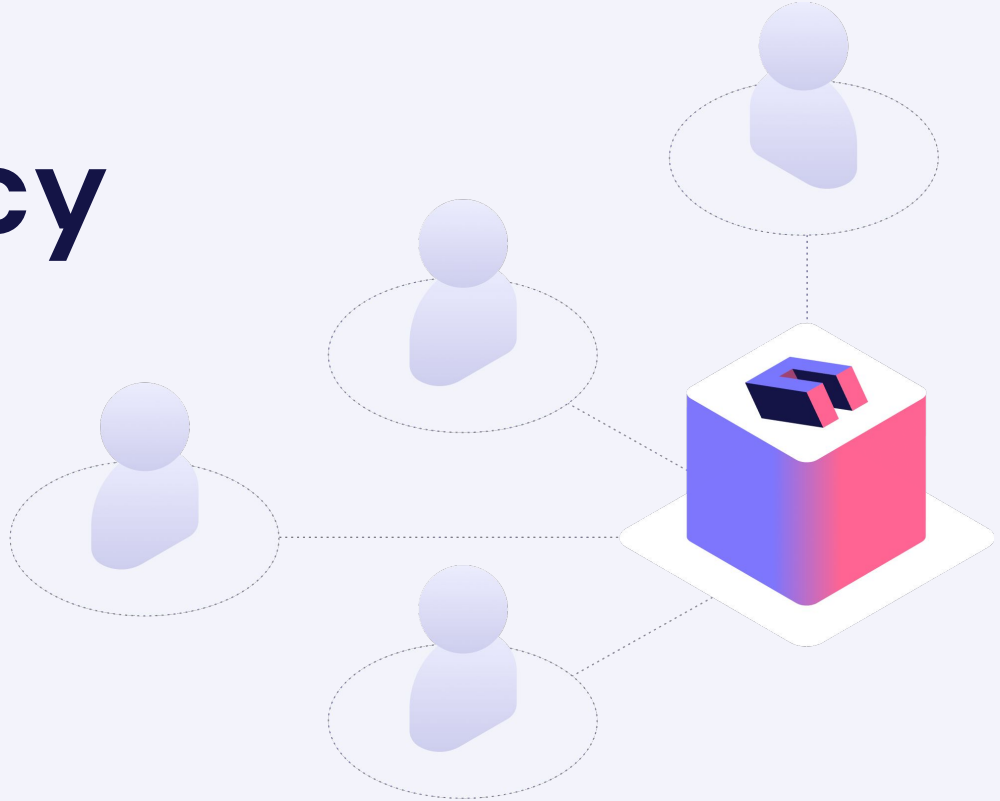
October 27, 2021

Multitenancy Workshop



cube

 Pavel Tiunov

 Ryan Pei

 Igor Lukanin

Community Code of Conduct

- We want to foster an open and welcoming environment where everyone feels they belong in the Cube.js community.
- The full text of our Code of Conduct is available at https://github.com/cube-js/cube.js/blob/master/CODE_OF_CONDUCT.md
- Any instances of inappropriate/unacceptable behavior can be reported to conduct@cube.dev.

Some quick notes 🗒️

- If you have any questions during the workshop, please feel free to type them in the “Q&A”.
- We will be using Cube Cloud during the demo/hands-on lab.
- Recording of today’s workshop will be posted on the Cube Dev YouTube channel.
- All attendees will receive a post-event survey and we’d appreciate your feedback to help us with future events.
    - In the survey, you can also provide your address so we can mail your Cube swags 😊

What we will discuss today

- Why do you need multitenancy?
- Getting started
- Use cases
- Resources
- Q&A

How does Cube define “multitenancy”?

