Your Cheat Code for API Authorization

Omri Gazitt
Co-founder & CEO, aserto.com

Aserto

{ AppDeveloperCon NORTH AMERICA }

November 12, 2024
Salt Lake City

About Me

Dev Tech, Cloud, IAM, startups, skiing

@omrig

omri@aserto.com

Authentication != Authorization

Authentication

Did the user prove they are who they say they are?

Standards:

Developer services:

Authorization

What can the user do in the context of this app?

Standards: ?

Developer services: ?

Problems:

- Bad security[1]
- Inconsistency
- Opportunity cost

OWASP TOP 10

Broken Access Control: #1 !!

(1) https://owasp.org/Top10/

Authorization is finally having its moment…

Zanzibar[1]

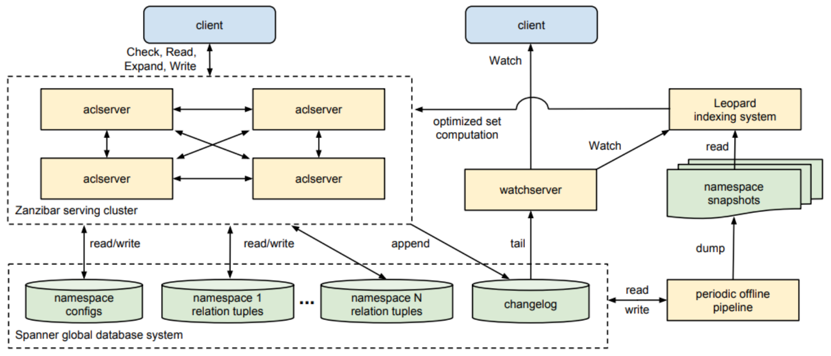
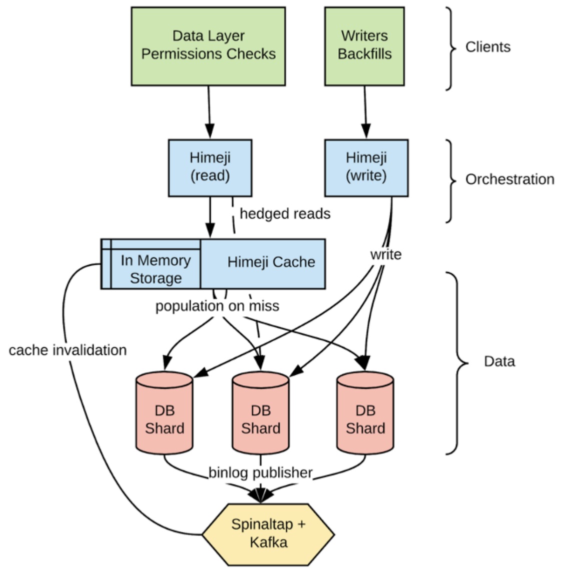
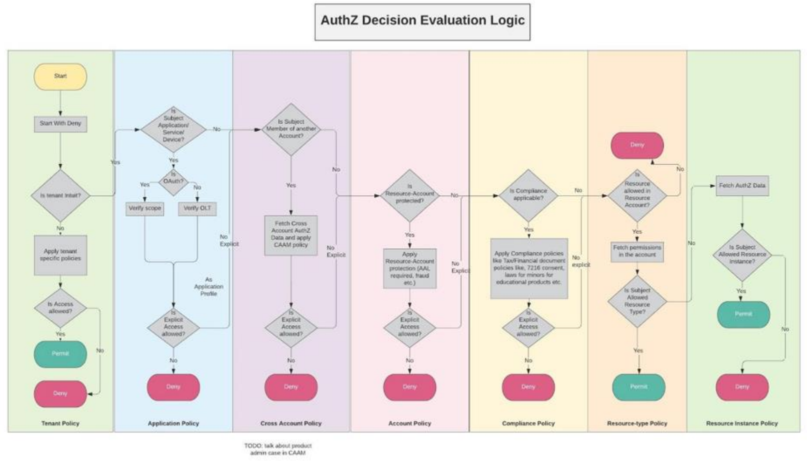


Figure 2: Zanzibar architecture. Arrows indicate the direction of data flow.

