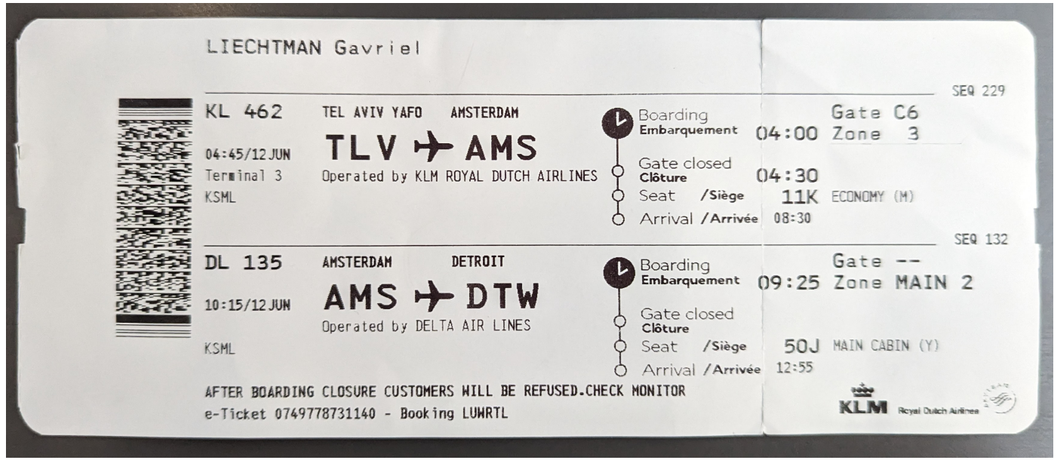
Love, Hate, and Policy Languages

Gabriel L. Manor @ DevOpsKC, August 2023

Permit.io

@gemanor

Find the Difference



Permit.io

@gemanor

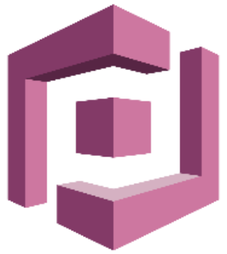
	Passport	Flight Ticket
Form of	Identity	Authority
Purpose	Verifies identity	Grants access to a flight
Scope	Global	Flight-specific
Issued by	Government	Airline desk, app, website, etc.
Information	Name, photo, birthdate, etc.	Name, flight number, seat, gate, etc.
Validity	Multiple years	One flight
Used	Once per flight	Multiple times per flight
Uniqueness	Unique to an individual	Unique to a flight and passenger
Changeable	No	Yes
Permissions	One (to travel)	Multiple (to board, to check bags, etc.)
Revocation granularity	All at once	One permission at a time
Transferable	No	Yes

	Authentication	Authorization
Purpose	Verifies user identity	Determines user permissions
Scope	Applies to all users	Specific to each user's role or status
Issued by	Identity provider	Any kind of data
Information	Username, social identity, biometrics, etc.	User roles, permissions, policy, external data, etc.
Validity	Until credentials change or are revoked	Per permissions check
Used	Once per session (typically)	Multiple times per session
Uniqueness	Unique to an individual user	Unique to a principal, action and resource
Changeable	Require session revocation	Yes (permissions can be updated, etc.)
Permissions	One (to access the system)	Multiple (to read, write, update, delete, etc.)
Revocation granularity	All at once (user is denied access)	One permission at a time
Transferable	No (credentials should not be shared)	Yes (policy can be applied to other users)

Authentication Advanced Features

- Multi-factor authentication
- Single sign-on / Social login / Passwordless
- User / account management
- Session management
- User registration / UI flows / customizations
- Account verification / recovery
- Audit / reporting / analytics / compliance
- Third party integrations

Auth0



clerk

STYTCH

frontegg

SuperTokens

Permit.io

@gemanor

We Will Do RBAC Later

- Every developer on every new application

Permit.io

@gemanor

The Auth Comfort Zone

Authentication 😌

Verify who the user is

```
if (!user) {
    return;
}
```

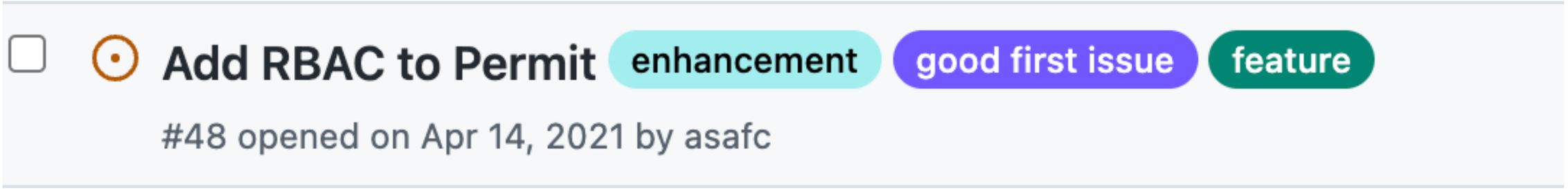
Authorization 😔

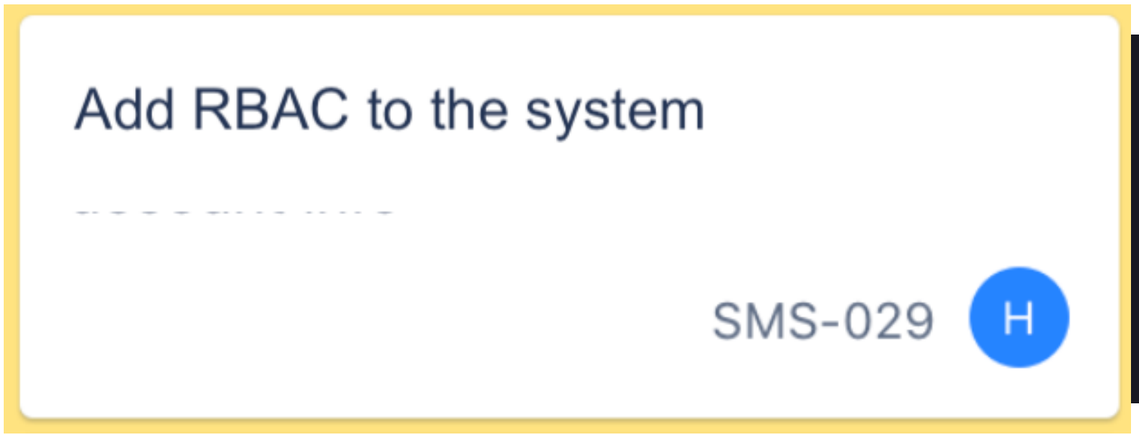
Check what a user can do

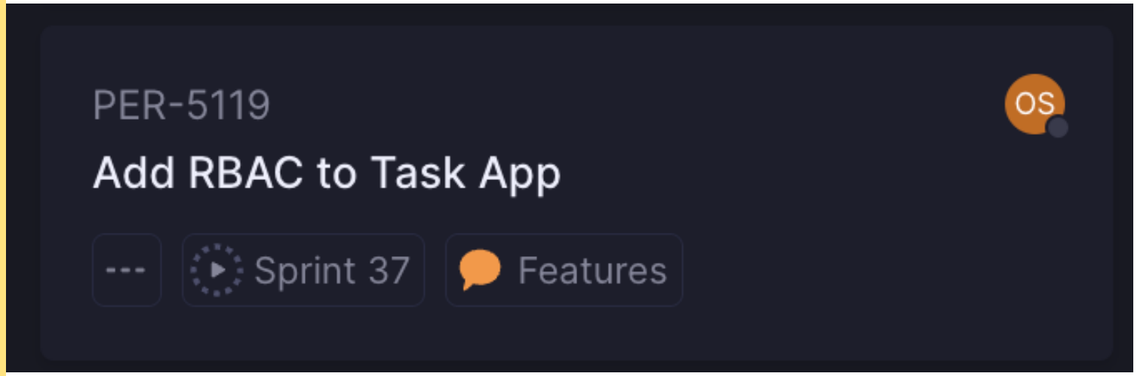
```
if (!allowed(
        user,
        action,
        resource
    )) {
    return;
}
```

Permit.io

@gemanor







Permit.io

@gemanor

“Nah, man. RBAC is pretty easy.”

