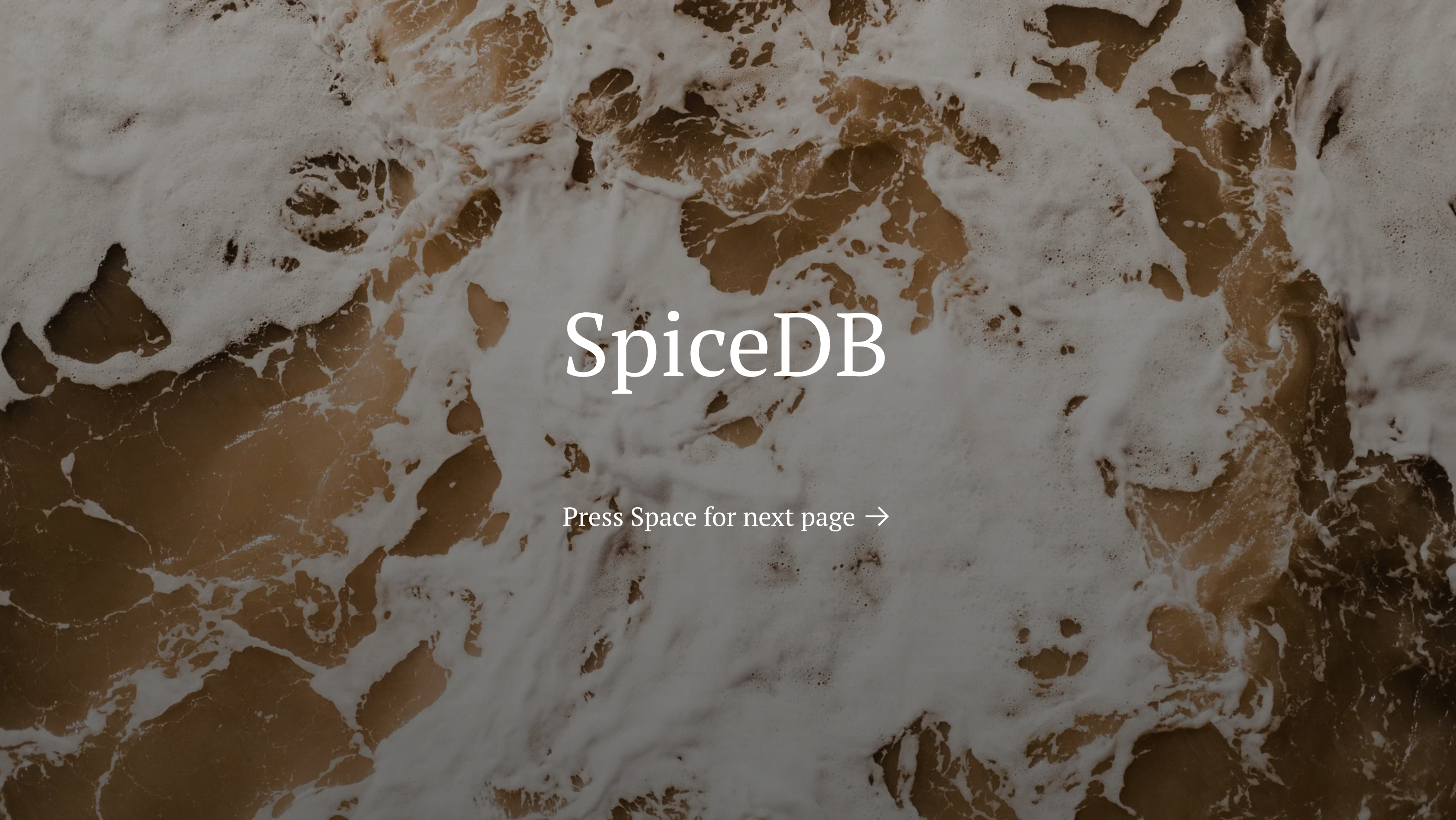


SpiceDB

Press Space for next page →

Agenda

- Identification, Authentication, **Authorization**
- Kind of **Authentication**
- Zanzibar
- SpiceDB as a solution

Identification, Authentication, Authorization

- **Identification** - Who are you? (@batazor)
- **Authentication** - How do you prove it? (password, token, etc)
- **Authorization** - What are you allowed to do? (RBAC, ABAC, etc)

Flow

Does `<subject>` have `<permission>` to `<object>`?

Example:

> Does `<@batazor>` have permission `<add star>` to `<@spicedb>` ?

