Fullstack Development

Authentication / Authorization

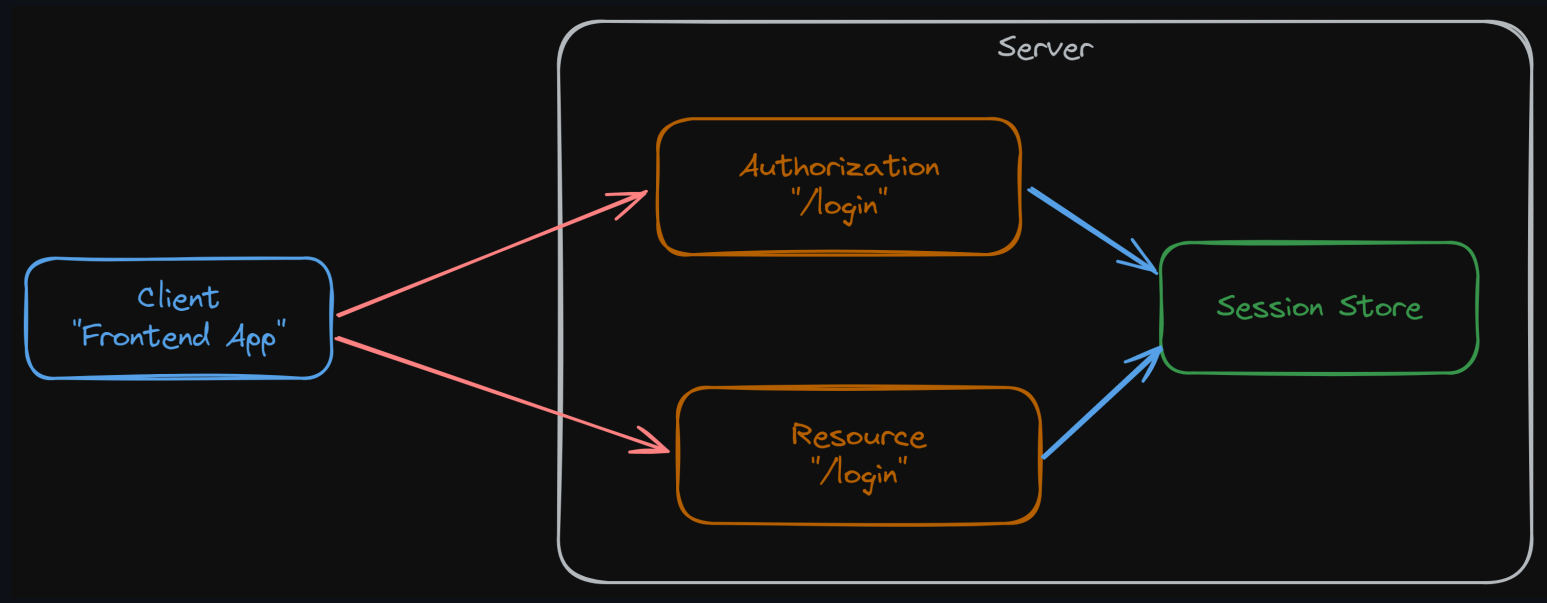
Part 3: Persisting auth's state

Session based

- Server is responsible for creating and maintaining the user's authentication state (i.e. in a database).
- After user sign-in, the server sets a cookie that contains the session ID and sends it to the browser.
  - The browser will include it in all further requests.
  - The server will use the cookie to identify the current user session from the database.

