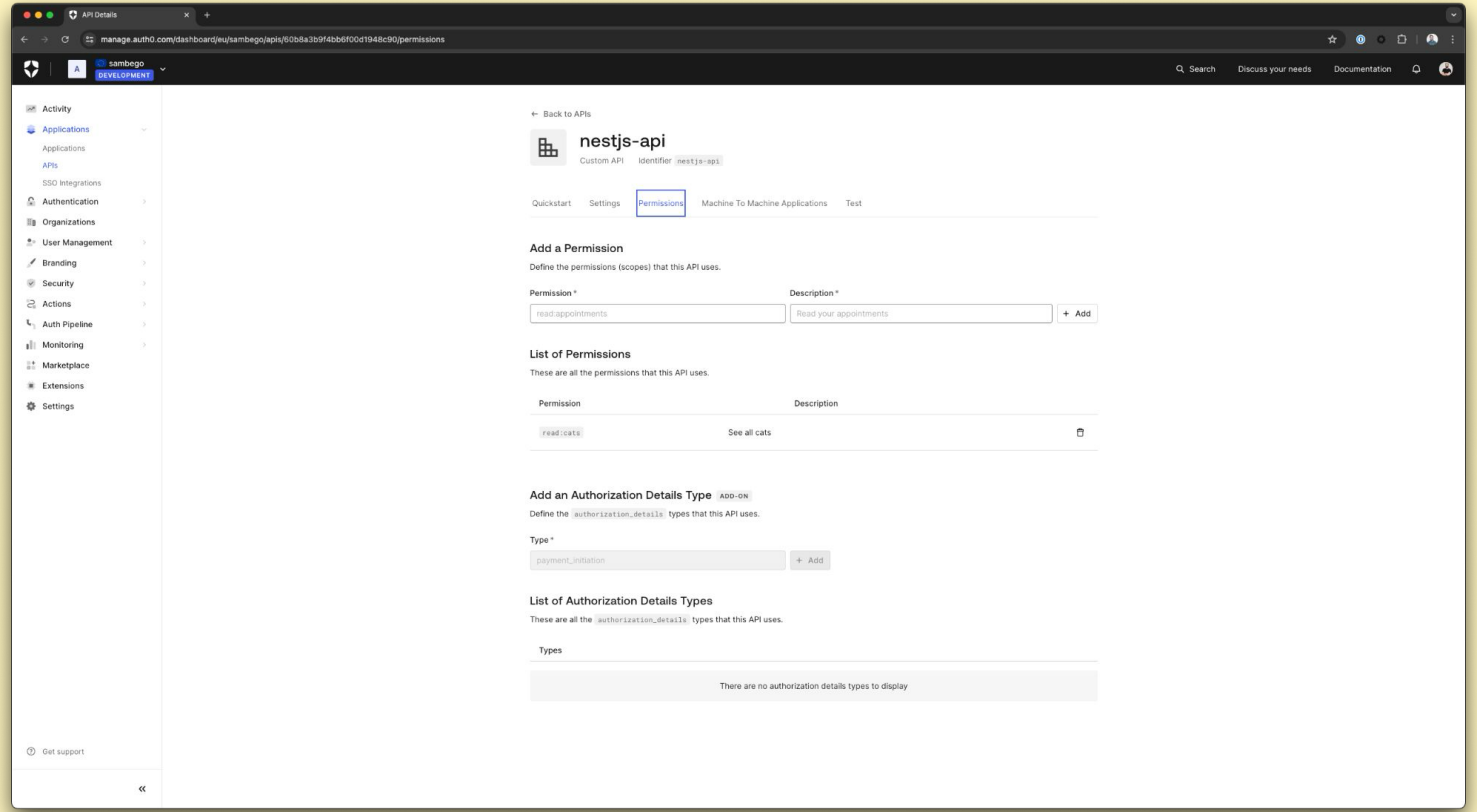
Manage ***roles*** for your users in the user management section of the dashboard

Paradigm
shift



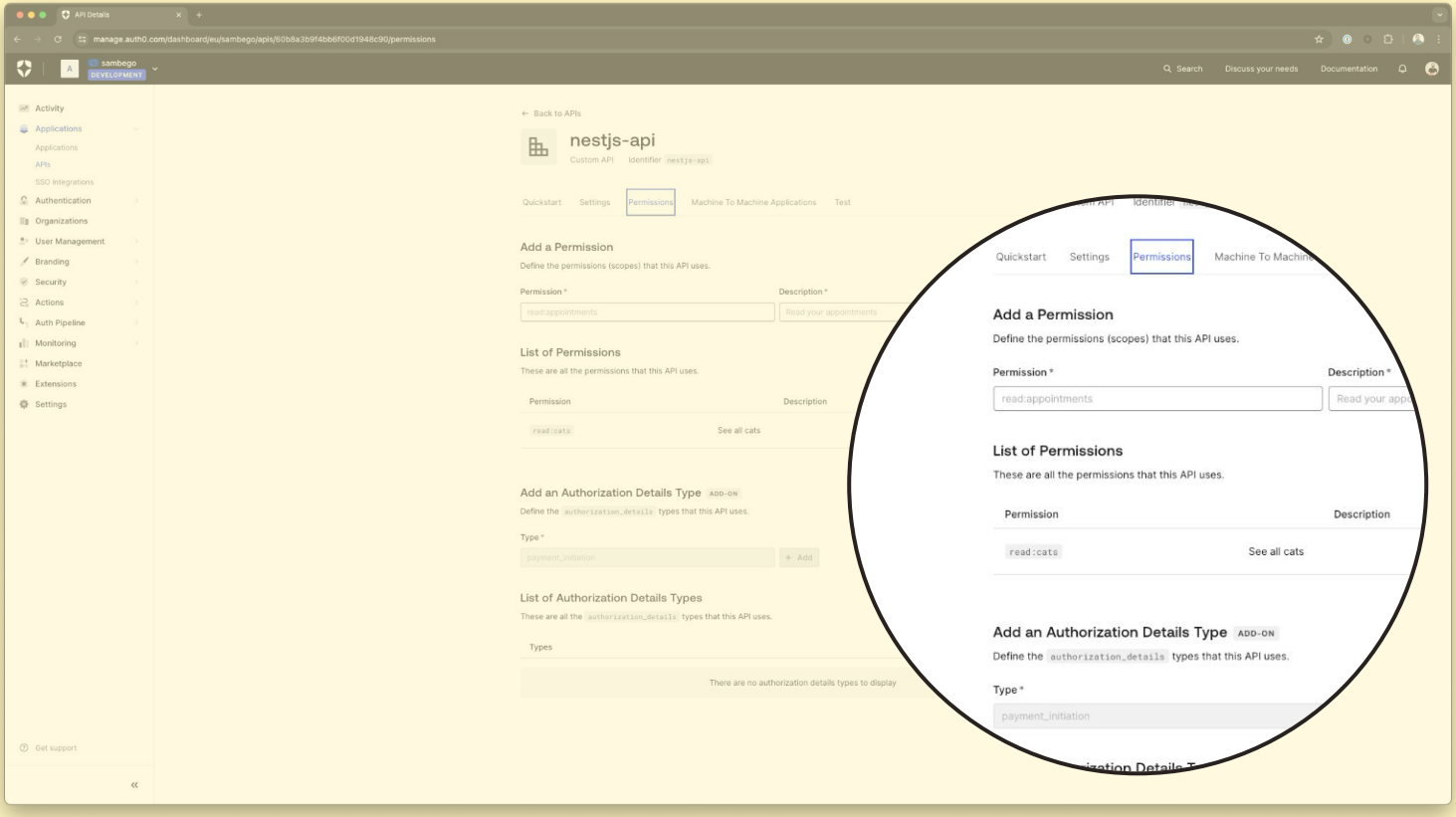
Manage ***roles*** for your users in the user management section of the dashboard