- Derek, age 24

Gabriel L. Manor

Director of DevRel @ Permit.io

🎖️ **Struggling with authorization for the last 8y**

Not an ethical hacker, zero awards winner, dark mode hater.

```
def delete_user(user_id):
    user = User.get(user_id)
    if user.role == 'admin':
        user.delete()
```

Permit.io

@gemanor

```python
# Middleware
def roles_required():
    ...
            if user.role == 'admin':
                return func(*args, **kwargs)
            else:
                raise Exception('User is not admin')
    ...

@roles_required('admin')
def delete_user(user_id):
    user = User.get(user_id)
    user.delete()
```

Permit.io

@gemanor

```python
# Middleware
def roles_required():
    ...
            if user.role == 'admin':
                return func(*args, **kwargs)
            else:
                raise Exception('User is not admin')
    ...
def permissions_required():
    ...
            if permissions == 'delete_user':
                return func(*args, **kwargs)
            else:
                raise Exception('User is not admin')
    ...
```

```python
# Business logic
@roles_required('admin')
@permissions_required('delete_user')
def delete_user(user_id):
    user = User.get(user_id)
    user.delete()
```

Permit.io

@gemanor

```python
# Middleware
def roles_required():
    ...
            if user.role == 'admin':
                return func(*args, **kwargs)
            else:
                raise Exception('User is not admin')
    ...

# Business logic
@roles_required('admin')
def enable_workflow(user_id):
    user = User.get(user_id)
    paid_tier = billing_service.get_user_tier(user_id) == 'paid'
    if not paid_tier:
        raise Exception('User is not on paid tier')
```

Permit.io

@gemanor

```
@roles_required('admin')
@permissions_required('enable_workflow')
def enable_workflow():
    user = User.get(user_id)
    step1 = workflow.run()
    step2 = workflow.run()
    if user.sms_enabled:
        send_sms_to_list(user.phone_number, 'Workflow is enabled')
```

Permit.io

@gemanor

Staging

```
@roles_required('admin')
@permissions_required('enable_workflow')
def enable_workflow():
    user = User.get(user_id)
    step1 = workflow.run()
```

Production

```
@roles_required('superadmin')
@permissions_required('enable_workflow')
def enable_workflow():
    user = User.get(user_id)
    step1 = workflow.run()
```

Permit.io

@gemanor

Django

```python
@permission_required('app_name.can_edit')
def my_view(request):
    # Your view logic here
    ...
```

Flask

```python
app = Flask(__name__)
login_manager = LoginManager(app)

class User(UserMixin):
    def __init__(self, id, role):
        self.id = id
        self.role = role

@login_manager.user_loader
def load_user(user_id):
    return User(user_id, 'admin')

@app.route('/admin')
@login_required
def admin():
    if current_user.role != 'admin':
        abort(403)
    ...
```

Permit.io

@gemanor

```python
if user.role == 'admin':
    if user.tier == 'paid':
        if user.sms_enabled:
            if user.phone_number:
                send_sms_to_list(user.phone_number, 'Workflow is enabled')
```

Authorization Best Practices

Declarative

Generic

Unified

Agnostic

Decoupled

Easy to audit

Permit.io

@gemanor

#1 Model

User | Action | Resource

Does *[Principal]* Allowed to Perform *[Action]* on *[Resource]*

Is a Monkey Allowed to Eat a Banana

Permit.io

@gemanor

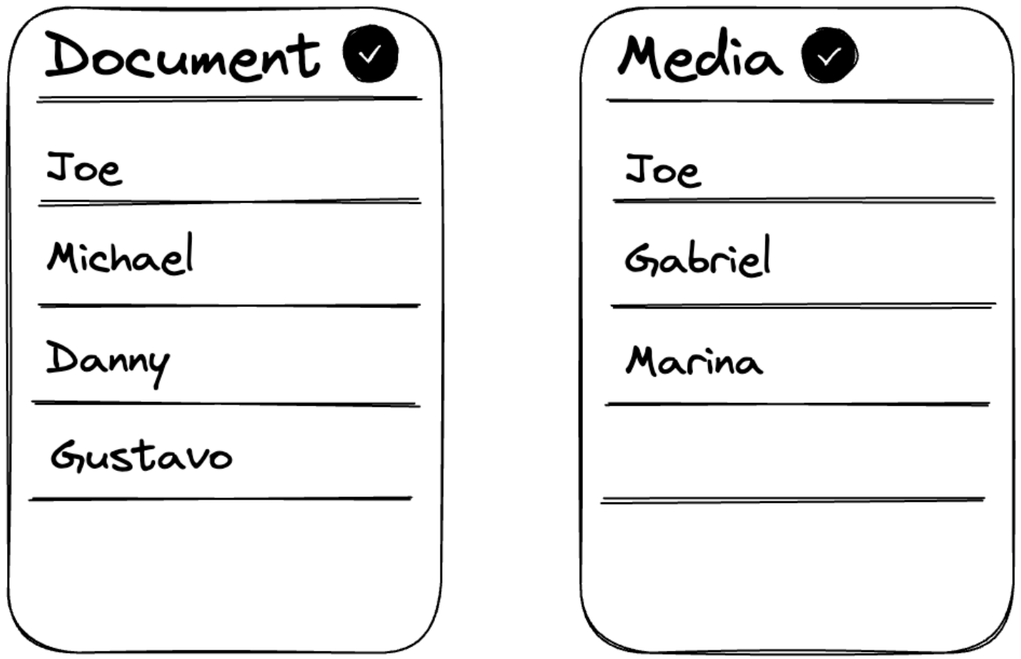
ACL - Access Control List

RBAC - Role-Based Access Control

ABAC - Attribute-Based Access Control

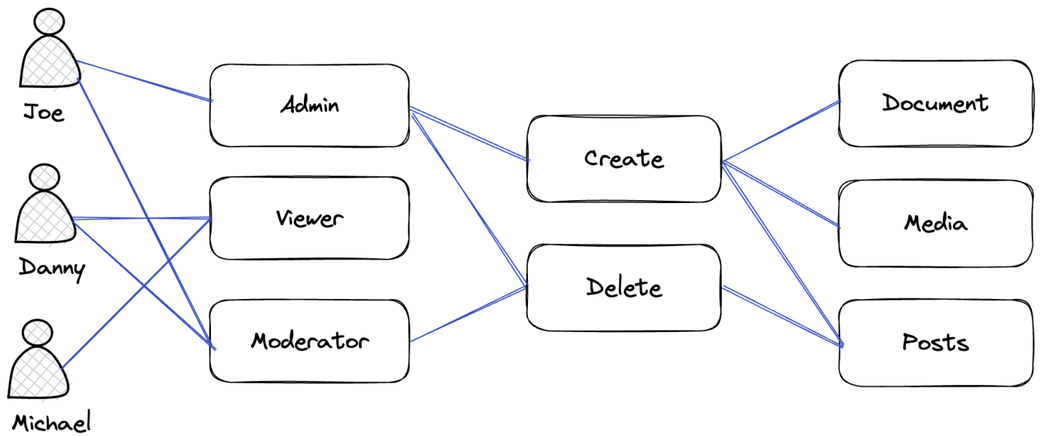
ReBAC - Relationship-Based Access Control

ACL - Access Control List



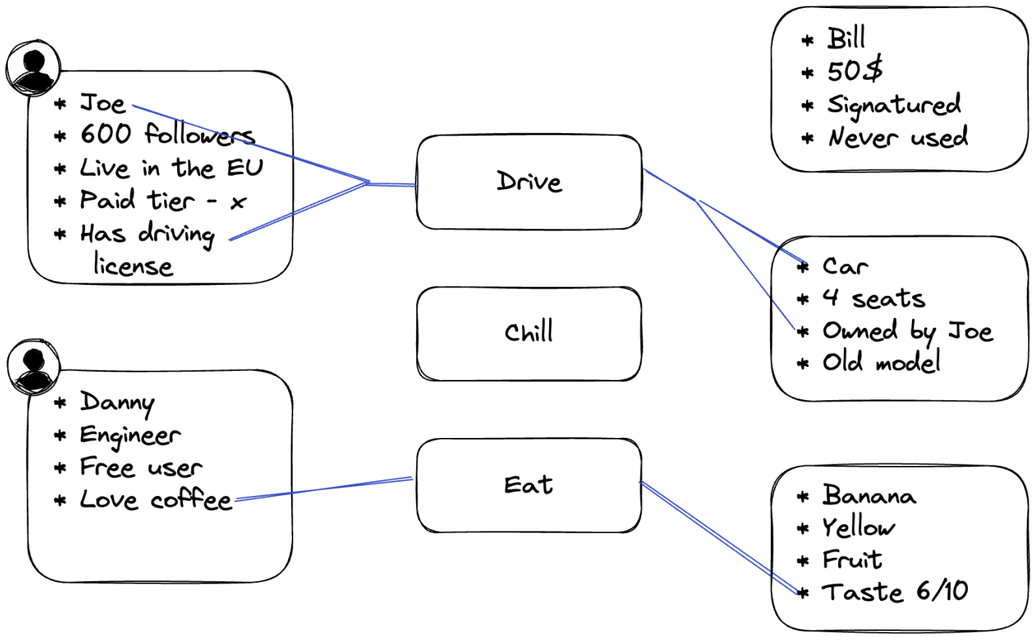
- EOL model
- Widely used in IT systems/ networks
- No segmentation/attribution support
- Hard to scale

RBAC - Role Based Access Control



- The widely-used model for app authorization
- 🤓 Easy to define, use and audit
- No resource inspection
- Limited scalablity

