Continuous Access Control with OPAL and Cedar

Gabriel L. Manor

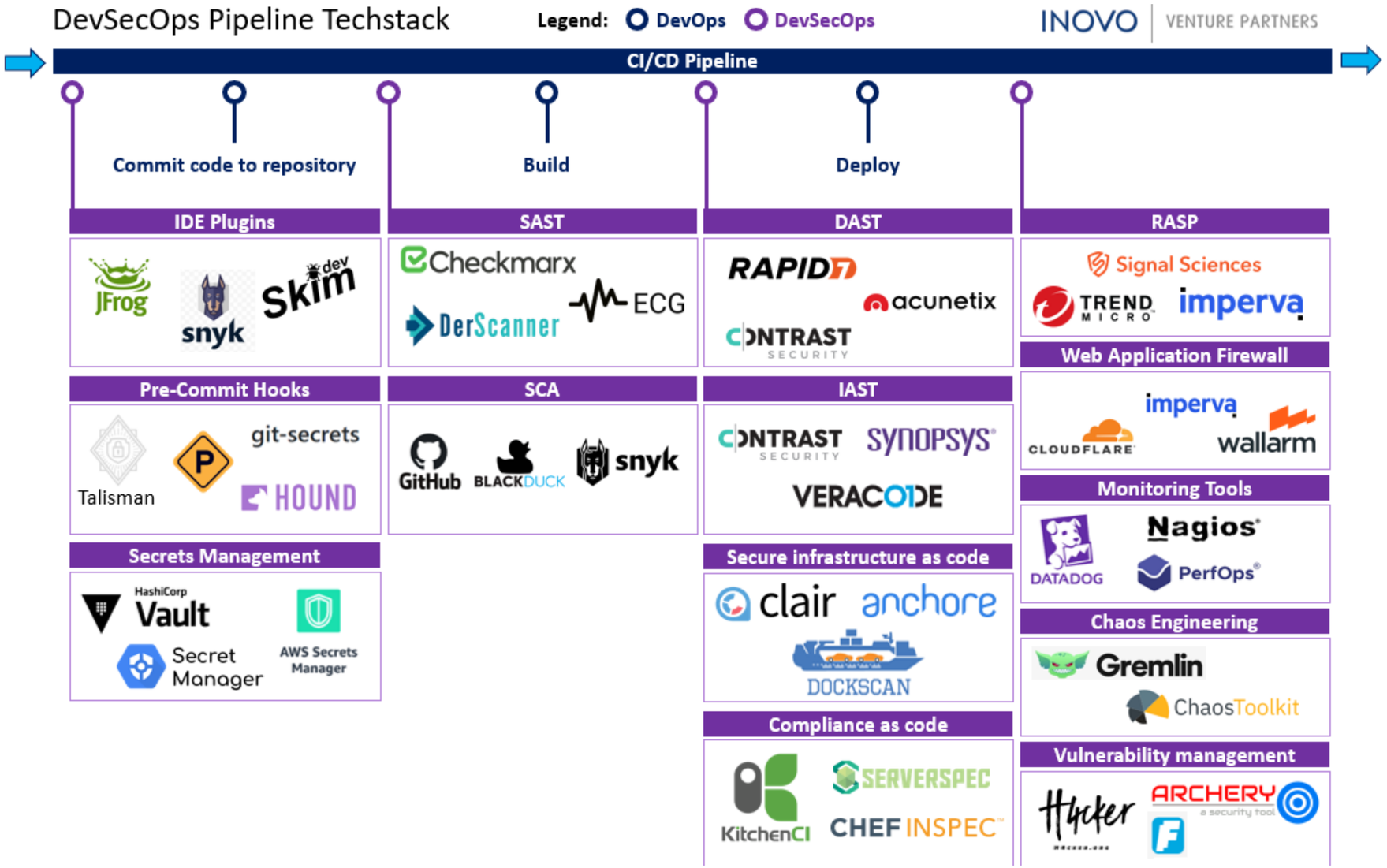
Permit.io

@gemanor

Shift Left

Shift left in security refers to *integrating* security practices and considerations earlier in the software development lifecycle, emphasizing proactive *measures* and reducing vulnerabilities.

— *ChatGPT*



Permit.io

@gemanor

~~Measure~~

Influence

Case Study

Application-Level Access Control

Permit.io

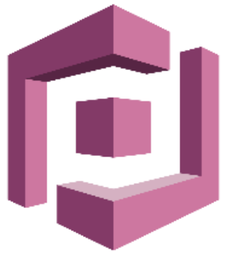
@gemanor

Gabriel L. Manor

Director of DevRel @ Permit.io

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

Auth0



clerk

STYTCH

frontegg

SuperTokens

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

```
// Middleware
if (req.user.roles.indexOf(role) === -1) {
    return res.send(403);
}

// Endpoint
@authz('admin')
const Document = () => {
    ...
}
```

Permit.io

@gemanor

```
// Middleware
const hasRole = roles.filter((r) => (
    req.user.roles.indexOf(role) > -1
))
if (!hasRole) {
    return res.send(403);
}

// Endpoint
@authz('admin')
@authz('paid')
const Document = () => {
    ...
}
```

Permit.io

@gemanor

```
// Middleware
if (args.length === 2 && req.user[args[0]] !== args[1]) {
    return res.send(403);
}
if (req.user.roles.indexOf(role) > -1) {
    return res.send(403);
}

// Endpoint
@authz('admin')
@authz('location', 'eu')
const Document = () => {
    ...
}
```

Permit.io

@gemanor

```
// Middleware
if (typeof args[0] === 'object') {
    return AuthzDesicion(args[0])
}
if (args.length === 2 && req.user[args[0]] !== args[1]) {
    return res.send(403);
}
if (req.user.roles.indexOf(role) > -1) {
    return res.send(403);
}

// Endpoint
@authz('admin')
@authz('location', 'eu')
@authz({
    ...
})
const Document = () => {
    ...
}
```