Paradigm *shift*

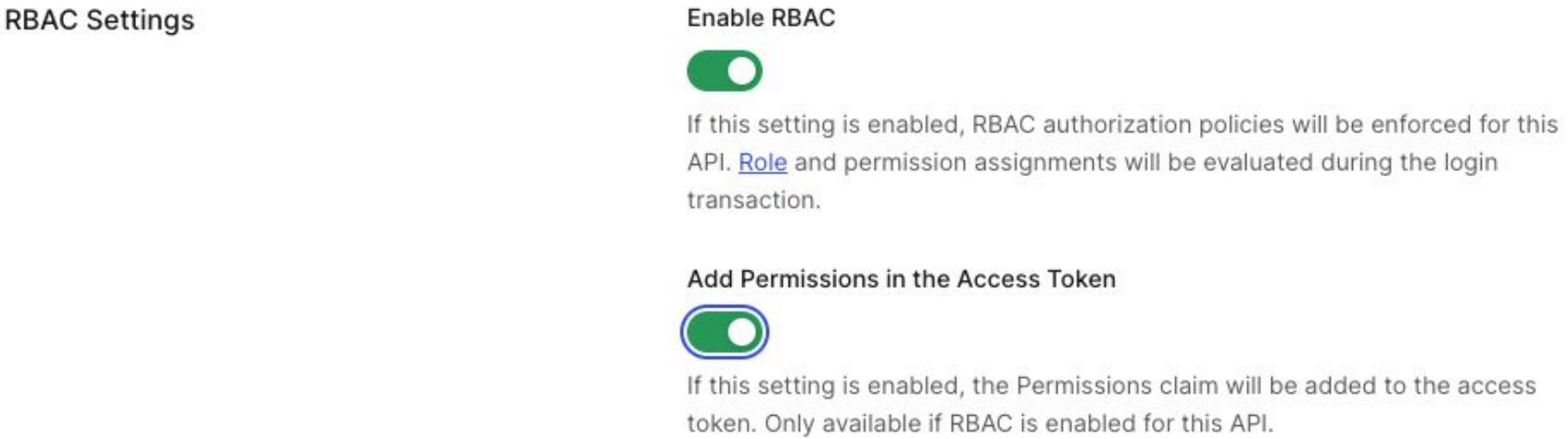


Manage *permissions* for your APIs

Paradigm
shift

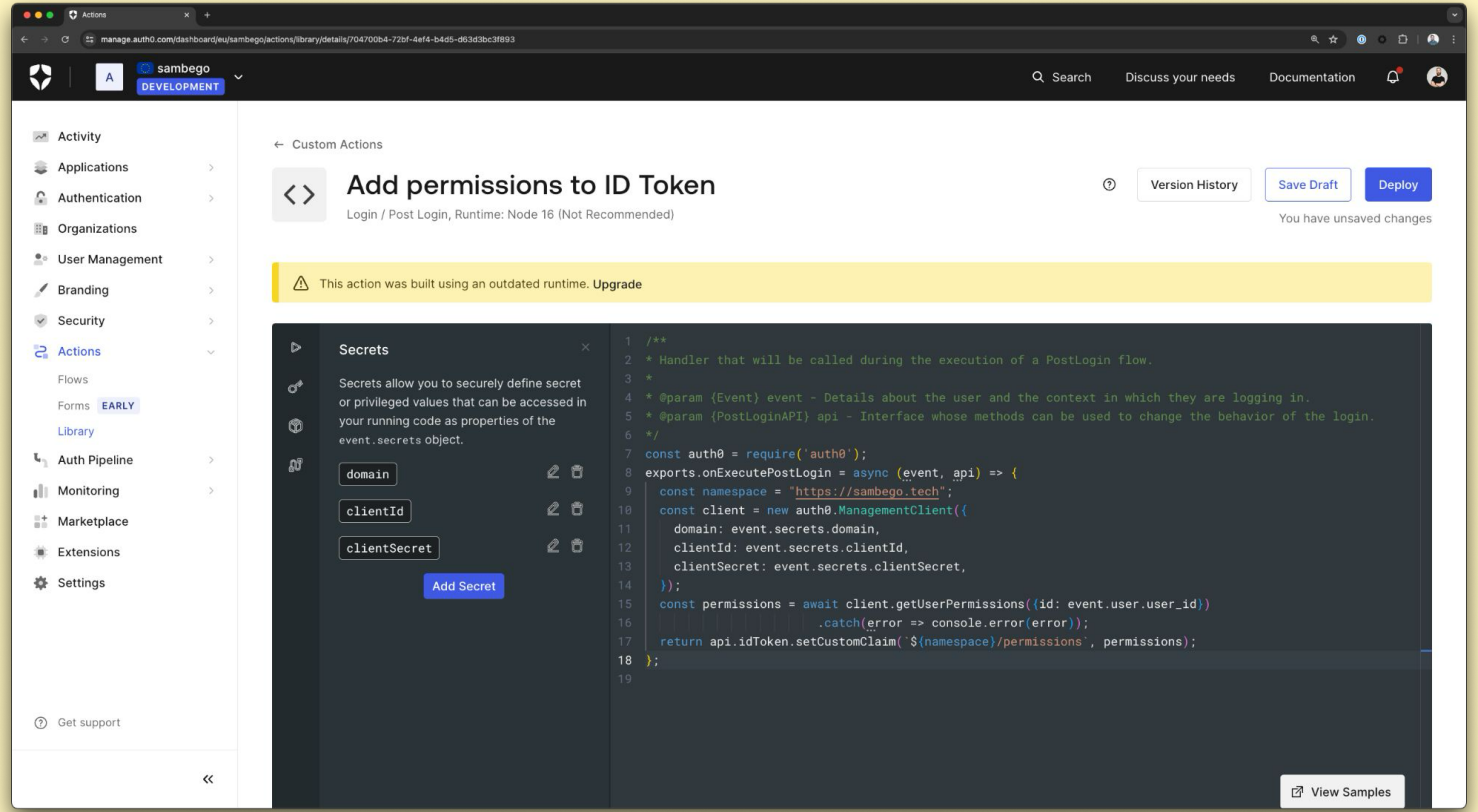


Manage *permissions* for your APIs



Add *permissions* to the access tokens issued by Auth0

Paradigm *shift*



Add *permissions* to the ID token trough Auth0 actions

Watch out for token bloat!

Paradigm
shift

ABAC

Attribute-based Access control

Different kinds of ***attributes*** influence the decision

Decisions are based on **attributes** related to the action being evaluated

Attributes

User

Environment

Resource

Action

User

Role

Organization

Security clearance

Environment

Time of day

Location of data

Code-freeze

Current threat level

Resource

Creation date

Owner

Data Sensitivity

Action

Read

Write

Delete

Paradigm
shift

PBAC

Policy-based Access control

Combine attributes
in ***policies***

Declare scenarios with ***attributes*** in one or more **policy**

A *policy engine* will evaluate access control decisions

Example

An accountant can upload an invoice, if it is during their working hours.

Example

An accountant can upload an invoice, if it is during their working hours.

User

Example

An **accountant** can **upload** an **invoice**, if it is **during** their working hours.

Action

Example

An **accountant** can **upload** an invoice, if it is **during** their working hours.