Authentication + Identification

- Basic Auth
- JWT (go-jwt, etc…)
- OAuth2 (go-oauth2, passportjs, etc…)
- OpenID Connect (go-oidc, etc…)

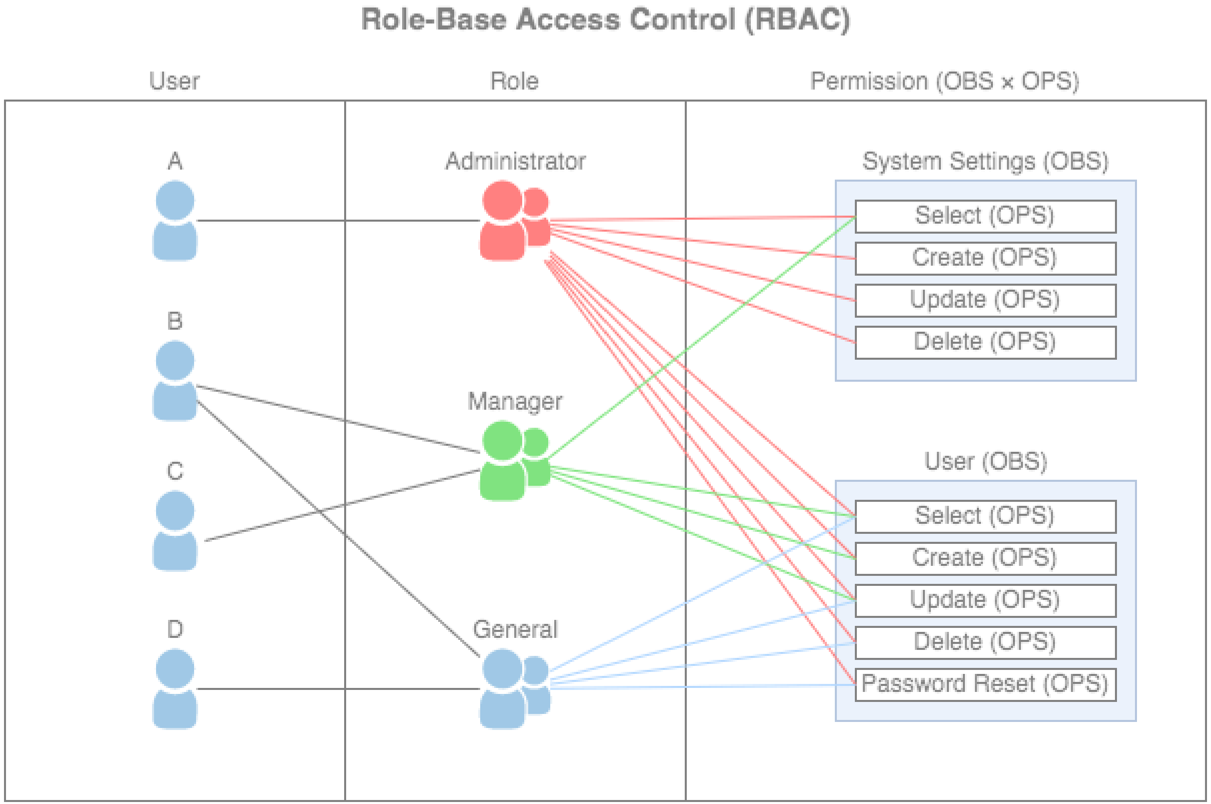
Authorization

- https://casbin.org/
  - RBAC, ABAC, etc…
  - Go, NodeJS, Python, etc…
- **usually it’s handmade**

Kind of Authentication

- RBAC
- ABAC
- ReBAC
- ACL
- other standards…

Kind of Authentication: RBAC (Role-Based Access Control)



Kind of Authentication: RBAC (Role-Based Access Control)

- Pros
  - Easy to implement
  - Easy to understand
- Cons
  - Not flexible
  - Not scalable
  - Not dynamic

Kind of Authentication: ABAC (Attribute-Based Access Control)

Kind of Authentication: ABAC (Attribute-Based Access Control)

- Pros
  - Flexible
  - Scalable
  - Dynamic
- Cons
  - Complex
  - Hard to implement
  - Hard to understand

Kind of Authentication: ReBAC (Relationship-Based Access Control)

Kind of Authentication: ReBAC (Relationship-Based Access Control)

- Pros
  - Flexible
  - Scalable
  - Dynamic
- Cons
  - Complex
  - Hard to implement
  - Medium to understand