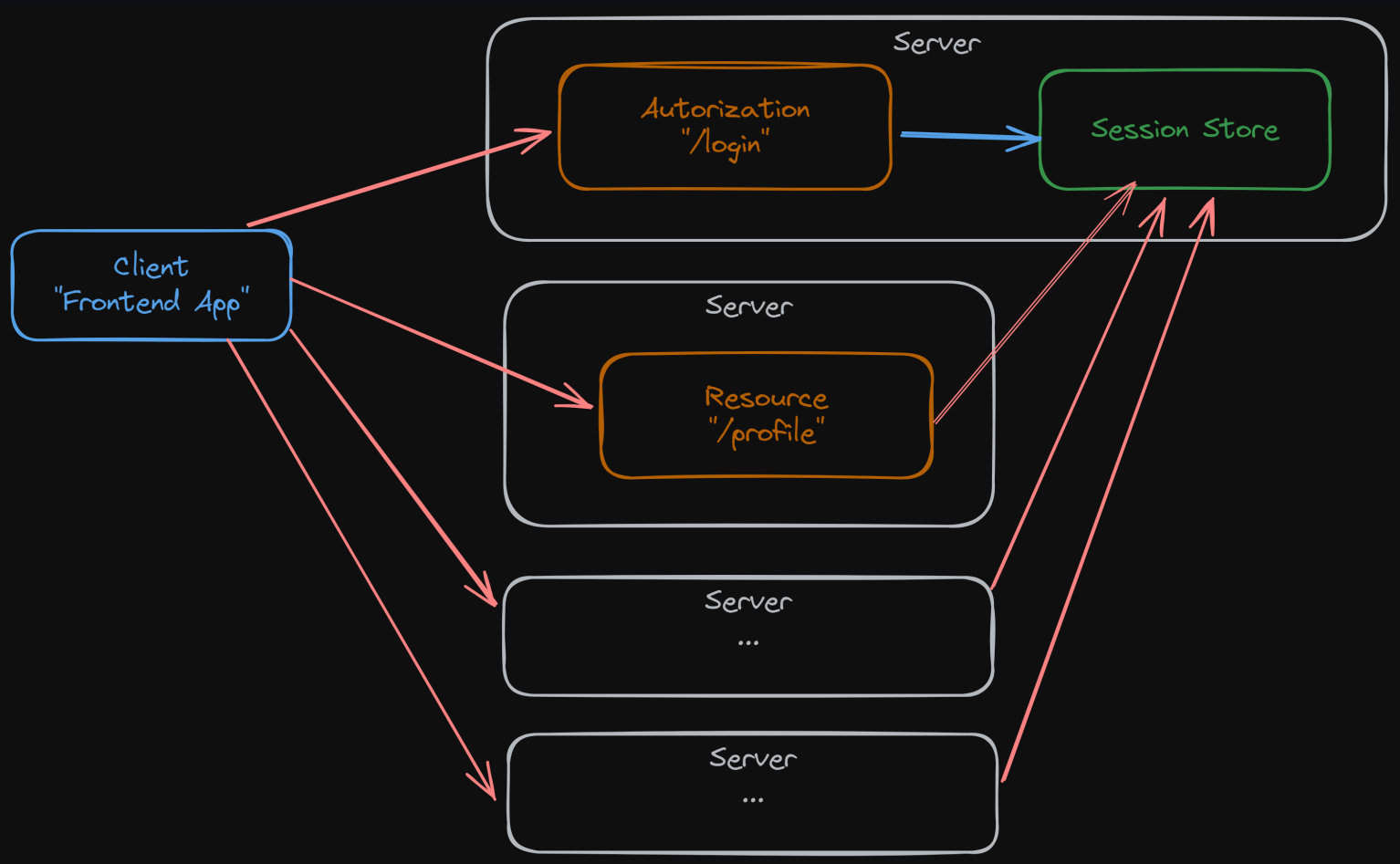
Session based

- Users' auth states are in DB.
- Need to query DB at every request.

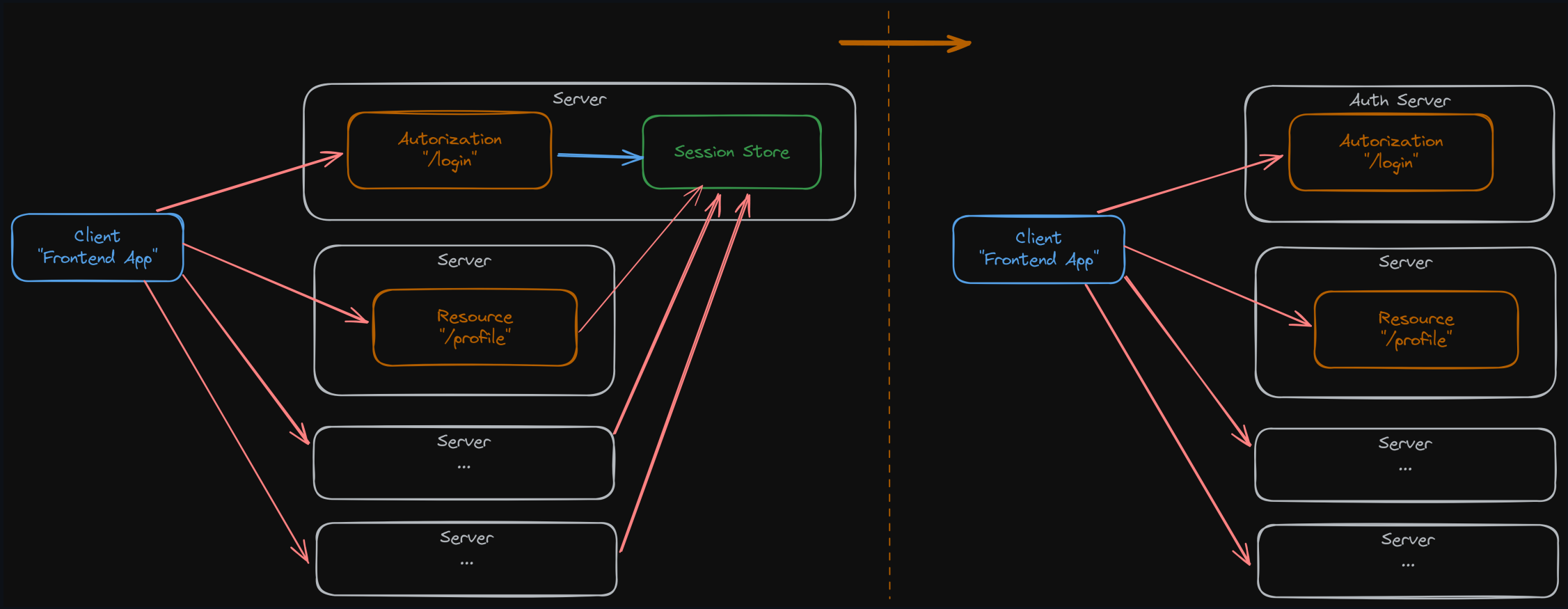


Session based

- This could be a problem in distributed system with centralized `auth` server.
- Session store could be overloaded.