Resource

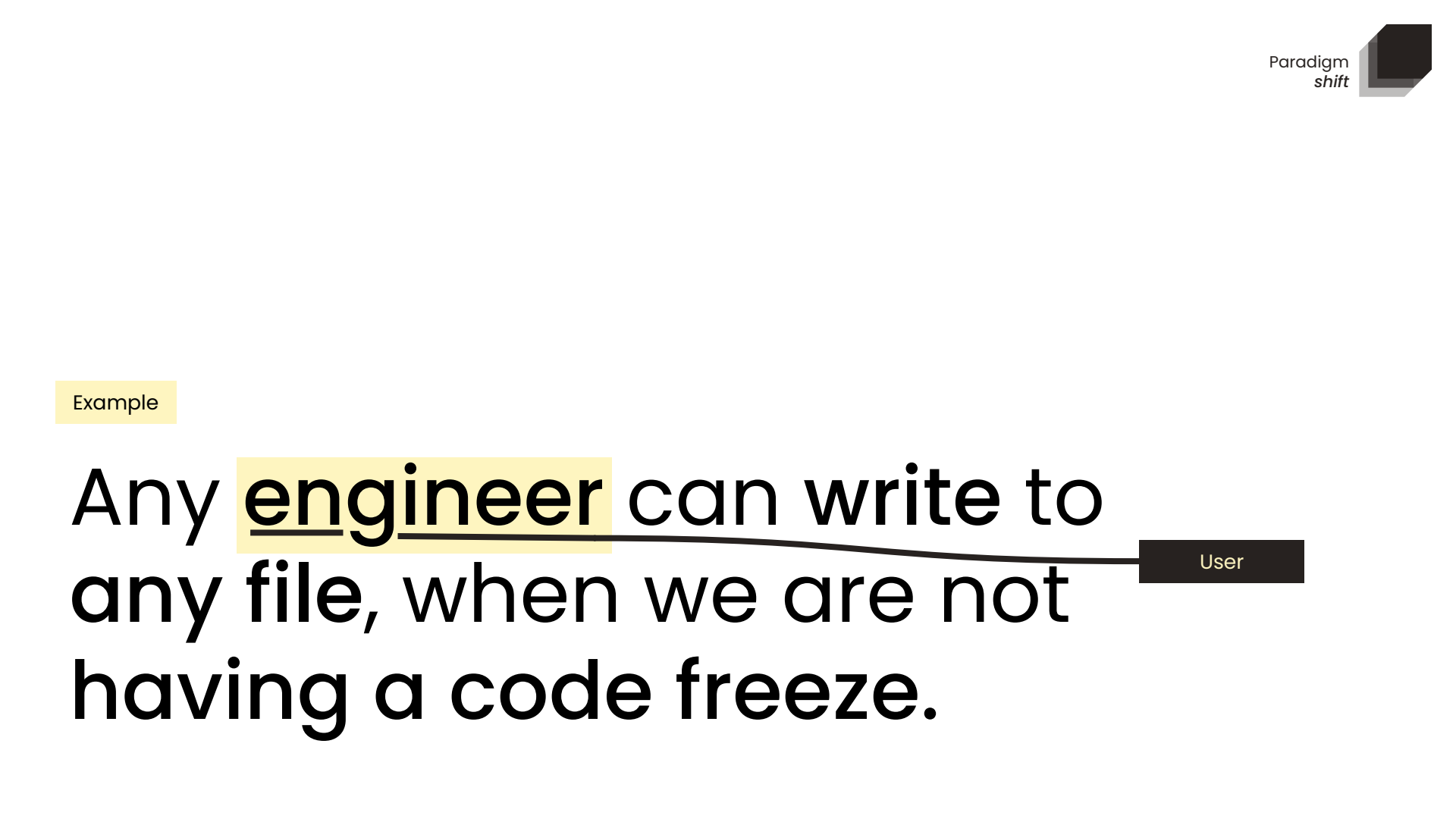
Example

An accountant can upload an invoice, if it is during their working hours.

Environment

Example

Any **engineer** can **write** to **any file**, when we are not **having a code freeze.**



Example

Any engineer can write to any file, when we are not having a code freeze.

User

Example

Any **engineer** can **write** to **any file**, when we are not **having a code freeze.**

Action

Example

Any **engineer** can **write** to any file, when we are not having a code freeze.

Resource

Example

Any **engineer** can **write** to **any file**, when we are not **having a code freeze**.

Environment

ABAC is often used for managing infrastructure access

Scenarios where decisions are made on a **limited set of data**, the ***attributes***

Paradigm
shift

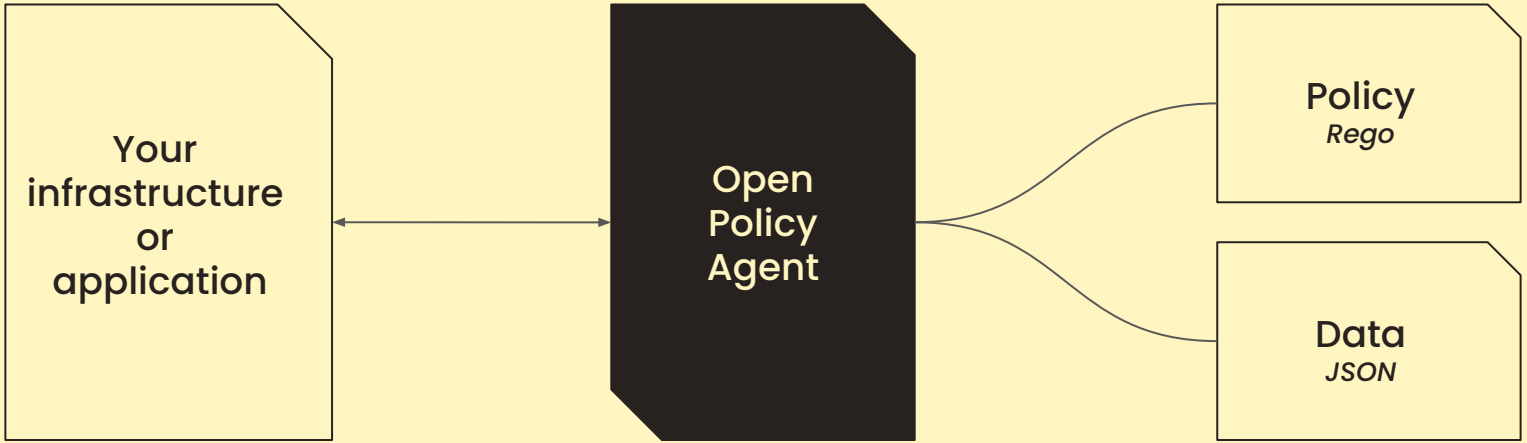
OPA

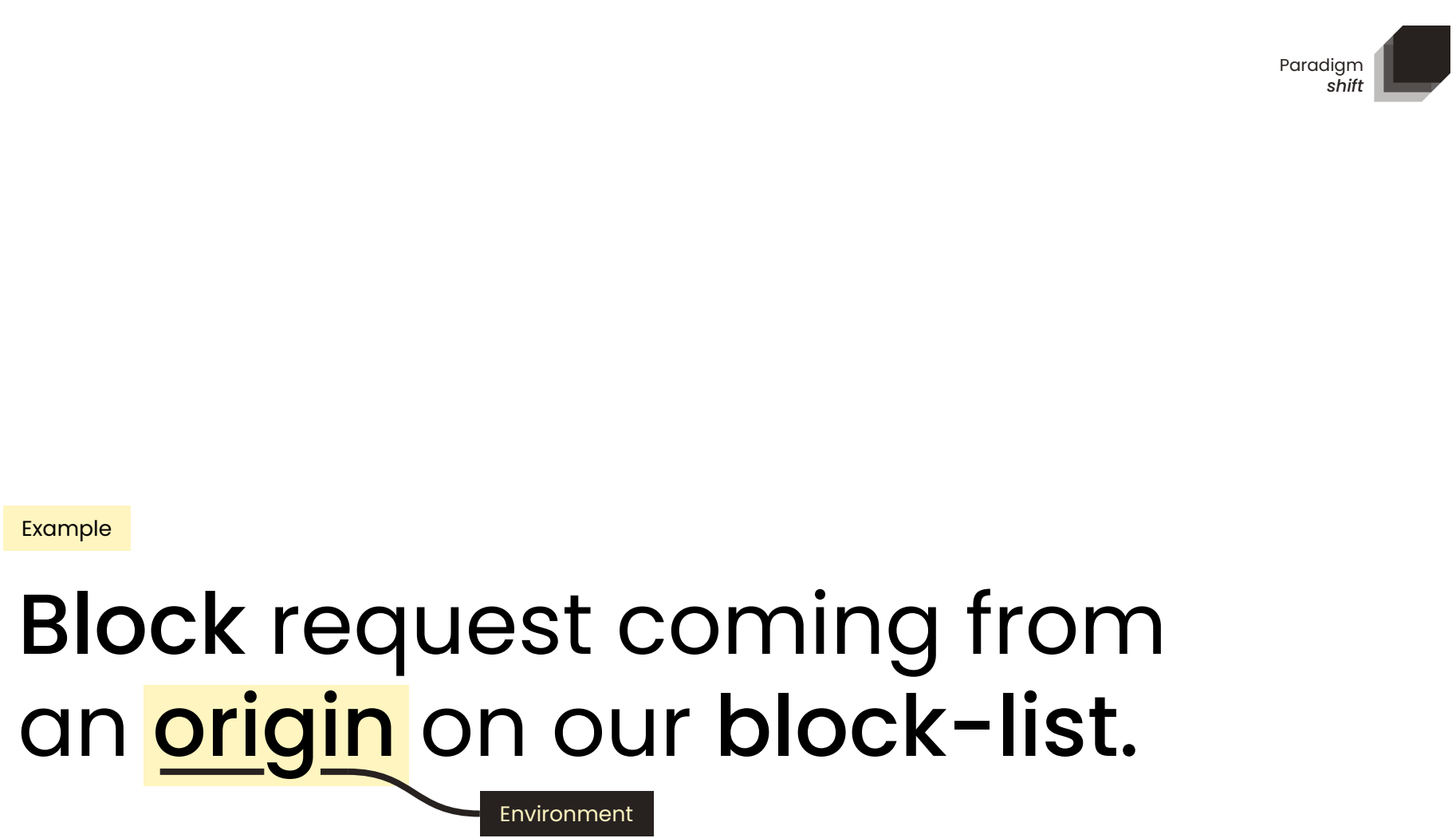
Open *Policy* Agent

Open-source ***policy engine***

OPA is a **decision-engine** that provides a way of declaratively writing **policies as code**.

It uses these *policies* as part of a **decision-making process**.