ABAC - Attribute Based Access Control

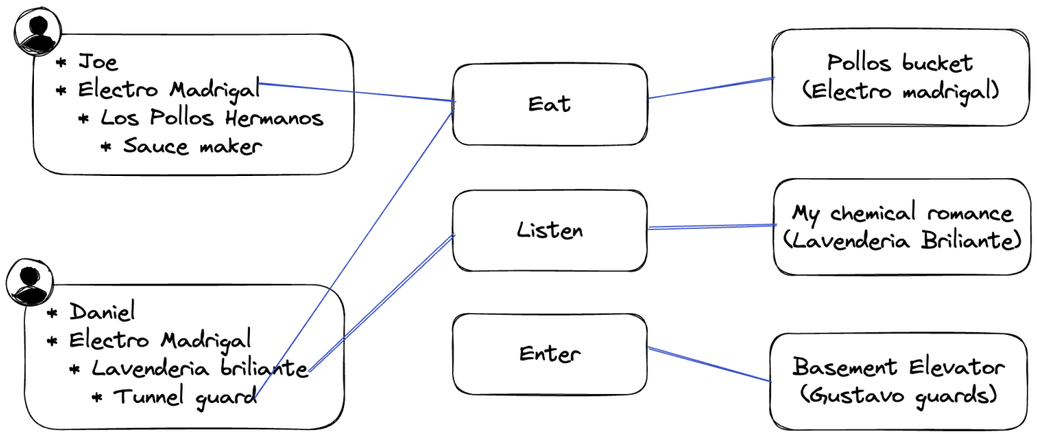


- The most robust model for inspection and desicion making
- Configuration could be hard
- Easy to handle multiple data sources
- 🚀 Highly scalable

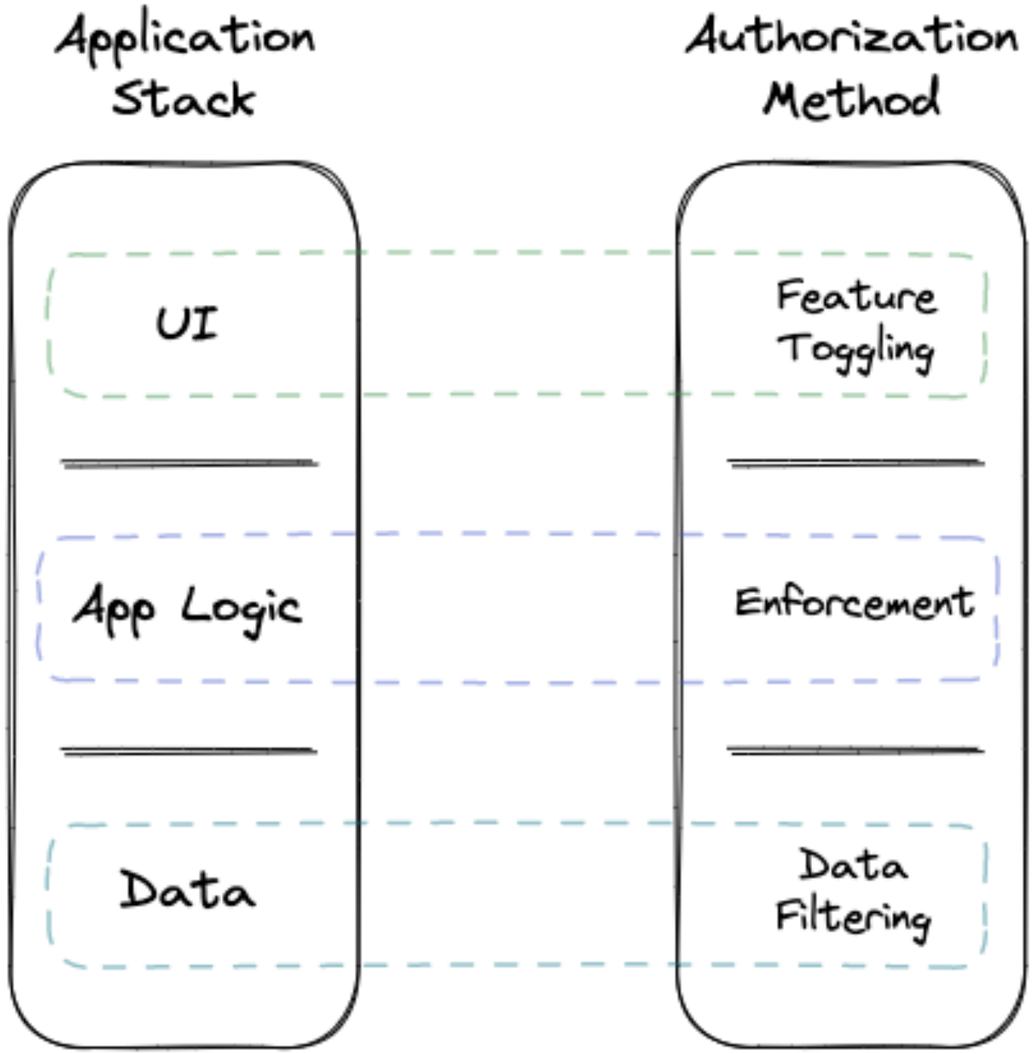
Permit.io

@gemanor

ReBAC - Relationship Based Access Control



- Best fit for consumer-style applications
- Support in reverse indices and search for allowed data
- Easy to scale for users (>1b) hard in desicion's performance



Permit.io

@gemanor

Authorization Along the Software Development Lifecycle

- Developer Permissions
- Admissions
- System to System
- End-User Permissions

#2 Author

Contracts Creates Better Relationships

👩🏽‍💻 *Especially in Human <> Machine Relationships* 🤖

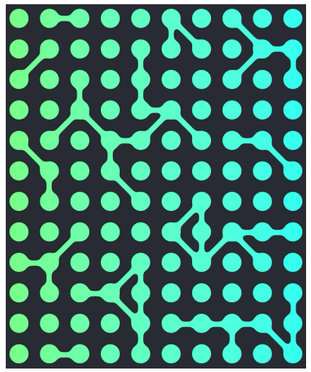

Permit.io

@gemanor

Open Policy Agent

AWS Cedar

Google Zanzibar



OPA

```
package app.rbac

import future.keywords.contains
import future.keywords.if
import future.keywords.in

default allow := false

allow if user_is_admin

allow if {
    some grant in user_is_granted
    input.action == grant.action
    input.type == grant.type
}

user_is_admin if "admin" in data.user_roles[input.user]

user_is_granted contains grant if {
    some role in data.user_roles[input.user]
    some grant in data.role_grants[role]
}
```

```
package app.abac

import future.keywords.if

default allow := false

allow if user_is_owner

allow if {
    user_is_employee
    action_is_read
}

allow if {
    user_is_employee
    user_is_senior
    action_is_update
}

allow if {
    user_is_customer
    action_is_read
    not pet_is_adopted
}

user_is_owner if data.user_attributes[input.user].title == "owner"

user_is_employee if data.user_attributes[input.user].title == "employee"

user_is_customer if data.user_attributes[input.user].title == "customer"

user_is_senior if data.user_attributes[input.user].tenure > 8

action_is_read if input.action == "read"

action_is_update if input.action == "update"

pet_is_adopted if data.pet_attributes[input.resource].adopted == true
```

Permit.io

@gemanor

Open Policy Agent

- 👍 Wide ecosystem
- 👍 Rego is comprehensive and robust
- Rego is complex 🤔
- Data cache replica
- Enforcement plugins


```
permit(
    principal in Role::"admin",
    action in [
        Action::"task:update",
        Action::"task:retrieve",
        Action::"task:list"
    ],
    resource in ResourceType::"task"
);
```

```
permit (
    principal,
    action in
        [Action::"UpdateList",
         Action::"CreateTask",
         Action::"UpdateTask",
         Action::"DeleteTask"],
    resource
)
when { principal in resource.editors };
```

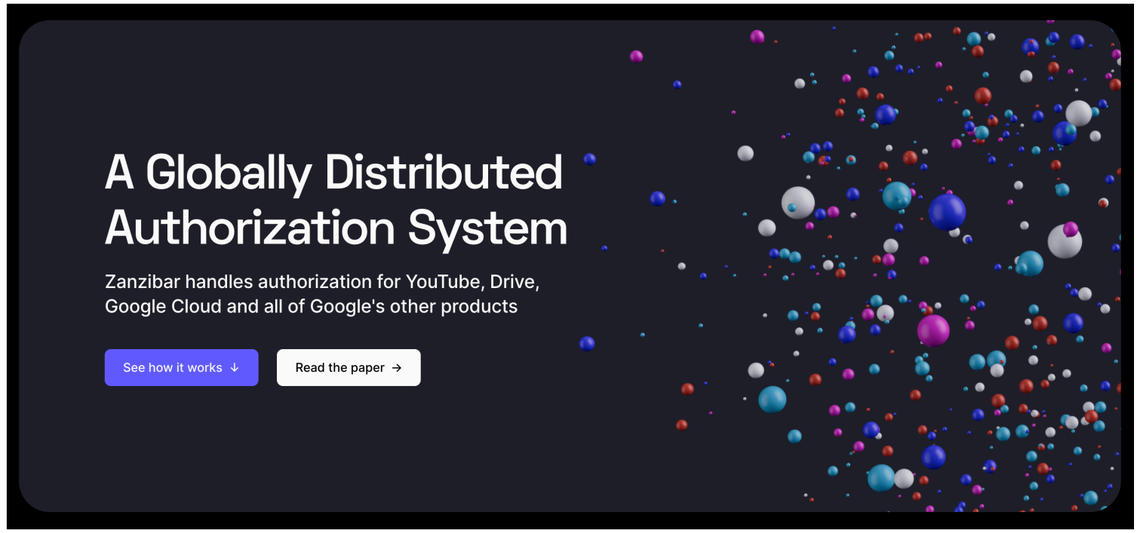
```
permit (
    principal,
    action,
    resource
)
when {
    resource has owner &&
    resource.owner == principal
};
```


