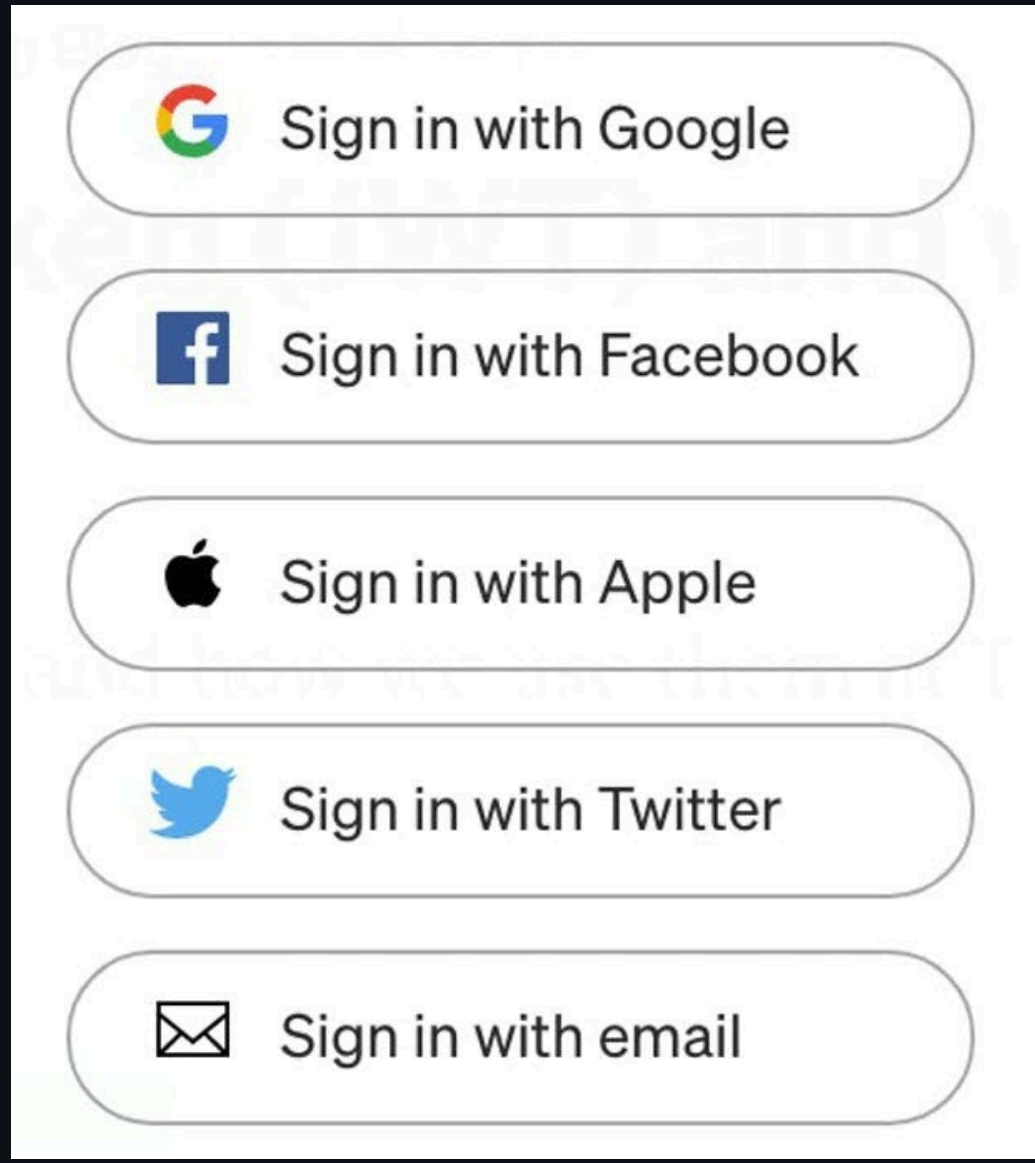
Fullstack Development

Authentication / Authorization

Part 2: Social signing up/in

Something like this

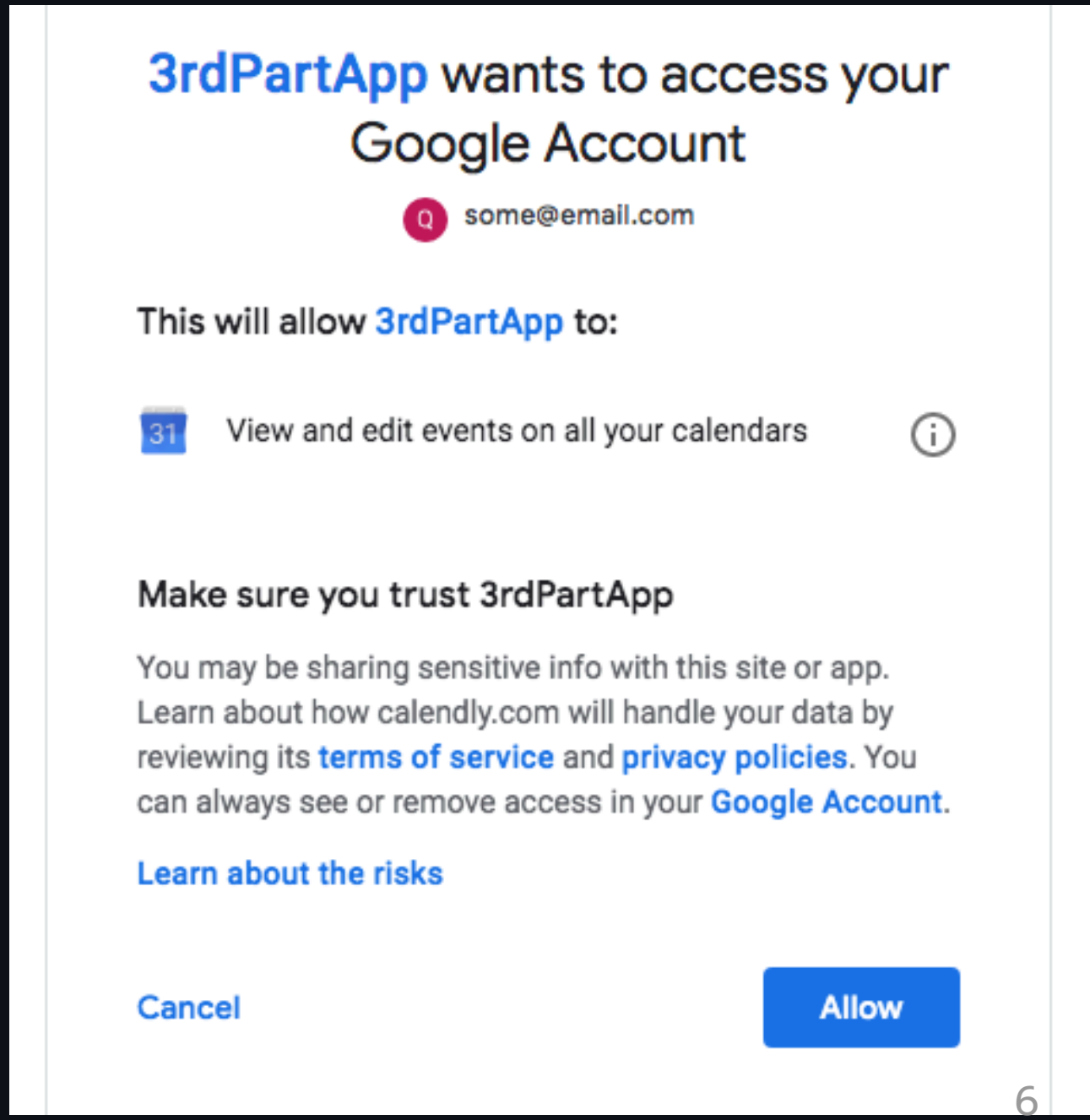
We need OAuth 2.0.



Part 2: Social signing up/in

Section 2A: OAuth 2.0

OAuth 2.0


3rdPartApp wants to access your Google Account
Q
some@email.com
This will allow 3rdPartApp to:
31
View and edit events on all your calendars
Make sure you trust 3rdPartApp
You may be sharing sensitive info with this site or app. Learn about how calendly.com will handle your data by reviewing its terms of service and privacy policies. You can always see or remove access in your Google Account.
Learn about the risks
Cancel
Allow

OAuth 2.0

- "Open Authorization"
- Standard designed to allow application to access resources hosted by other web apps on behalf of a user.
  - Standard for `author`
  - Not for `authen`
- Replaced OAuth 1.0 in 2012.

OAuth 2.0

- Specifies many "flows"
  - **Authorization Code Flow**
  - Client Credentials Flow
  - Refresh Token Flow
  - JWT Bearer Flow
  - Device Code Flow
- We will use "Authorization Code Flow" for social login.

Recommended resources

- https://engineering.backmarket.com/oauth2-explained-with-cute-shapes-7eae51f20d38
- https://developer.okta.com/blog/2019/10/21/illustrated-guide-to-oauth-and-oidc?utm_source=pocket_shared
- https://youtu.be/8aCyojTIW6U?si=YPxkcLPcAoK5jixI
- https://youtu.be/t18YB3xDfXI?si=pD1JnFP0GrnBXW2v

Wait

> Are we using OAuth (standard for `author`) and **authorization** code flow for `authen`?

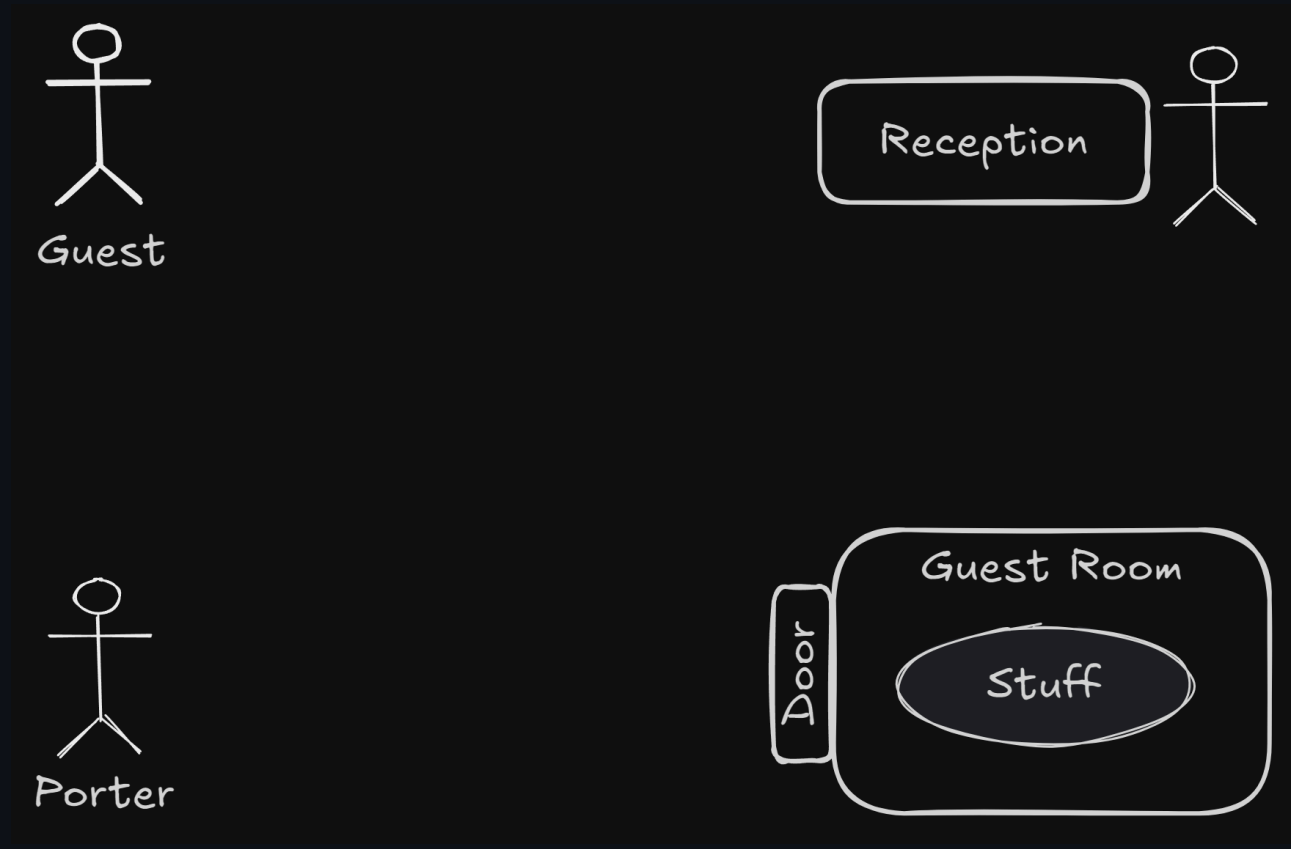
> Yes, we kind of "misusing" it.