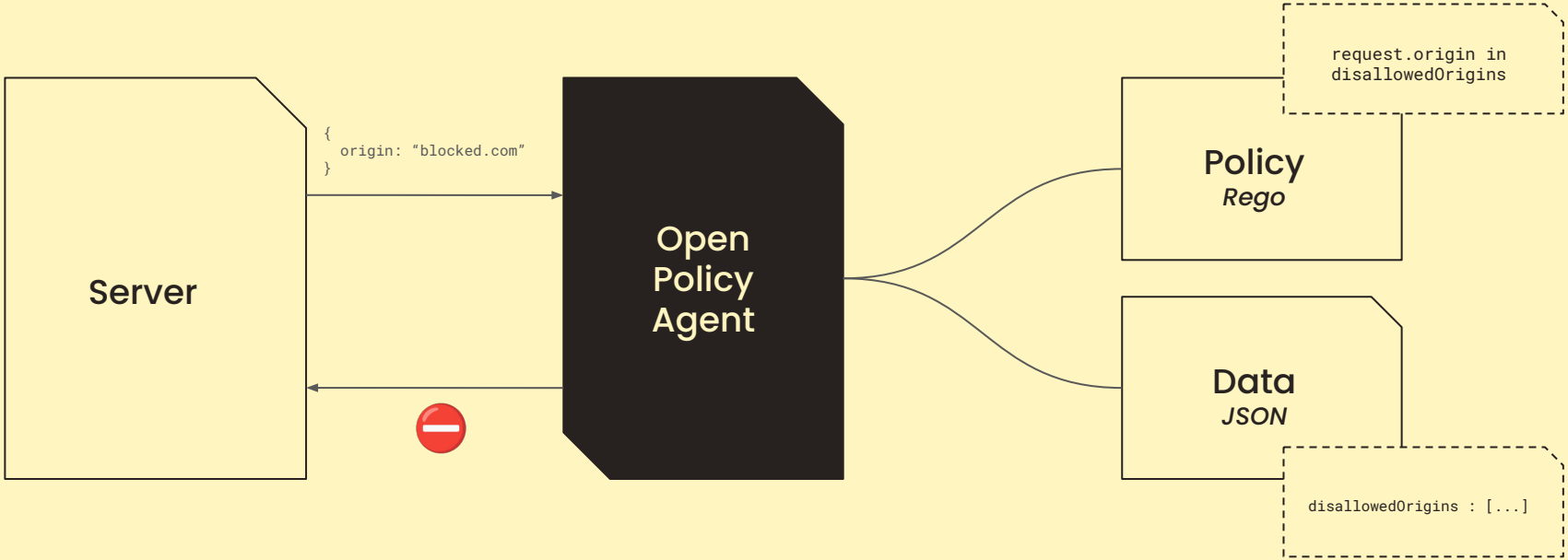

Your infrastructure or application
Open Policy Agent
Policy
Rego
Data
JSON


Example
Block request coming from
an origin on our block-list.
Environment

```
deny[reason] {
    disallowedOrigins := ["blocked.com","malicious.com"]

    some input.request.origin in disallowedOrigins
    reason := sprintf("origin %s has been blocked", [input.request.origin])
}
```

A policy written in the Rego language
used by OPA

Paradigm
shift



ABAC isn't optimized for large amounts of data that's continually changing.

Paradigm
shift

ReBAC

Relationship-based Access control

Look for ***relationships*** between resources

Access control decisions are based on relationships between a consumer and a resource

A ***consumer*** can be a user, group, folder, another resource...

Example

Sam is the owner of this slidedeck

Example

Sam is the owner of this slidedeck

Consumer

Example

Sam is the **owner** of **this slidedeck**

Resource

Example

Sam is the owner of this slidedeck

Relation

There are direct and indirect *relationships*.

Direct relationships are explicitly defined

Indirect relationships are derived from the direct relationships

Example

Sam is a **member of Auth0**, and can therefor **view all slide decks in our organisation**.

Example

Sam is **indirectly related** to this slide deck.

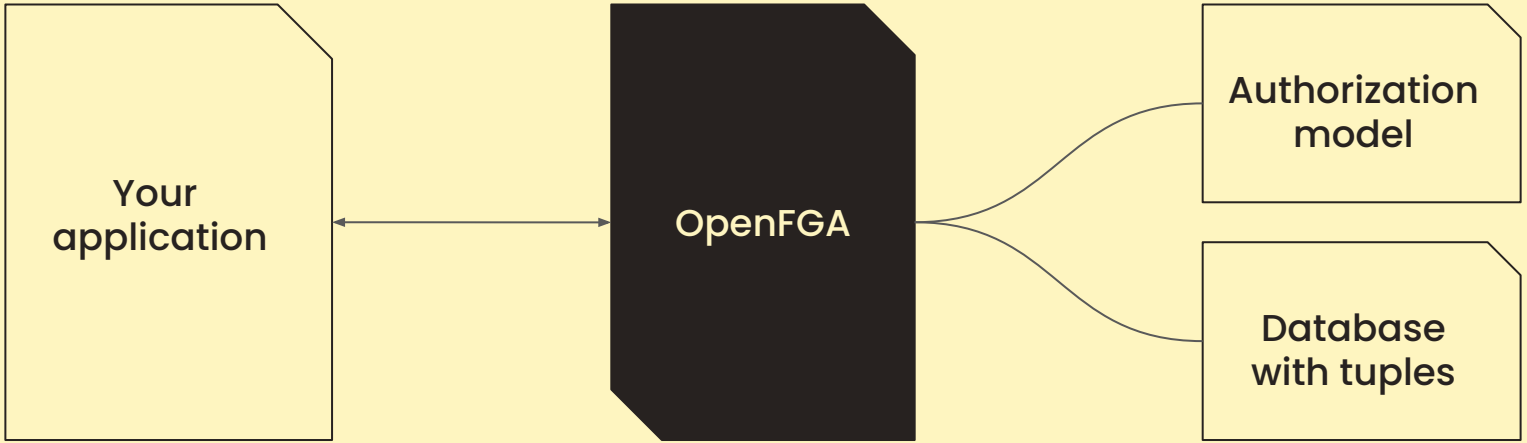
Paradigm
shift

OpenFGA

An ***open-source ReBAC solution***, a CNCF sandbox project

OpenFGA is a **decision-engine** that makes decisions based on an **authorization model** in combination with **tuples saved in a database**.

These *tuples* are the **direct relationships**.


Your application
OpenFGA
Authorization model
Database with tuples

Example

Check if a user can view a GitHub repository

Relation

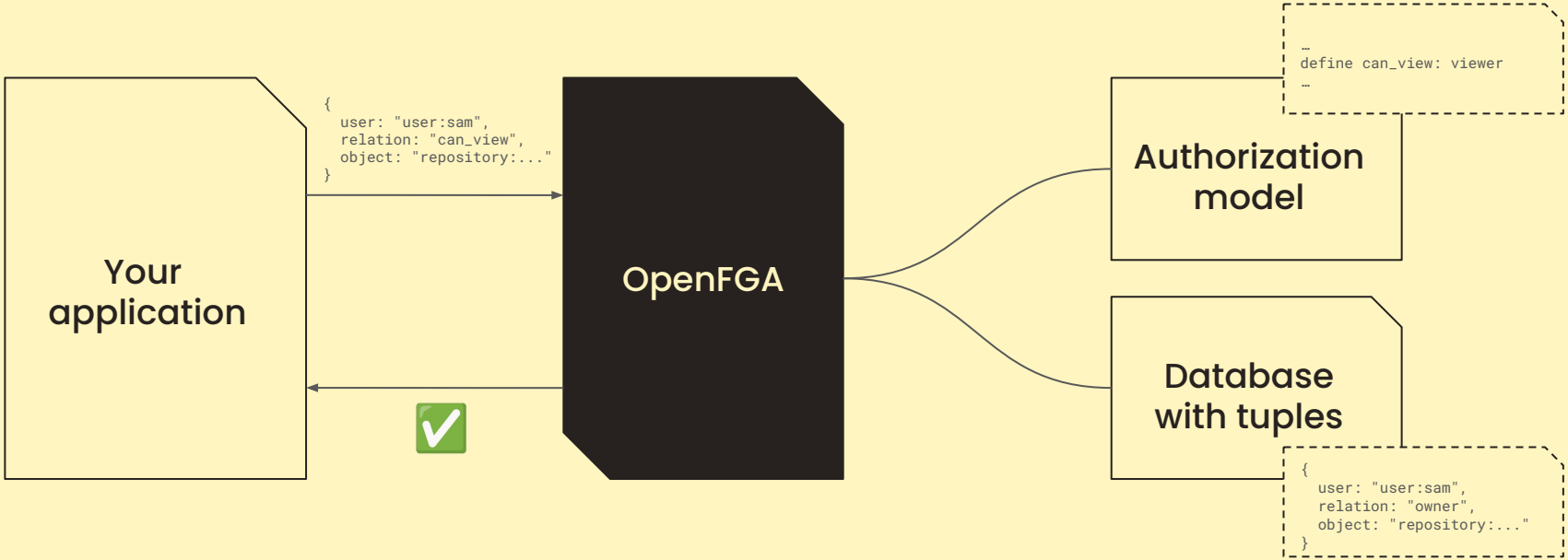
```
type user
type repository
  relations
    define owner: [user]
    define viewer: [user] or owner
    define can_view: viewer or owner
```