Permit.io

@gemanor

AWS Cedar

- 👍 First desgined as application authz language
- 👍 Backed by AWS
- Just released 🤔
- ReBAC via ABAC
- Benchmark winner



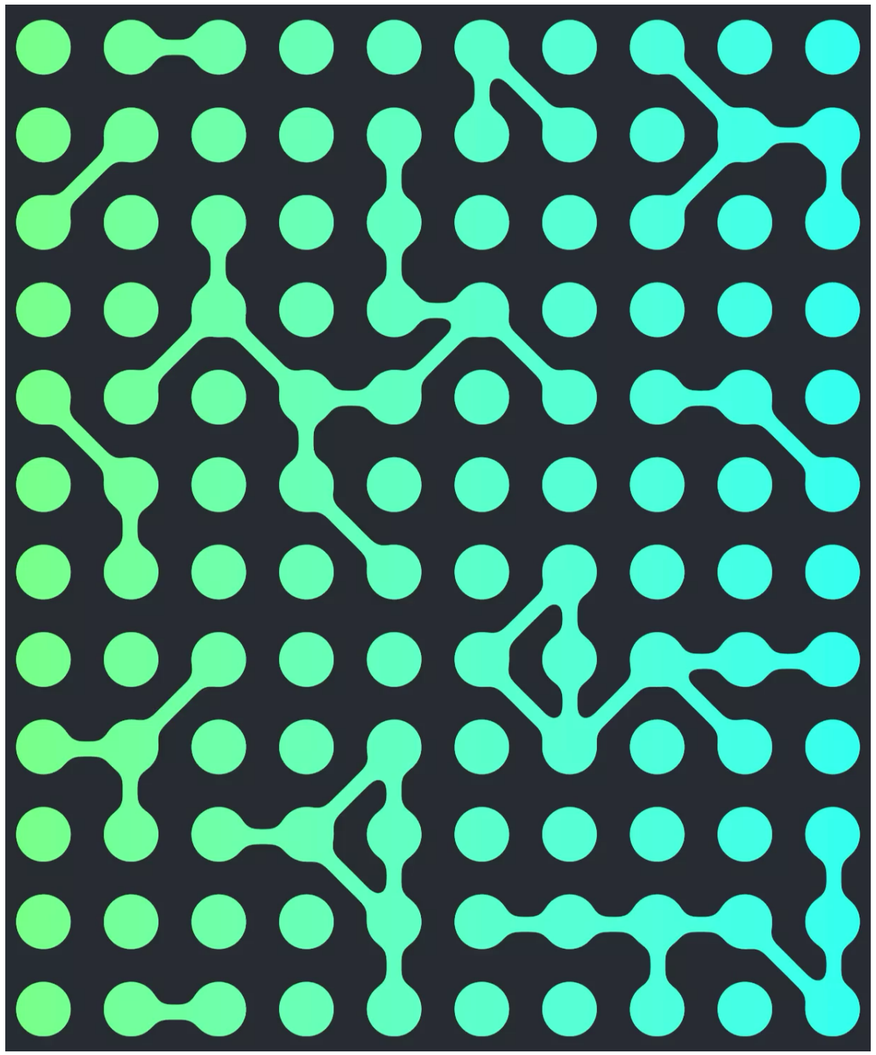

```
name: "doc"

relation { name: "owner" }

relation {
 name: "editor"
 userset_rewrite {
  union {
   child { _this {} }
   child { computed_userset { relation: "owner" } }
} } }

relation {
 name: "viewer"
 userset_rewrite {
  union {
   child { _this {} }
   child { computed_userset { relation: "editor" } }
   child { tuple_to_userset {
     tupleset { relation: "parent" }
     computed_userset {
       object: $TUPLE_USERSET_OBJECT  # parent folder
       relation: "viewer"
   } } }
} } }
```

Permit.io

@gemanor

OpenFGA Implementation

```
{
  "schema_version": "1.1",
  "type_definitions": [{
      "type": "user"
    }, {
      "type": "document",
      "relations": {
        "reader": {"this": {}},
        "writer": {"this": {}},
        "owner": {"this": {}}
      },
      "metadata": {
        "relations": {
          "reader": {
            "directly_related_user_types": [{"type": "user"}]
          },
          "writer": {
            "directly_related_user_types": [{"type": "user"}]
          },
          "owner": {
            "directly_related_user_types": [{"type": "user"}]
          }
        }
      }
    }
  ]
}
```



Permit.io

@gemanor

Google Zanzibar - OpenFGA

- Simple ReBAC
- Complex RBAC/ABAC
- Reverse indices support
- Backed by OKTA
- Stateful

#3 Analyze

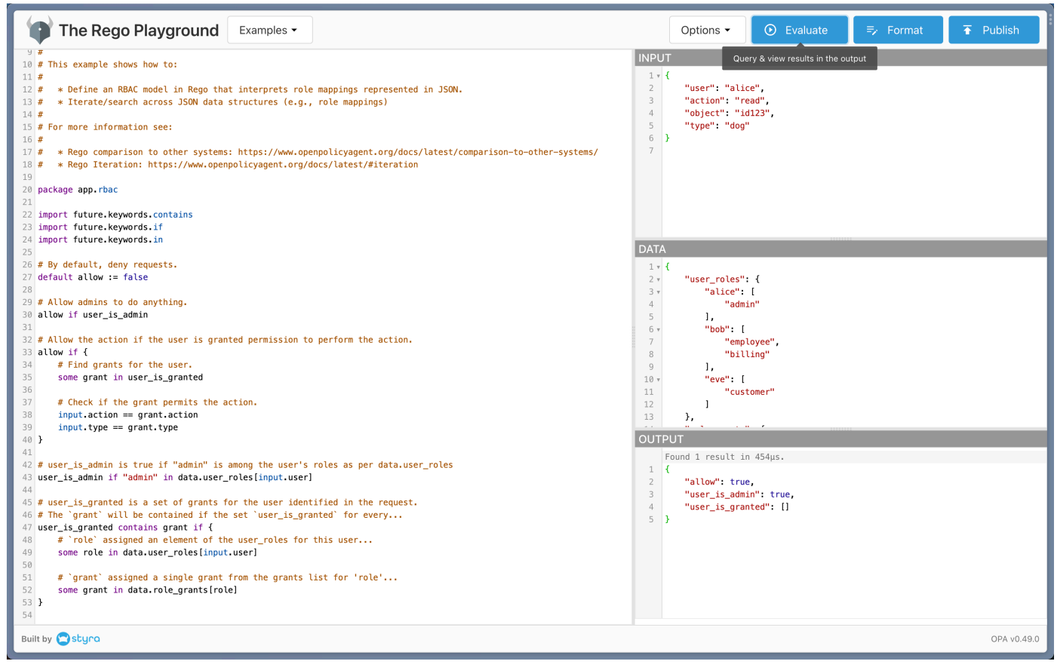
Cedar Agent

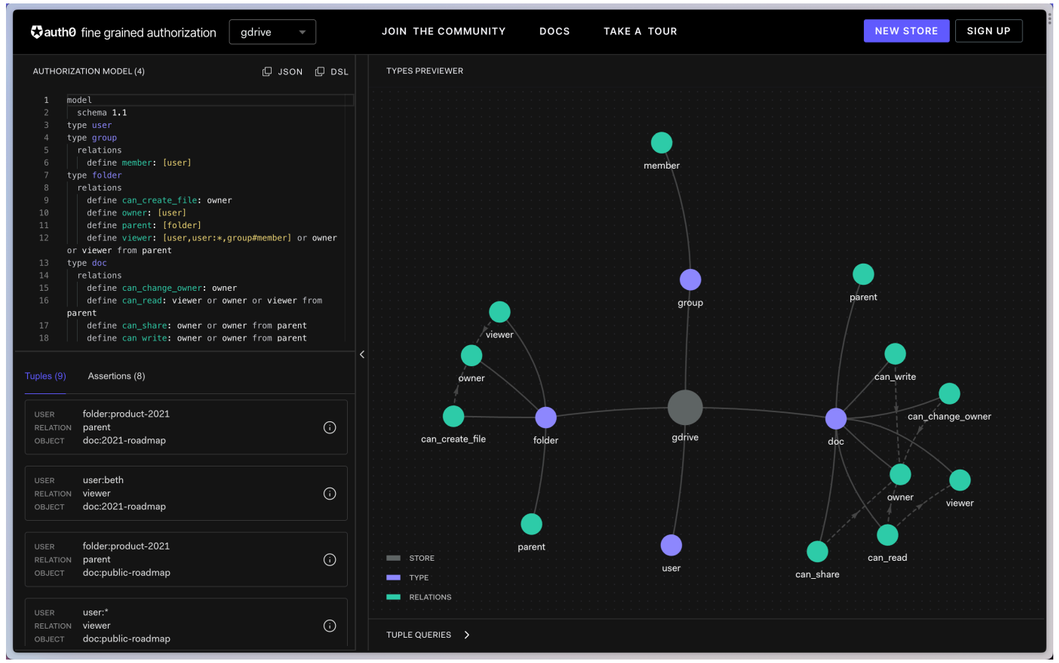
- Policy decision maker
- Decentralized continer, run as a sidecar to applications
- Monitored and audited
- Focused in getting very fast decisions >10ms