Entities:

- Subject
- Action
- Object
- Relationship

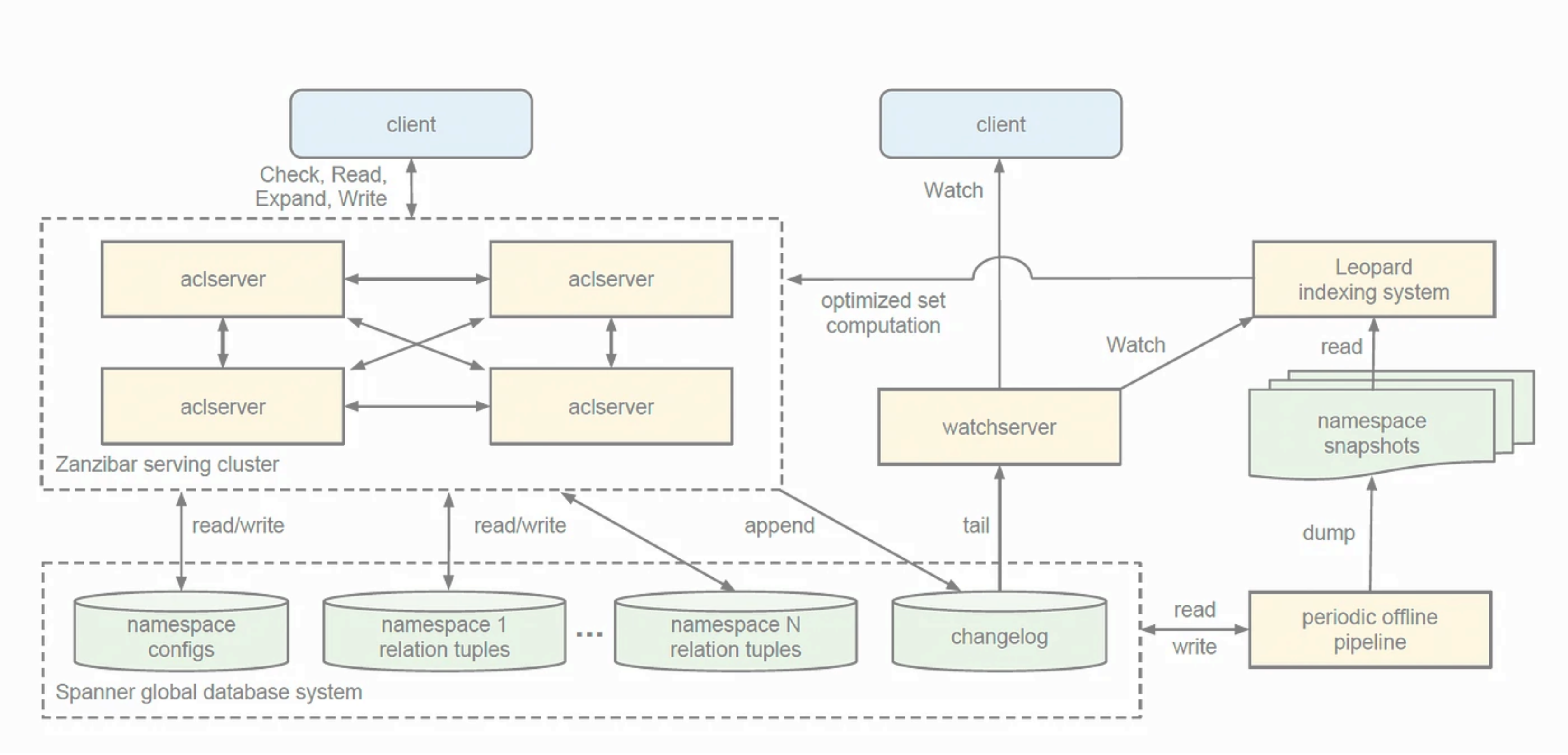
Zanzibar

- Zanzibar: Google’s Consistent, Global Authorization System
- Authzed: zanzibar
- Understanding Google Zanzibar: A Comprehensive Overview
- Jake Moshenko on Zanzibar: Google’s Consistent, Global Authorization System

Zanzibar: goals

- Correctness (step-by-step)
- Flexibility (policy)
- Low latency (3mc; 99% -> 20mc)
- High availability (2 minutes for 3 years)
- Scalability (20m+ rps)