Paradigm *shift*

```
{
  user: "user:sam",
  relation: "owner",
  object: "repository:fga-demo"
}
```

An *OpenFGA authorization model* and tuple

Paradigm
shift



Example

Check if a user **can commit** to a GitHub repository

Relation

```
type user

type organization
  relations
    define member: [user]
    define repo_writer: [user] or member

type repository
  relations
    define owner: [organization]
    define can_commit: repo_writer from owner
```

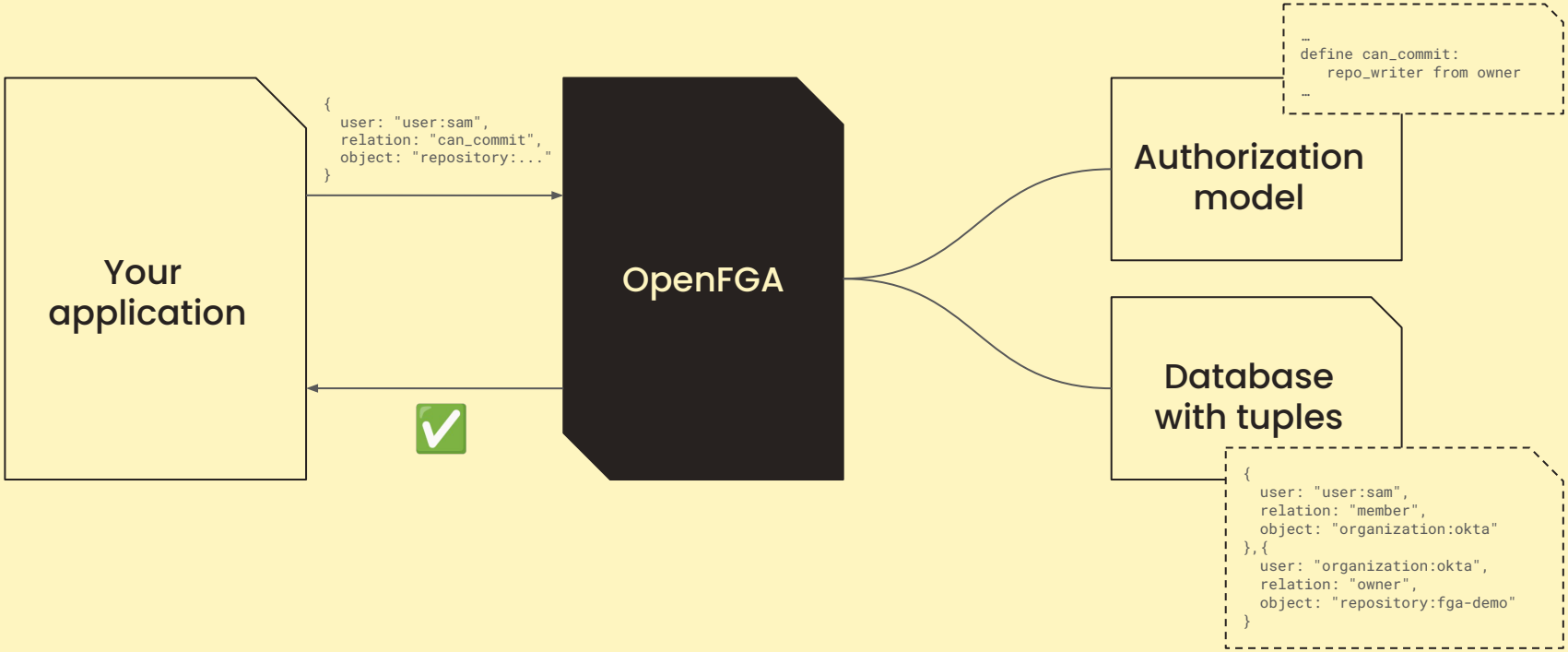
An OpenFGA *authorization model*

Paradigm
shift

```
{
  user: "user:sam",
  relation: "member",
  object: "organization:okta"
},{
  user: "organization:okta",
  relation: "owner",
  object: "repository:fga-demo"
}
```

OpenFGA *tuples*

Paradigm
shift



Example

Retrieval-augmented generation (RAG): Retrieve financial documents assigned to a user

Relation

Paradigm
shift

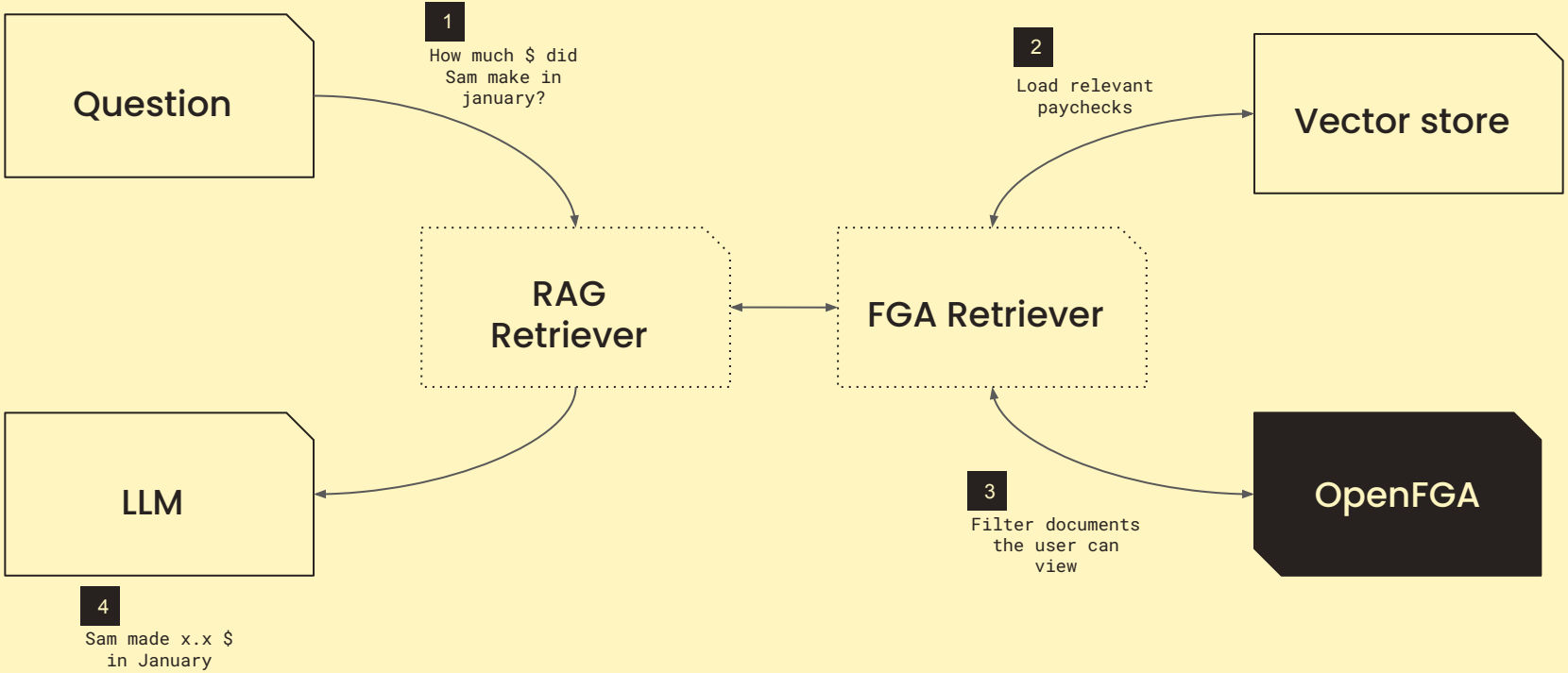
```
type user

type document
  relations
    define owner: [user]
    define can_view: [user] or owner
```

An OpenFGA *authorization model*

```
{
  user: "user:sam",
  relation: "owner",
  object: "document:paycheck-jan-2024-sam"
},{
  user: "user:hr-rudy",
  relation: "can_view",
  object: "document:paycheck-jan-2024-sam"
},{
  user: "user:jane",
  relation: "owner",
  object: "document:paycheck-jan-2024-jane"
},{
  user: "user:hr-rudy",
  relation: "can_view",
  object: "document:paycheck-jan-2024-jane"
}
```

OpenFGA *tuples*

Paradigm *shift*



Paradigm
shift

There’s an SDK for that!