Permit.io

@gemanor

Policy Playgrounds





play.openpolicyagent.org

play.fga.dev

Permit.io

@gemanor

#4 Enforce

Enforcing Authorization Policies

```
# Call authorization service
# In the request body, we pass the relevant request information
allowed = requests.post('http://host.docker.internal:8180/v1/is_authorized', json={
    "principal": f"User::\"{user}\"",
    "action": f"Action::\"{method.lower()}\"",
    "resource": f"ResourceType::\"{original_url.split('/')[1]}\"",
    "context": request.json
}, headers={
    'Content-Type': 'application/json',
    'Accept': 'application/json'
})
```

Permit.io

@gemanor

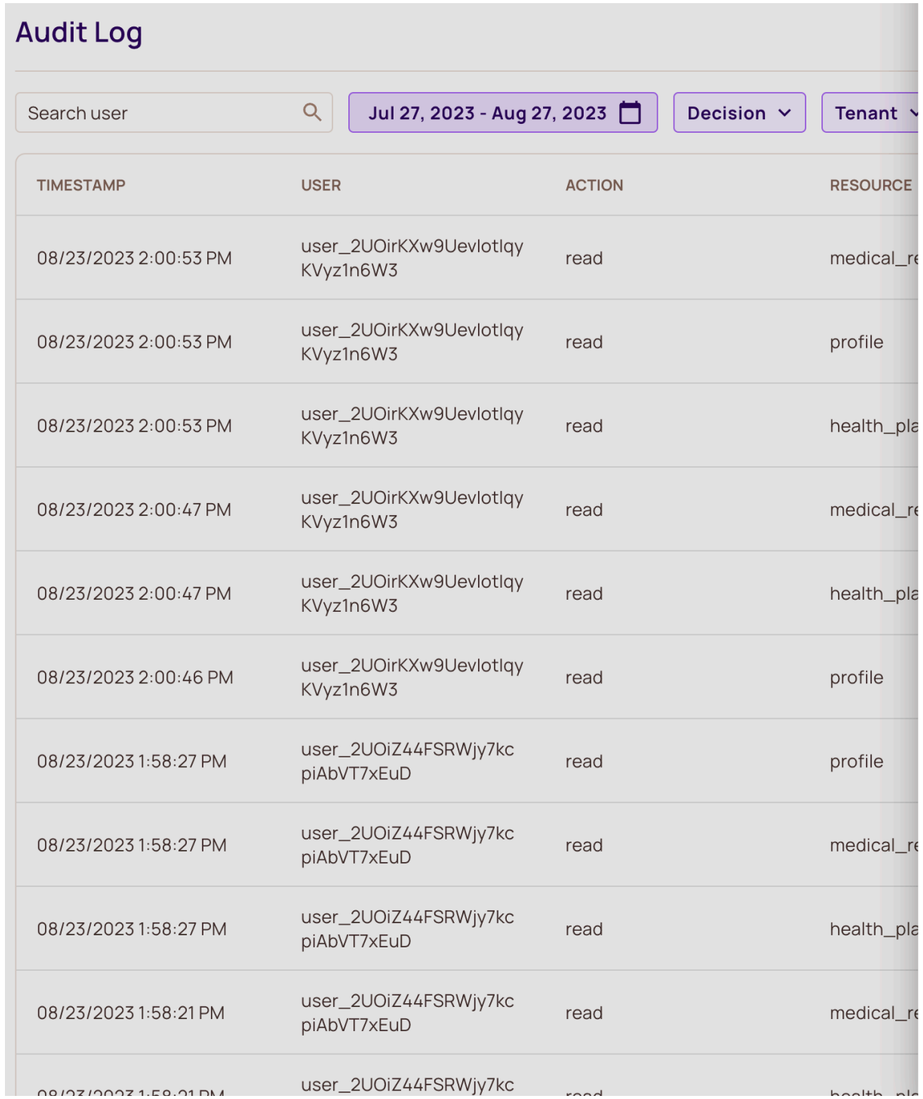
CASL - The Frontend Feature Toggling SDK

```
import { createMongoAbility, AbilityBuilder } from '@casl/ability';

// define abilities
const { can, cannot, build } = new AbilityBuilder(createMongoAbility);

can('read', ['Post', 'Comment']);
can('manage', 'Post', { author: 'me' });
can('create', 'Comment');

// check abilities
const ability = build();

ability.can('read', 'Post') // true
```

Permit.io

@gemanor

#5 Audit



Audit Event

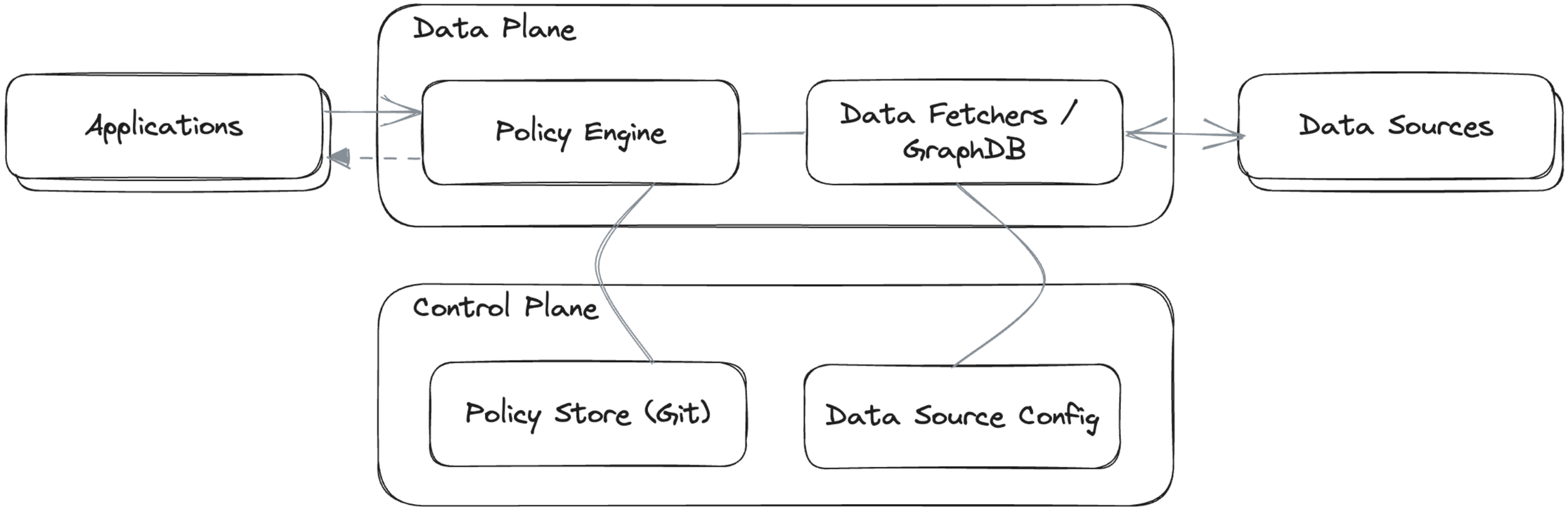
Timestamp	08/23/2023 2:00:53 PM
User	user_2UOirKXw9UevlotIqyKVyz1n6W3
Tenant	Default Tenant
Resource	Profile
Action	read
Decision	DEFAULT PDP CONFIG · PERMITTED


```
{
  "allow": true,
  "debug": {
    "rbac": {
      "code": "no_user_roles",
      "allow": false,
    "reason":"no roles assigned to user                ,
             'user_2UOirKXw9UevIotIqyKVyz1n6W3'"
"support_link":"https://docs.permit.io/errors/no_user_role
    },
    "rebac": {
      "code": "allow",
      "allow": true,
"reason":"user 'user_2UOirKXw9UevIotIqyKVyz1n6W3' has the
         permission on resource
         'profile:profile_user_2UOirKXw9UevIotIqyKVyz1n6W3
         tenant 'default', granted by
         'profile:profile_user_2UOirKXw9UevIotIqyKVyz1n6W3
         which is granted by
         'member:member_user_2UOirKXw9UevIotIqyKVyz1n6W3#o
      "allowing_roles": [
```