Multitenancy for Cube

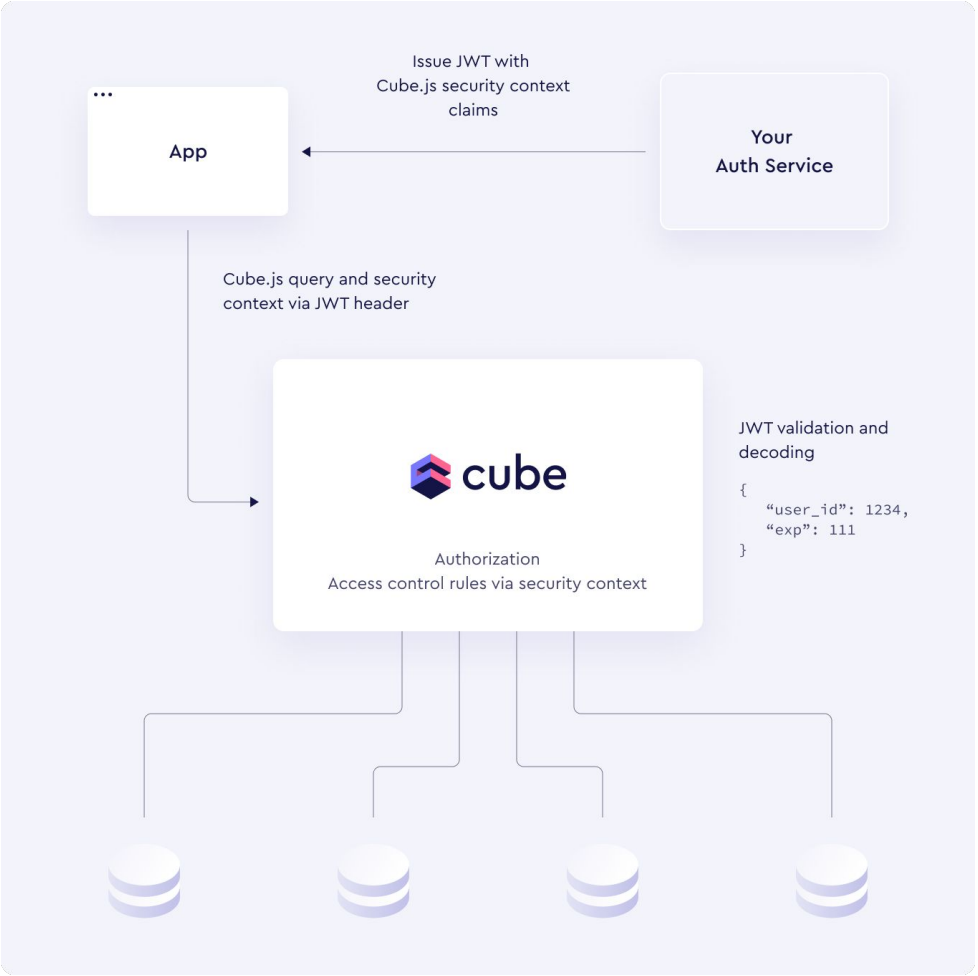
Cube is your app's API for data.

Multitenancy allows to control how different users access the data.

Each user or set of users with their own scope of data access is a **tenant**.

Front-end

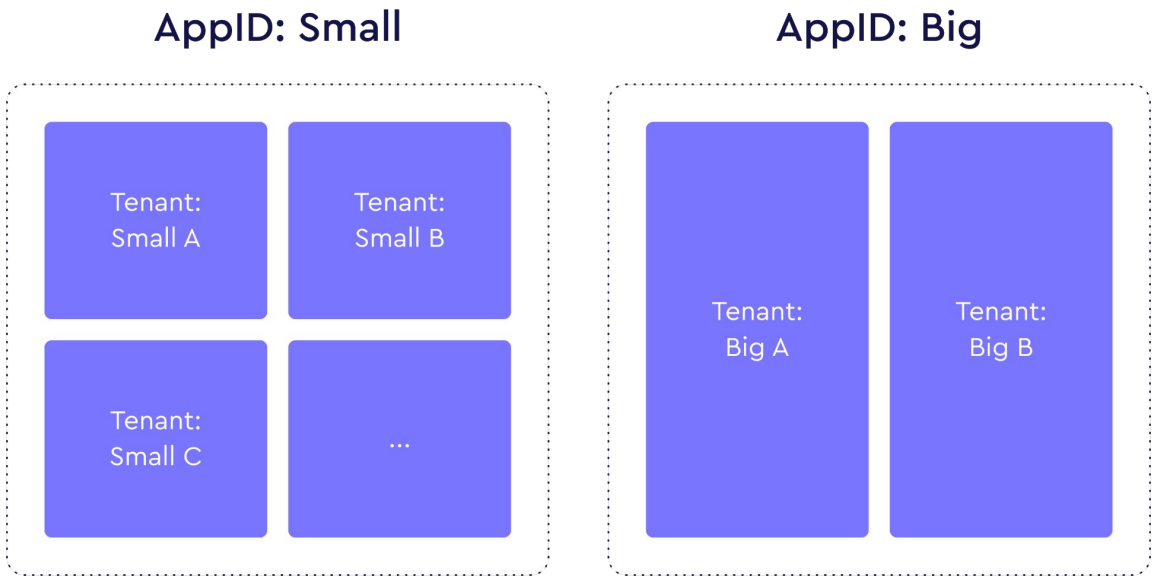
On the client side of Cube, tenants are recognized via auth tokens, e.g. JWT



Backend controls to scope tenants in Cube

contextToAppId

queryRewrite



contextToAppId

Cube treats `appId` as indivisible, nothing should be shared between the cluster using different appId's.