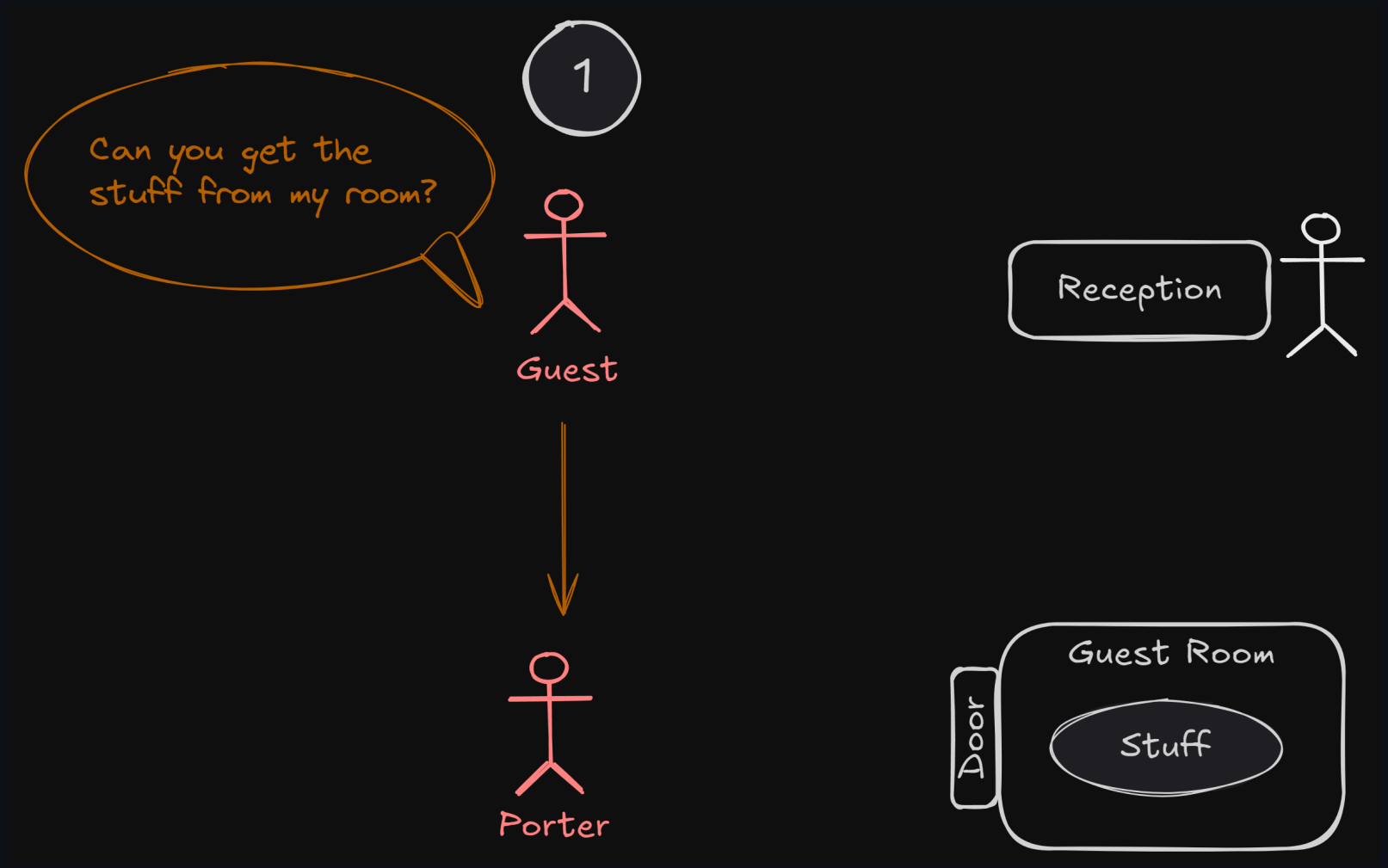
Authorization code flow

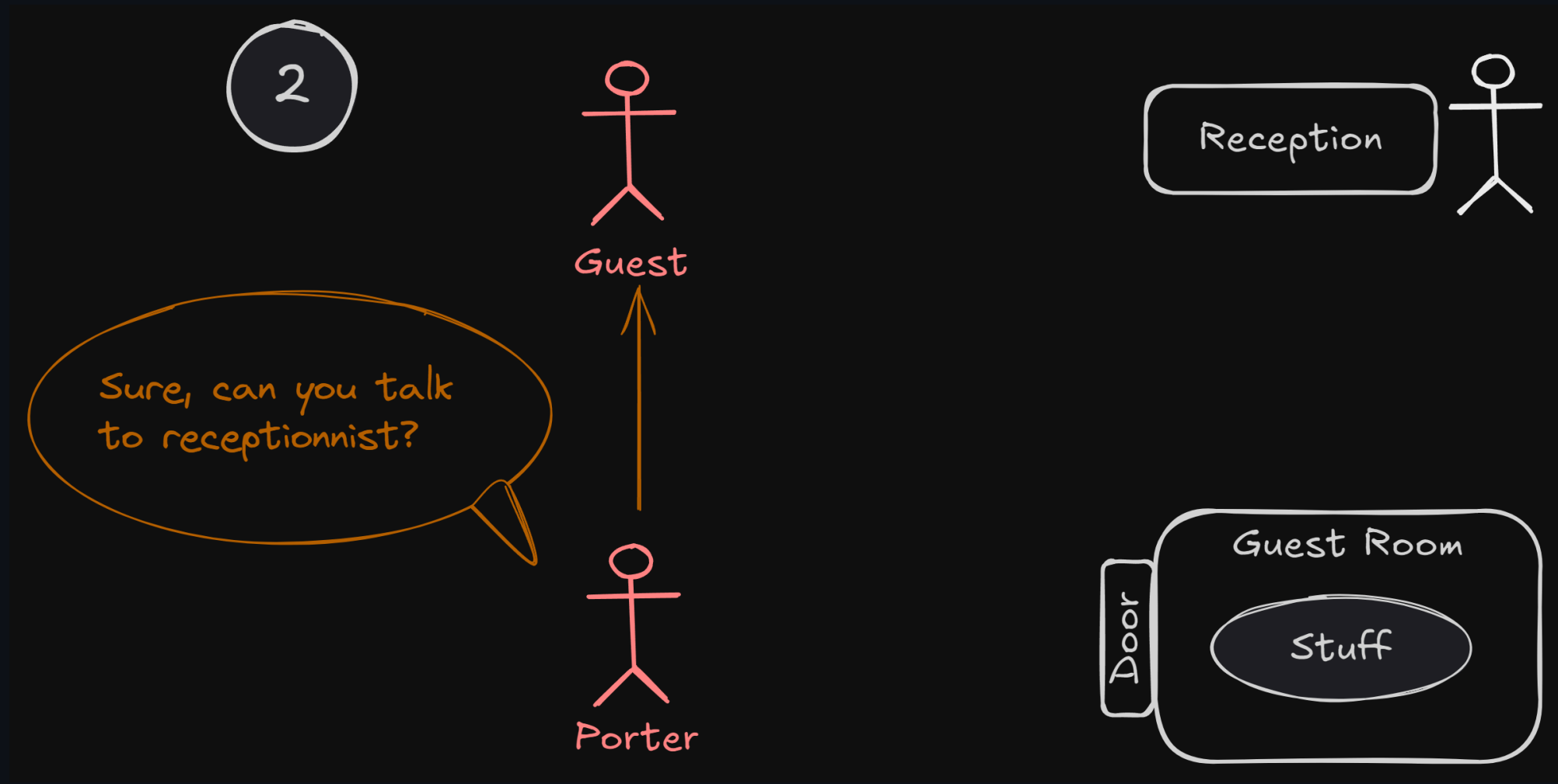
> In real life

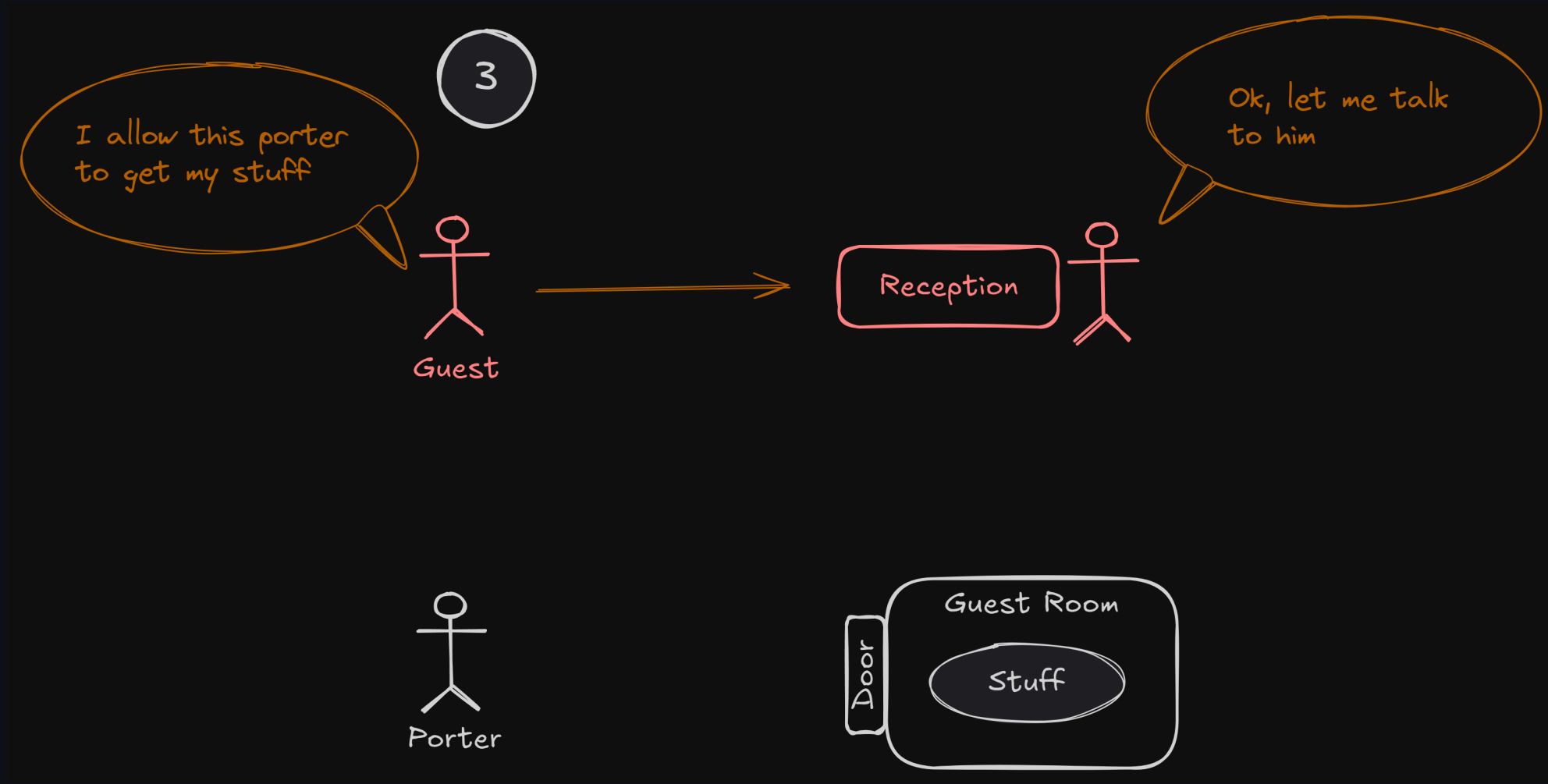
Setup

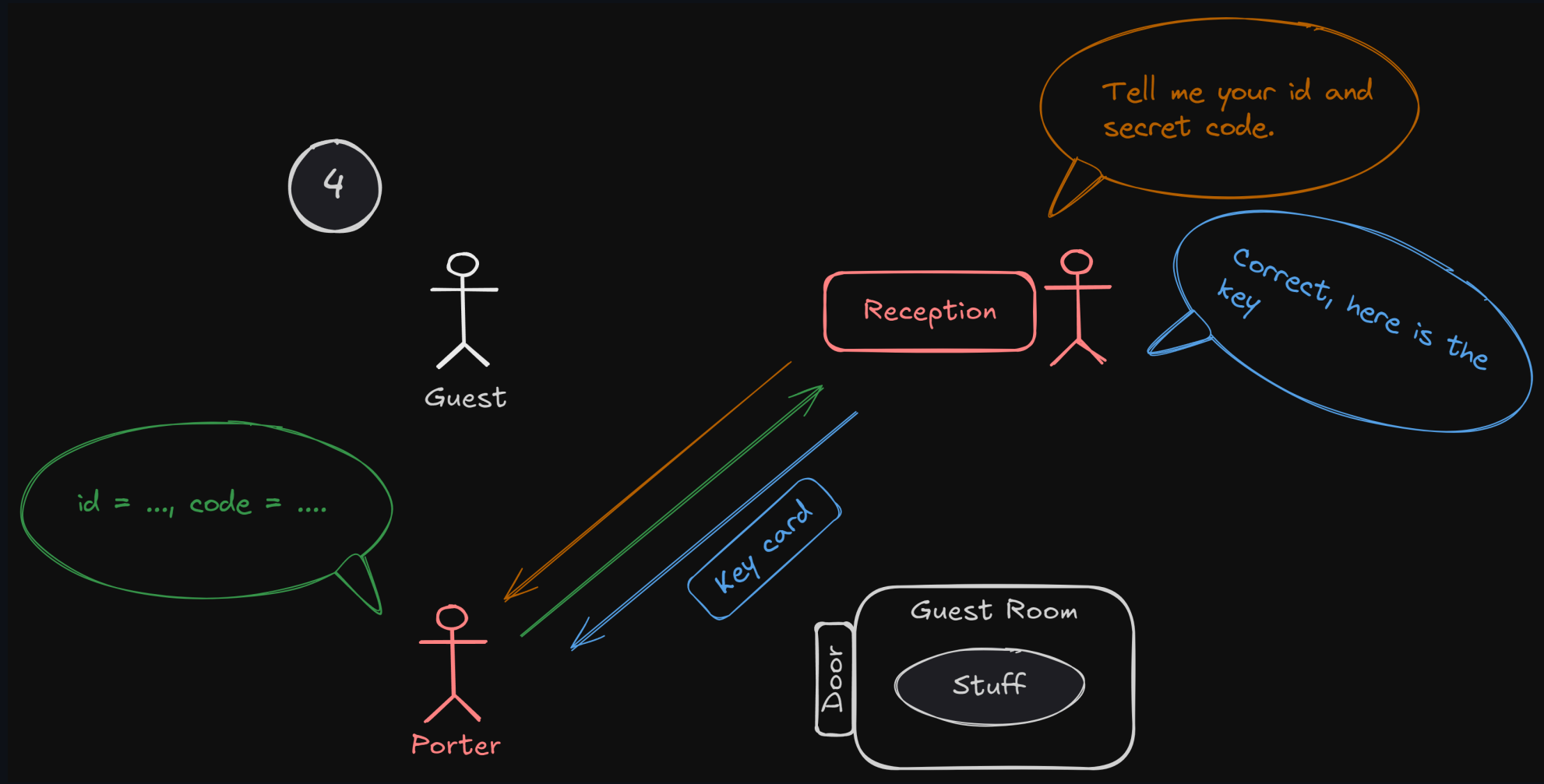
- You are a guest at a hotel.
- You already checked out.
- You forgot your stuff in the room.
- You want a porter to get your stuff for you.