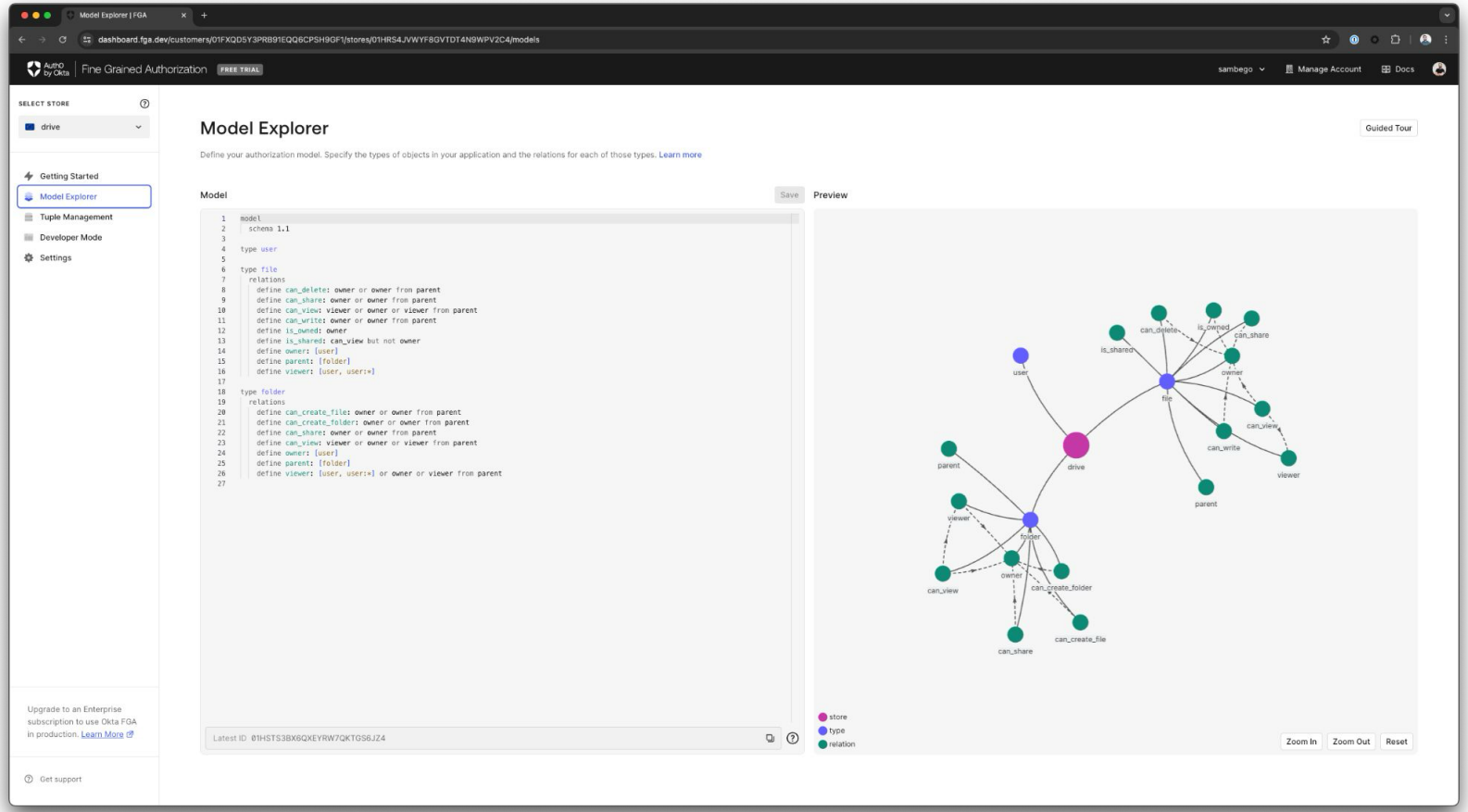
a0.to/ato-fga-ai

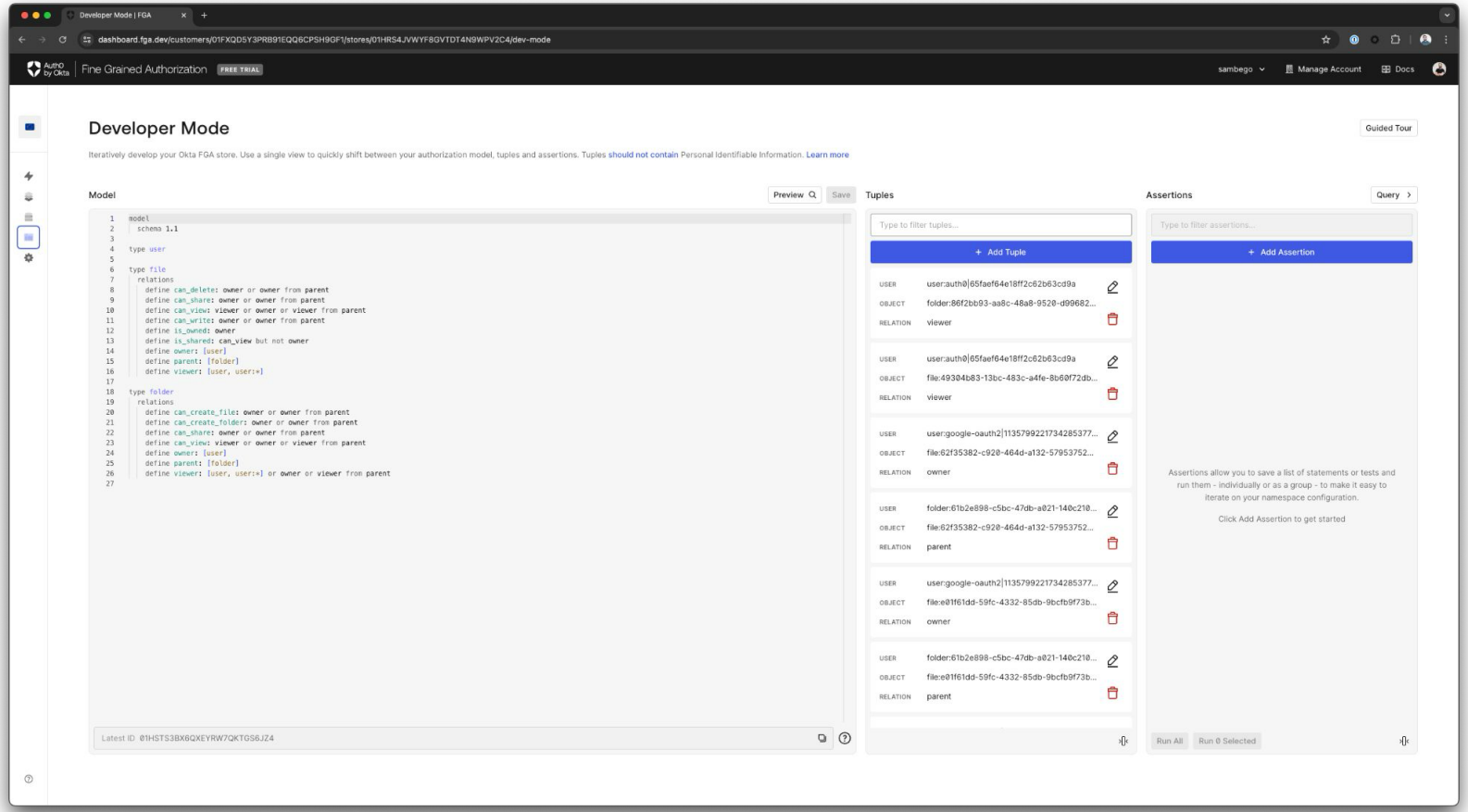
Paradigm
shift

Okta FGA

A ***managed*** OpenFGA service

Paradigm
shift




Developer Mode | FGA
dashboard.fga.dev/customers/01FXQD5Y3PRB91EQQ6CPSH9GF1/stores/01HRS4JVWYF8GVTDT4N9WPV2C4/dev-mode
Auth0 by Okta | Fine Grained Authorization
FREE TRIAL
sambego
Manage Account
Docs
Developer Mode
Guided Tour
Iteratively develop your Okta FGA store. Use a single view to quickly shift between your authorization model, tuples and assertions. Tuples should not contain Personal Identifiable Information. Learn more
Model
Preview
Save
model
schema 1.1
type user
type file
relations
define can_delete: owner or owner from parent
define can_share: owner or owner from parent
define can_view: viewer or owner or viewer from parent
define can_write: owner or owner from parent
define is_owned: owner
define is_shared: can_view but not owner
define owner: [user]
define parent: [folder]
define viewer: [user, user:*]
type folder
relations
define can_create_file: owner or owner from parent
define can_create_folder: owner or owner from parent
define can_share: owner or owner from parent
define can_view: viewer or owner or viewer from parent
define owner: [user]
define parent: [folder]
define viewer: [user, user:*] or owner or viewer from parent
Latest ID 01HSTS3BX6QXEYRW7QKTGS6JZ4
Tuples
Type to filter tuples...
+ Add Tuple
USER user:auth0|65faef64e18ff2c62b63cd9a
OBJECT folder:86f2bb93-aa8c-48a8-9520-d99682...
RELATION viewer
USER user:auth0|65faef64e18ff2c62b63cd9a
OBJECT file:49304b83-13bc-483c-a4fe-8b60f72db...
RELATION viewer
USER user:google-oauth2|113579922173428537...
OBJECT file:62f35382-c920-464d-a132-57953752...
RELATION owner
USER folder:61b2e898-c5bc-47db-a021-140c210...
OBJECT file:62f35382-c920-464d-a132-57953752...
RELATION parent
USER user:google-oauth2|113579922173428537...
OBJECT file:e01f61dd-59fc-4332-85db-9bcfb9f73b...
RELATION owner
USER folder:61b2e898-c5bc-47db-a021-140c210...
OBJECT file:e01f61dd-59fc-4332-85db-9bcfb9f73b...
RELATION parent
Assertions
Query
Type to filter assertions...
+ Add Assertion
Assertions allow you to save a list of statements or tests and run them - individually or as a group - to make it easy to iterate on your namespace configuration.
Click Add Assertion to get started
Run All
Run 0 Selected