Zanzibar: architecture



Zanzibar: features

- Relation Tuples (Subject, Action, Object)
- Namespace Configuration (Domain)
- Good API (Check, Write, Read, Watch)

Implementation of Zanzibar

- SpiceDB
- Ory Keto
- and more…

SpiceDB

- Open Source
- 4.7k stars
- UI (graph visualization + editor + test cases + save to file)
- SpiceDB Operator for Kubernetes
- Extentention for VSCode (Language Server)
- Wildcard policy

References

- ABAC on SpiceDB: Enabling Netflix's Complex Identity Types

SpiceDB: Schema

```
// user represents a user that can be granted role(s)
definition user {}

// document represents a document protected by Authzed.
definition document {}
```

SpiceDB: Schema

```
// user represents a user that can be granted role(s)
definition user {}

// document represents a document protected by Authzed.
definition document {
    // writer indicates that the user is a writer on the document.
    relation writer: user

    // reader indicates that the user is a reader on the document.
    relation reader: user
}
```

SpiceDB: Schema

```
// user represents a user that can be granted role(s)
definition user {}

// document represents a document protected by Authzed.
definition document {
    // writer indicates that the user is a writer on the document.
    relation writer: user

    // reader indicates that the user is a reader on the document.
    relation reader: user

    // edit indicates that the user has permission to edit the document.
    permission edit = writer

    // view indicates that the user has permission to view the document, if they
    // are a `reader` *or* have `edit` permission.
    permission view = reader + edit
}
```

SpiceDB: Tuple

Flow

Does `<subject>` have `<permission>` to `<object>`?

Format

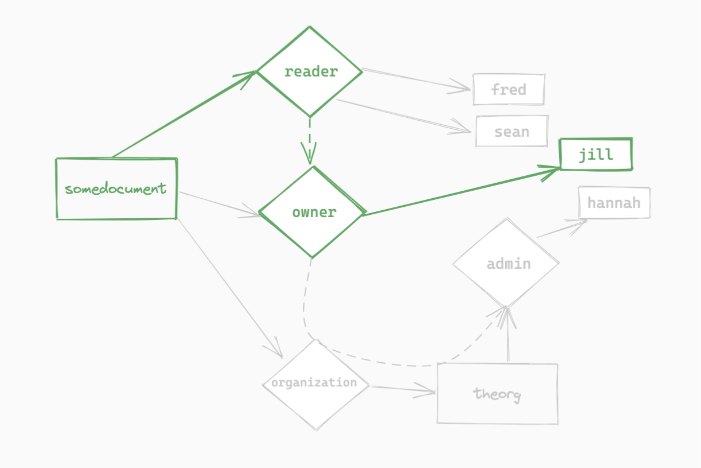
```
<resource>:<id>#<relation>@<subject>:<id>
```

Example

```
document:firstdoc#writer@user:tom
document:firstdoc#reader@user:fred
document:seconddoc#reader@user:tom
```

SpiceDB: ACL-filtering

```
SELECT FROM resources WHERE resource.created_by = $1
```



References

- ycombinator - pre/post filtering
- ACL Filtering in Authzed/SpiceDB
- The Challenge of ACL Filtering in Relational Databases

SpiceDB: In real life

1. Load schema to SpiceDB by API
2. Load tuples to SpiceDB by API
3. Check permission by API
4. PROFIT

Golang: example

Buf Schema registry

```go
relationship := &permission.WriteRelationshipsRequest{
    Updates: []*permission.RelationshipUpdate{{
        Relationship: &permission.Relationship{
            Resource: &permission.ObjectReference{
                ObjectType: "document",
                ObjectId:   document.GetId(),
            },
            Relation: "writer",
            Subject: &permission.SubjectReference{
                Object: &permission.ObjectReference{
                    ObjectType: "user",
                    ObjectId:   user.GetId(),
}}}}}}

authClient.ReadSchema(ctx, ...)
authClient.WriteSchema(ctx, ...)

authClient.WriteRelationships(ctx, relationship)
authClient.DeleteRelationships(ctx, relationship)
authClient.CheckPermission(ctx, relationship)
authClient.LookupResources(ctx, relationship)
authClient.WatchResources(ctx, relationship)
```

SpiceDB: Playgrounds

Open Playground

Learn More

- SpiceDB Docs
- ABAC on SpiceDB: Enabling Netflix’s Complex Identity Types
- Почему авторизация сложно и причем здесь Занзибар? -Максим Горозий, Тинькофф

Thank you!

- email
- telegram
- github