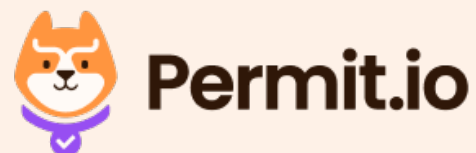


Building Authorization with Node.js

Gabriel L. Manor @ Conf42: JavaScript 2023



@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

Authentication Advanced Features

- Multi-factor authentication

Authentication Advanced Features

- Multi-factor authentication
- Single sign-on / Social login / Passwordless

Authentication Advanced Features

- Multi-factor authentication
- Single sign-on / Social login / Passwordless
- User / account management

Authentication Advanced Features

- Multi-factor authentication
- Single sign-on / Social login / Passwordless
- User / account management
- Session management

Authentication Advanced Features

- Multi-factor authentication
- Single sign-on / Social login / Passwordless
- User / account management
- Session management
- User registration / UI flows / customizations

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

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

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

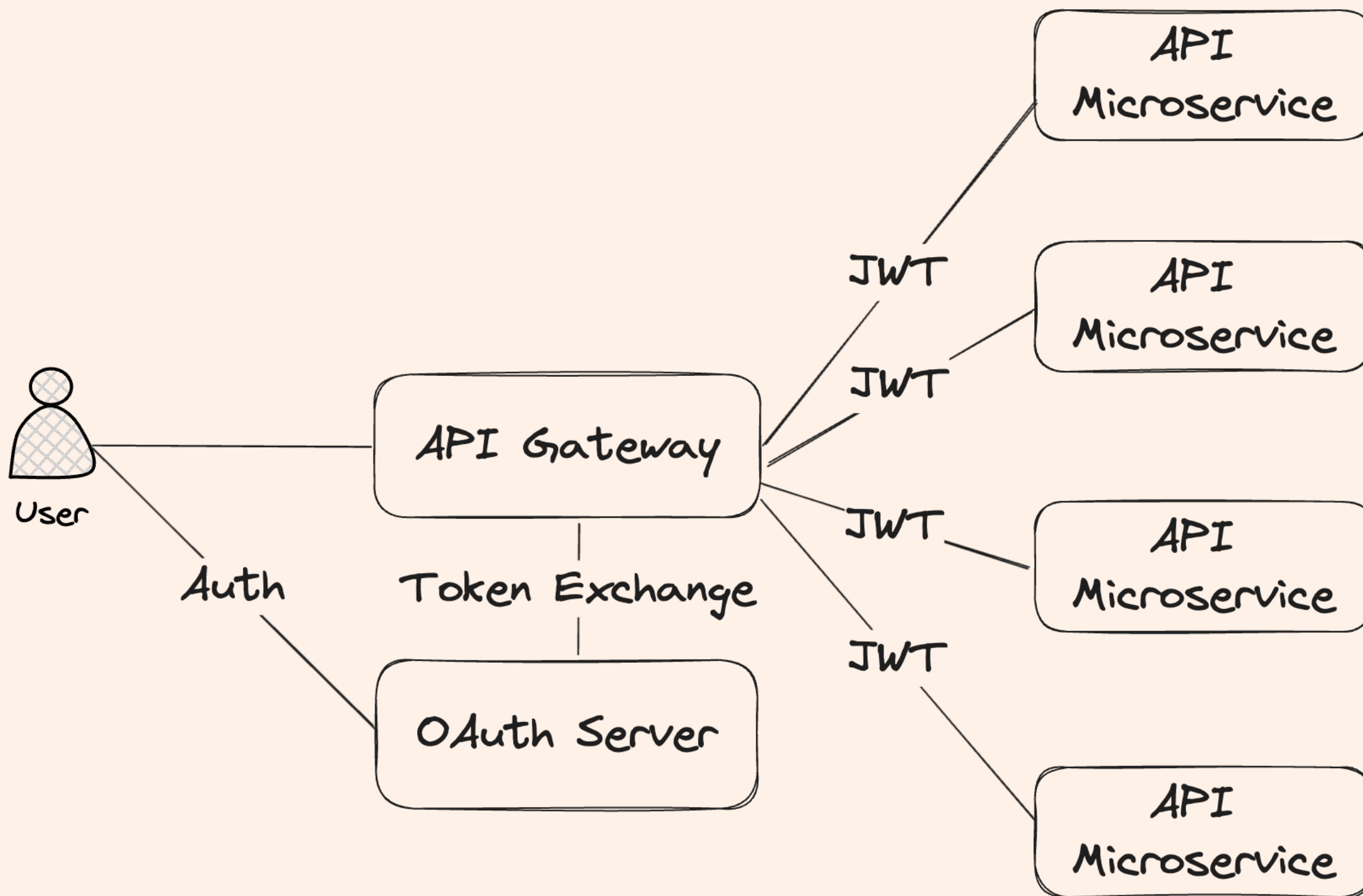




Gabriel L. Manor

Director of DevRel @ Permit.io

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



```
function deleteUser(user_id) {  
  let user = User.get(user_id);  
  if (user.role === "admin") {  
    user.delete();  
  }  
}
```