Permit.io

@gemanor

Implement RBAC to an Application

Permit.io

@gemanor

~~Implement RBAC to an Application~~

Enforce Permissions in Applications

Permit.io

@gemanor

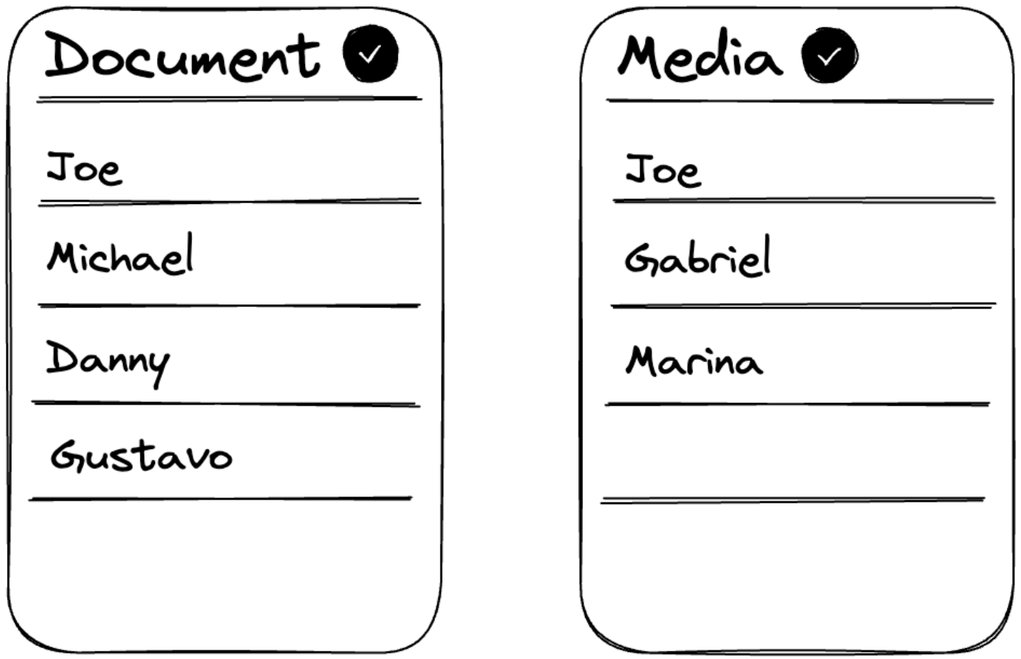
ACL - Access Control List

RBAC - Role-Based Access Control

ABAC - Attribute-Based Access Control

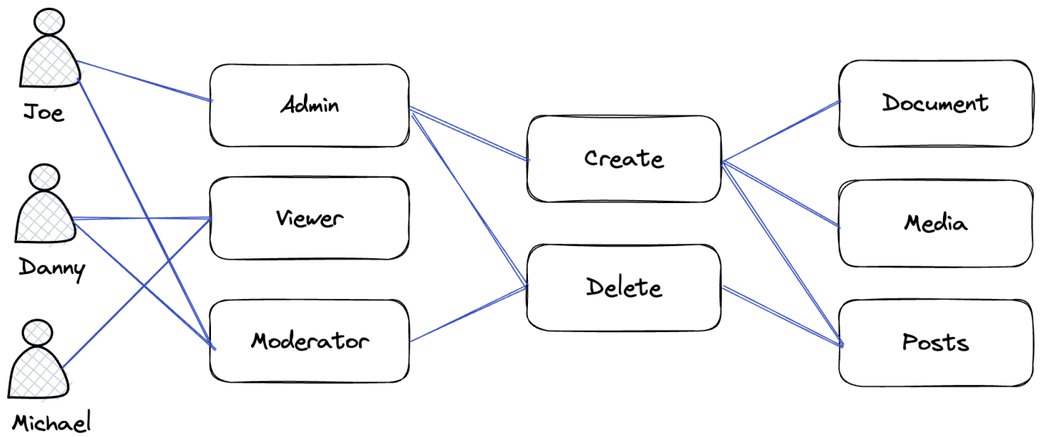
ReBAC - Relationship-Based Access Control

ACL - Access Control List



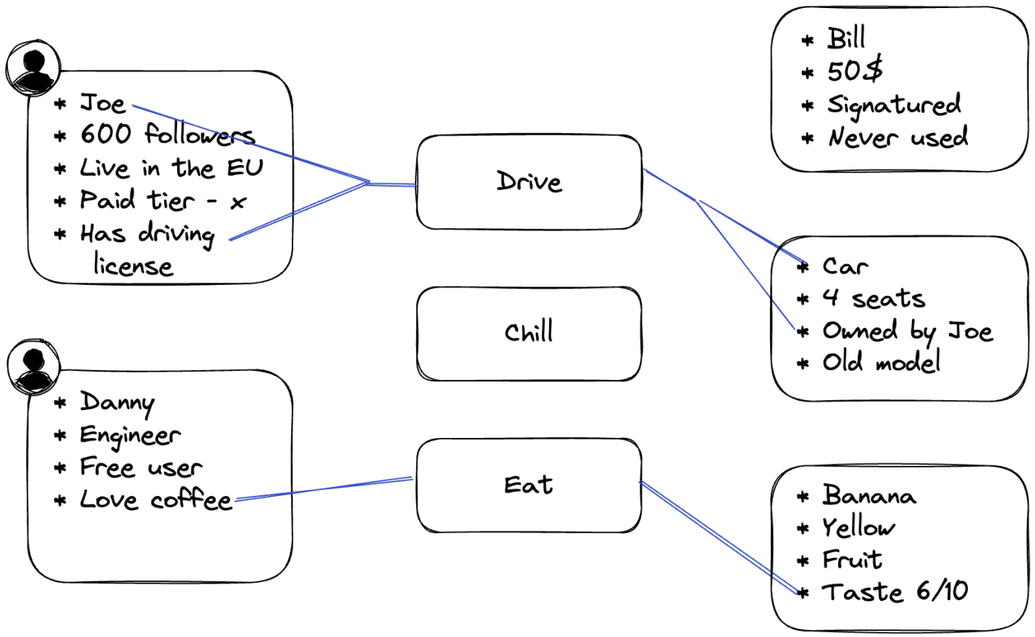
- EOL model
- Widely used in IT systems/ networks
- No segmentation/attribution support
- Hard to scale