Permit.io

@gemanor

Authorization System Building Blocks



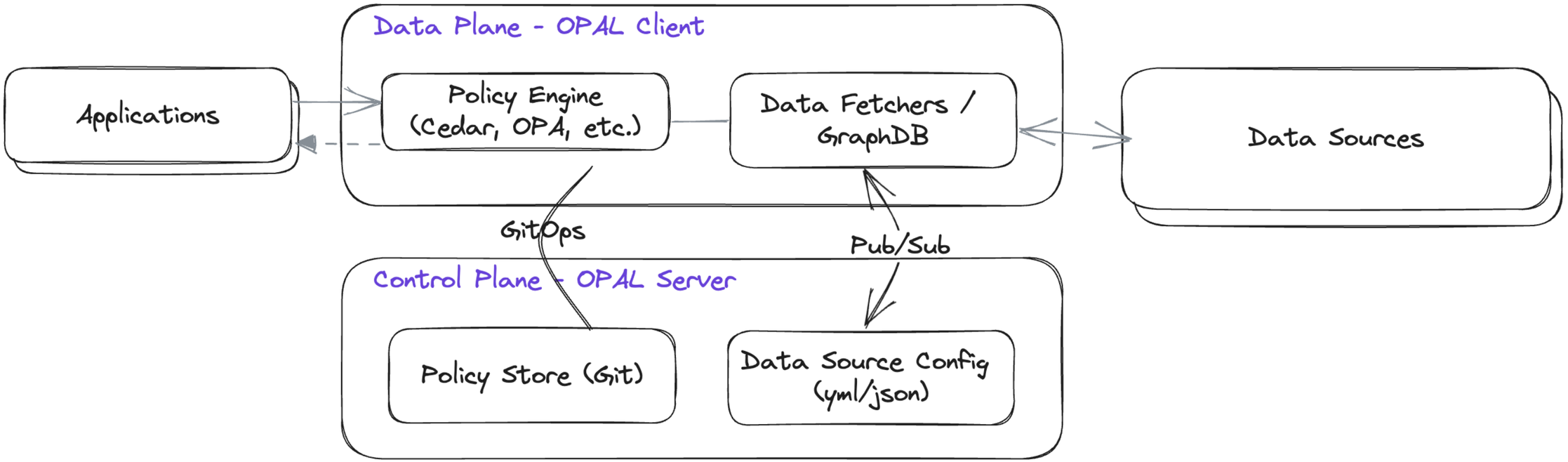
OPAL - Open Policy Administration Layer

- Open Source, Written in Python
- Sync decision points with data and policy stores
- Auto-scale for engines
- Centralized services such as Audit
- Unified APIs for the enforcement point
- Extensible for any kind of data source
- Supports OPA, Cedar (and soon to be announced more)
- Used by Tesla, Zapier, Cisco, Accenture, Walmart, NBA and thousands more

Permit.io

@gemanor

OPAL Based Authorization Architecture



Permit.io

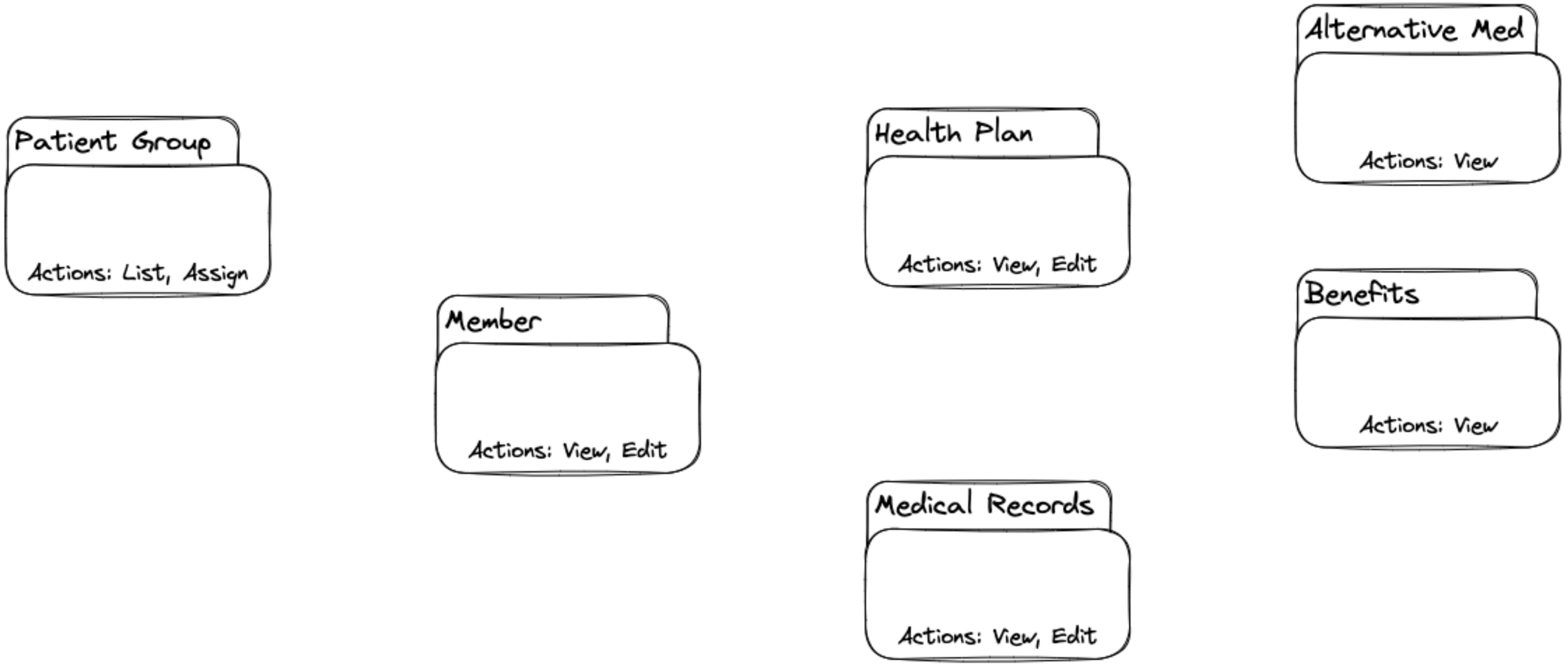
@gemanor

Demo time 🍿

Galactic Health Corporation - Requirements

- Each member can see their health plan and medical records
- Each member can belong to a member group of potential caregivers
- Each member can assign permissions to caregivers to view their own medical records or health plan
- The caregiver "membership" can be limited by time

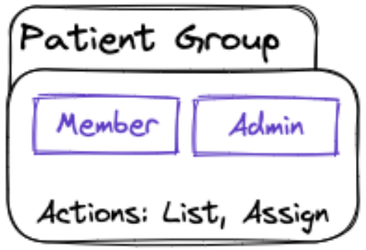
Galactic Health Corporation - System Resources