ReBAC is a great solution for applications that deal with user-generated content, that is constantly changing

Paradigm
shift

Centralized access control

Where should your authorization decisions be made?

Modern ABAC and ReBAC solutions have a decision engine that centralizes decisions from your applications

This is useful for audits!

All applications will use the same decisions

*Hybrid**
access control

Modern tools can support ***multiple paradigms***

Some access control solutions have support for multiple methods, working around limitations inherent to their original strategy

Limitation

ABAC does not work well with dynamic data

OPAL adds an administration layer to *OPA*, to keep **policies and data in sync** across agents

TOPAZ brings “**real-time,** policy based access control for **applications and APIs**” to *OPA*

Limitation

ReBAC does not take **context or environment** into consideration

OpenFGA has ABAC support through contextual tuples and conditions

Standards?

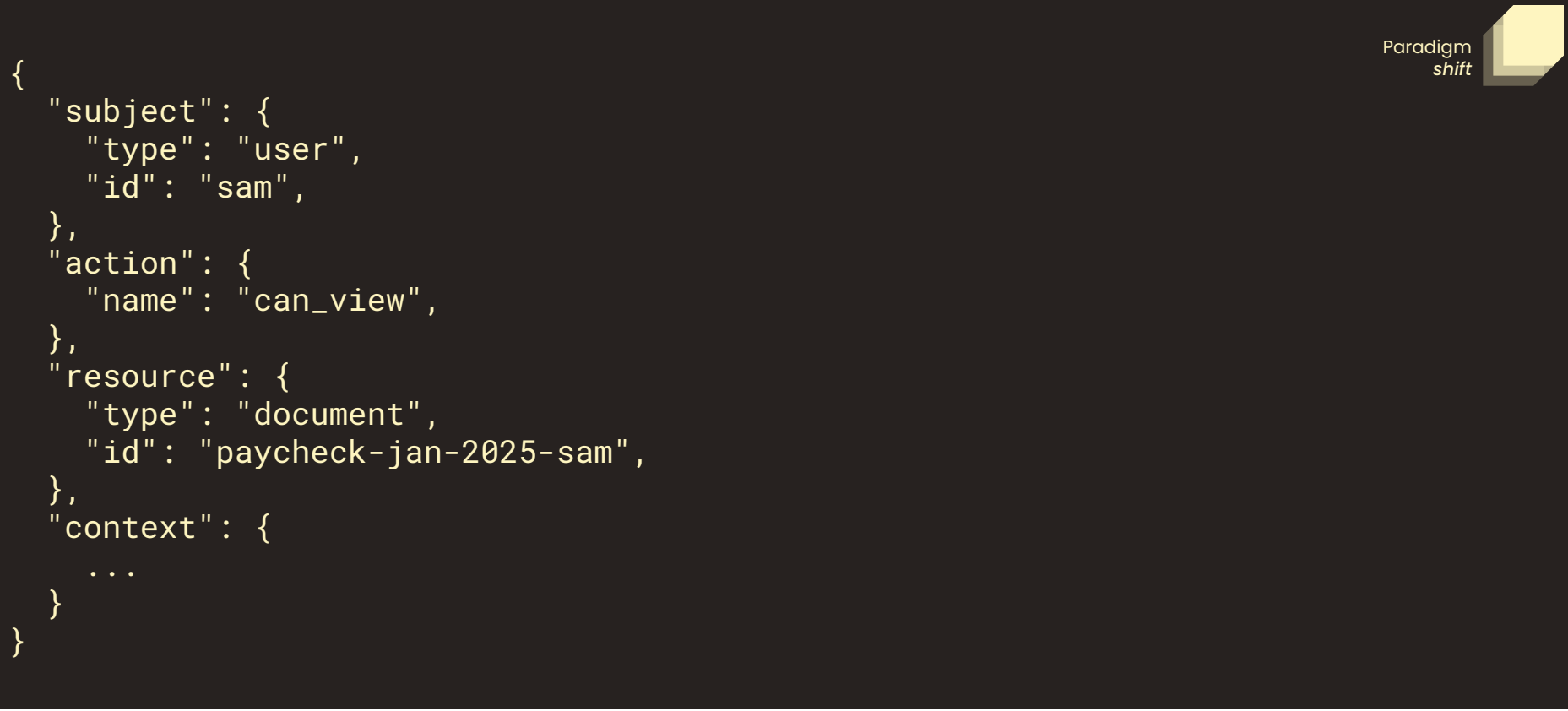
Available *authorization standards*

XACML, extensible access control markup language

~~*XACML*, extensible access control markup language~~

AuthZEN provide standard mechanisms to **communicate authorization related information** from different entities

AuthZEN is currently a fourth (and final?) draft at the *OpenID Foundation*.



Paradigm *shift*

AuthZEN evaluation payload

`pdp.example.com/access/evaluation`

```
{
  "decision": true
}
```

AuthZEN evaluation response

`pdp.example.com/access/evaluation`

Paradigm *shift*

```
{
  "decision": false,
  "context": {
    "id": "0",
    "reason_admin": {
      "en": "No relation between user and object"
    },
    "reason_user": {
      "en-403": "Not allowed. Contact your administrator",
      "it-403": "Non consentito. Contatta l'amministratore"
    }
  }
}
```

AuthZEN evaluation response

`/evaluation`

Paradigm *shift*

```
{
  "subject": {
    "type": "user",
    "id": "sam",
  },
  "action": {
    "name": "can_view",
  },
  "resource": {
    "type": "document",
    "id": "paycheck-jan-2025-sam",
  },
  "context": {
    ...
  }
}
```

User

Relation

Object

AuthZEN evaluation payload for OpenFGA

Paradigm *shift*

```
{
  "subject": {
    "type": "user",
    "id": "sam",
  },
  "action": {
    "name": "can_view",
  },
  "resource": {
    "type": "document",
    "id": "paycheck-jan-2025-sam",
  },
  "context": {
    ...
  }
}
```

User attribute

Resource attribute

AuthZEN evaluation payload for OPA

```
{
  "subject": {
    "type": "user",
  },
  "action": {
    "name": "can_view",
  },
  "resource": {
    "type": "document",
    "id": "paycheck-jan-2025-sam",
  },
  "context": {
    ...
  }
}
```