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

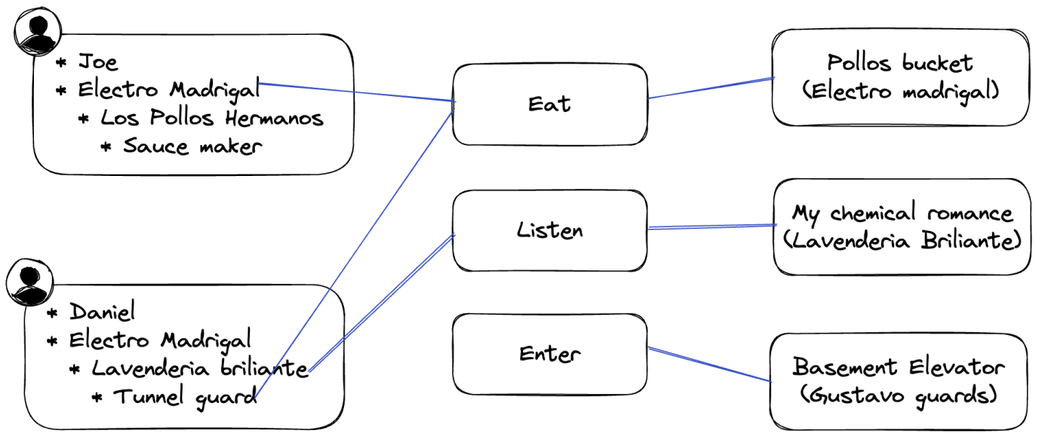


- The most robust model for inspection and desicion making
- Configuration could be hard
- Easy to handle multiple data sources
- 🚀 Highly scalable

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

```
// Middleware
if (typeof args[0] === 'object') {
    return AuthzDesicion(args[0])
}
if (args.length === 2 && req.user[args[0]] !== args[1]) {
    return res.send(403);
}
if (req.user.roles.indexOf(role) > -1) {
    return res.send(403);
}

// Endpoint
@authz('admin')
@authz('location', 'eu')
@authz({
    ...
})
const Document = () => {
    ...
}
```

Permit.io

@gemanor

```
// Middleware
if (typeof args[0] === 'object') {
    return AuthzDesicion(args[0])
}
if (args.length === 2 && req.user[args[0]] !== args[1]) {
    return res.send(403);
}
if (req.user.roles.indexOf(role) > -1) {
    return res.send(403);
}

// Endpoint
@authz('admin')
const Document = () => {
    const enrichedUser = getSocialData(req.user);
    if (enrichedUser.lastGeolocation !== 'eu') {
        return;
    }
    ...
}
```

Permit.io

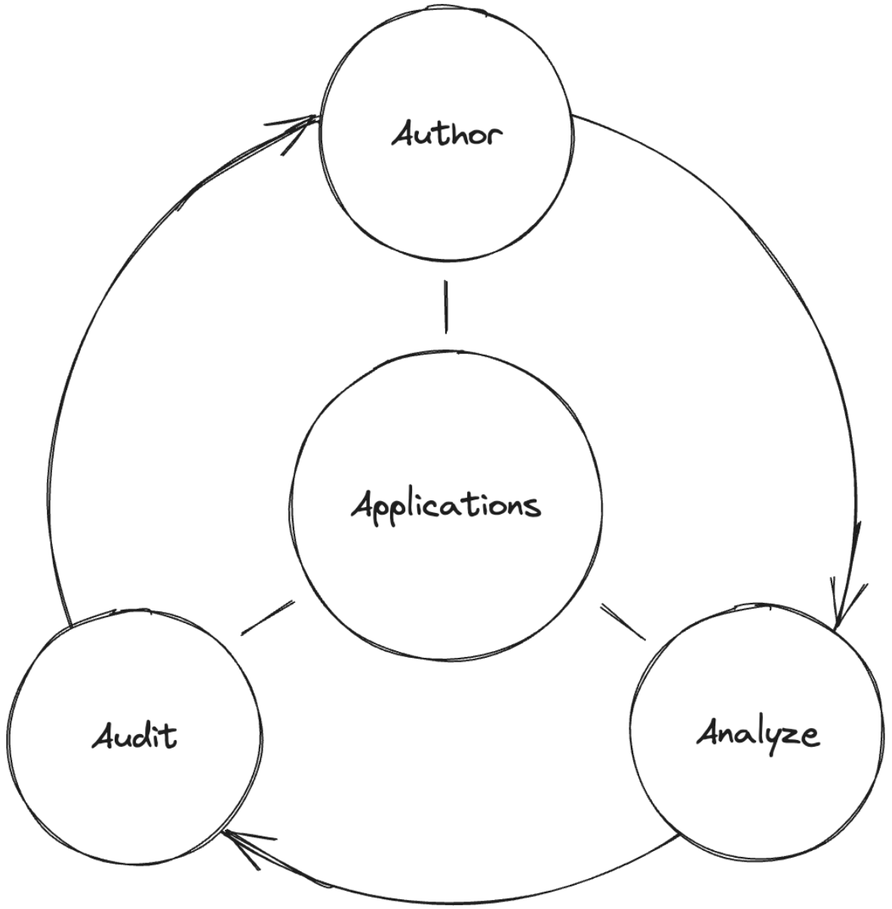
@gemanor

```
// Middleware
if (typeof args[0] === 'object') {
    return AuthzDesicion(args[0])
}
if (args.length === 2 && req.user[args[0]] !== args[1]) {
    return res.send(403);
}
if (req.user.roles.indexOf(role) > -1) {
    return res.send(403);
}

// Endpoint
@authz('admin')
const Document = () => {
    const enrichedUser = getSocialData(req.user);
    if (enrichedUser.lastGeolocation !== 'eu') {
        return;
    }
    ...
}
```