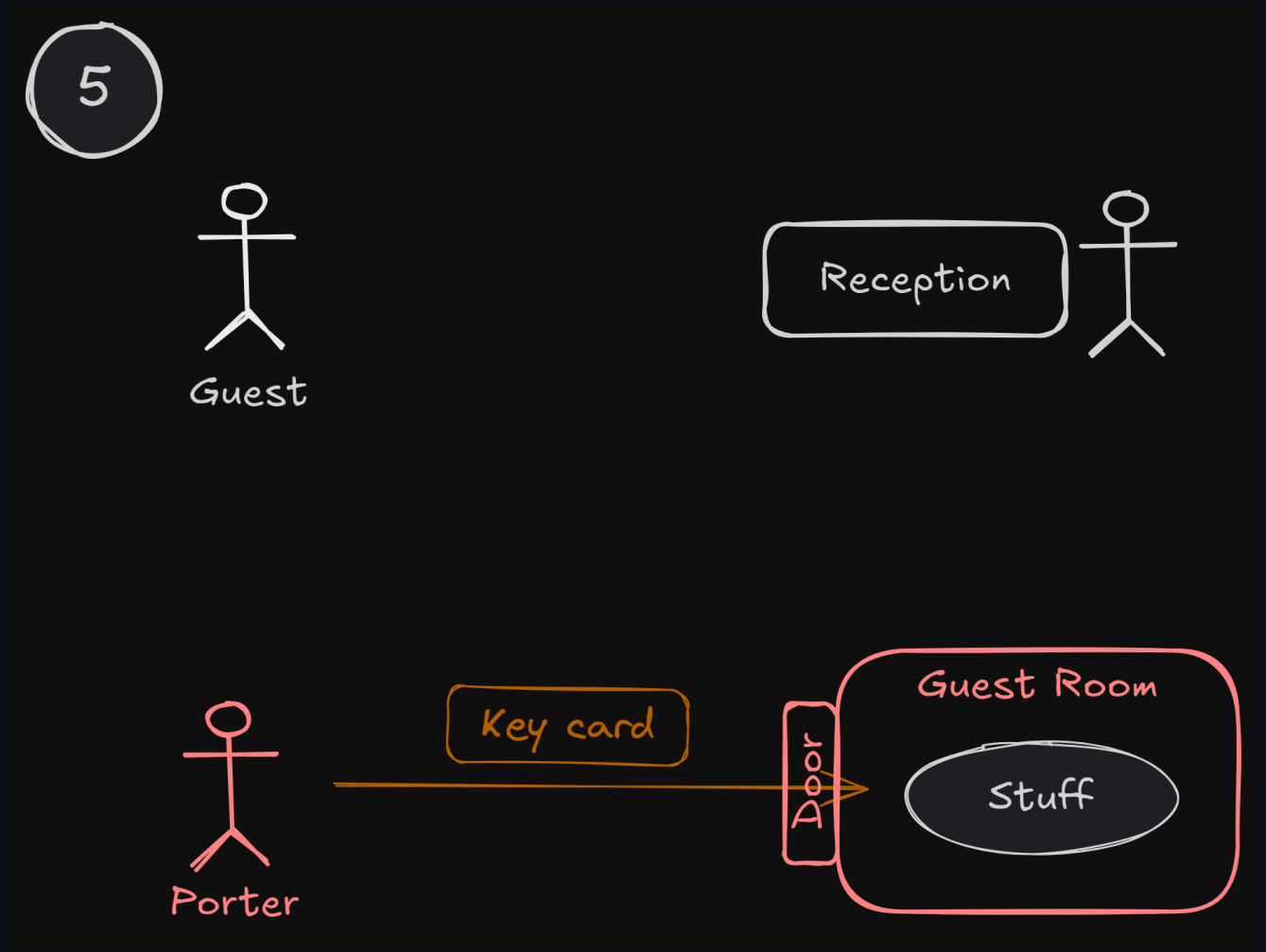



1
Can you get the stuff from my room?
Guest
Reception
Porter
Guest Room
Door
Stuff


2
Guest
Sure, can you talk to receptionnist?
Porter
Reception
Guest Room
Door
Stuff


3
I allow this porter to get my stuff
Guest
Reception
Ok, let me talk to him
Porter
Guest Room
Door
Stuff


4
Guest
Tell me your id and secret code.
Reception
Correct, here is the key
id = ..., code = ...
Key card
Porter
Guest Room
Door
Stuff

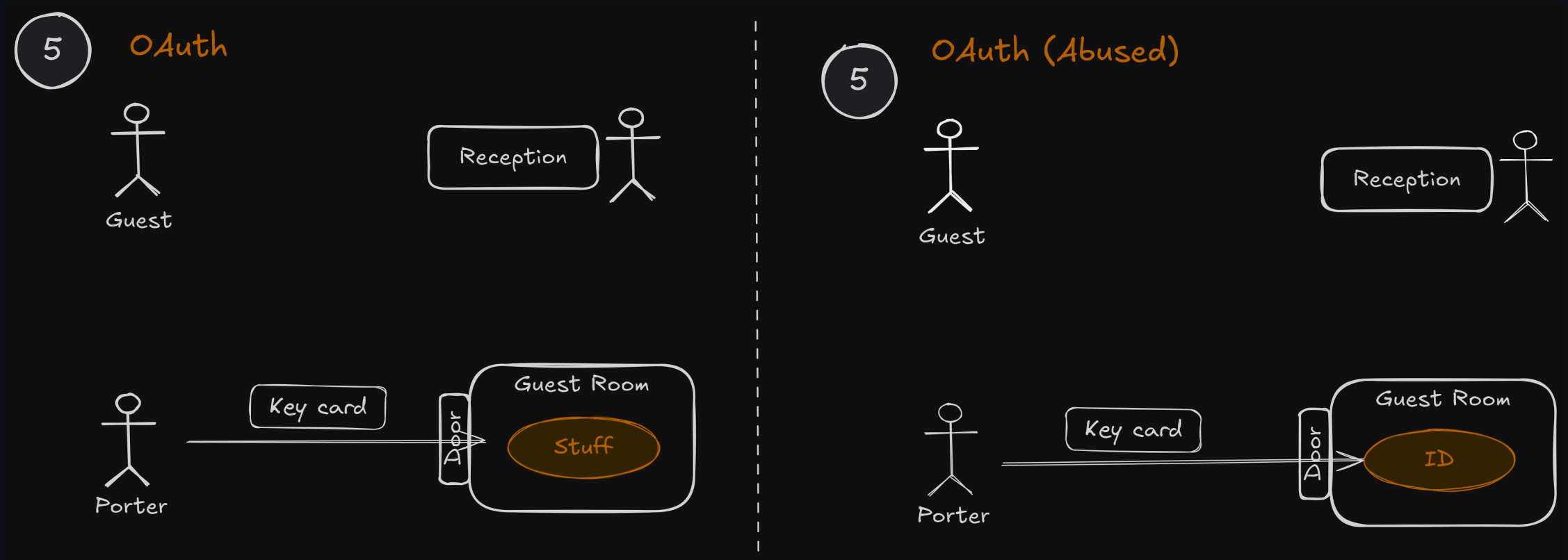

5
Guest
Reception
Porter
Key card
Door
Guest Room
Stuff

Authorization code flow

- You (`guest`) authorize `porter` to access your resource.
- `porter` does not need to know who you are.
- The keycard reader at the door also doesn't need to have your information.

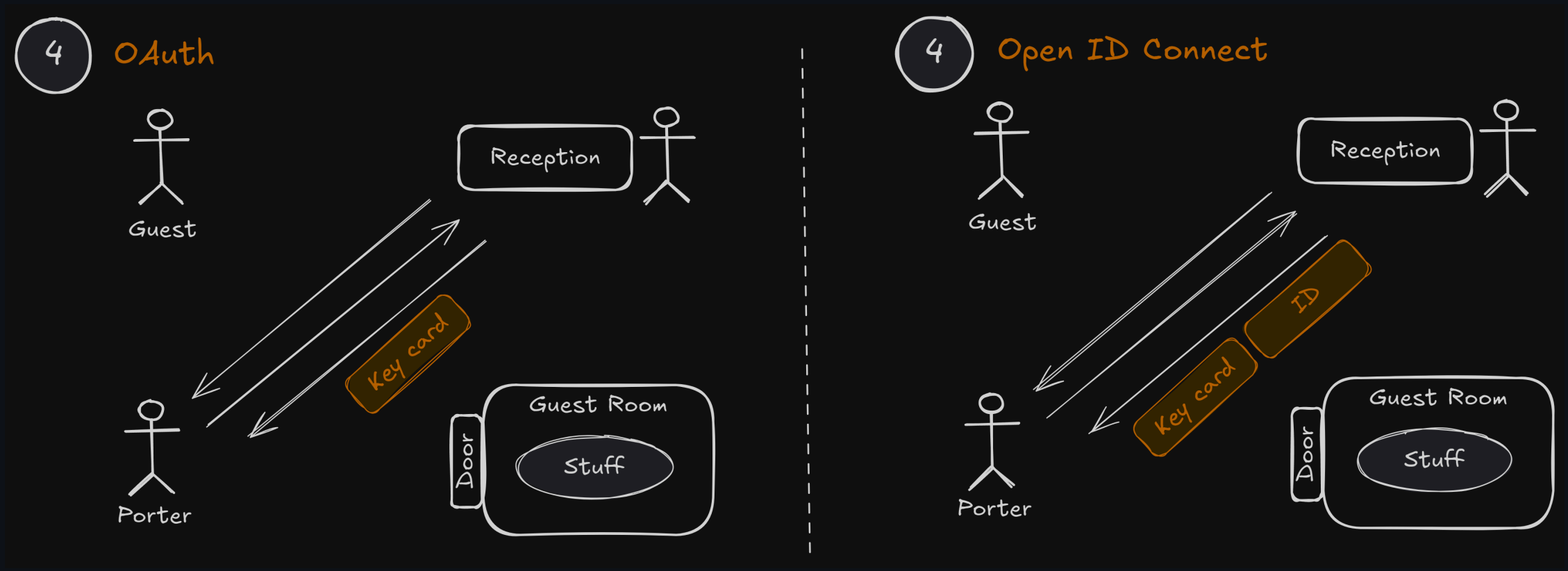
Authentication?

- But what if the porter wants to know who you are.
- There are two ways.



This is what we are using.

Is there a better way?

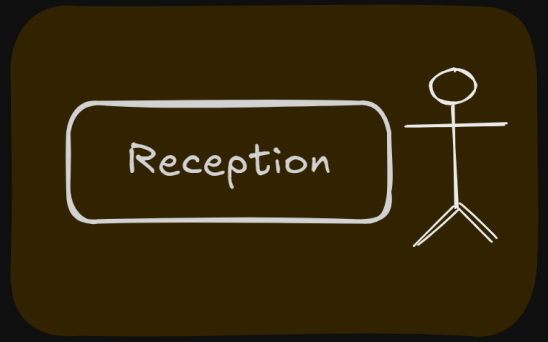

4
OAuth
Guest
Reception
Key card
Porter
Guest Room
Door
Stuff
4
Open ID Connect
Guest
Reception
ID
Key card
Porter
Guest Room
Door
Stuff

OpenID Connect (OIDC)

- Thin layer that sits on top of OAuth 2.0
  - Adds login and profile information about the person who is logged in.
- When a "Authorization Server" supports OIDC, it is sometimes called an "Identity Provider".
- Not all servers support OIDC.