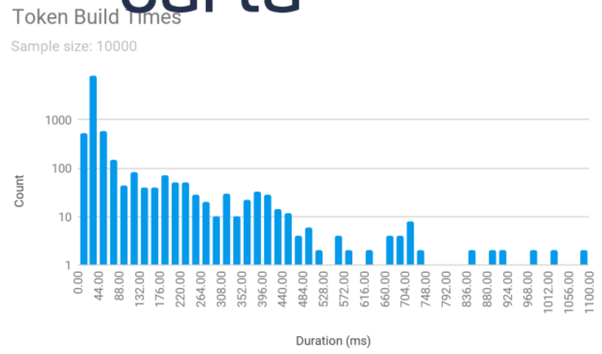
Himeji[3]



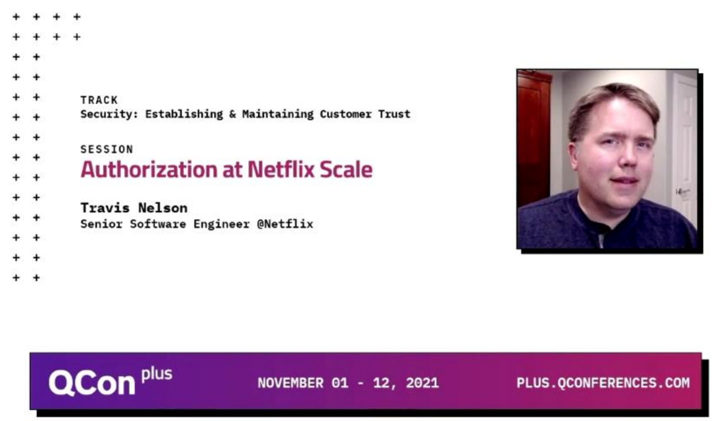
AuthZ[2]



AuthZ[4]



Product Access Service[5]


TRACK
Security: Establishing & Maintaining Customer Trust

SESSION
Authorization at Netflix Scale

Travis Nelson
Senior Software Engineer @Netflix

QCon plus NOVEMBER 01 - 12, 2021 PLUS.QCONFERENCES.COM

(1) https://research.google/pubs/pub48190/
(2) https://medium.com/intuit-engineering/authz-intuits-unified-dynamic-authorization-system-bea554d18f91
(3) https://medium.com/airbnb-engineering/himeji-a-scalable-centralized-system-for-authorization-at-airbnb-341664924574
(4) https://medium.com/building-carta/authz-cartas-highly-scalable-permissions-system-782a7f2c840f
(5) https://www.infoq.com/presentations/authorization-scalability/

Cloud-native authorization changes the game

	"Old-school"	Cloud-native	
WHAT	Coarse-grained, tenant-level permissions	**Fine-grained**: resource-level permissions	... Principle of least privilege
HOW	Authorization "spaghetti logic" embedded in the application	**Policy-based**: authorization logic extracted out of the application	... Separation of duties
WHEN	Permissions evaluated at login time, scopes embedded in access token	**Real-time**: permissions evaluated before granting access to resource	... Continuous enforcement

Fine-grained access control evolution

Access Control Lists: 1980's-1990's (UNIX, NT CACL)

ACL

Does Alice have read access to this file?

Role-based Access Control: 1990's-2000's (LDAP, AD)

RBAC

Is Bob in the sales-admin role?

Attribute-based Access Control: 2000's-2010's (XACML)

ABAC

Is Mallory in the sales department, is the document in the sales folder, and is it currently working hours in the US?