AuthZEN subject search payload

`pdp.example.com/access/subject`

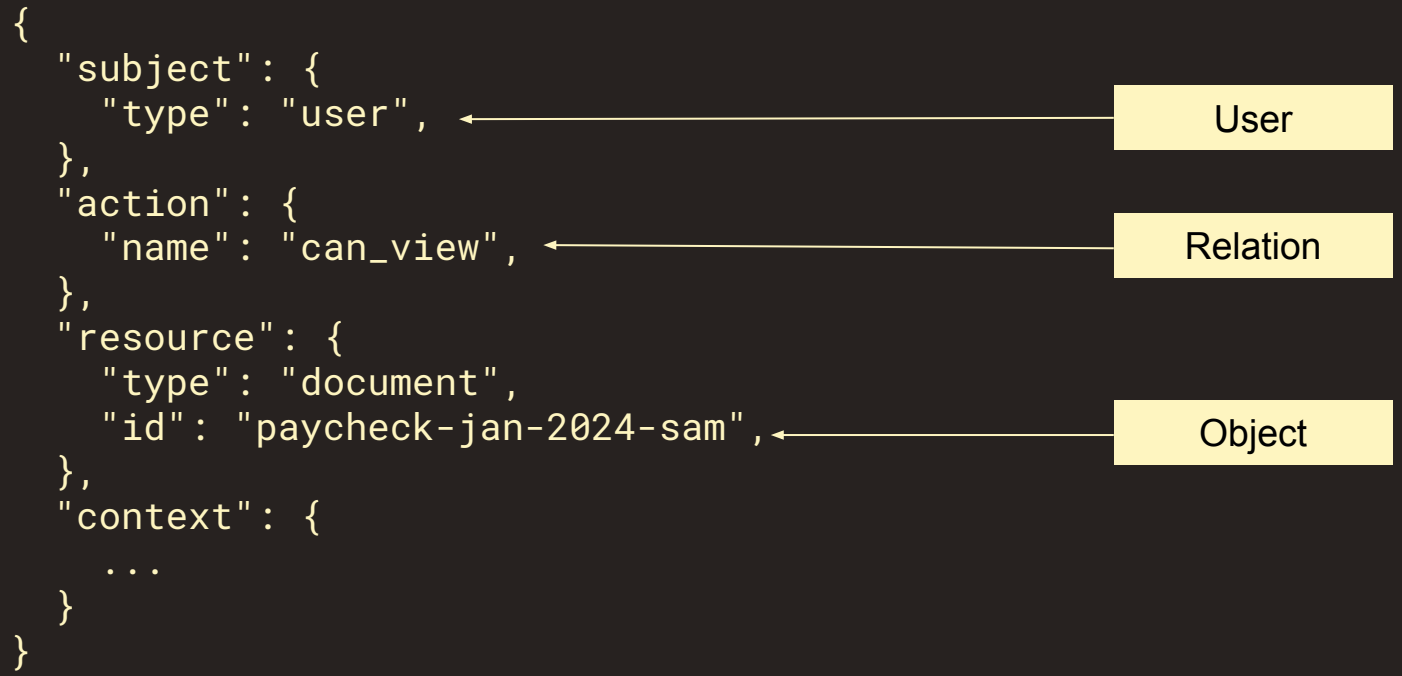
Paradigm
shift

```
{
  "results": [
    {
      "type": "user",
      "id": "sam"
    },
    {
      "type": "user",
      "id": "jane"
    }
  ]
}
```

AuthZEN subject search response

`pdp.example.com/access/subject`

Paradigm *shift*



AuthZEN subject search payload for OpenFGA

Paradigm *shift*

```
{
  "subject": {
    "type": "user",
  },
  "action": {
    "name": "can_view",
  },
  "resource": {
    "type": "document",
    "id": "paycheck-jan-2024-sam",
  },
  "context": {
    ...
  }
}
```

User attribute

Resource attribute

AuthZEN subject search payload for OPA

```
{
  "subject": {
    "type": "user",
    "id": "sam"
  },
  "action": {
    "name": "can_view",
  },
  "resource": {
    "type": "document",
  },
  "context": {
    ...
  }
}
```

AuthZEN resource search payload

`pdp.example.com/access/resource`

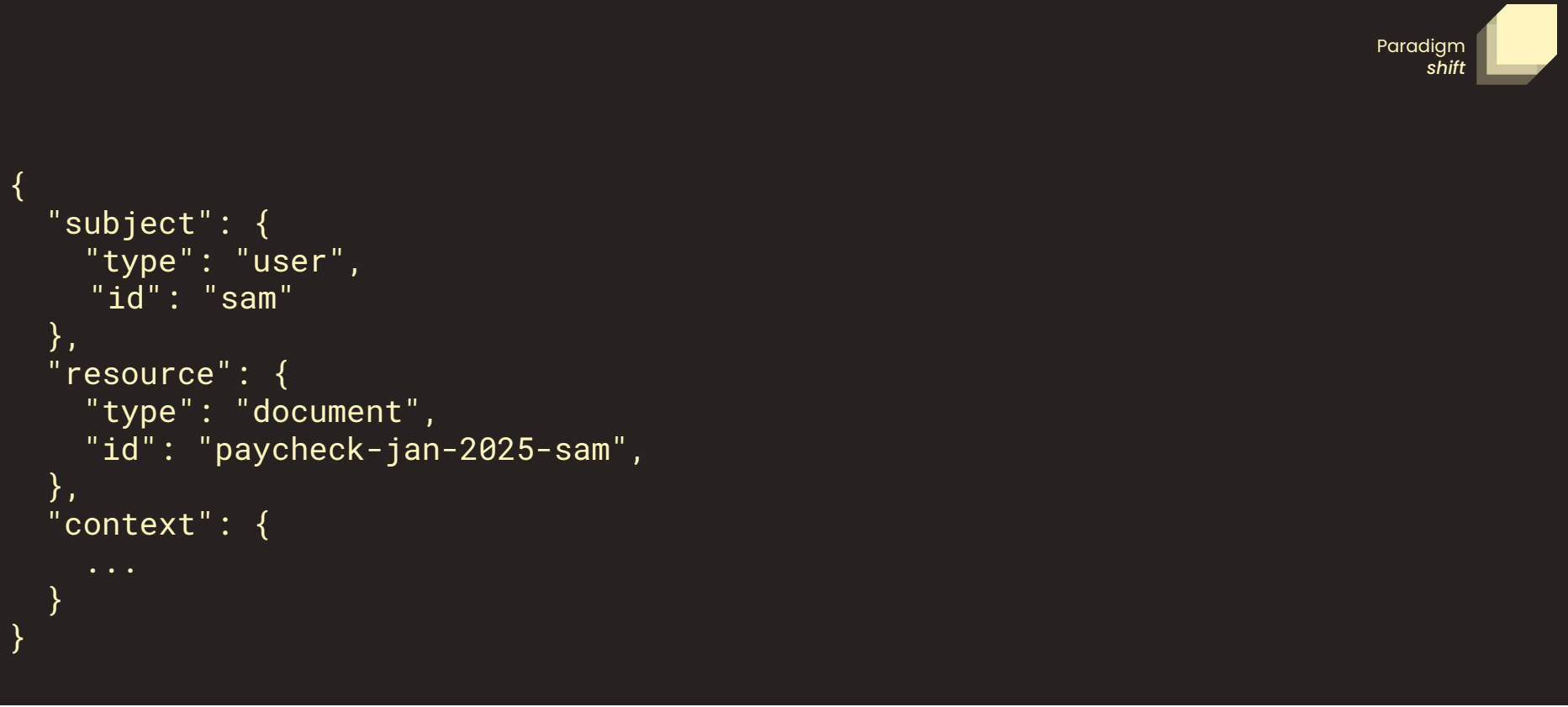
Paradigm *shift*

```
{
  "results": [
    {
      "type": "document",
      "id": "paycheck-jan-2025-sam"
    },
    {
      "type": "document",
      "id": "paycheck-feb-2025-sam"
    }
  ]
}
```

AuthZEN resource search response

`pdp.example.com/access/resource`

Paradigm *shift*



```
{
  "subject": {
    "type": "user",
    "id": "sam"
  },
  "resource": {
    "type": "document",
    "id": "paycheck-jan-2025-sam",
  },
  "context": {
    ...
  }
}
```

AuthZEN action search payload

`pdp.example.com/access/action`

Paradigm
shift

```
{
  "results": [
    {
      "name": "can_view"
    },
  ]
}
```

AuthZEN action search response

`pdp.example.com/access/action`

Let’s *recap*

TL;DR

There are *different access control strategies*, **they all have their purpose!**

RBAC is good for applications that don't let their **users generate content**.

ABAC is great for fine-grained control based on a **limited set of attributes**.

ReBAC can evaluate a **large database of direct relationships**, and **deduct indirect ones** from this dataset.

This is ***perfect*** for complex applications with **an ever changing set of data** that influences the access control decision.

AuthZEN is an **open standard** to provide a common way to relay access control decisions, **regardless of strategy** or decision engine

Paradigm
shift

Sam Bellen

Principal developer advocate at Auth0

@sambego
sambego.tech

Paradigm
shift

Learn more

a0.to/ato-fga

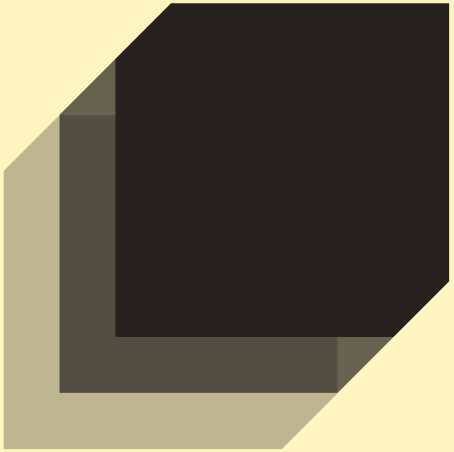
@sambego
sambego.tech

Find these slides

slides.sambego.tech/paradigm-shift

@sambego
sambego.tech

Thank *you*!



Paradigm *shift*

Moving beyond roles and permissions
to a **fine-grained** access control