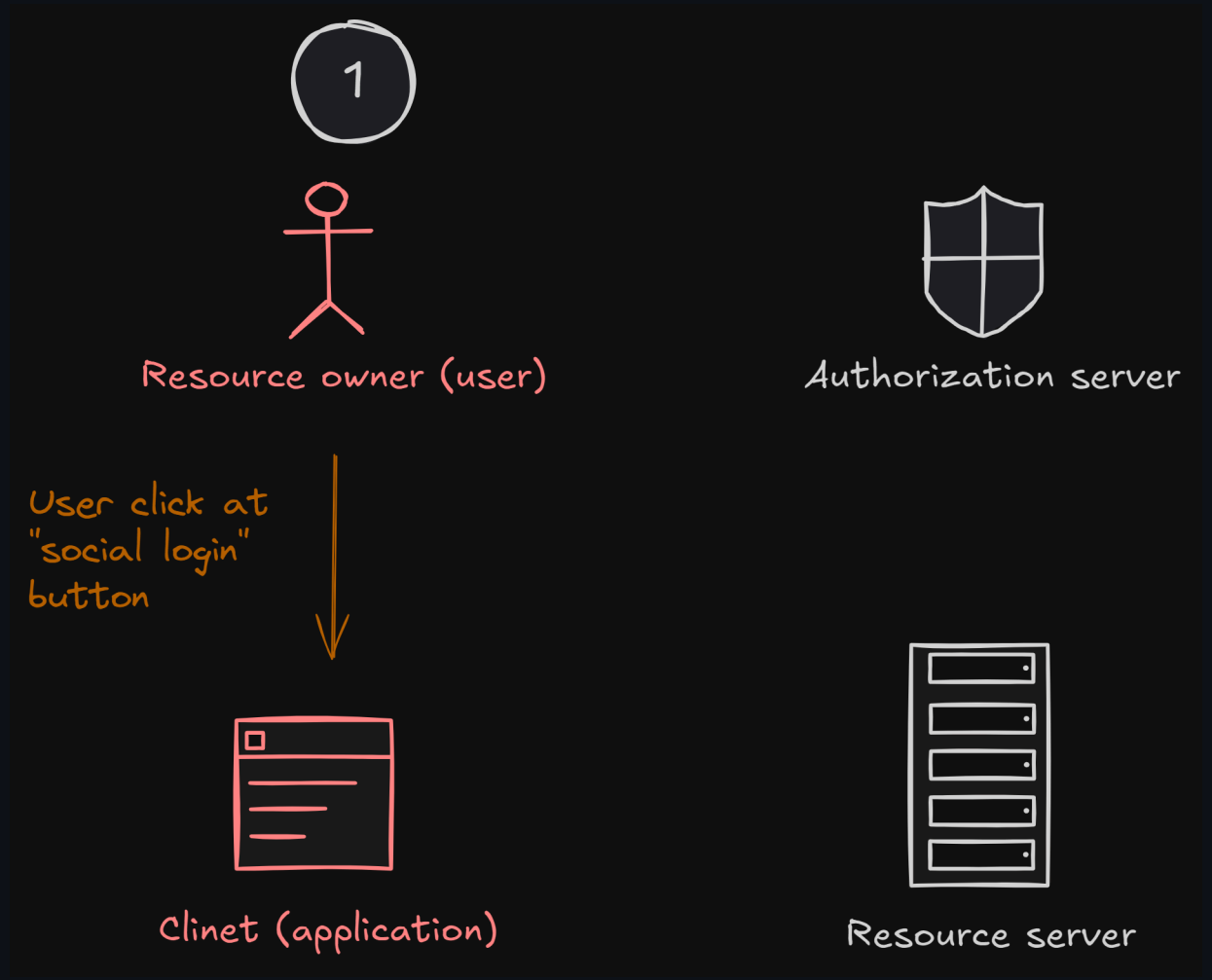
Resource owner (user)

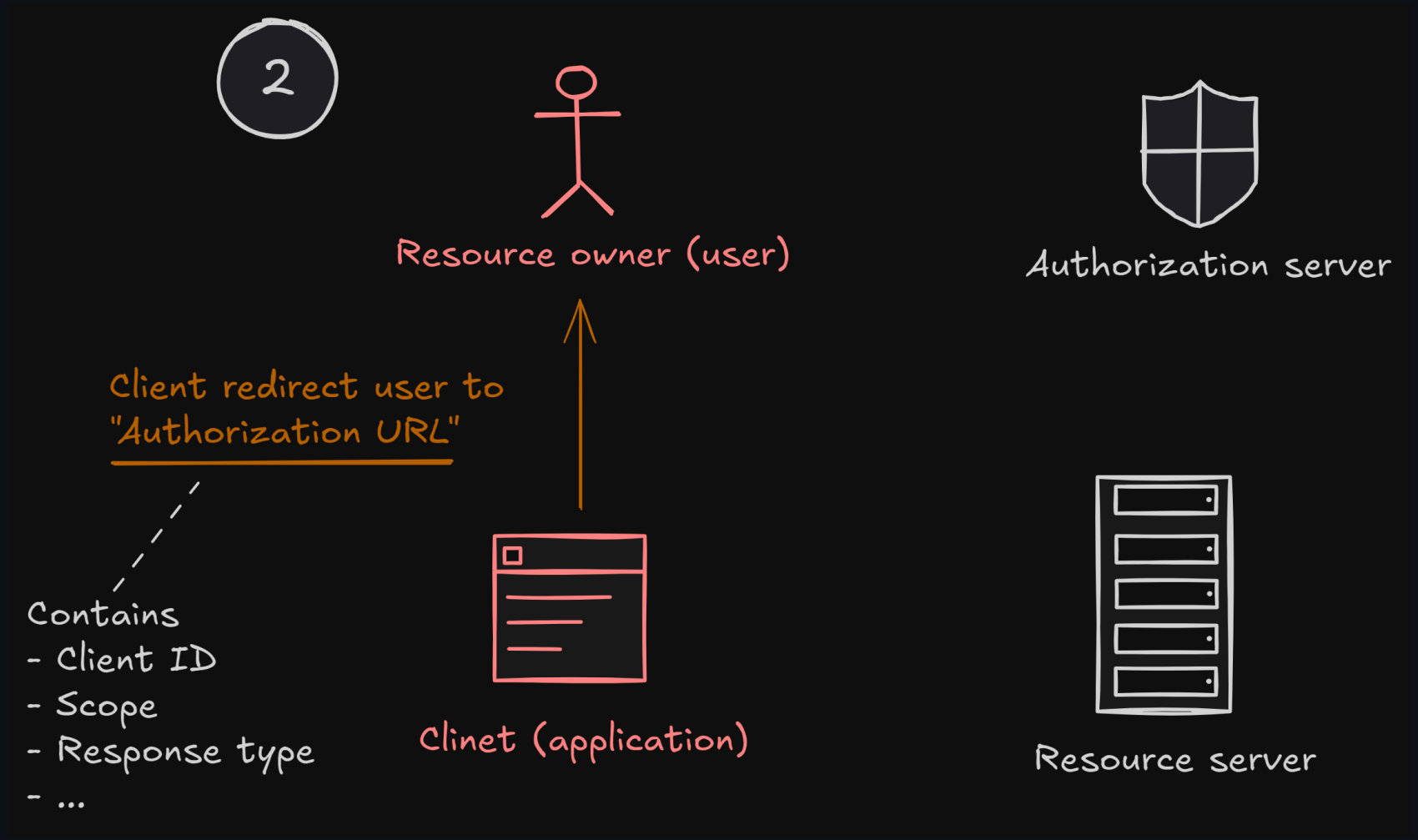
Authorization server

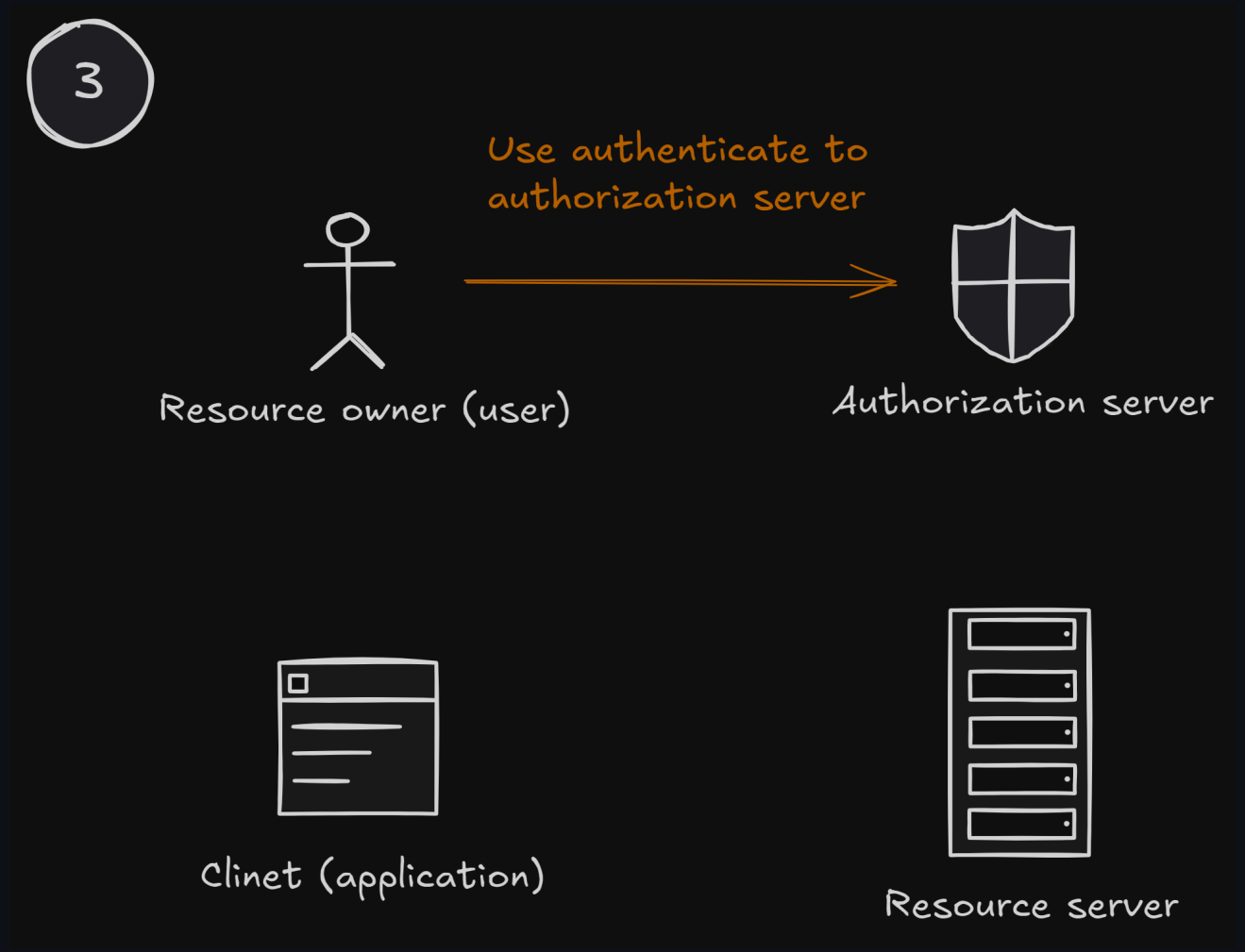


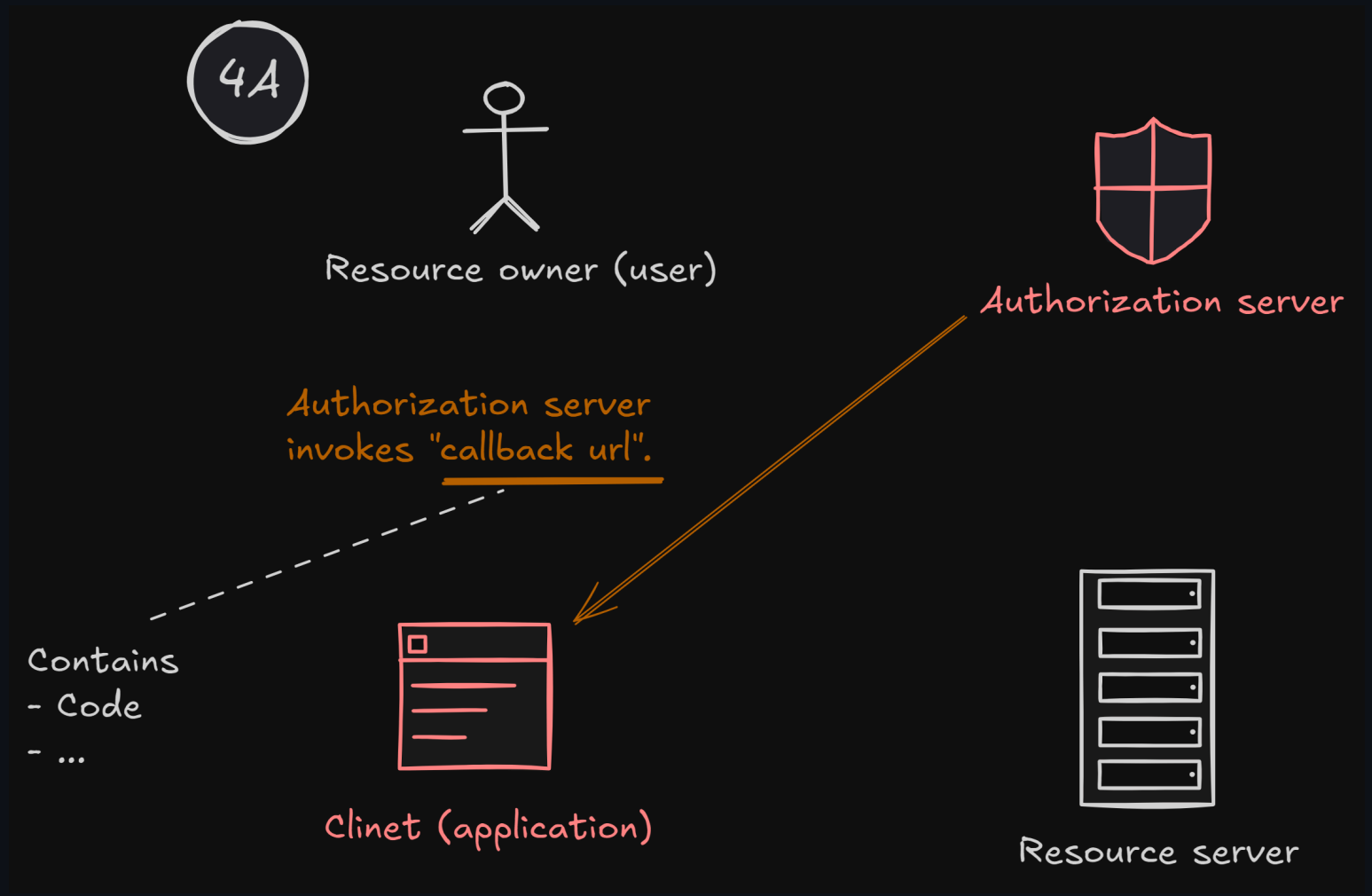
Clinet (application)