Permit.io

@gemanor

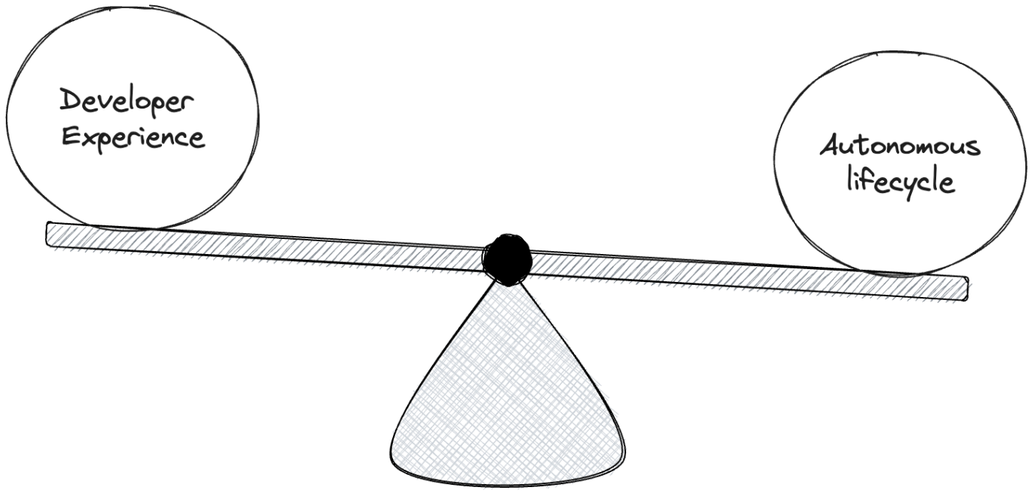
The Autonomous Authorization Lifecycle



Permit.io

@gemanor

Security 🤝 DevEx

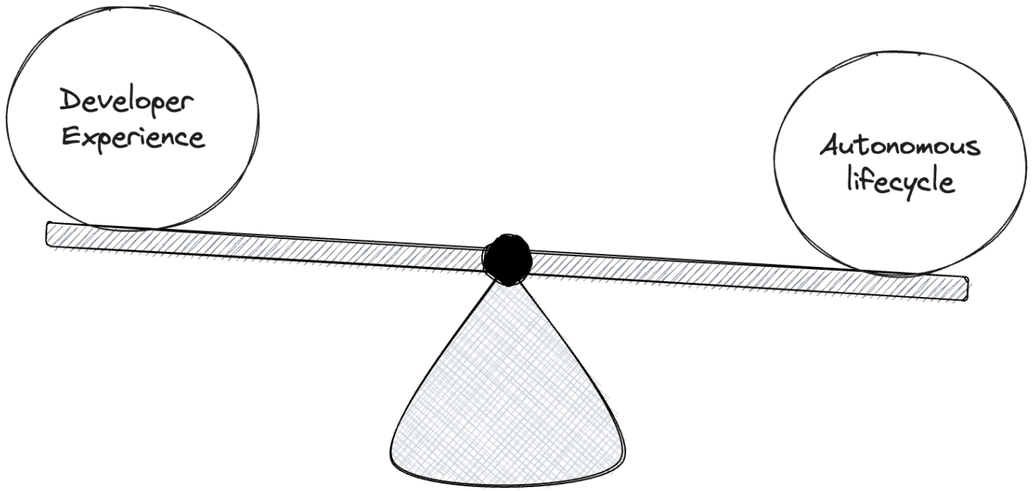


Permit.io

@gemanor

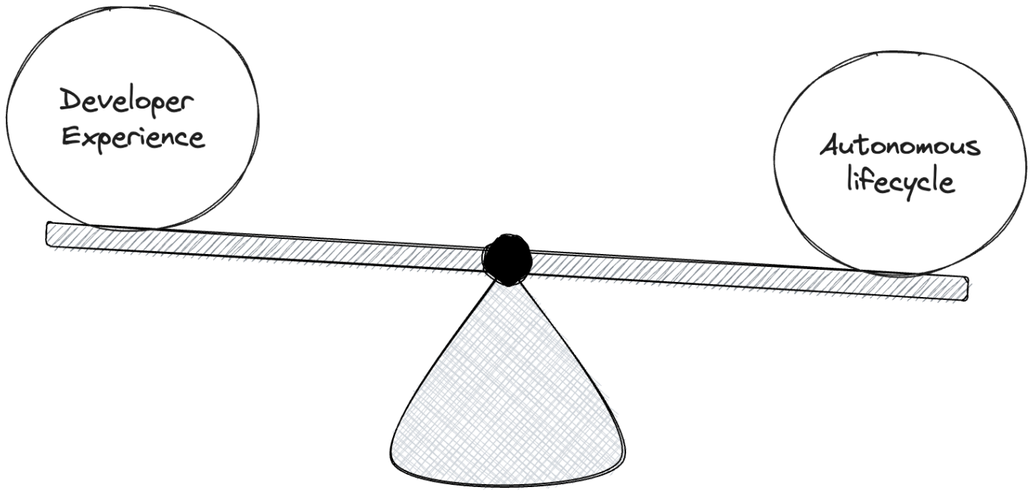
Security 🤝 DevEx

- Cross-platform enforcement



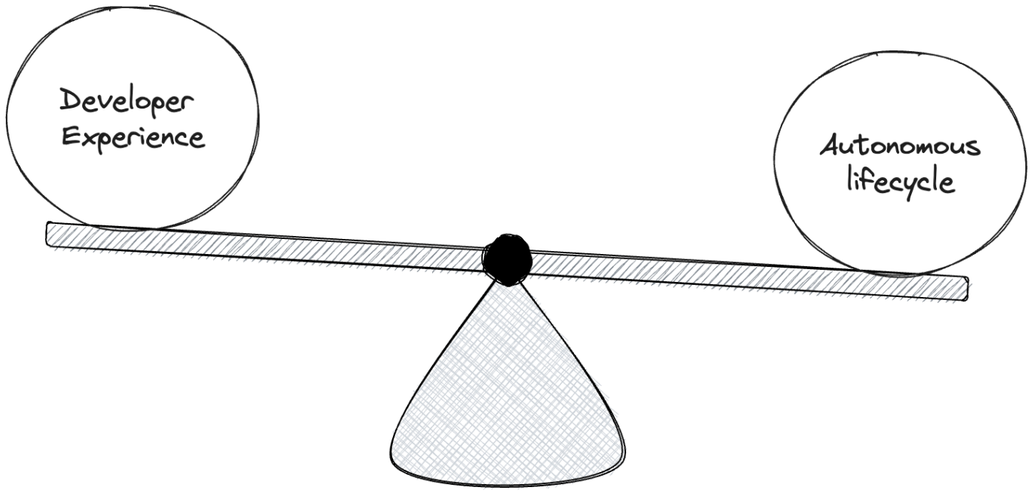
Security 🤝 DevEx

- Cross-platform enforcement
- Performance-aware system



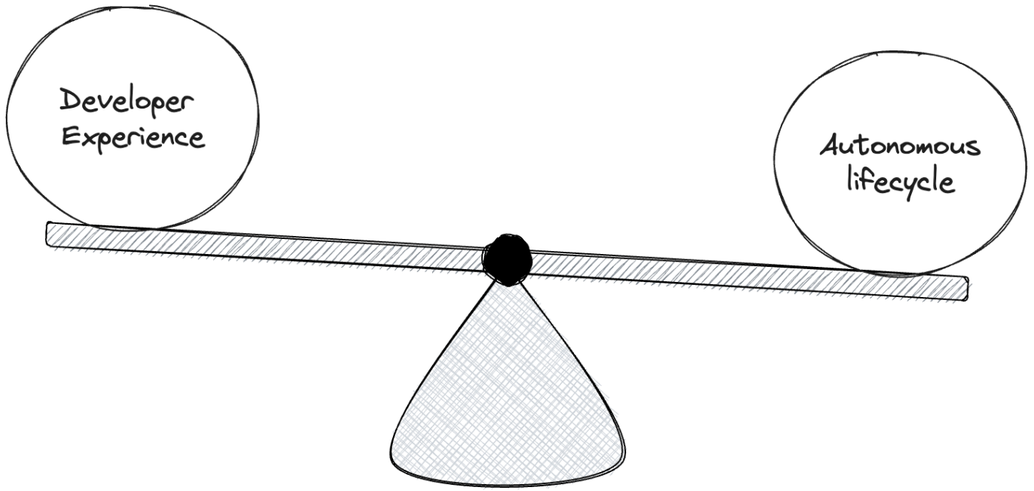
Security 🤝 DevEx

- Cross-platform enforcement
- Performance-aware system
- Zero-effort deployment



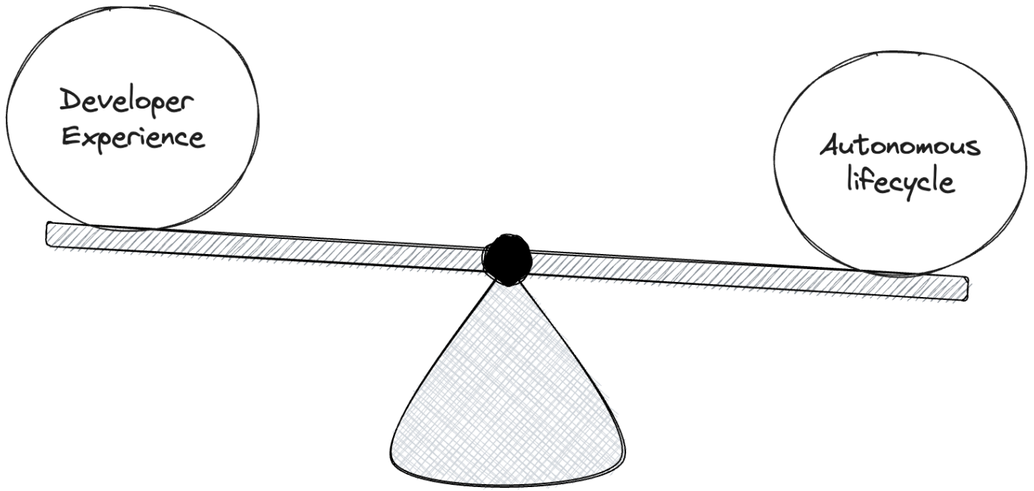
Security 🤝 DevEx

- Cross-platform enforcement
- Performance-aware system
- Zero-effort deployment
- Decentralized architecture



Security 🤝 DevEx

- Cross-platform enforcement
- Performance-aware system
- Zero-effort deployment
- Decentralized architecture
- Seamless data and configuration syncing



Permit.io

@gemanor

Best-Practices for Secure Authorization System

- Agnostic to the permissions model
- Decouple policy from code
- Influenced by security
- DevEx in first mind
- Centralized lifecycle control

Contracts Creates Better Relationships

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


Permit.io

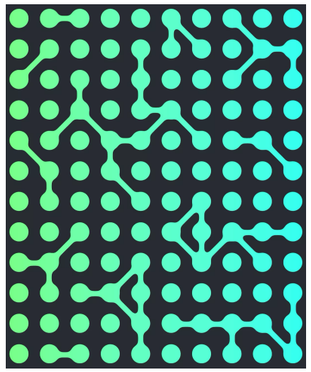
@gemanor

Configuration 🤝 Execution

Open Policy Agent

AWS Cedar

Google Zanzibar



```rego
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]
}
```