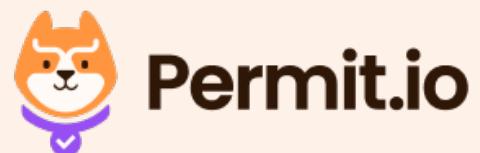
```
// Middleware
function rolesRequired(role) {
  return function (req, res, next) {
    if (req.user.role === role) {
      next();
    } else {
      res.status(403).send("Forbidden");
    }
  };
}

app.delete("/users/:id", rolesRequired("admin"), function (req, res) {
  let user = User.get(req.params.id);
  user.delete();
  res.send("User deleted successfully");
});
```

```
// Middleware
function rolesRequired(role) {
  return function (req, res, next) {
    if (req.user.role === role) {
      next();
    } else {
      res.status(403).send("Forbidden");
    }
  };
}

function permissionsRequired(permission) {
  return function (req, res, next) {
    if (req.user.permissions.includes(permission)) {
      next();
    } else {
      res.status(403).send("Forbidden");
    }
  };
}
```

```
// Business logic
app.delete(
  "/users/:id",
  rolesRequired("admin"),
  permissionsRequired("delete_user"),
  function (req, res) {
    let user = User.get(req.params.id);
    user.delete();
    res.send("User deleted successfully");
  }
);
```

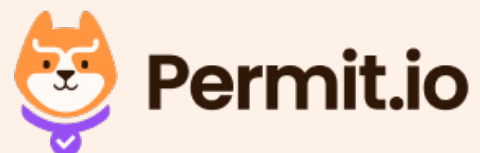


@gemanor

```
// Middleware
function rolesRequired(role) {
  return function (req, res, next) {
    if (req.user.role === role) {
      next();
    } else {
      res.status(403).send("Forbidden");
    }
  };
}
```

```
// Business logic
app.get(
  "/enable_workflow/:user_id",
  rolesRequired("admin"),
  function (req, res) {
    const { userId } = req.params;
    let user = User.get(userId);
    let tier = bs.getTier(userId);
    if (tier !== "paid") {
      res.status(403).send("User is not paid user");
    } else if (user.sms_enabled) {
      sms(user.phone_number, "Workflow is enabled");
      res.send("Workflow enabled successfully");
    }
  }
);
```

```
// Business logic
app.get(
  "/enable_workflow/:user_id",
  rolesRequired("admin"),
  permissionsRequired("enable_workflow"),
  function (req, res) {
    const { user_id } = req.params;
    let user = User.get(user_id);
    let step1 = workflow.run();
    let step2 = workflow.run();
    if (user.sms_enabled) {
      send_sms_to_list(user.phone_number, "Workflow is enabled");
    }
    res.send("Workflow enabled successfully");
  }
);
```



@gemanor

Staging

```
// Business logic
app.get(
  "/enable_workflow/:user_id",
  rolesRequired("admin"),
  permissionsRequired("enable_workflow"),
  function (req, res) {
    const { user_id } = req.params;
    let user = User.get(user_id);
    let step1 = workflow.run();
    ...
  });
```

Production

```
// Business logic
app.get(
  "/enable_workflow/:user_id",
  rolesRequired("superadmin"),
  permissionsRequired("enable_workflow"),
  function (req, res) {
    const { user_id } = req.params;
    let user = User.get(user_id);
    let step1 = workflow.run();
    ...
  });
```

Express

```
app.get('/my_view', [  
  rolesRequired('admin'),  
  permissionsRequired('app_name.can_edit')  
], function (req, res) {  
  // Your view logic here  
  ...  
});
```

Flask

```
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)  
    ...
```



```
if (
  user.role === "admin" &&
  user.tier === "paid" &&
  user.sms_enabled &&
  user.phone_number
) {
  send_sms_to_list(user.phone_number, "Workflow is enabled");
}
```

Authorization Best Practices



Declarative



Generic



Unified



Agnostic



Decoupled



Easy to audit

#1 Model

User | Action | Resource

Does *[Principal]* Allowed to Perform *[Action]* on *[Resource]*
Is a Monkey Allowed to Eat a Banana