Note: `appId` is required for different databases (different connections) and only for different repositories of schema (repository factory) if those different schemas map to different tenants. It's also (usually) required if you want tenants to have their own tables for pre-aggregations.

queryRewrite

Restricts data access with query filters

How does Cube configure “multitenancy”?

The default configuration

With an empty cube.js file, a Cube app is configured like this:

- a **single-tenant** app — all requests are treated the same way
- **no queries** are filtered or rewritten
- a **single database** connection (of a **single type**)
- a **single file-based repository** for the **static data schema**

  with a **default data source** and **no tenant-dependent adjustments**
- a **single database schema** for pre-aggregations

Configuration options for multitenancy

Please check the docs page where you can find these options:

- `contextToAppId`
- `dbType`
- `driverFactory`
- `repositoryFactory`
- `preAggregationsSchema`
- `queryRewrite`

contextToAppId

Called on each request.

Used to tell Cube which **tenant** is making the **current request**.

```
contextToAppId: ({ securityContext })
  => `CUBE_APP_${securityContext.user_id}`
```

```
contextToAppId: ({ securityContext })
  => `CUBE_APP_${securityContext.admin ? ‘admin’ : ‘user’}`
```

dbType

Called once per tenant.

Used to tell Cube which **database type** is used to store data for a **tenant**.

```
dbType: ({ dataSource })
  => dataSource === `production` ? `bigquery` : `postgres`

dbType: ({ securityContext })
  => dbTypes[securityContext.user_id] || `snowflake`
```

driverFactory

Called once per data source.

Used to tell Cube which **database driver** is used for a **data source**.

```
driverFactory: ({ dataSource })
  => dataSource === `production`
    ? new PostgresDriver(...)
    : new PostgresDriver(...)
```

repositoryFactory

Called once per tenant.

Used to tell Cube which **data schema** files to use for a **tenant**.

```
repositoryFactory: ({ securityContext })
  => new FileRepository(`schema/${securityContext.user_id}`)

repositoryFactory: ({ securityContext })
  => ({ dataSchemaFiles: async () => ... })
```

preAggregationsSchema

Called once per tenant.

Used to tell Cube which **database schema** to use to store pre-aggregations for a **tenant**.

```
preAggregationsSchema: ({ securityContext })
  => `pre_aggregations_${securityContext.user_id}`
```

queryRewrite

Called on each request.

Used to validate or update a request before it's executed. Often, used to control **data access** for each **tenant**.

```
queryRewrite: (query, { securityContext }) => {
  if (securityContext.role !== `admin`) throw new Error('...');

  query.filters.push(...);
}
```

Options that play great with multitenancy

- `scheduledRefreshContexts`
- `COMPILE_CONTEXT`
- `dataSource`

scheduledRefreshContexts

Called by a refresh worker.

Allows to build pre-aggregations for all **tenants**.

```
scheduledRefreshContexts: () => [
  { securityContext: { tenant: ‘avocado’ } },
  { securityContext: { tenant: ‘mango’ } },
]

scheduledRefreshContexts: async ()
  => await fetchTenantContexts()
```

COMPILE_CONTEXT

Available in the data schema during its compilation. Usually, provided via `scheduledRefreshContexts`.

Allows to adjust the data schema to each **tenant**.

```
const {
  securityContext: { company_id }
} = COMPILE_CONTEXT;
```

dataSource

Provided in the data schema. Play well with `COMPILE_CONTEXT`.

Allows to set a specific **data source** for each **tenant**.

```
cube(`Facts`, {
  sql: `...`,

  dataSource: dataSources[COMPILE_CONTEXT.securityContext.id]
})
```

Let’s get started!