Relationship-based Access Control – 2020 (Zanzibar)

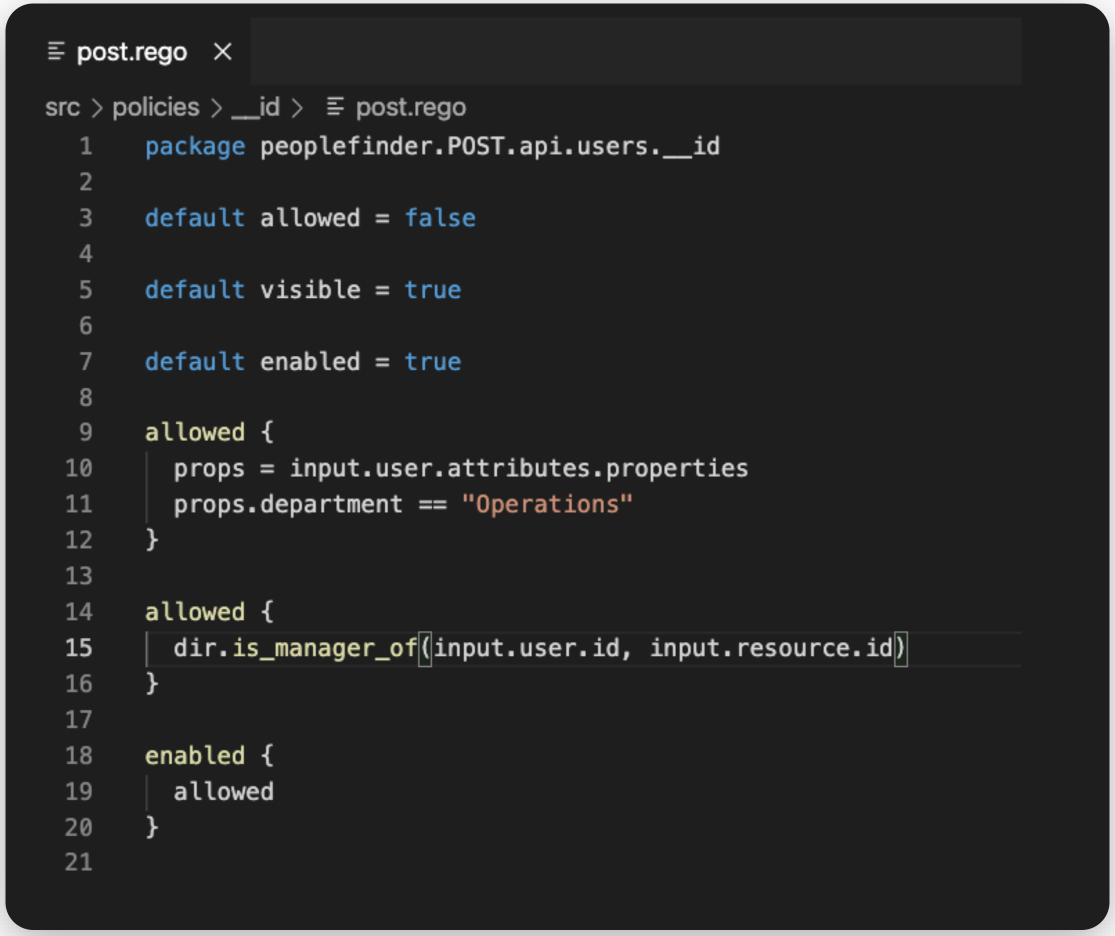
ReBAC

Does Eve have read access to this document if she's in the sales group, the document is in the sales folder, and the sales group is in the "editor" relation on the sales folder?

Policy-based access management

Lift access control logic out of the application and into its own policy-as-code artifact

Policy written in Rego (Open Policy Agent / Topaz)