Application Stack



Authorization Method



ACL - Access Control List

RBAC - Role-Based Access Control

ABAC - Attribute-Based Access Control

ReBAC - Relationship-Based Access Control

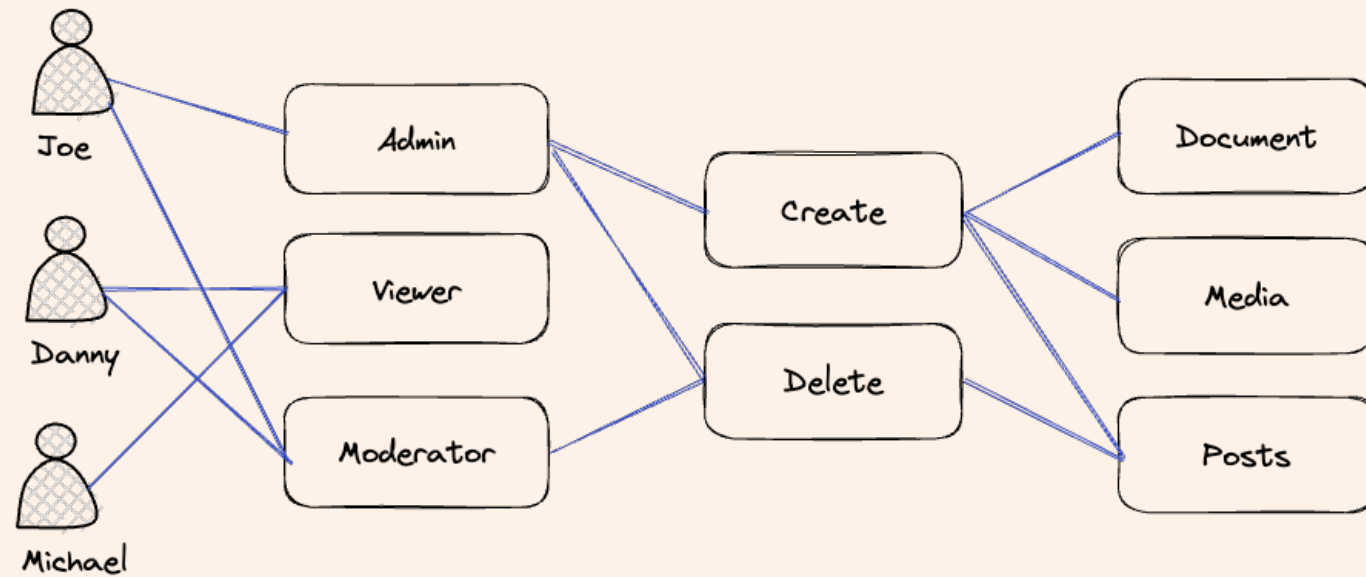
ACL - Access Control List

Document
Joe
Michael
Danny
Gustavo

Media
Joe
Gabriel
Marina

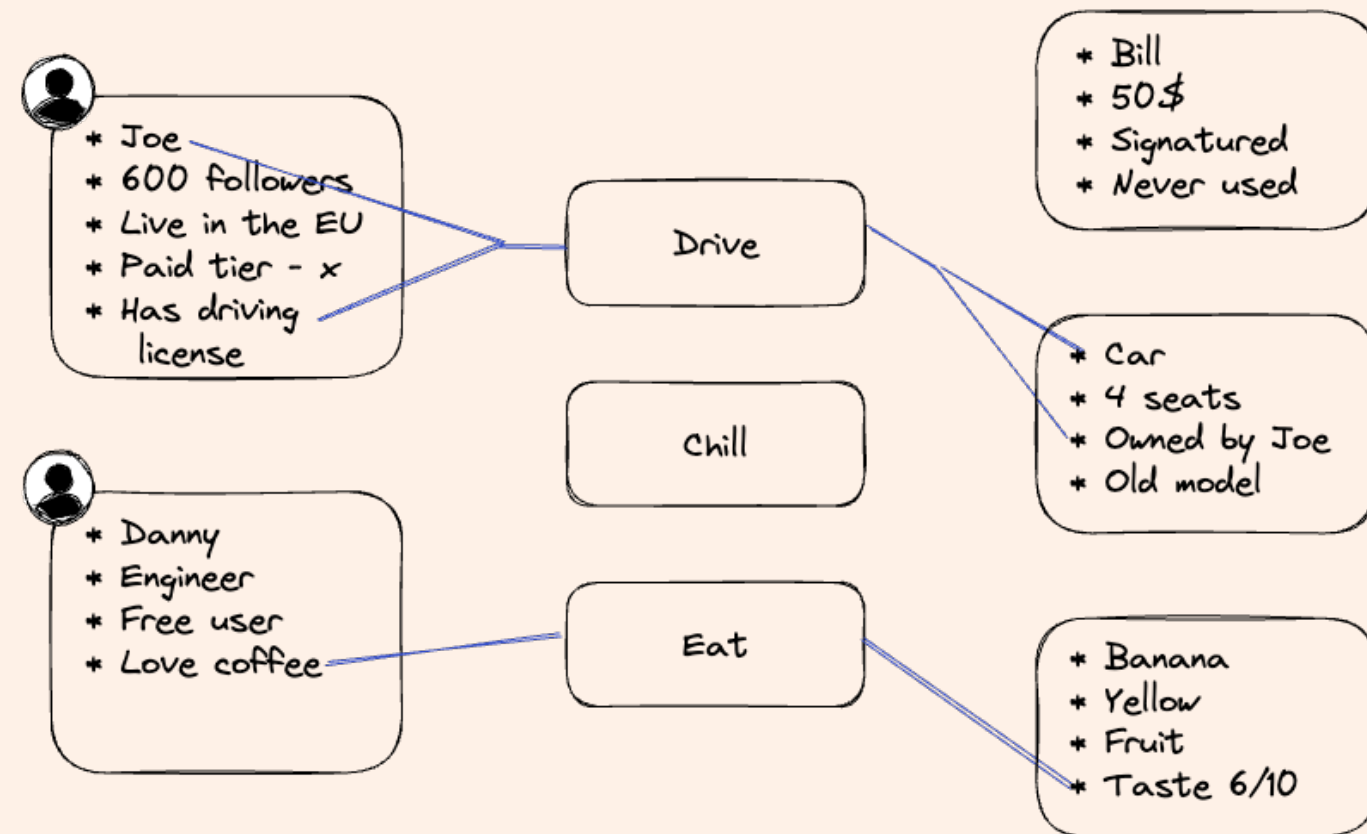
- EOL model
- Widely used in IT systems/networks
- No segmentation/ attribution support
- Hard to scale