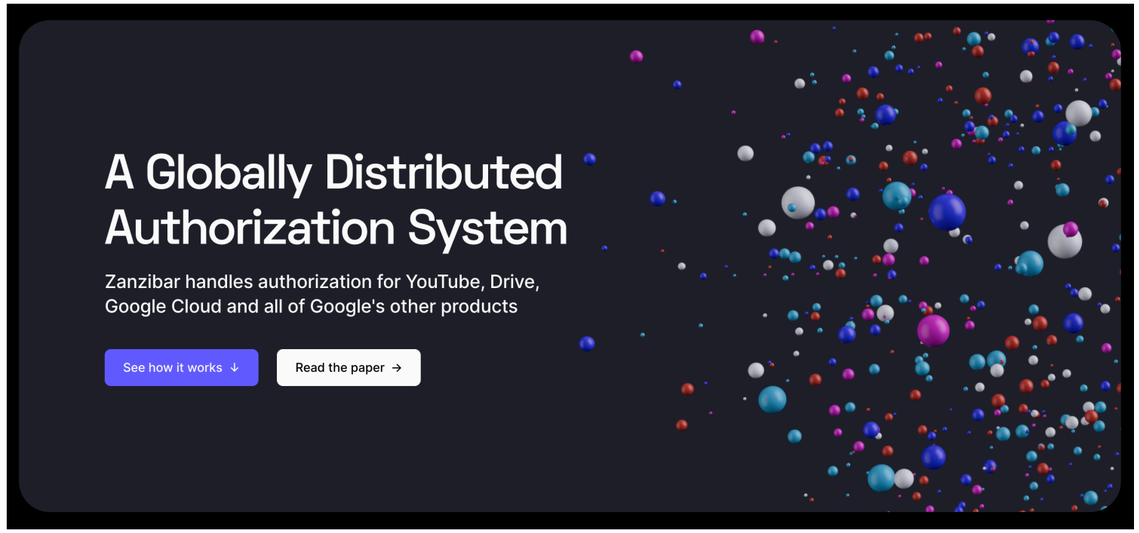
```rego
package app.abac

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



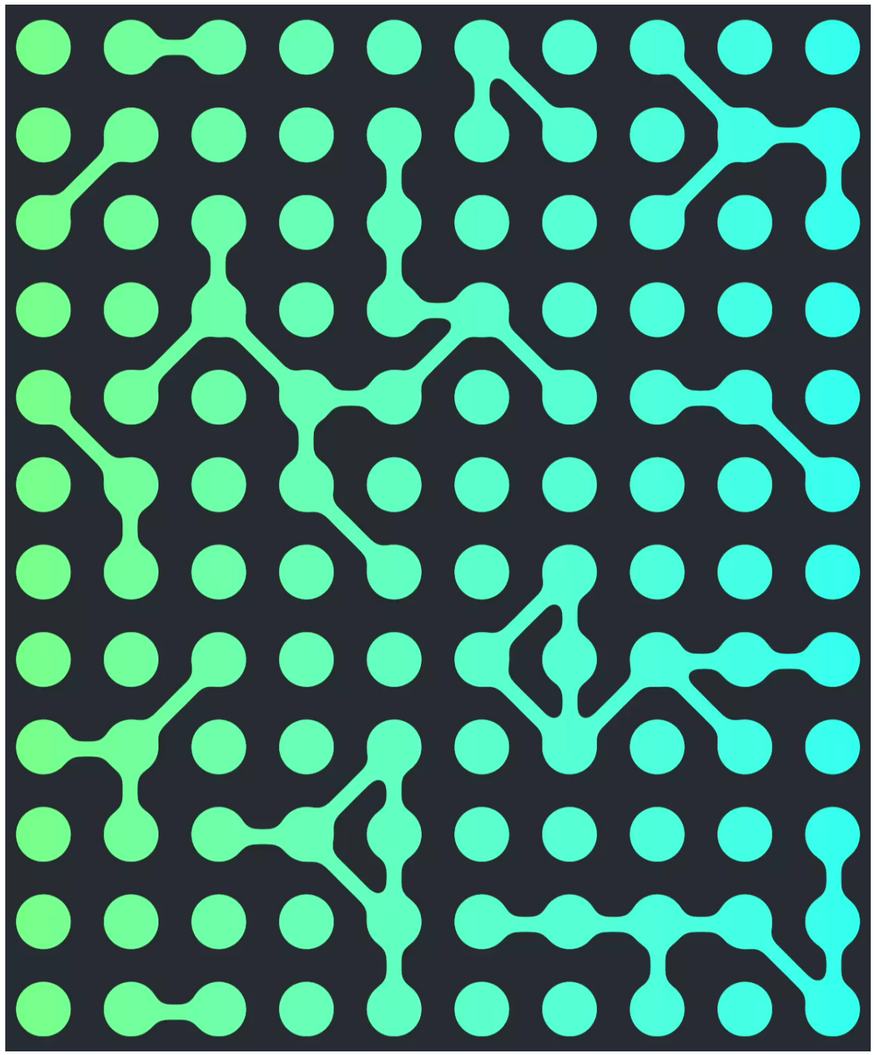

```
name: "doc"

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


```
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

Cedar

Best of Both Worlds

Permit.io

@gemanor

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

Is a Monkey Allowed to Eat a Banana

```
permit(
    principal,
    action,
    resource
);
```

Permit.io

@gemanor

Cedar Entities

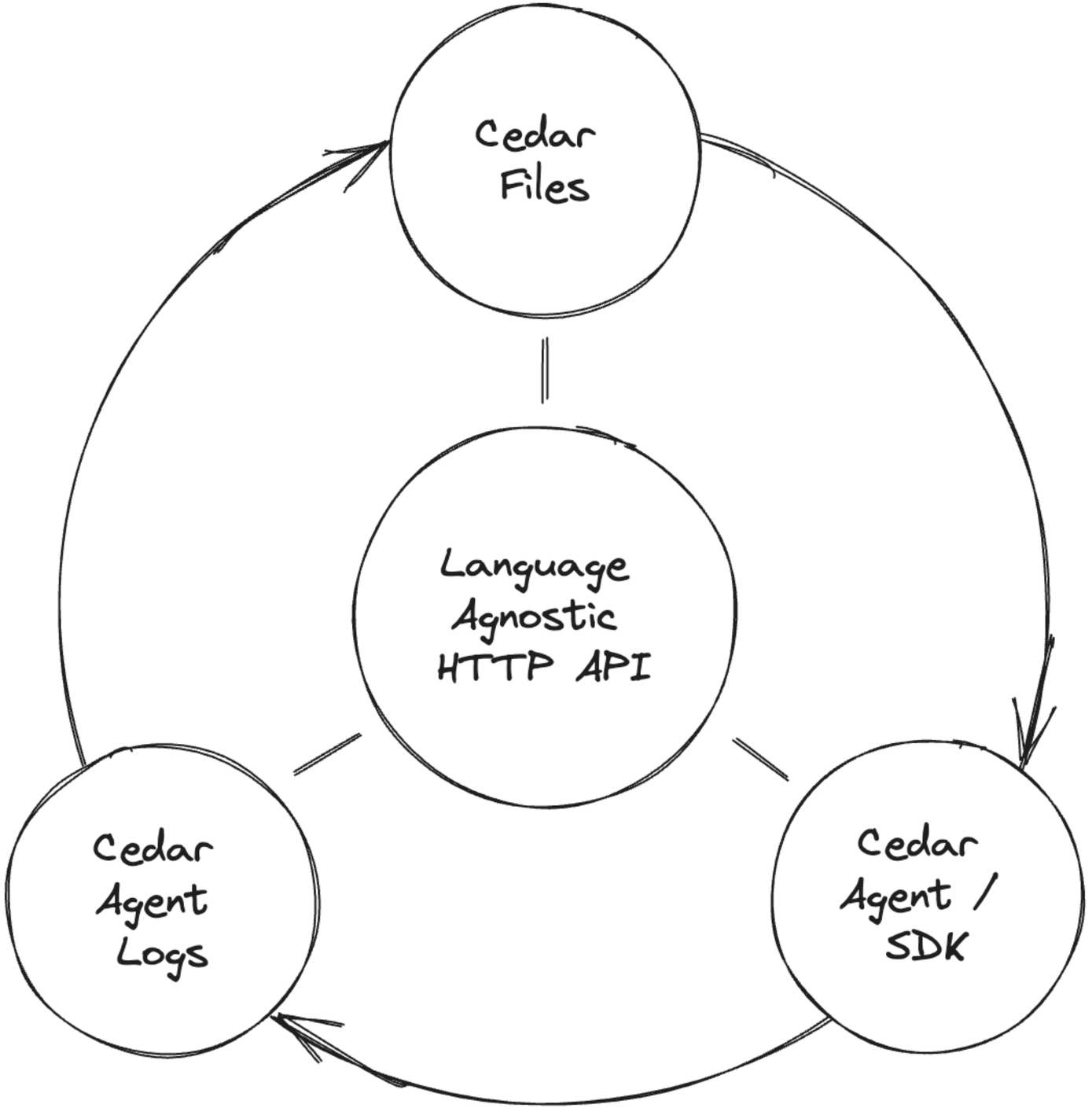
```
...
{
    "attrs": {},
    "parents": [
        {
            "id": "admin",
            "type": "Role"
        }
    ],
    "uid": {
        "id": "admin@blog.app",
        "type": "User"
    }
},
...
```