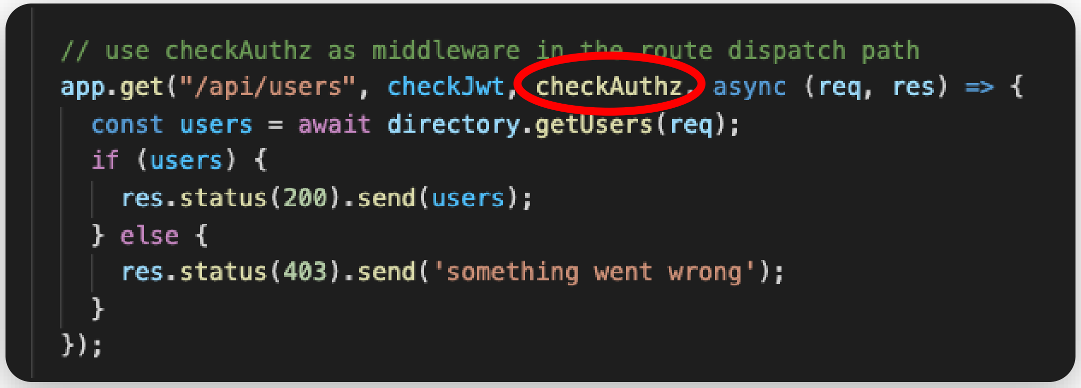


post.rego

src > policies > __id > post.rego

```
package peoplefinder.POST.api.users.__id

default allowed = false

default visible = true

default enabled = true

allowed {
  props = input.user.attributes.properties
  props.department == "Operations"
}

allowed {
  dir.is_manager_of(input.user.id, input.resource.id)
}

enabled {
  allowed
}
```

Application code uses middleware to call authz service



```
// use checkAuthz as middleware in the route dispatch path
app.get("/api/users", checkJwt, checkAuthz, async (req, res) => {
  const users = await directory.getUsers(req);
  if (users) {
    res.status(200).send(users);
  } else {
    res.status(403).send('something went wrong');
  }
});
```

Store and version policy just like application code

Every policy change is part of a git changelog

Policy can be evolved by security team, decoupled from app

Policy can be built into an immutable image and signed (https://github.com/opcr-io/policy)

Real-time access checks

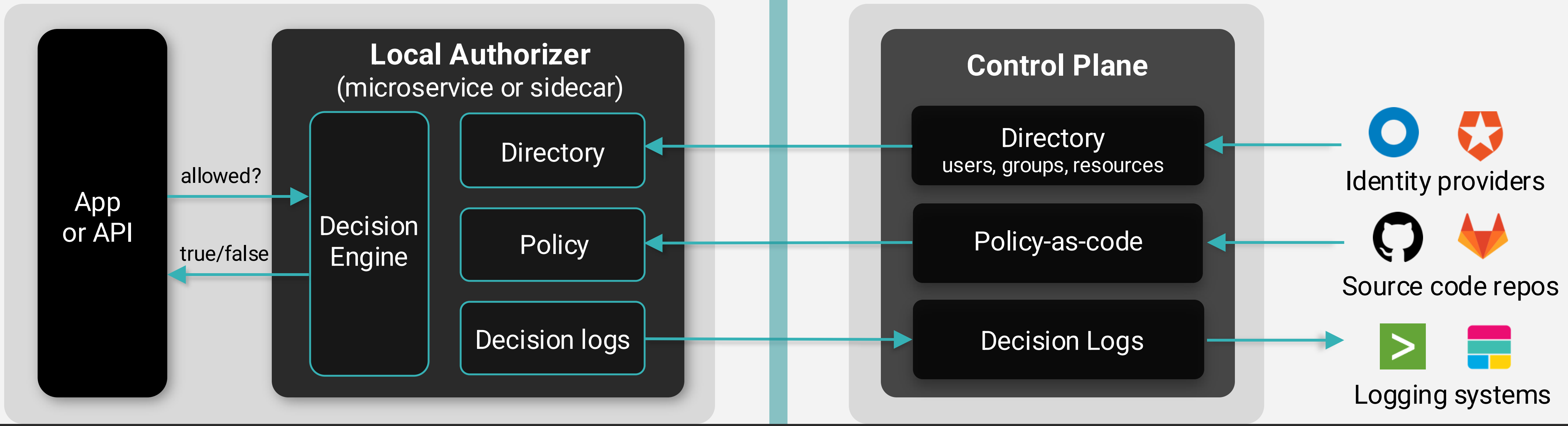
Done correctly, authorization is a distributed systems problem