RBAC - Role Based Access Control



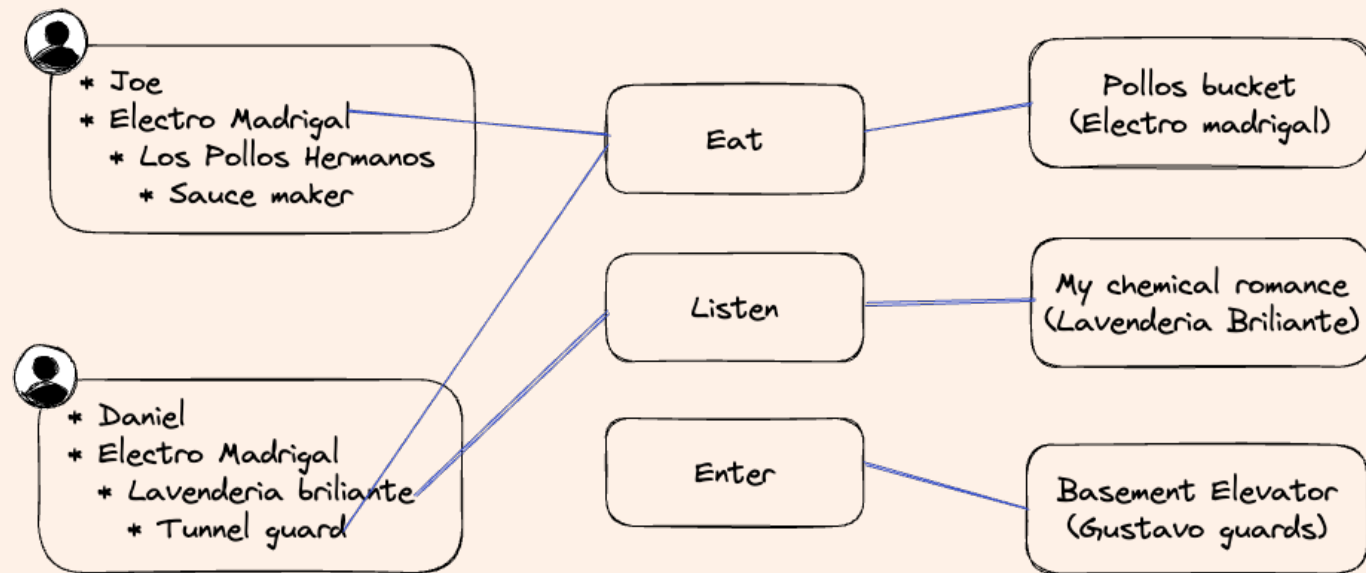
- The widely-used model for app authorization
- 🧐 Easy to define, use and audit
- No resource inspection
- Limited scalability

ABAC - Attribute Based Access Control



- The most robust model for inspection and decision making
- Configuration could be hard
- Easy to handle multiple

ReBAC - Relationship Based Access Control



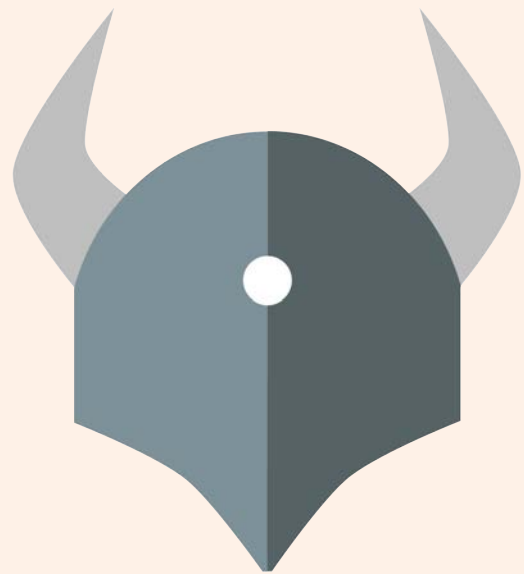
- Best fit for consumer-style applications
- Support in reverse indices and search for allowed data
- Easy to scale for users (>1b) hard in desicion's