```
...
{
    "attrs": {},
    "parents": [],
    "uid": {
        "id": "article",
        "type": "ResourceType"
    }
},
{
    "attrs": {},
    "parents": [],
    "uid": {
        "id": "put",
        "type": "Action"
    }
},
...
```

```
...
{
    "attrs": {},
    "parents": [],
    "uid": {
        "id": "admin",
        "type": "Role"
    }
},
...
```

Permit.io

@gemanor



Permit.io

@gemanor

Cedar Agent

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

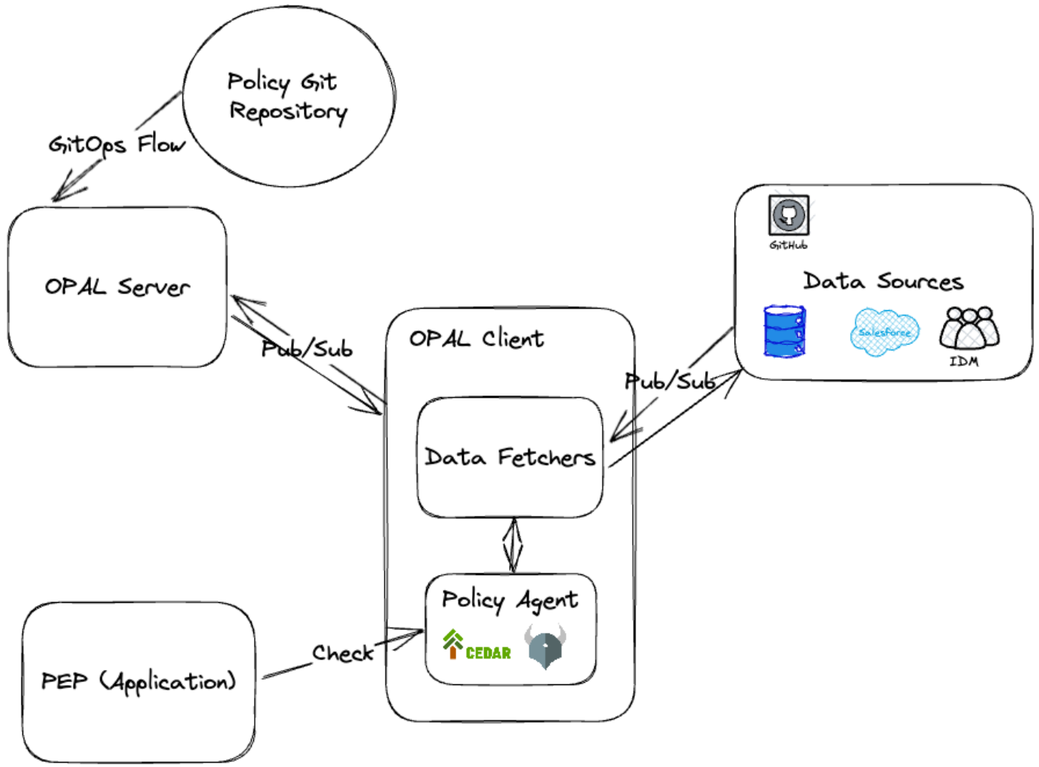
Policy Store

- Usually a Git repository
- Uses Git features, such as branching and version control
- Immutable versioning
- Use IaC principles for permissions management

Data Store

- Any data we need for policy execution
- Traditionally - IDM
- Currently - multiple sources
- In Cedar, transformed to Cedar entities

OPAL - Open Policy Administration Layer



- Sync decision points with data and policy stores
- Auto-scale for engines
- Centralized services such as Audit
- Unified APIs for the enforcement point

Permit.io

@gemanor

Enforcement Points

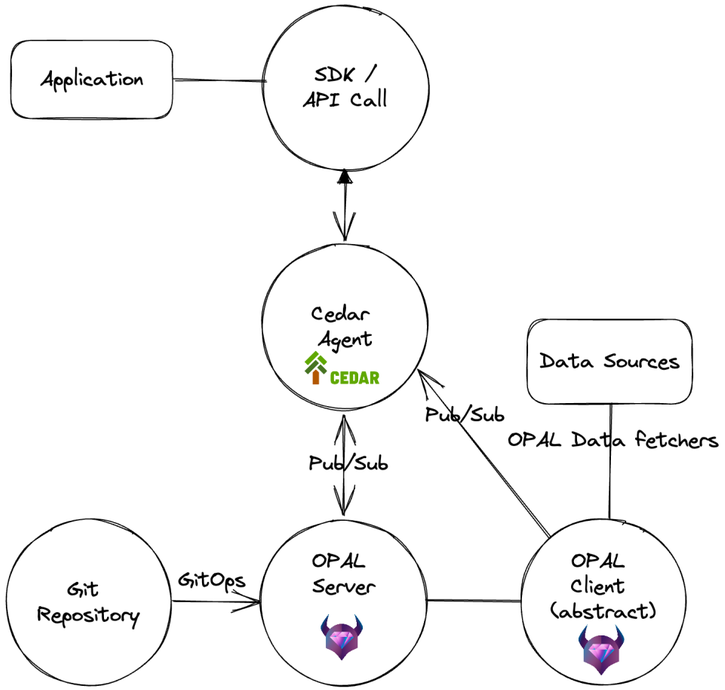
```python
# Call authorization service
# In the request body, we pass the relevant request information
response = requests.post('http://host.docker.internal:8180/v1/is_authorized', json={
    "principal": f"User::\"{user}\"",
    "action": f"Action::\"{method.lower()}\"",
    "resource": f"ResourceType::\"{original_url.split('/')[1]}\"",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

```javascript
// Call the authorization service (Decision Point)
const response = await fetch('http://host.docker.internal:8180/v1/is_authorized', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
        'Accept': 'application/json',
    },
    // The body of the request is the authorization request
    body: JSON.stringify({
        "principal": `User::\"${user}\"`,
        "action": `Action::\"${method.toLowerCase()}\"`,
        "resource": `ResourceType::\"${originalUrl.split('/')[1]}\"`,
        "context": body
    })
});
```