Authorize locally

- Authorization is in the critical path of every application request
- Requires 100% availability at milliseconds of latency
- Must be deployed right next to the application / microservice
- Compute decision using cached policy, user, and resource data

Manage centrally

- Control plane manages policies, user directory, resource data
- Decision log collection and aggregation
- High-speed event and data fabric between control plane and edge



Cloud-native authorization: open source landscape

“Policy as code” (ABAC)

Open Policy Agent

+ CNCF graduated project, one single OSS implementation
+ general-purpose, flexible decision engine
+ built for PBAM & ABAC (similar to XACML)

- Datalog-derived logic-based language, high learning curve
- no help with modeling application authz (FGA / ReBAC)
- no help with getting user / resource data to the engine

Alternatives: Casbin CEDAR cerbos

“Policy as data” (ReBAC)

Zanzibar-inspired projects

+ opinionated authorization model
+ can model a relationship graph between subjects & objects
+ built for Google Docs style authorization (RBAC + ACL)

- at least half a dozen competing OSS implementations
- no common schema / data language
- hard to go outside strict ReBAC (e.g. attributes)

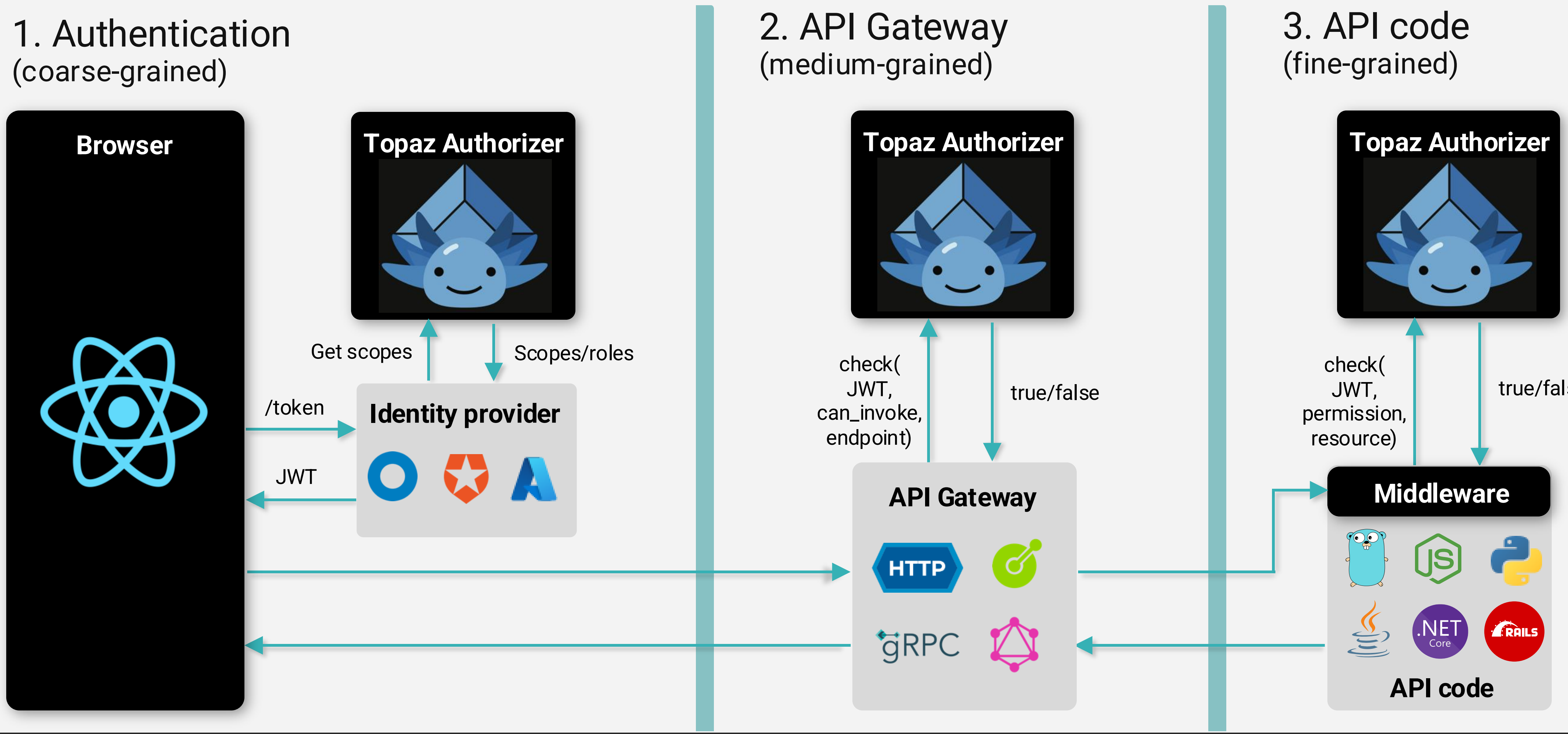
Alternatives: OpenFGA™ spicedb oso

Topaz: “The best of both”

🌟 🌟 https://github.com/aserto-dev/topaz 🌟 🌟

Enforcement points

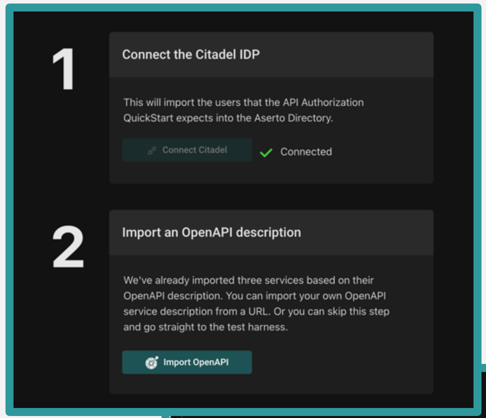
Zero-trust means defense-in-depth



Demo – API Authorization

Using Topaz/Aserto & Zuplo API Gateway

1. Import OpenAPI spec

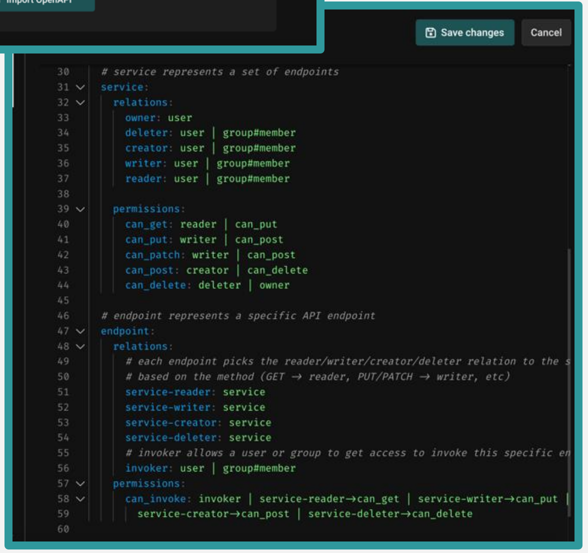


1 **Connect the Citadel IDP**

This will import the users that the API Authorization QuickStart expects into the Aserto Directory.