You can follow along with the code here:

https://descriptive-reply-0b7.notion.site/Multitenancy-workshop-code-58f4e4bedc3b44b791e104f7be6375e5

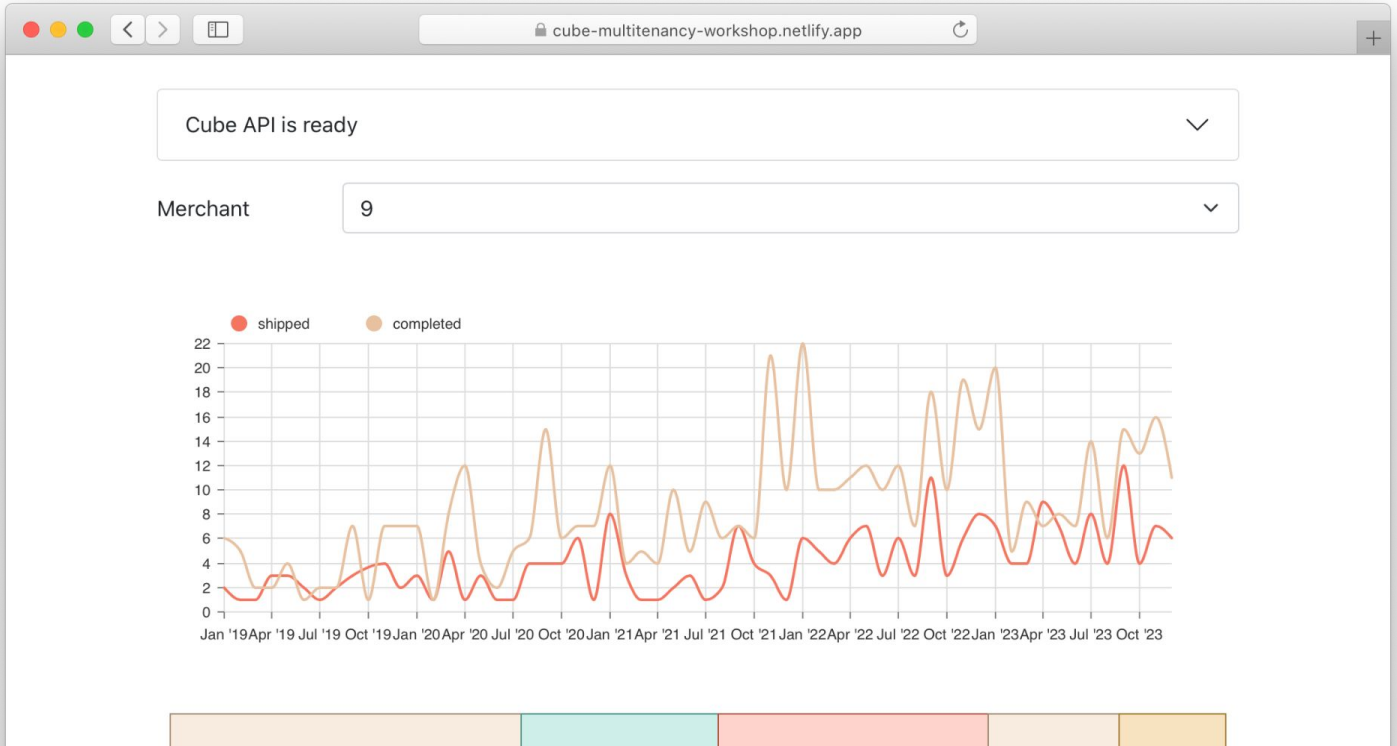
(This link is at the bottom of the workshop prep instructions we sent out earlier; we'll also post this in the Zoom chat)

All the code referenced is here:

https://github.com/cube-js/cube.js/tree/master/examples/multitenancy-workshop

Imagine we're an ecommerce platform...

https://cube-multitenancy-workshop.netlify.app



Imagine we're an ecommerce platform...

Our merchants want a report/dashboard for their orders.

For now, we rely on a merchant ID on an `Orders` table to tell us whose orders are whose.

Now a new merchant comes along...

We just entered a partnership with a new major merchant.

Now we have a tenant with its own DB.

When a measure/dimension is tenant-specific

Our new partner merchant has a new order status type, “Processed”, for which they will need to see counts, too.

And we don’t want other merchants to see “Processed” orders (they have none).

Front-end example

You can checkout https://cube-multitenancy-workshop.netlify.app/

Or, run the code yourself:

Resources

To learn more about multitenancy in Cube

Docs reference for most of what we discussed today:

- Multitenancy on Cube

Please check out relevant recipes:

- Using Multiple Data Sources
- Using Different Schemas for Tenants

Don’t hesitate to ask any questions on Slack.


cube