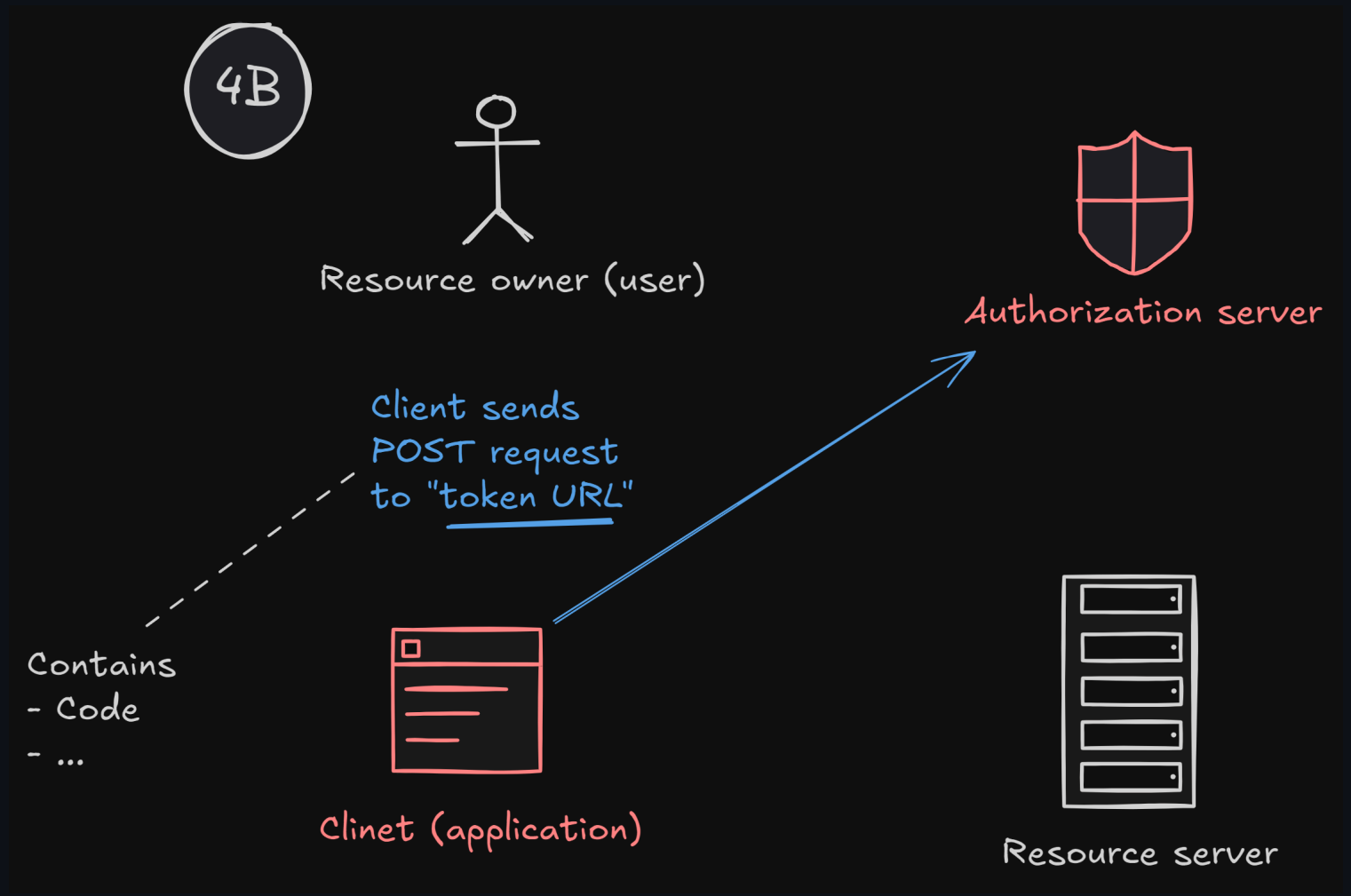
Resource server

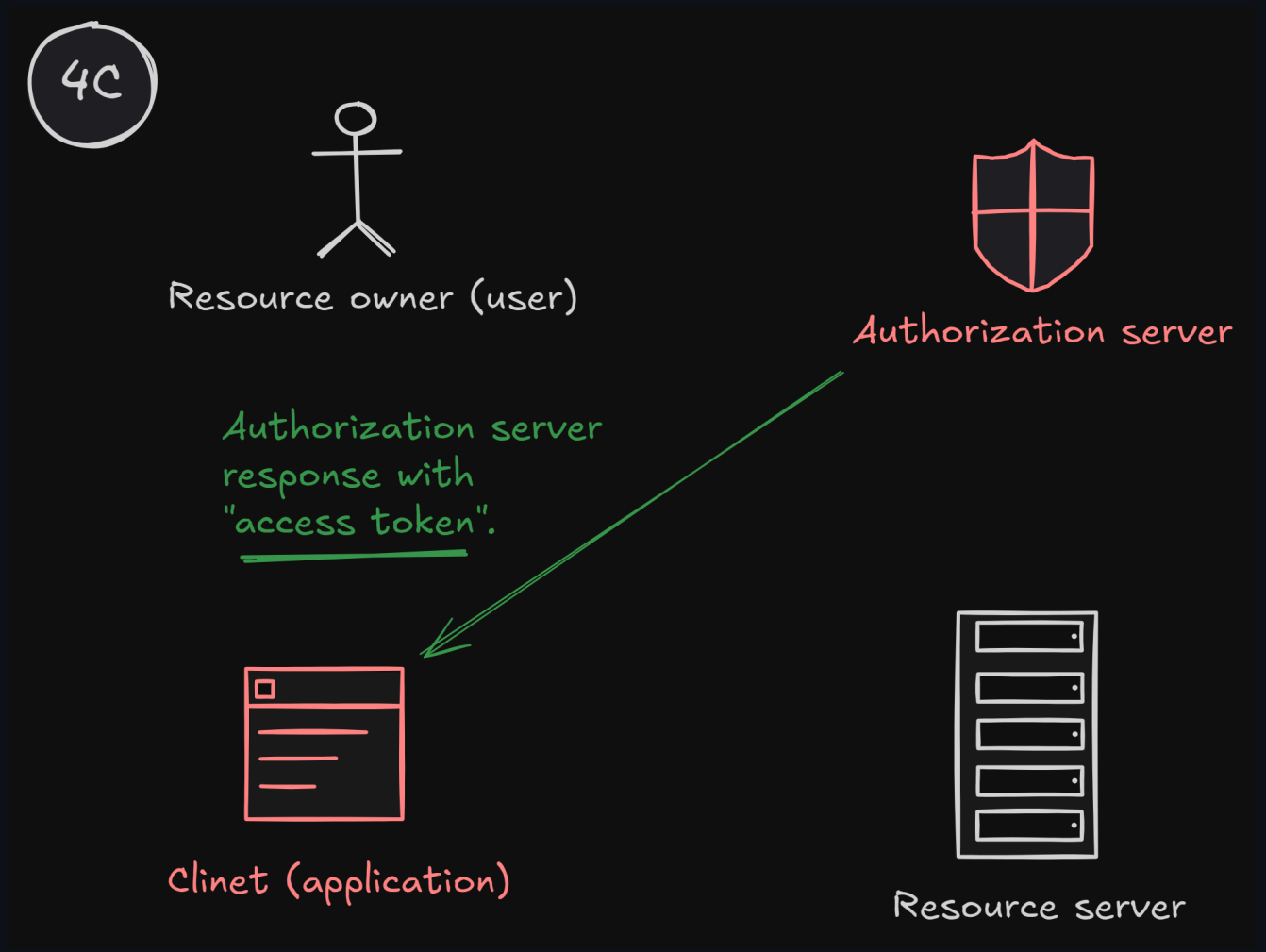

1
Resource owner (user)
Authorization server
User click at "social login" button
Clinet (application)
Resource server

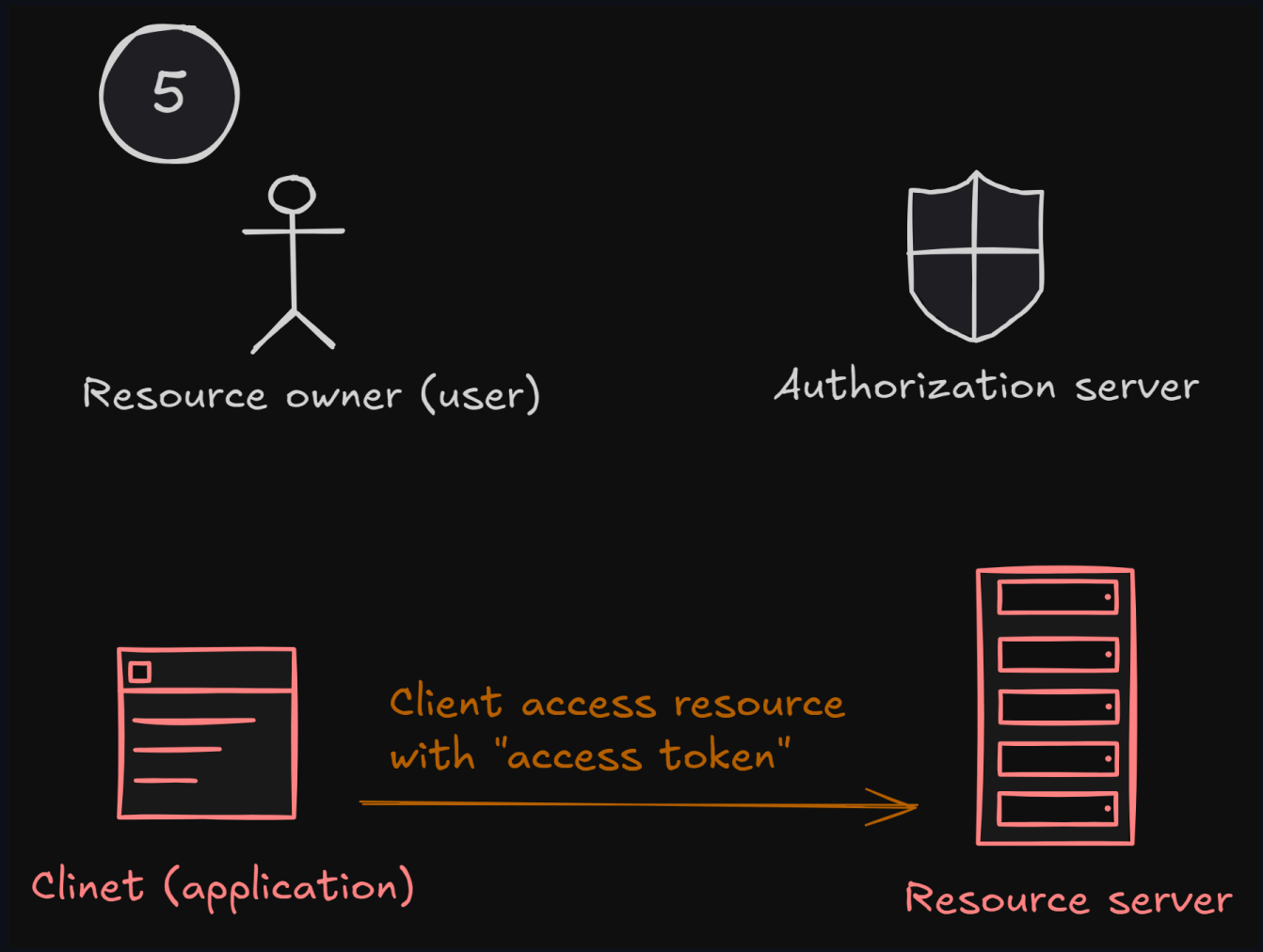

2
Resource owner (user)
Authorization server
Client redirect user to
"Authorization URL"
Contains
- Client ID
- Scope
- Response type
- ...
Clinet (application)
Resource server


3
Use authenticate to
authorization server
Resource owner (user)
Authorization server
Clinet (application)
Resource server


4A
Resource owner (user)
Authorization server
Authorization server
invokes "callback url".
Contains
- Code
- ...
Clinet (application)
Resource server


4B
Resource owner (user)
Authorization server
Client sends
POST request
to "token URL"
Contains
- Code
- ...
Clinet (application)
Resource server


4C
Resource owner (user)
Authorization server
Authorization server response with "access token".
Clinet (application)
Resource server


5
Resource owner (user)
Authorization server
Client access resource
with "access token"
Clinet (application)
Resource server