#2 Author

Contracts Creates Better Relationships

 *Especially in Human <> Machine Relationships* 

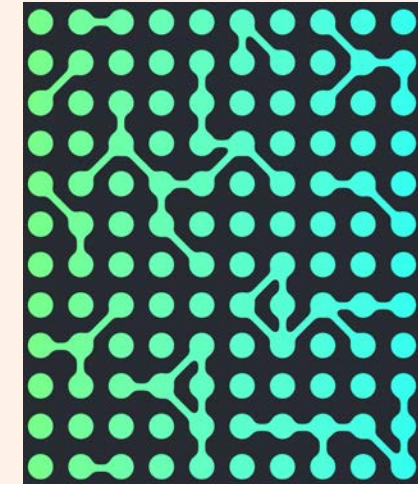
Open Policy
Agent



AWS
Cedar



Google
Zanzibar -
Open FGA



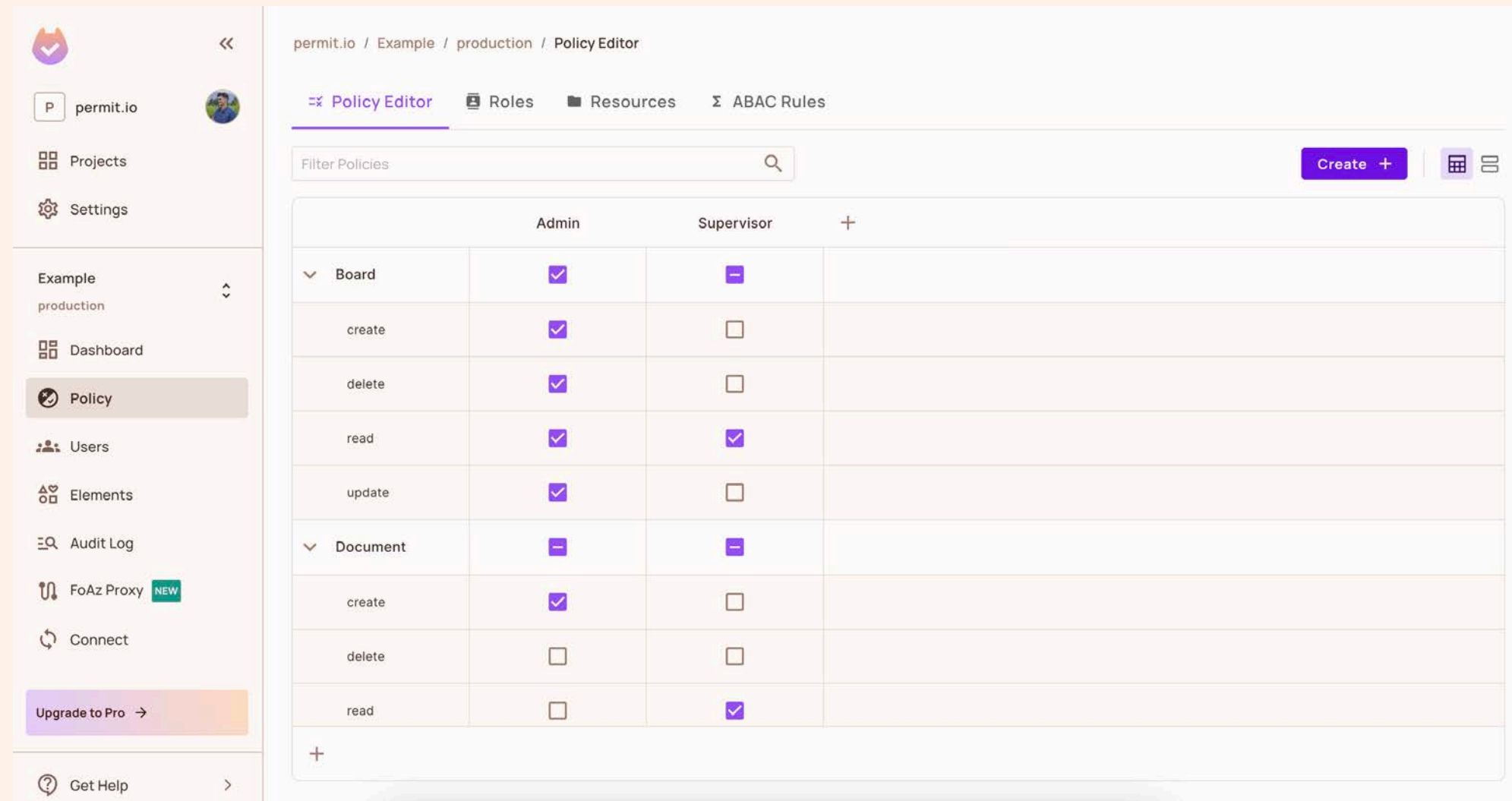


```
permit(  
  principal in Role::"admin",  
  action in  
    [Action::"task:update", Action::"task:retrieve", Action::"task:list"],  
  resource in ResourceType::"task"  
);
```

```
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  
};
```