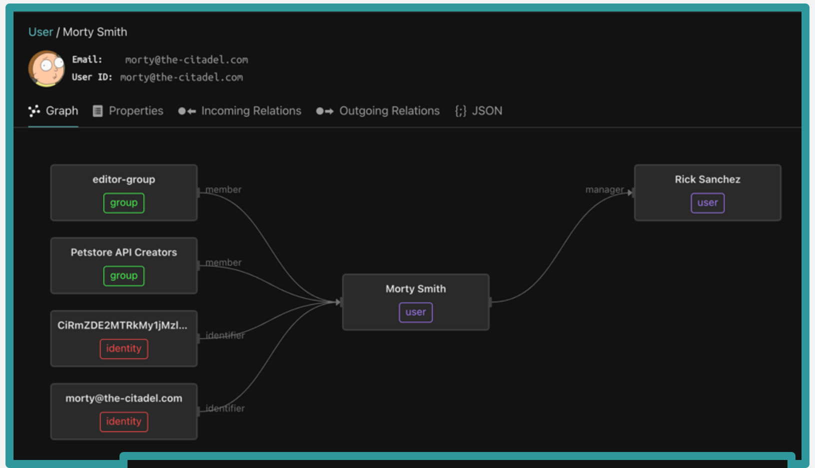
2 **Import an OpenAPI description**

We've already imported three services based on their OpenAPI description. You can import your own OpenAPI service description from a URL. Or you can skip this step and go straight to the test harness.



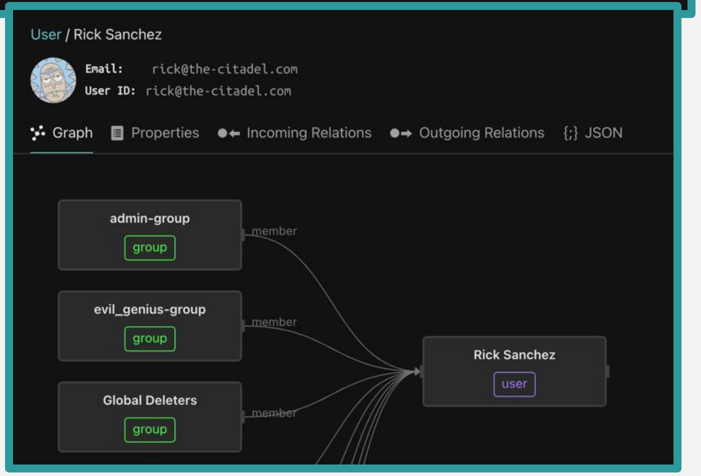
```
# service represents a set of endpoints
service:
  relations:
    owner: user
    deleter: user | group#member
    creator: user | group#member
    writer: user | group#member
    reader: user | group#member

  permissions:
    can_get: reader | can_put
    can_put: writer | can_post
    can_patch: writer | can_post
    can_post: creator | can_delete
    can_delete: deleter | owner

# endpoint represents a specific API endpoint
endpoint:
  relations:
    # each endpoint picks the reader/writer/creator/deleter relation to the s
    # based on the method (GET → reader, PUT/PATCH → writer, etc)
    service-reader: service
    service-writer: service
    service-creator: service
    service-deleter: service
    # invoker allows a user or group to get access to invoke this specific en
    invoker: user | group#member
  permissions:
    can_invoke: invoker | service-reader→can_get | service-writer→can_put |
      service-creator→can_post | service-deleter→can_delete

```

2. Set up relationships



User / Morty Smith

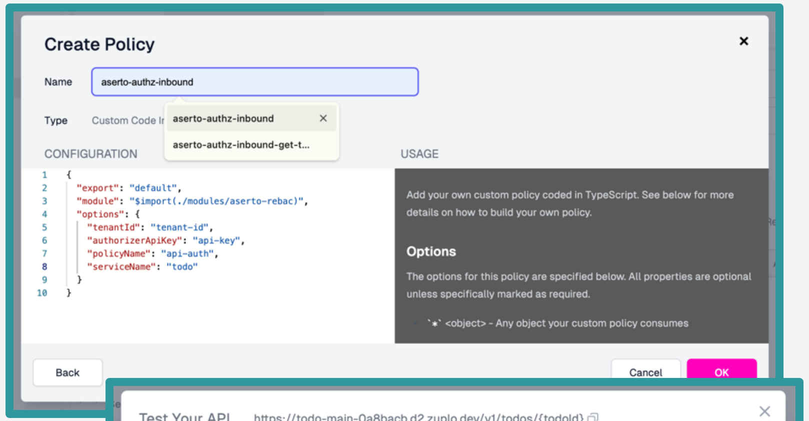
Email: morty@the-citadel.com
User ID: morty@the-citadel.com