Permit.io

@gemanor

A Cedar-Based Authorization System

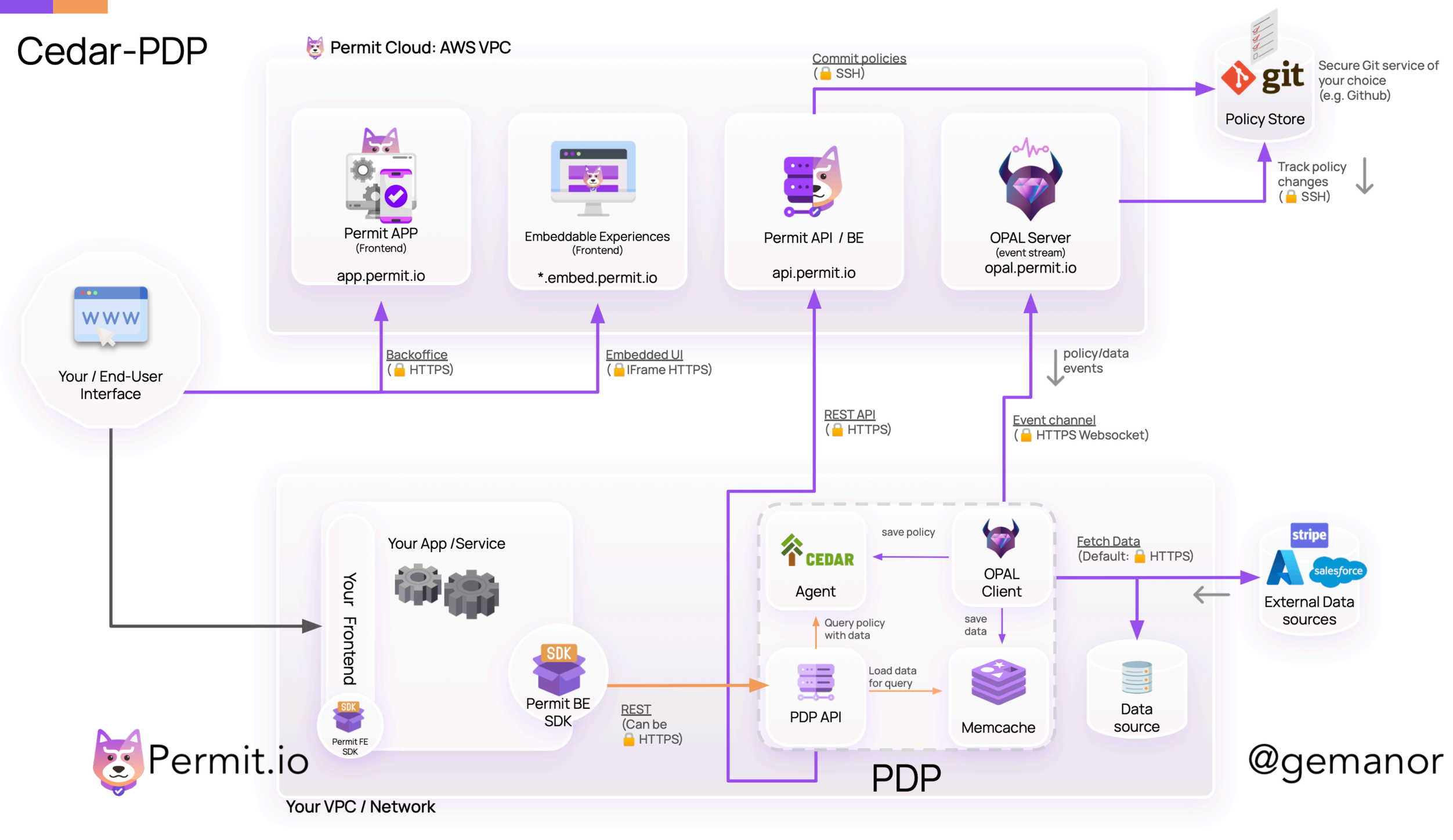


Permit.io

@gemanor

Demo time 🍿

Cedar-PDP


Permit Cloud: AWS VPC
Commit policies
(SSH)
Secure Git service of your choice (e.g. Github)
Policy Store
Track policy changes
(SSH)
Permit APP
(Frontend)
app.permit.io
Embeddable Experiences
(Frontend)
*.embed.permit.io
Permit API / BE
api.permit.io
OPAL Server
(event stream)
opal.permit.io
Your / End-User Interface
Backoffice
(HTTPS)
Embedded UI
(IFrame HTTPS)
policy/data events
REST API
(HTTPS)
Event channel
(HTTPS Websocket)
Your App /Service
Your Frontend
SDK
Permit FE SDK
SDK
Permit BE SDK
REST
(Can be HTTPS)
CEDAR
Agent
save policy
OPAL Client
Fetch Data
(Default: HTTPS)
stripe
salesforce
External Data sources
Query policy with data
save data
PDP API
Load data for query
Memcache
Data source
PDP
Permit.io
Your VPC / Network
@gemanor

Thank You 🙏

Show your love to Cedar-agent
with a GitHub Star ⭐ 👉

Permit.io

@gemanor