Generate Code from UI



The screenshot displays the Permit.io Policy Editor interface. The left sidebar contains navigation options: permit.io, Projects, Settings, Example production (with sub-items: Dashboard, Policy, Users, Elements, Audit Log, FoAz Proxy, Connect), Upgrade to Pro, and Get Help. The main content area shows the breadcrumb path: permit.io / Example / production / Policy Editor. Below this are tabs for Policy Editor, Roles, Resources, and ABAC Rules. A search bar labeled 'Filter Policies' and a 'Create +' button are visible. The main table lists permissions for two roles: Admin and Supervisor. The table has columns for the role and checkboxes for each permission. The 'Board' resource has permissions: create (Admin: checked, Supervisor: unchecked), delete (Admin: checked, Supervisor: unchecked), read (Admin: checked, Supervisor: checked), and update (Admin: checked, Supervisor: unchecked). The 'Document' resource has permissions: create (Admin: checked, Supervisor: unchecked), delete (Admin: unchecked, Supervisor: unchecked), and read (Admin: unchecked, Supervisor: checked).

	Admin	Supervisor	
Board	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
create	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
delete	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
read	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
update	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Document	<input type="checkbox"/>	<input type="checkbox"/>	
create	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
delete	<input type="checkbox"/>	<input type="checkbox"/>	
read	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

#3 Analyze

Cedar Agent



- Policy decision maker
- Decentralized container, run as a sidecar to applications
- Monitored and audited
- Focused in getting very

#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'
})
```

CASL – 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
```

#5 Audit

Audit Log

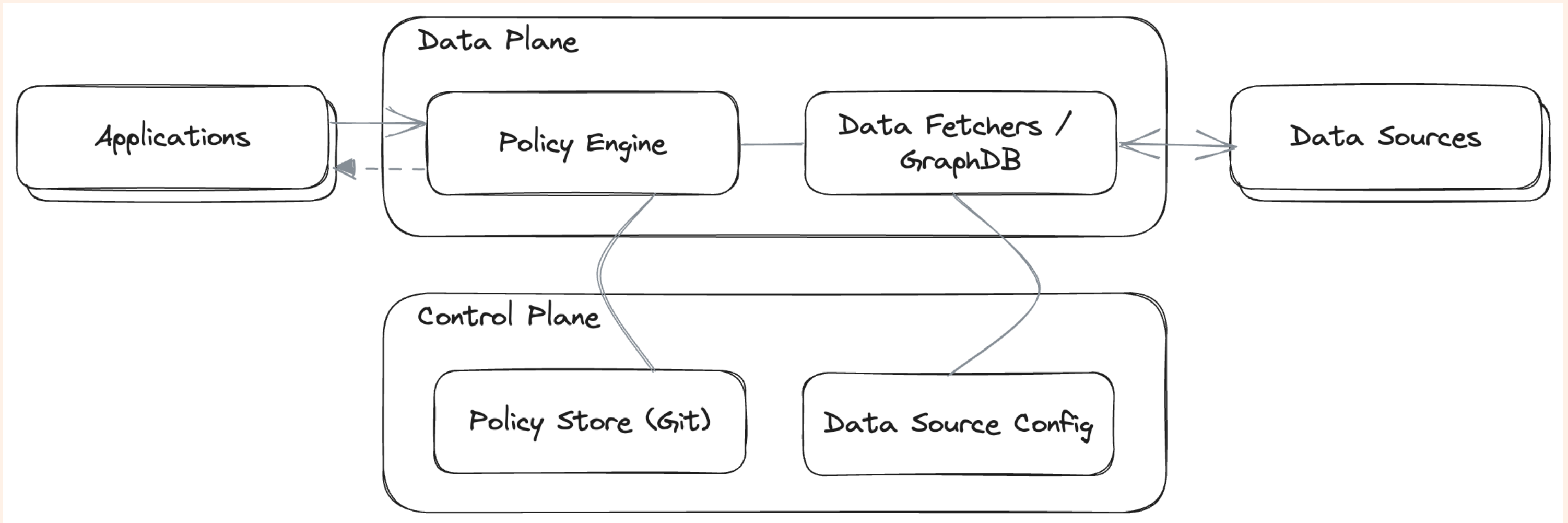
TIMESTAMP	USER	ACTION	RESOURCE
08/23/2023 2:00:53 PM	user_2U0irKXw9UevlotIqyKVyz1n6W3	read	medical_re
08/23/2023 2:00:53 PM	user_2U0irKXw9UevlotIqyKVyz1n6W3	read	profile
08/23/2023 2:00:53 PM	user_2U0irKXw9UevlotIqyKVyz1n6W3	read	health_pla
08/23/2023 2:00:47 PM	user_2U0irKXw9UevlotIqyKVyz1n6W3	read	medical_re
08/23/2023 2:00:47 PM	user_2U0irKXw9UevlotIqyKVyz1n6W3	read	health_pla
08/23/2023 2:00:46 PM	user_2U0irKXw9UevlotIqyKVyz1n6W3	read	profile
08/23/2023 1:58:27 PM	user_2U0iZ44FSRWjy7kcpiAbVT7xEuD	read	profile
08/23/2023 1:58:27 PM	user_2U0iZ44FSRWjy7kcpiAbVT7xEuD	read	medical_re
08/23/2023 1:58:27 PM	user_2U0iZ44FSRWjy7kcpiAbVT7xEuD	read	health_pla
08/23/2023 1:58:21 PM	user_2U0iZ44FSRWjy7kcpiAbVT7xEuD	read	medical_re
08/23/2023 1:58:21 PM	user_2U0iZ44FSRWjy7kcpiAbVT7xEuD	read	health_pla