Can do something like this?



Note that the right system is not exactly what you want to do.

Token-based

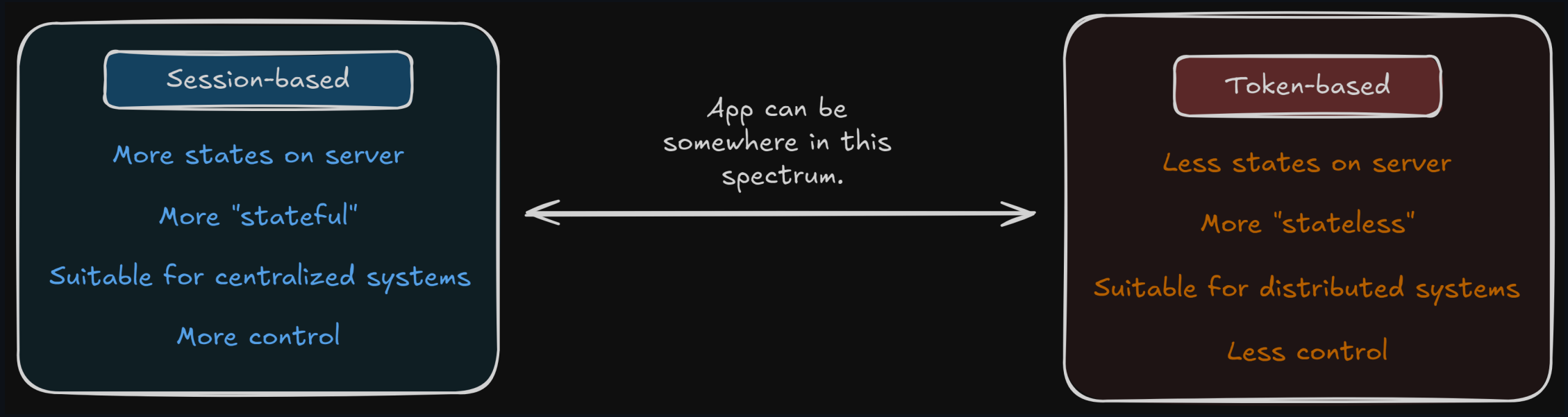
- `token` is a cryptographically signed piece of data that contains information about the authenticated user and their access permissions.
- The server will only have to verify the validity of the token rather than having it stored in a database.
  - Reduces the amount of state that needs to be stored on the server.
- While other token formats exist, JSON Web Tokens (JWTs) have become the prevailing standard for token-based approach.

JWT Test

- `git clone -b jwt https://github.com/fullstack-67/auth-mpa-v2.git auth-jwt`
- `pnpm i`
- `npx tsx ./src/test.ts`

Clarification

- It is better to think about where you put users' `auth` state.
  - `Session-based`: more states in server (*"stateful"*)
  - `Token-based`: more states in client (*stateless*)
- Using JWTs does not automatically means you are using token-based approach.
  - You can put JWTs in session cookie.
- The system can contain both approaches.



- When going token-based approach, you are losing **control** over user's state and you are making your system **less secured**.

Please do not do this.

- It is tempting to go **100% stateless** using token-based approach (JWT) to avoid dealing to storing information on server.
  - **You don't know who is using your system!**
- Also, be aware of these concerns (Ref1, Ref2).
  - Cannot really log out users.
  - Cannot really block users.
  - Stale data
  - Limited storage
  - JWT could be decrypted at some point.

Considering token-based approch?

- Do you have distributed system with centralized auth server?
  - If no, go session-based.
- You are concerned about overloading your database.
  - Have you considered `redis`?

Considering token-based approach?

- Have you consider the fact that modern token secuity is quite complex (*and will require database anyway*)?
  - Refresh tokens (revokable)
  - Allowed/Revoked lists
  - Token rotation
  - Token behavior detection

Bottom line

> If you don't have database table storing `auth` states, your system lacks **visibility** and **security response** against cyber attacks.

> Part 3: Persisting auth's state

Using `session` for session management.

Setup

```
git clone -b session https://github.com/fullstack-67/auth-mpa-v2.git auth-session
```

```
pnpm i
```

```
npm run db:reset
```

```
npm run dev
```

Highlighted package

`package.json`

```
{
  "express-session": "^1.18.0"
}
```