> Part 2: Social signing up/in

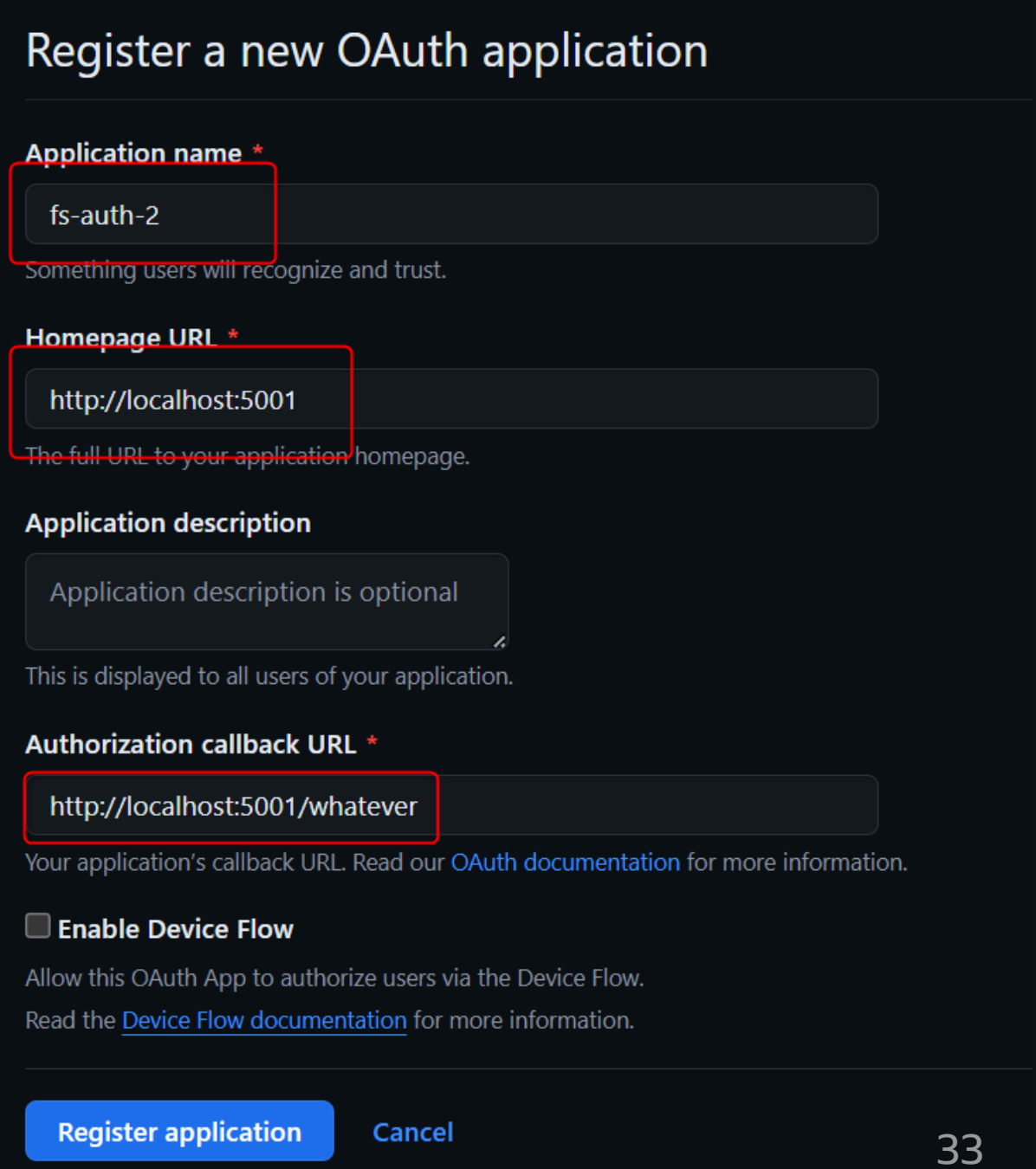
Section 2B: Oauth 2.0 with Github (Lab)

You will need

- `Client ID = ...`
- `Client Secret = ...`
- `Callback URL = http://localhost:5001/whatever`
- `Authoriation URL = ...`
- `Token URL = ...`
- `Resource URL`
  - `https://api.github.com/user`
  - `https://api.github.com/user/emails`
- `Access Token = ...`

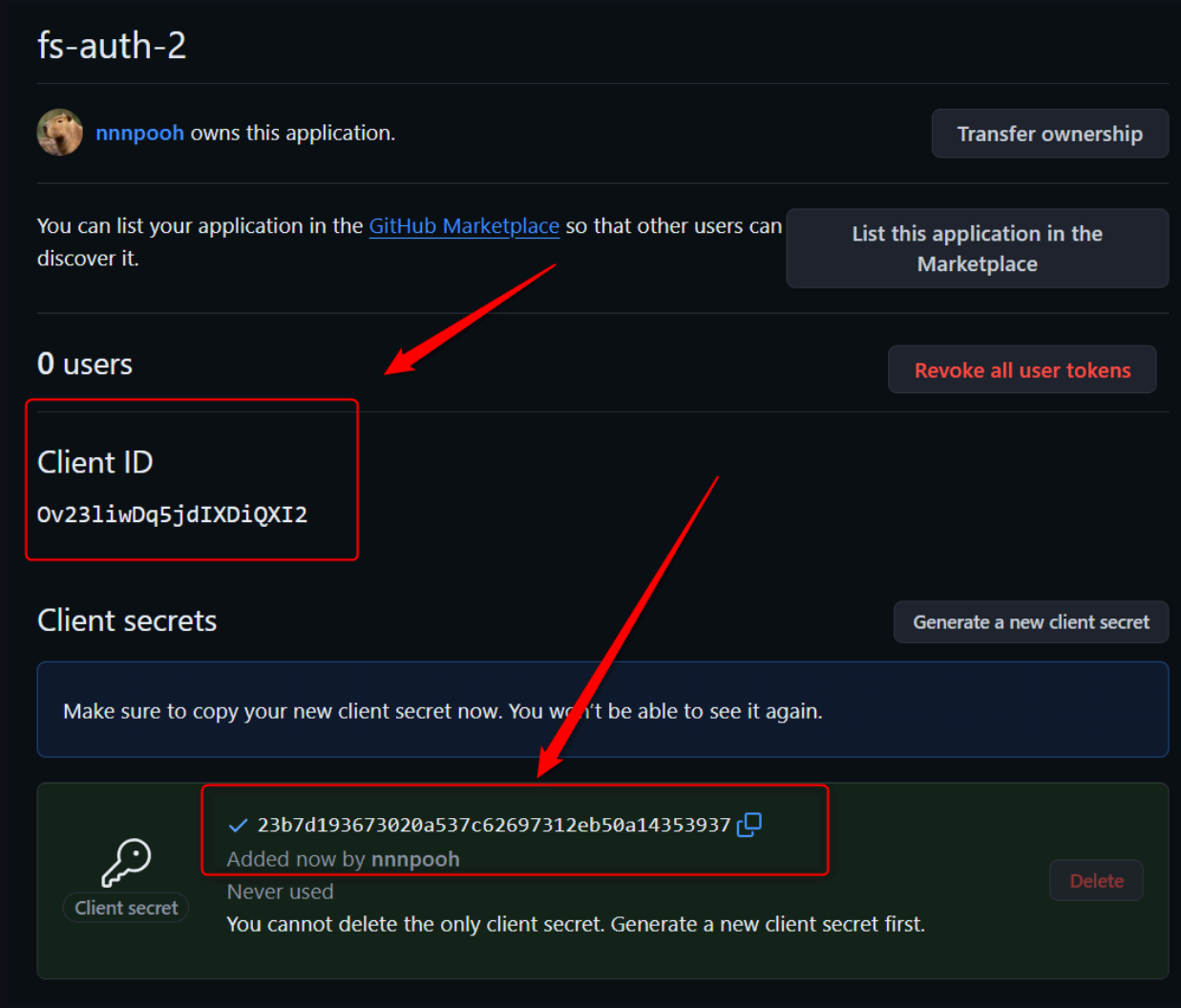
Setup

- Register your app here.
- `Homepage URL` and `Callback URL` can be whatever for now.



Setup

- Get `Client ID` and `Client Secret`



Setup

- Choose scope.
  - `scope=user,user:email`
- Contruct `Authorization URL`
  - `https://github.com/login/oauth/authorize?client_id=CLIENT_ID&redirect_uri=REDIRECT_URL&response_type=code&scope=SCOPE`

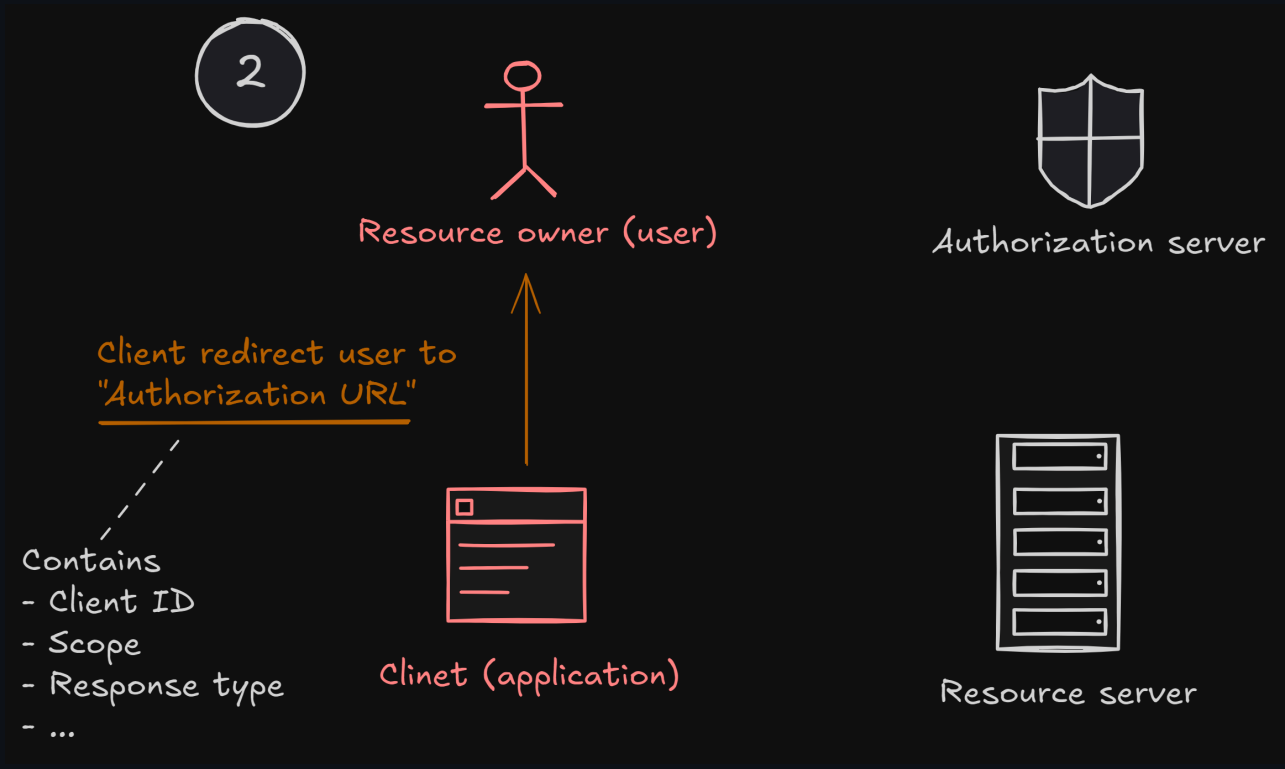
Setup

- Construct `Token URL` *(incompleted)*
  - `https://github.com/login/oauth/access_token?client_id=CLIENT_ID&client_secret=CLIENT_SECRET&code=CODE&redirect_uri=CALLBACK_URL`