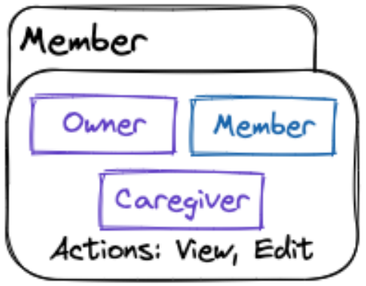


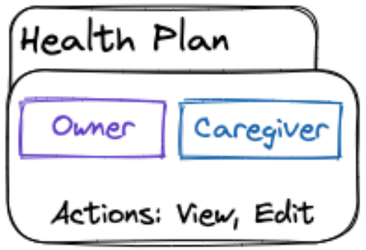
Permit.io

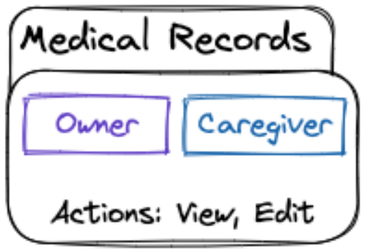
@gemanor

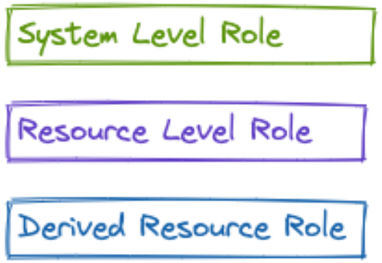
Galactic Health Corporation - Roles







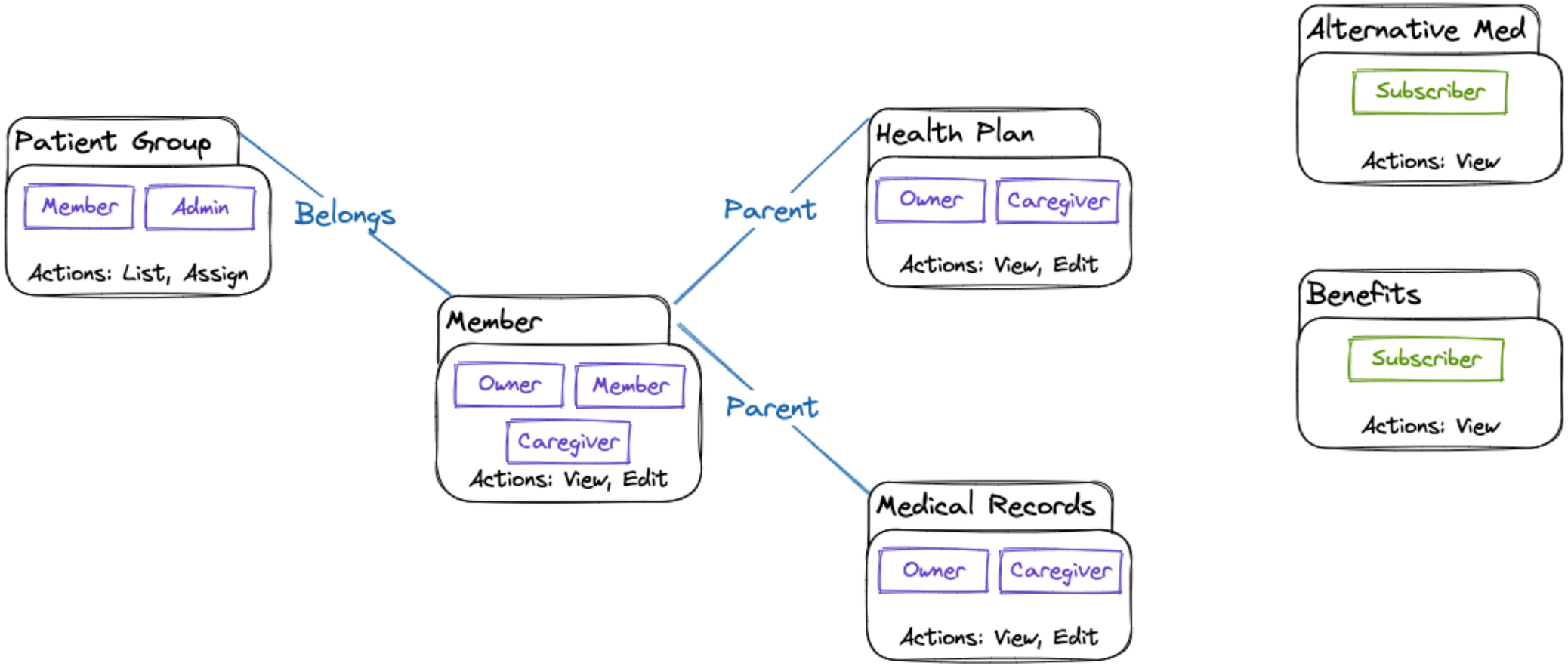




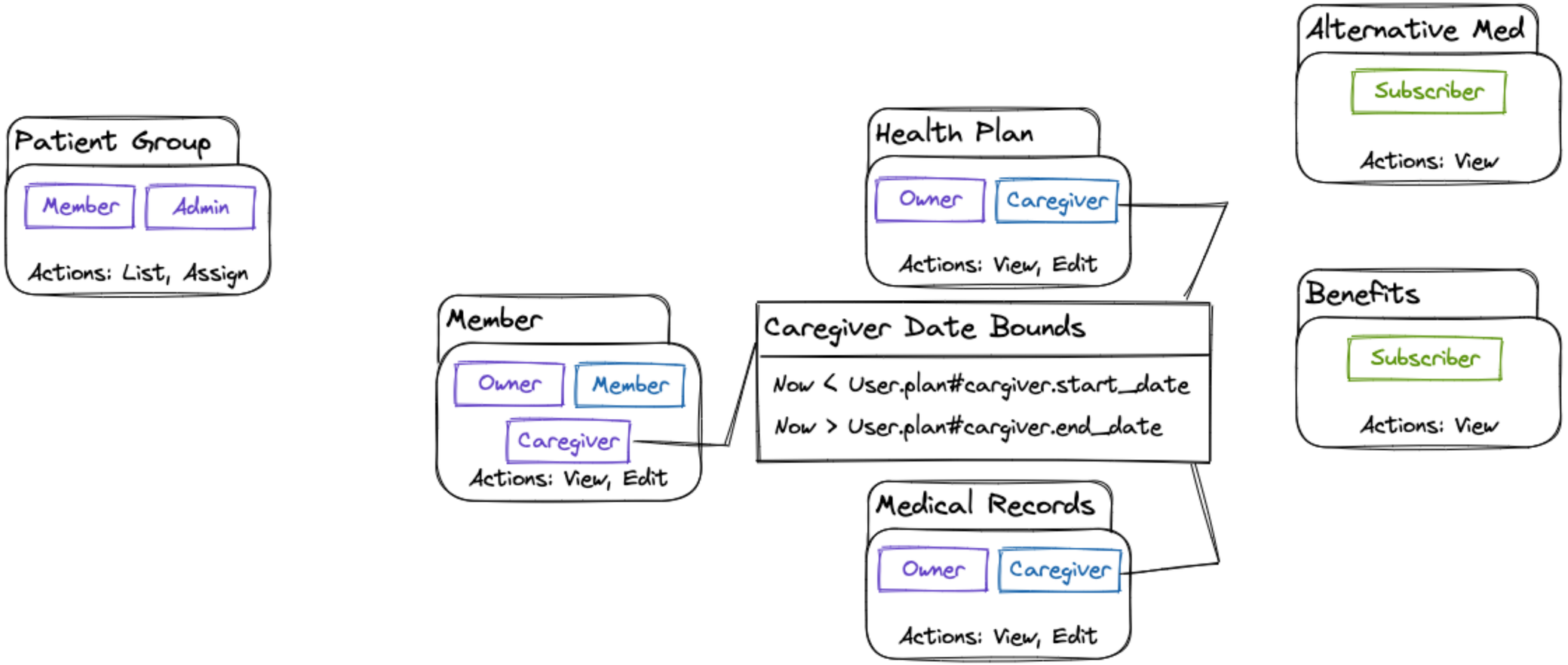
Permit.io

@gemanor

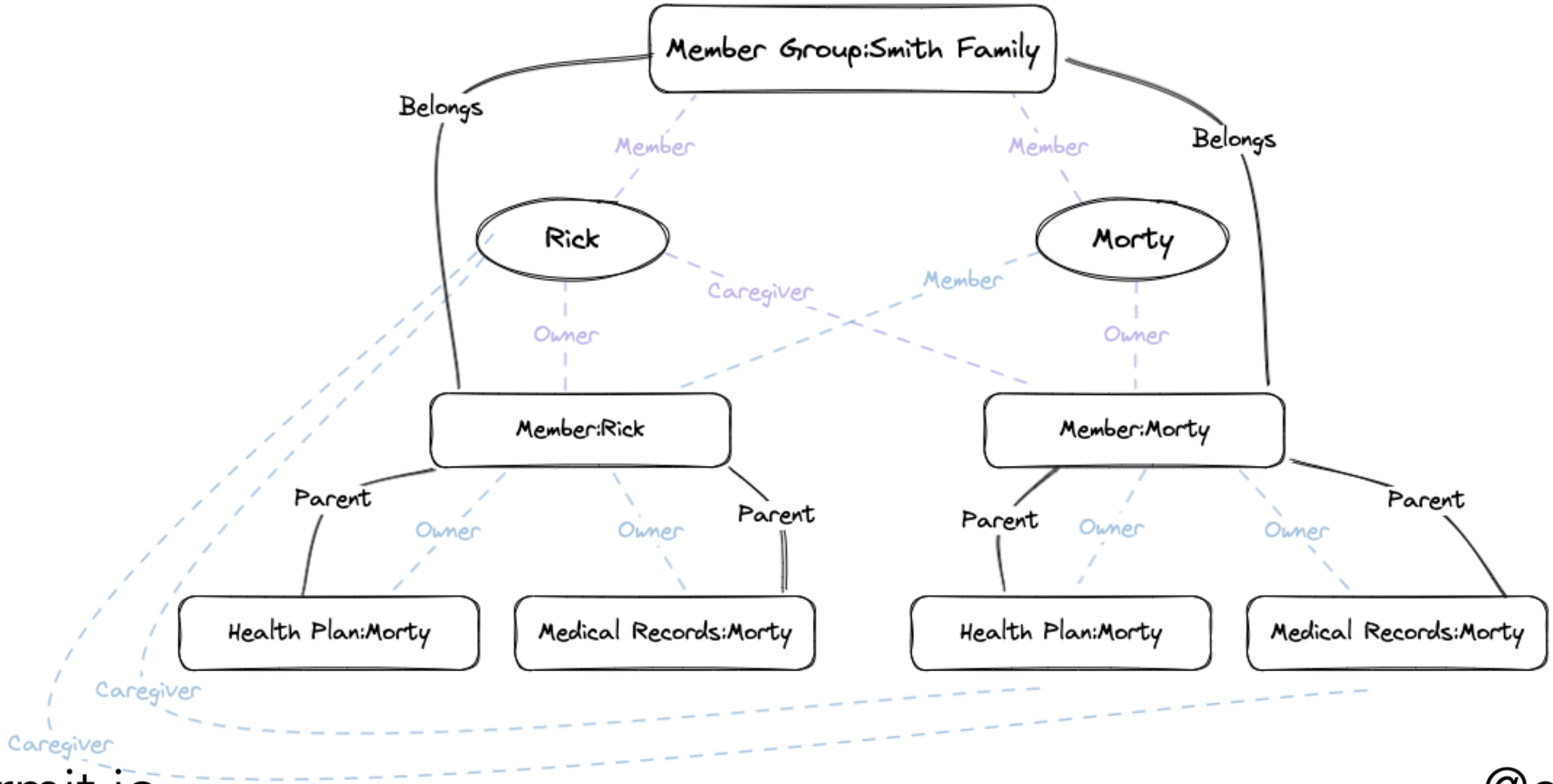
Galactic Health Corporation - Resource Relations