Audit Event

Timestamp	08/23/2023 2:00:53 PM
User	user_2U0irKXw9UevlotIqyKVyz1n6W3
Tenant	Default Tenant
Resource	Profile
Action	read
Decision	DEFAULT PDP CONFIG PERMITTED

```
1 {
2   "allow": true,
3   "debug": {
4     "rbac": {
5       "code": "no_user_roles",
6       "allow": false,
7       "reason": "no roles assigned to user
8         'user_2U0irKXw9UevIotIqyKVyz1n6W3'",
9     },
10    "rebac": {
11      "code": "allow",
12      "allow": true,
13    "reason": "user 'user_2U0irKXw9UevIotIqyKVyz1n6W3' has the
14      permission on resource
15      'profile:profile_user_2U0irKXw9UevIotIqyKVyz1n6W3
16      tenant 'default', granted by
17      'profile:profile_user_2U0irKXw9UevIotIqyKVyz1n6W3
18      which is granted by
19      'member:member_user_2U0irKXw9UevIotIqyKVyz1n6W3#o
20    "allowing_roles": [
```

Authorization System Building Blocks

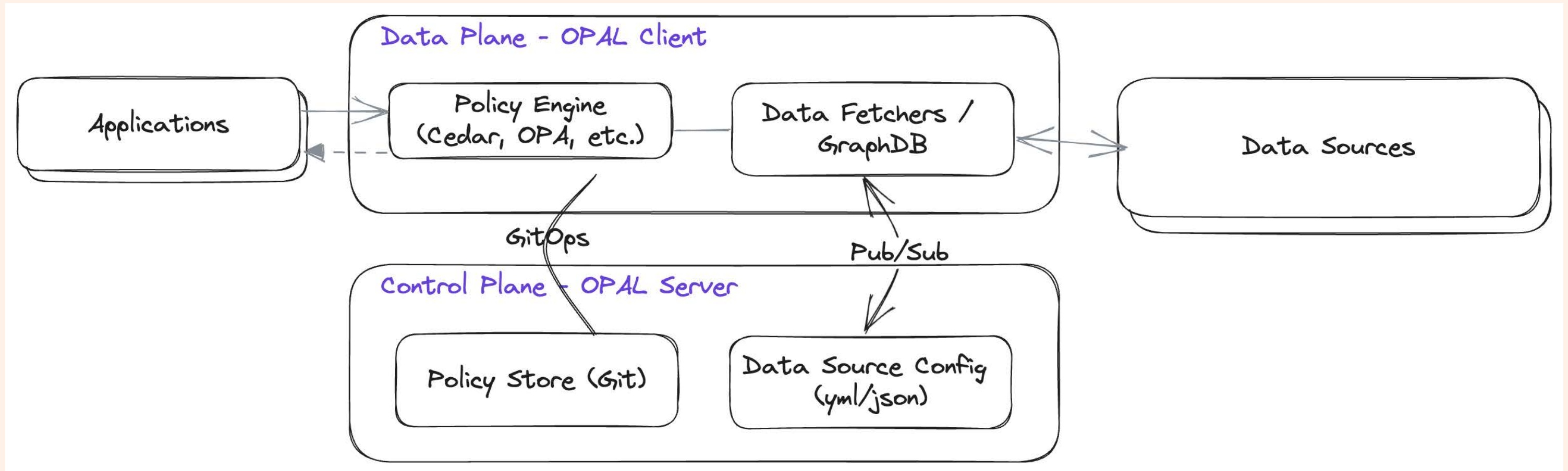


OPAL – Open Policy Administration Layer

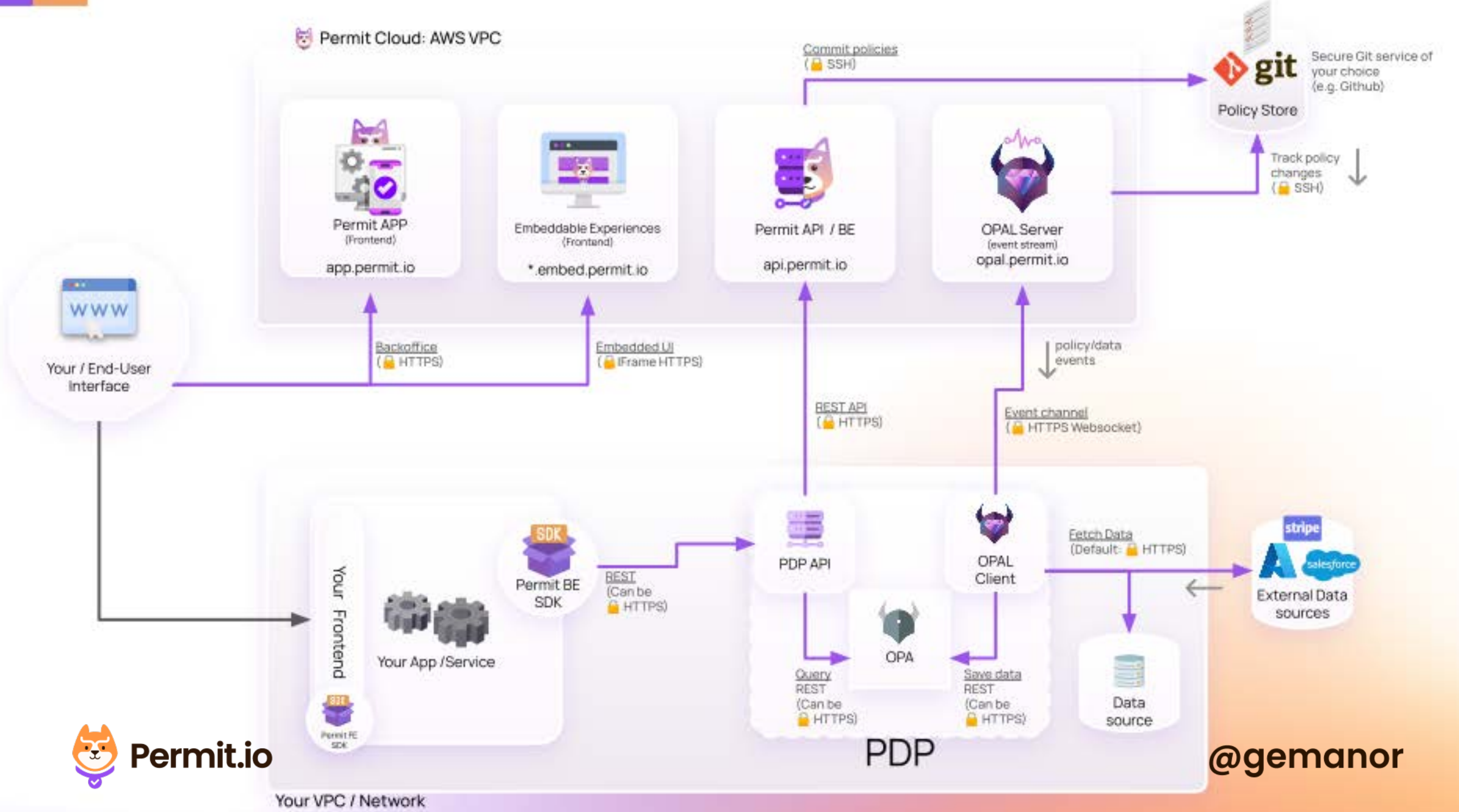


- Open Source, Written in Python
- Sync decision points with data and policy stores
- Auto-scale for engines
- Centralized services such as Audit
- Unified APIs for the enforcement point
- Extensible for any kind of data source
- Supports OPA, Cedar (and soon to be announced more)
- Used by Tesla, Zapier,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

OPAL Based Authorization Architecture



 Demo Time 

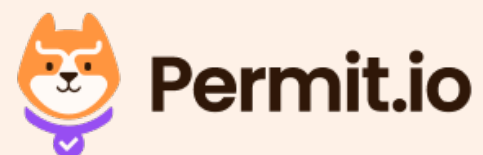


Thank You 🙏

Show your love to OPAL with a
GitHub Star 🌟 ➡️

Find more about OPAL on opal.ac

Follow me on Twitter [@gemanor](https://twitter.com/gemanor)



[@gemanor](https://twitter.com/gemanor)