User / Rick Sanchez

Email: rick@the-citadel.com
User ID: rick@the-citadel.com

3. Enforce from gateway



Create Policy

Name: aserto-authz-inbound

Type: Custom Code In

CONFIGURATION

```
{
  "export": "default",
  "module": "$import(./modules/aserto-rebac)",
  "options": {
    "tenantId": "tenant-id",
    "authorizerApiKey": "api-key",
    "policyName": "api-auth",
    "serviceName": "todo"
  }
}
```

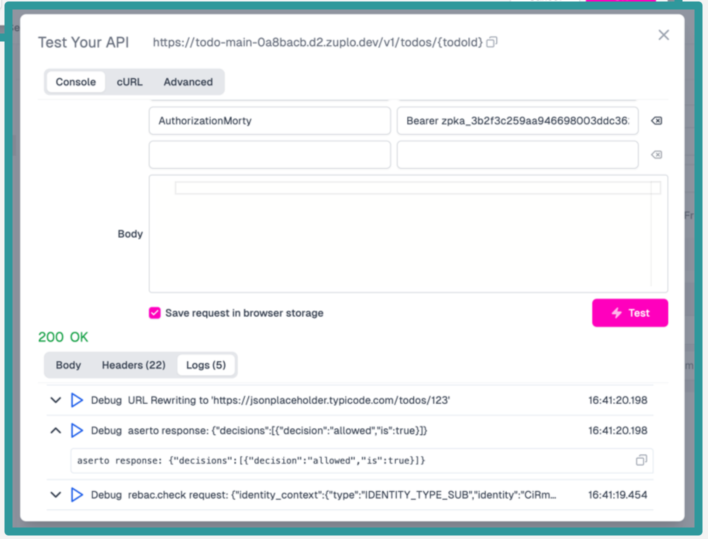
USAGE

Add your own custom policy coded in TypeScript. See below for more details on how to build your own policy.

Options

The options for this policy are specified below. All properties are optional unless specifically marked as required.

- `*` <object> - Any object your custom policy consumes



Test Your API https://todo-main-0a8bacb.d2.zuplo.dev/v1/todos/{todoId}

AuthorizationMorty	Bearer zpka_3b2f3c259aa946698003ddc36

Body

[✓] Save request in browser storage

200 OK

```
Debug  URL Rewriting to 'https://jsonplaceholder.typicode.com/todos/123'                     16:41:20.198
Debug  aserto response: {"decisions":[{"decision":"allowed","is":true}]}                      16:41:20.198
  aserto response: {"decisions":[{"decision":"allowed","is":true}]}
Debug  rebac.check request: {"identity_context":{"type":"IDENTITY_TYPE_SUB","identity":"CiRm...   16:41:19.454
```

The principles of cloud-native authorization

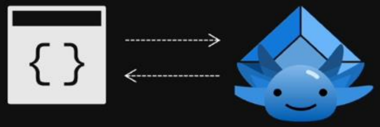
Fine-grained	Support a consistent model (RBAC, ABAC, ReBAC) that fits the application domain
Policy-based	Extract policy out of the app and into its own repo, and build into a signed image
Real-time	Authorization is a local call, executing over fresh user / resource data
Centrally managed	Policy and directory/resource data are centrally managed
Compliance & forensics	Decision logs are aggregated and stored centrally
Developer-centric	Authorization with a single line of code
Integrates easily	Identity providers, source code repos, artifact registries, logging systems, API gateways
Cloud-native and open	Ecosystem effects of using k8s-native technologies like Open Policy Agent, Topaz, OCI

TOPAZ

★Star the repo!★

The best of **OPA** and **Zanzibar**

Fast



~1ms authorization

Flexible

RBAC ABAC ReBAC

Any model, any cloud

Easy

Get started in under 5 minutes

Questions? Connect with me this week!

- AuthZEN: the "OpenID Connect" of Authorization: Wed 11:15am
- Open Policy Containers: Project Pavilion 17B: Thu 1:30-5pm
- The Policy Engines Showdown: Fri 2pm

@omrig

omri@aserto.com

aserto.com/slack

aserto.com/contact