Galactic Health Corporation - Attribute Conditions



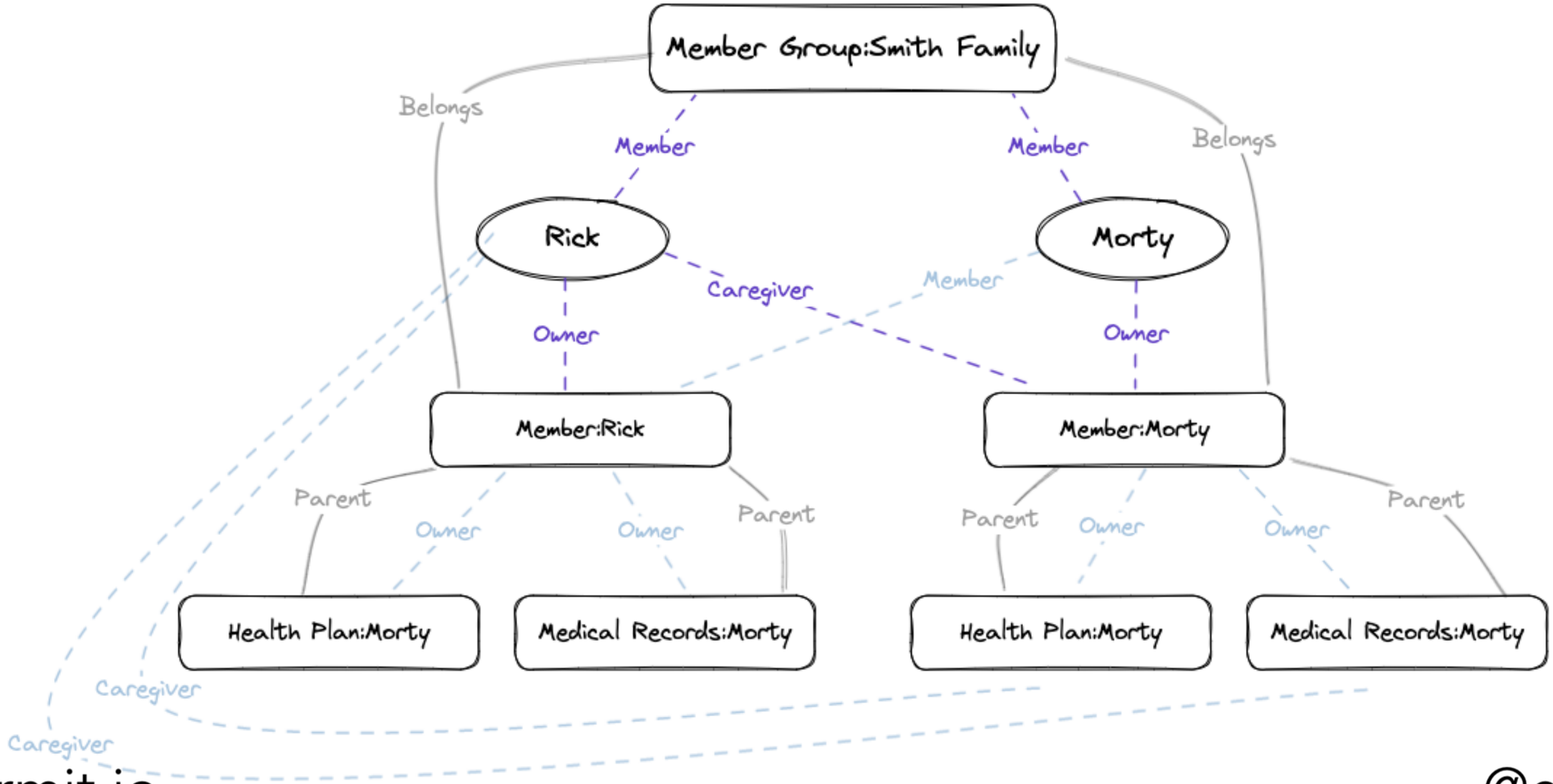
User Modeling - Relationship Tuples



Permit.io

@gemanor

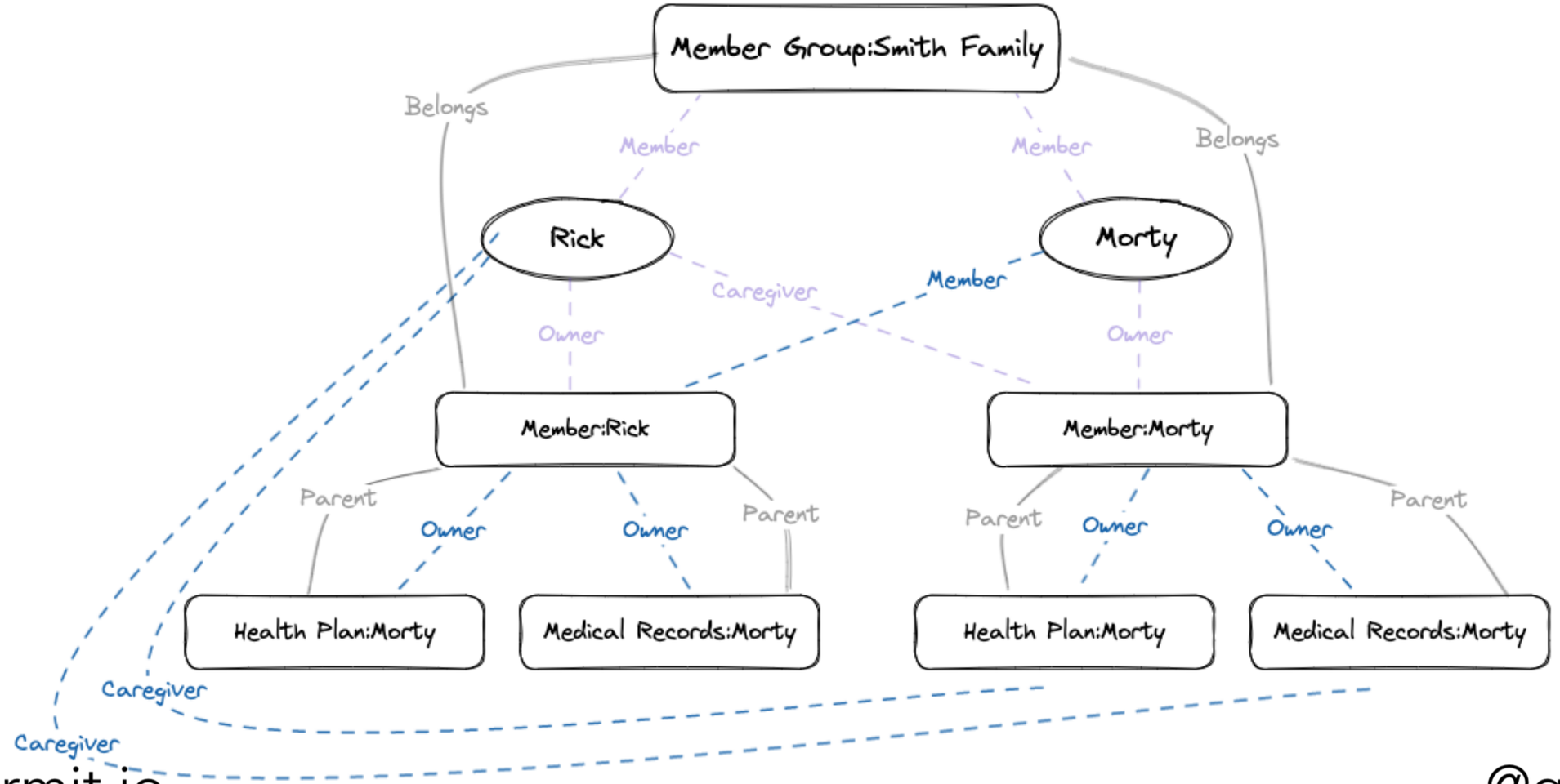
User Modeling - Role Assignments



Permit.io

@gemanor

User Modeling - Role Derviations



Permit.io

@gemanor

Thank You 🙏

Show your love to OPAL with a GitHub Star ⭐ 👉

Find more about OPAL on opal.ac

Follow me on Twitter @gemanor

Permit.io

@gemanor