Let's go

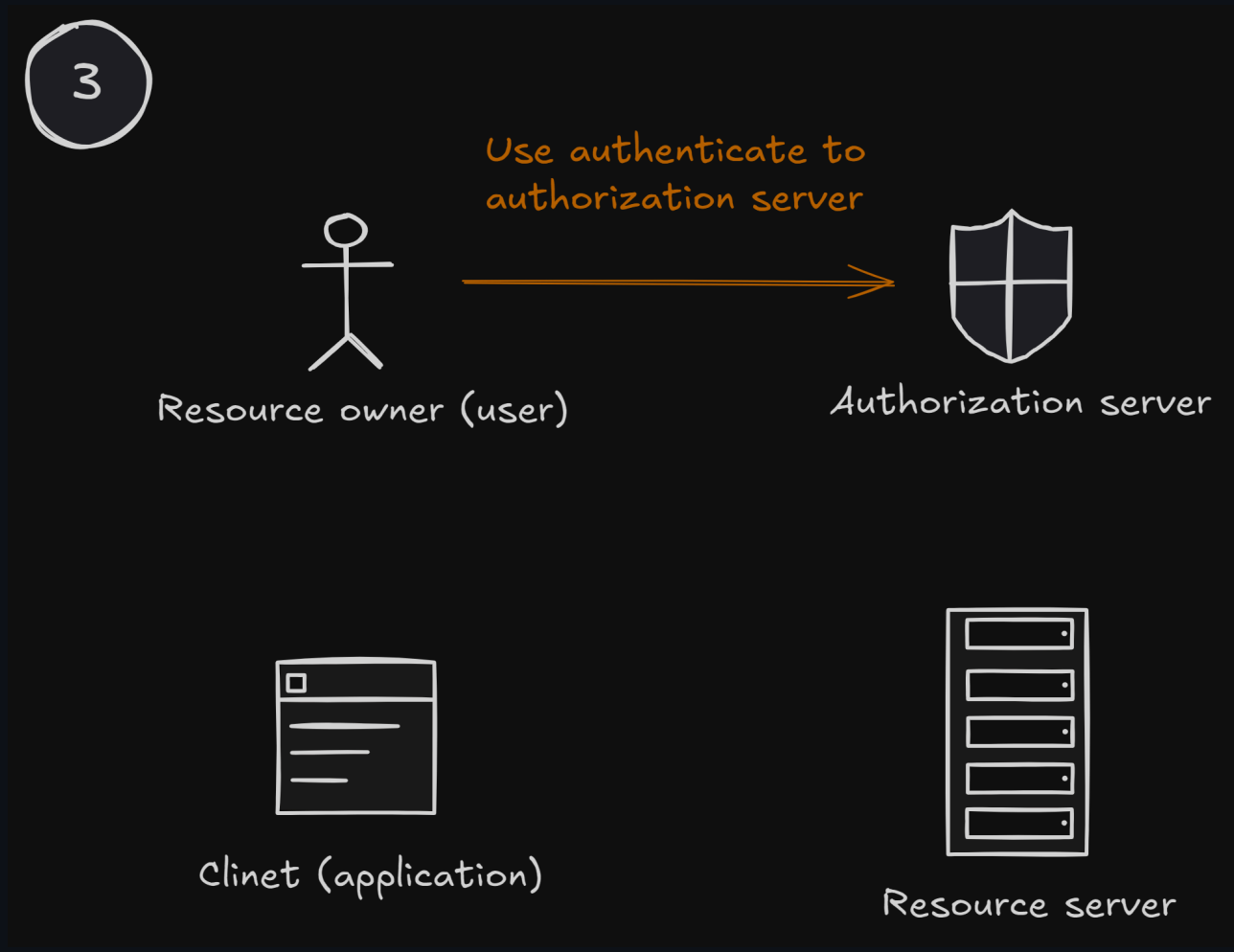
Step 2

- *(Skip 1 for now since there is no app.)*
- Goto at `Authorization URL`



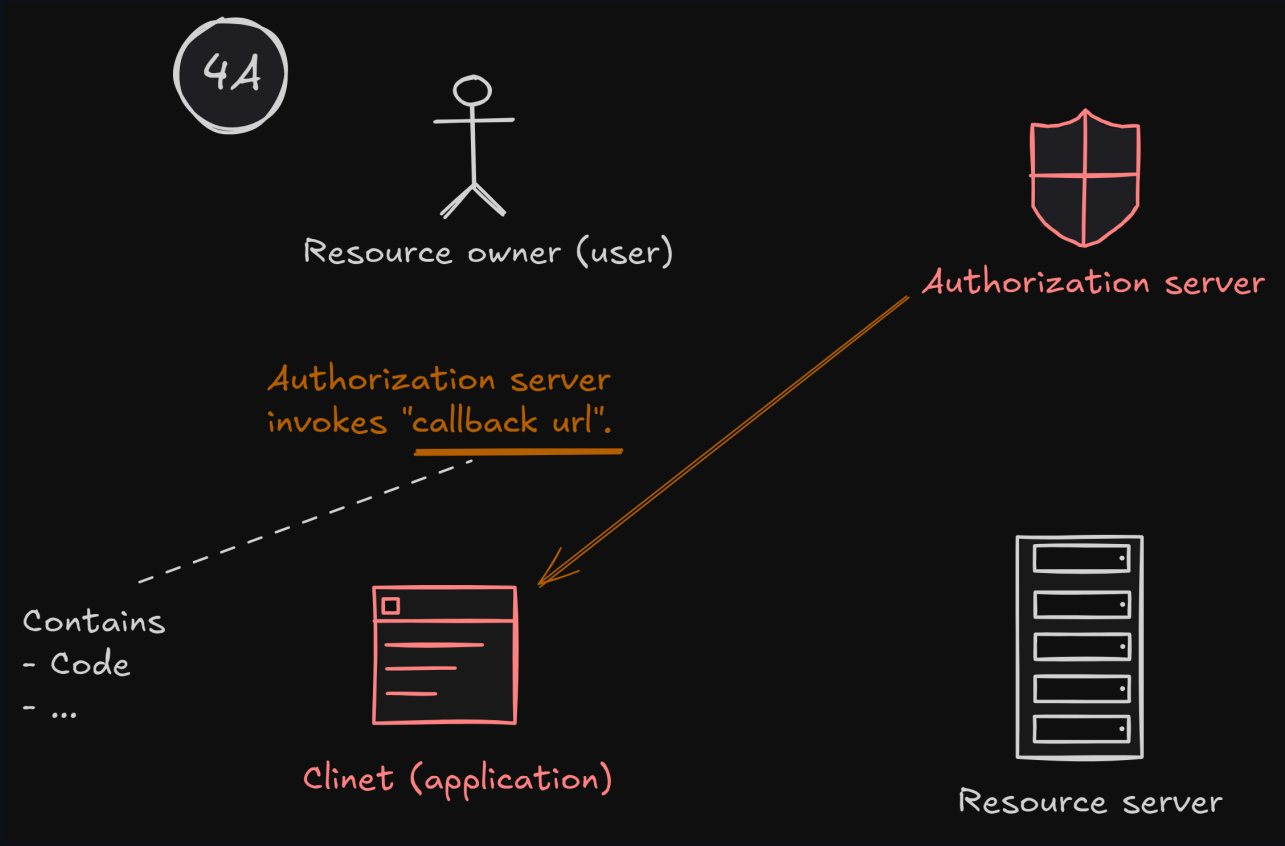
Step 3

- Authenticate with Github



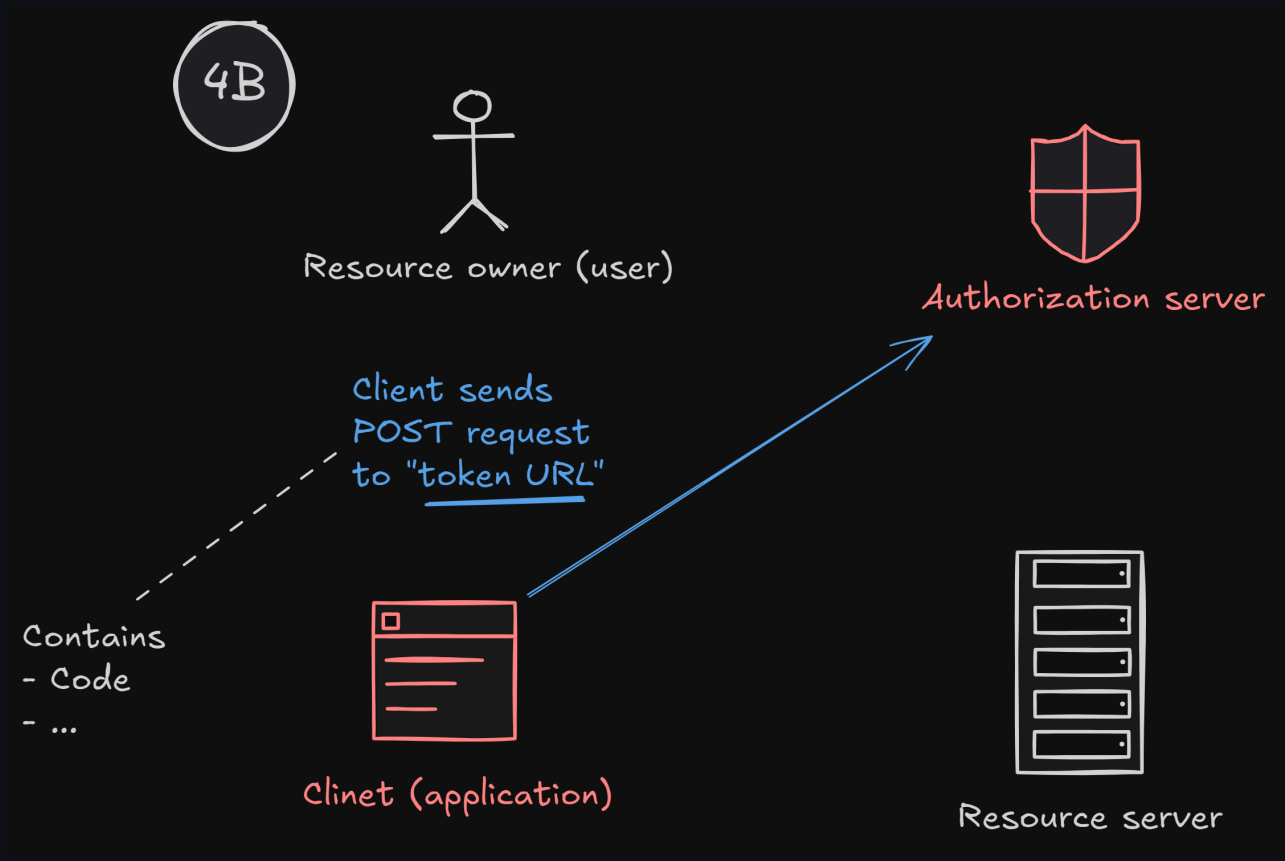
Step 4A

- Extract `code` and keep it.
- `code` is usually very short-lived.



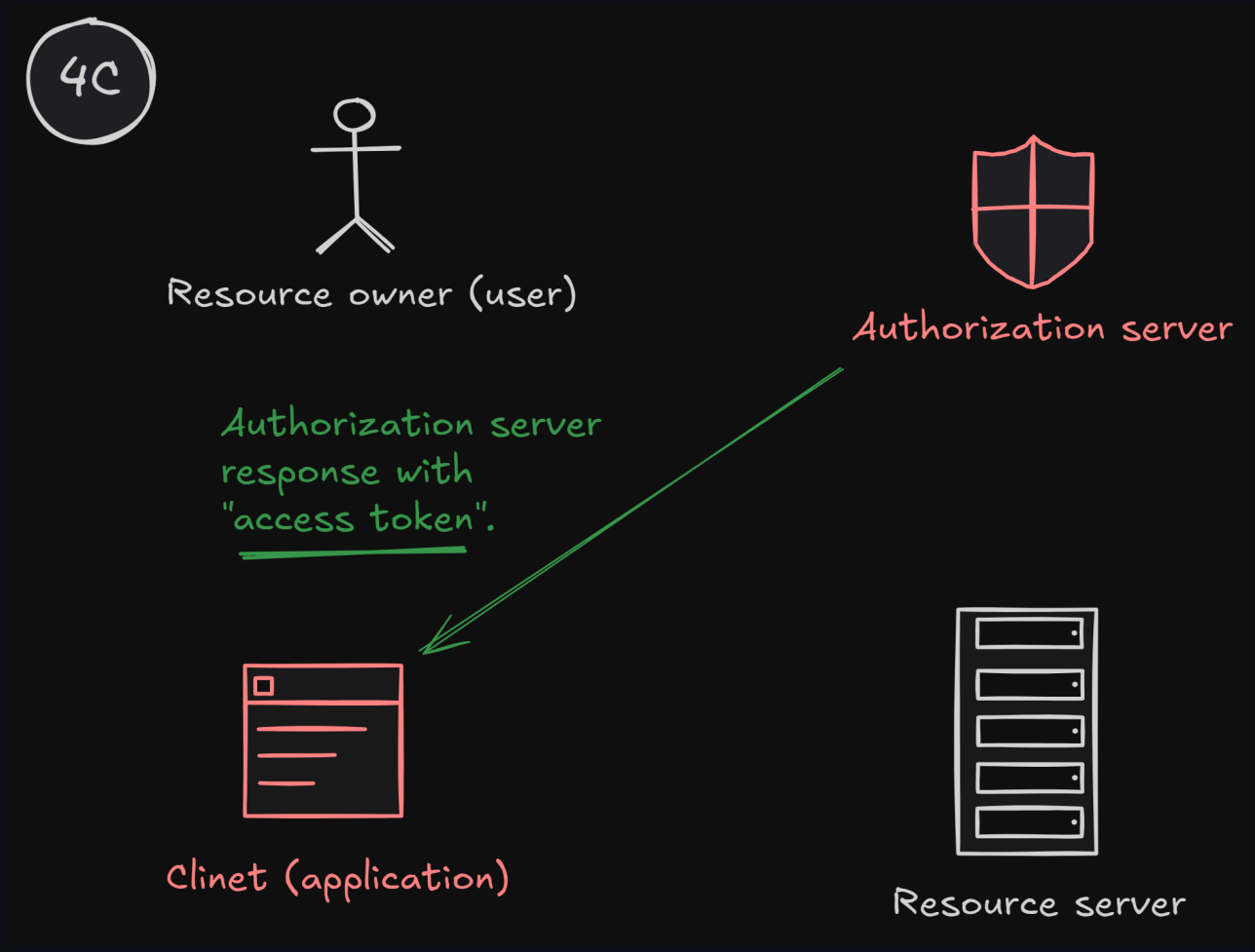
Step 4B

- Send `POST` request to `Token URL` with actual `Code`.
- Reference



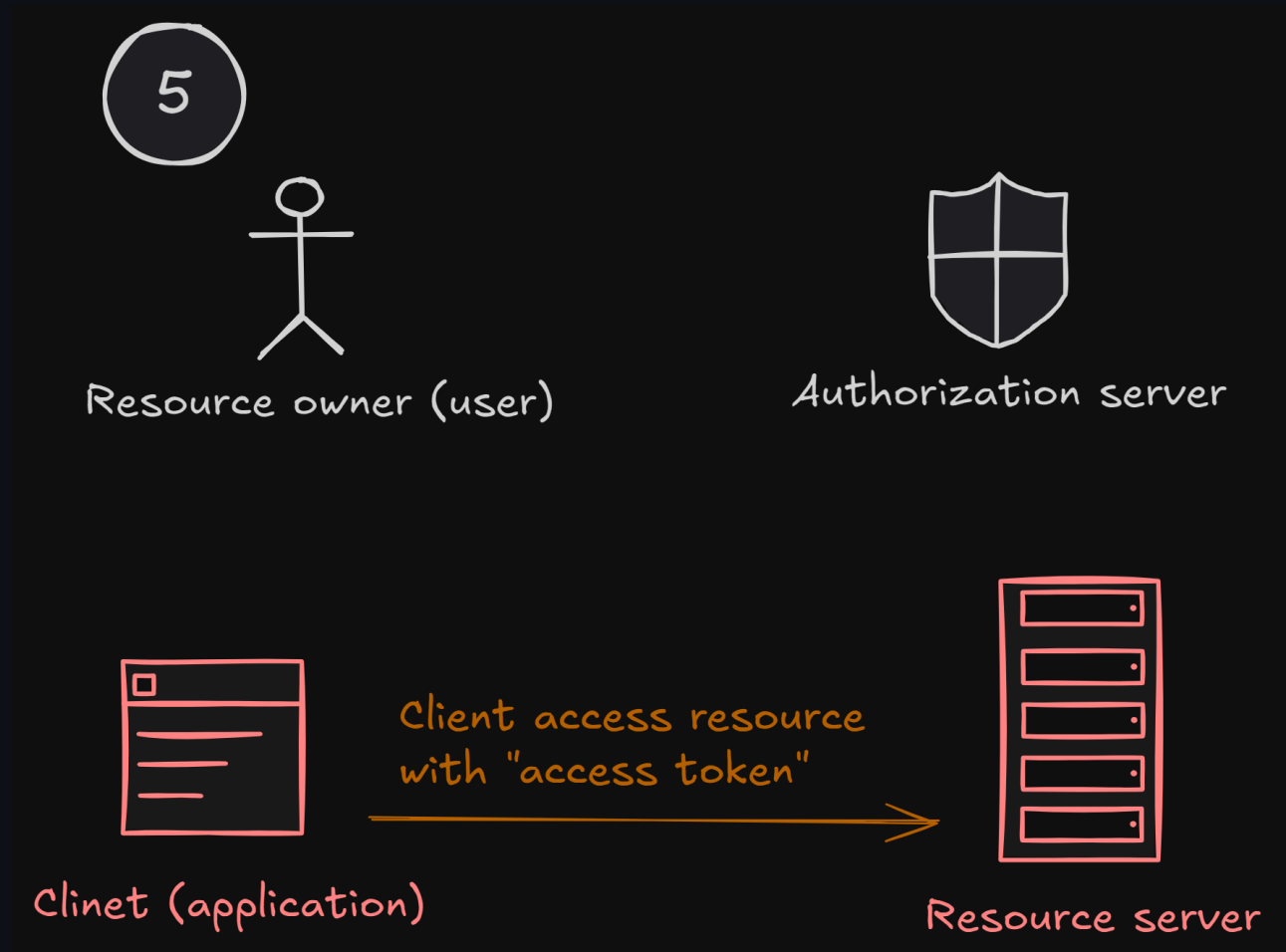
Step 4C

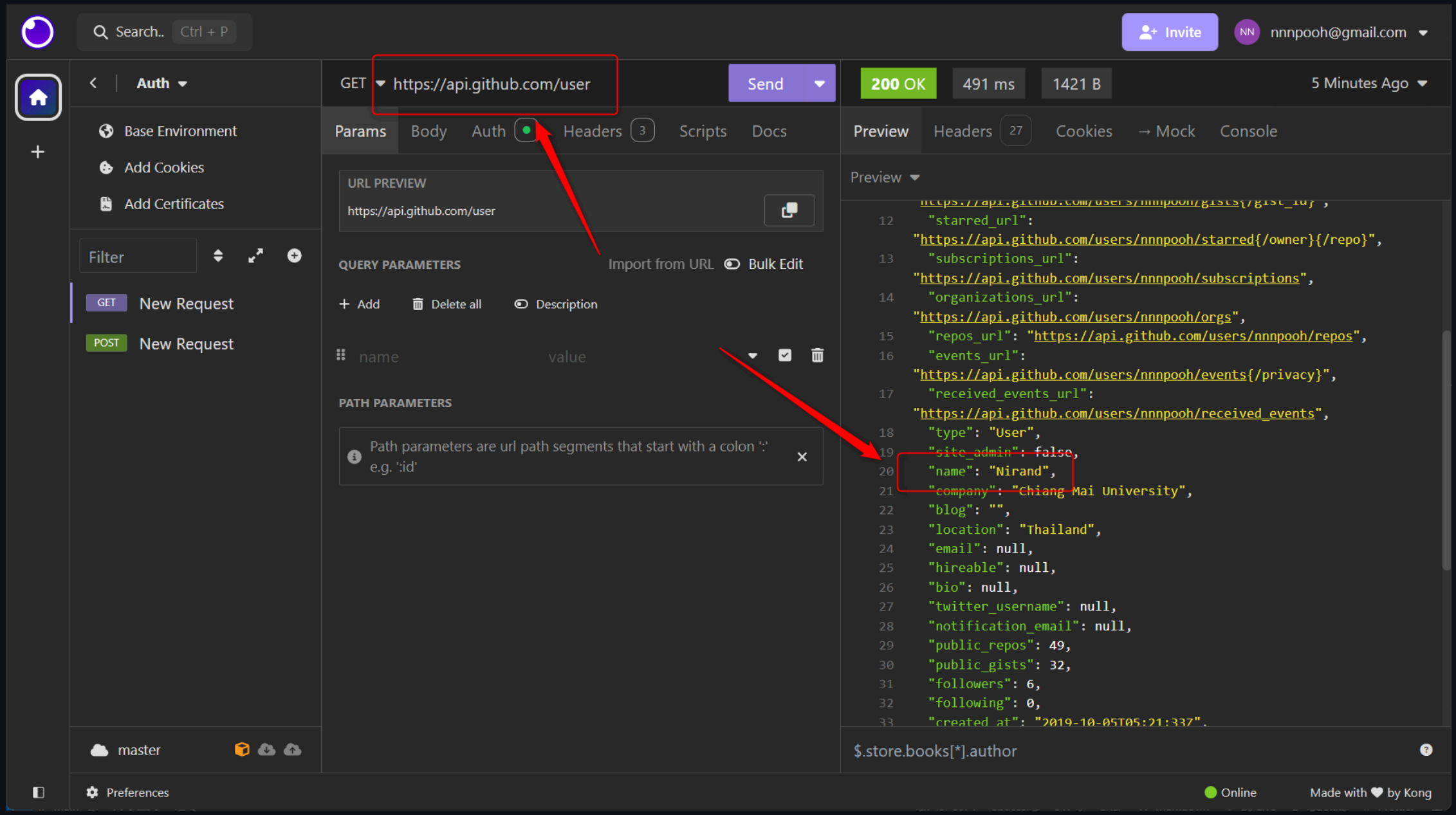
- Keep `Access Token` from the response.



Step 5

- Send `GET` requestion to resouces.
- Use `Access Token` as bearer token.
- Reference

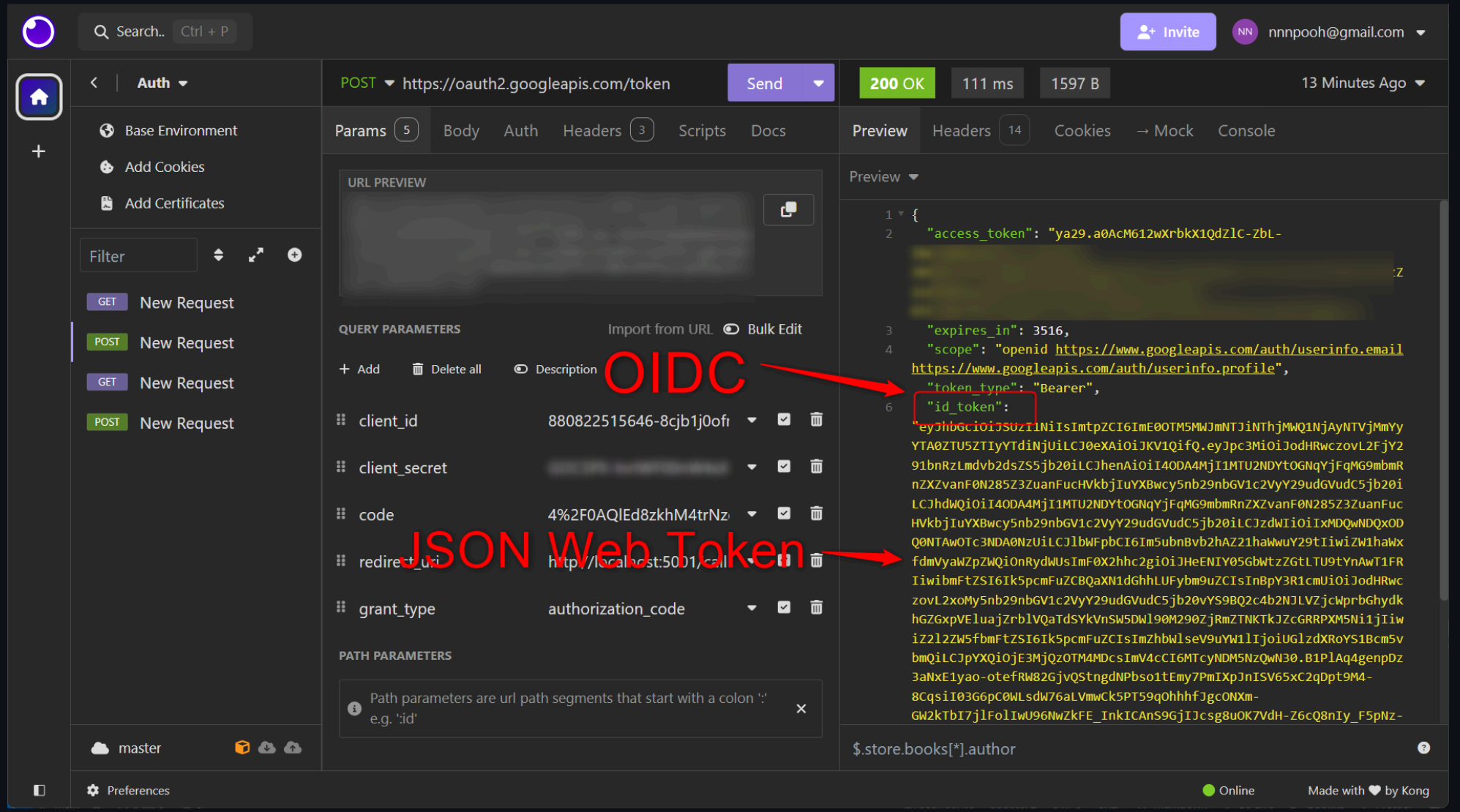



Search.. Ctrl + P
Invite
nnnpooh@gmail.com
Auth
Base Environment
Add Cookies
Add Certificates
Filter
GET New Request
POST New Request
master
Preferences
GET https://api.github.com/user
Send
Params
Body
Auth
Headers 3
Scripts
Docs
URL PREVIEW
https://api.github.com/user
QUERY PARAMETERS
Import from URL
Bulk Edit
+ Add
Delete all
Description
name
value
PATH PARAMETERS
Path parameters are url path segments that start with a colon ':'
e.g. ':id'
200 OK
491 ms
1421 B
5 Minutes Ago
Preview
Headers 27
Cookies
→ Mock
Console
Preview
"starred_url": "https://api.github.com/users/nnnpooh/starred{/owner}{/repo}",
"subscriptions_url": "https://api.github.com/users/nnnpooh/subscriptions",
"organizations_url": "https://api.github.com/users/nnnpooh/orgs",
"repos_url": "https://api.github.com/users/nnnpooh/repos",
"events_url": "https://api.github.com/users/nnnpooh/events{/privacy}",
"received_events_url": "https://api.github.com/users/nnnpooh/received_events",
"type": "User",
"site_admin": false,
"name": "Nirand",
"company": "Chiang Mai University",
"blog": "",
"location": "Thailand",
"email": null,
"hireable": null,
"bio": null,
"twitter_username": null,
"notification_email": null,
"public_repos": 49,
"public_gists": 32,
"followers": 6,
"following": 0,
"created_at": "2019-10-05T05:21:33Z",
$.store.books[*].author
Online
Made with ♥ by Kong

Google OAuth 2.0

- `Authoriation URL = https://accounts.google.com/o/oauth2/v2/auth?client_id=CLIENT_ID&redirect_uri=REDIRECT_URI&response_type=code&scope=openid+https://www.googleapis.com/auth/userinfo.email+https://www.googleapis.com/auth/userinfo.profile`
- `Token URL = https://oauth2.googleapis.com/token?client_id=CLIENT_ID&client_secret=CLIENT_SECRET&code=CODE&redirect_uri=CALLBACK_URL&grant_type=authorization_code`
- `Resource URL = https://www.googleapis.com/oauth2/v2/userinfo`

- Google supports OIDC. You will receive `id_token`.



> Part 2: Social signing up/in

Section 2C: `passport` implementation

Setup

- `git clone -b signin-oauth https://github.com/fullstack-67/auth-mpa-v2.git auth-signin-oauth`
- `pnpm i`
- `npm run db:reset`
- `npm run dev`

Note

- Database tables
- Types of response objects
- Add type definition to `req.user`.

Highlighed packages

```
{
  "passport": "^0.7.0",
  "passport-oauth2": "^1.8.0"
}
```

```
// * Passport
passport.use("github", github);
passport.use("google", google);
app.use(passport.initialize());
```

```
export const github = new OAuthStrategy(
  {
    // Option
  },
  async function (
    accessToken: string,
    refreshToken: string,
    profile: any,
    done: VerifyCallback
  ) {
    // Do something with accessToken
  }
);
```