HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD

www.heig-vd.ch

Wrap up and F.A.Q

Answers to your questions

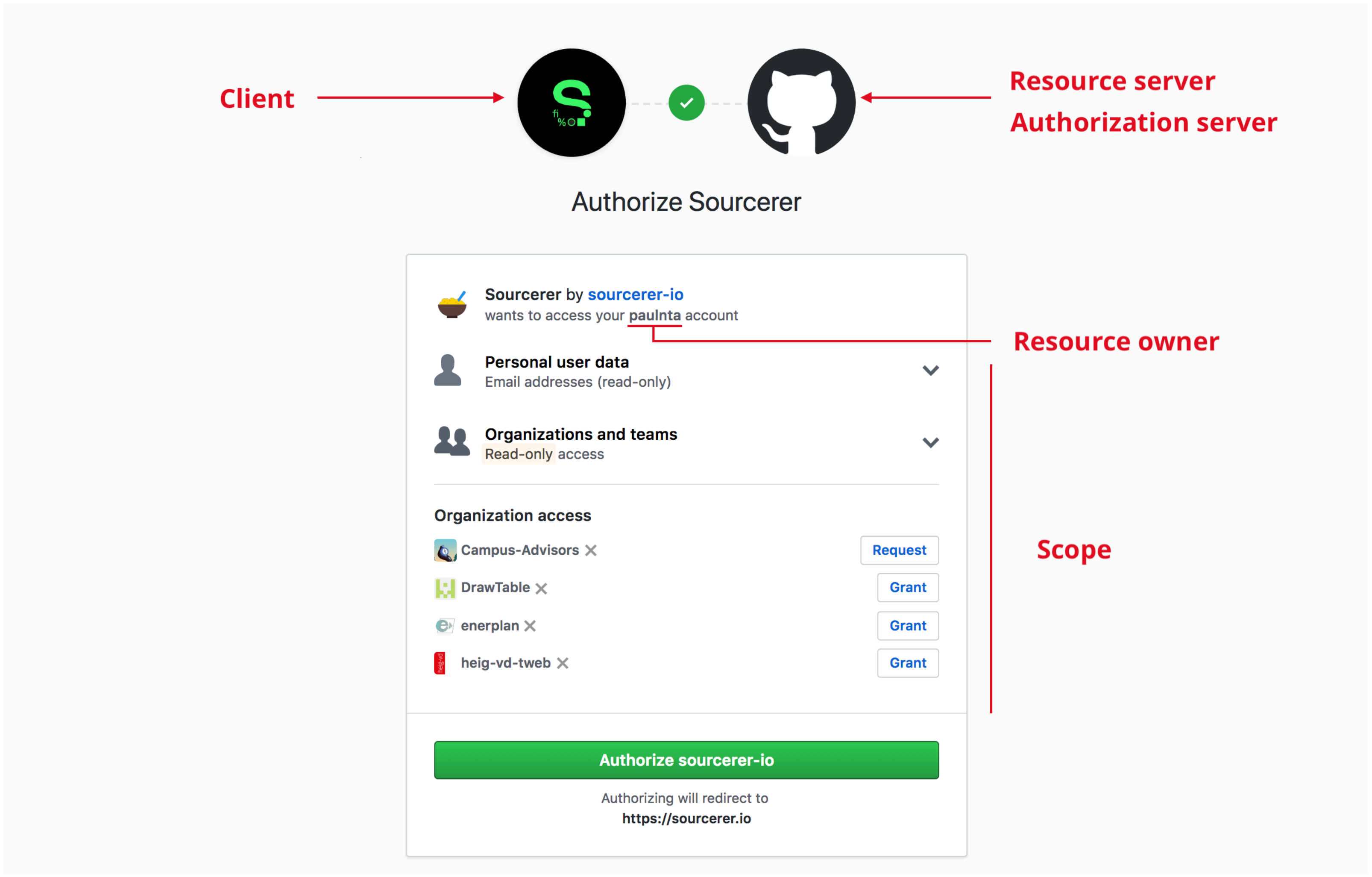
Daily menu

- What is OAuth ?
- How to implement OAuth web flow ?
- How many unit tests should I write ?
- How to prepare for the test ?

What is OAuth ?

- OAuth is an authorization framework / protocol.
- It allows users to grant limited access to their resources on a site
- A third party application (your github-analytics app) can access protected github resources (private email, repos) on behalf of a user
- Instead of using the user's credential, the client obtains an access token

In practice



Authorize Sourcerer

Sourcerer by sourcerer-io
wants to access your paulnta account

Personal user data
Email addresses (read-only)

Organizations and teams
Read-only access

Organization access

Campus-Advisors ✕	Request
DrawTable ✕	Grant
enerplan ✕	Grant
heig-vd-tweb ✕	Grant

Authorizing will redirect to
https://sourcerer.io

Roles

Resource owner: (paulnta)
The end-user, capable of granting access to a protected resource.

Resource server: (github)
Where protected data are stored. An access token is required to access protected data.

Client: (sourcerer.io)
The application that needs to access protected resources on behalf of the *resource owner* (paulnta)

Authorization server: (github, again)
Issues access token to the *client* after obtaining authorizations from the *resource owner*

OAuth flow in theory

```
+--------+                               +---------------+
|        |--(A)- Authorization Request ->|   Resource    |
|        |                               |     Owner     |
|        |<-(B)-- Authorization Grant ---|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(C)-- Authorization Grant -->| Authorization |
| Client |                               |     Server    |
|        |<-(D)----- Access Token -------|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(E)----- Access Token ------>|    Resource   |
|        |                               |     Server    |
|        |<-(F)--- Protected Resource ---|               |
+--------+                               +---------------+
```

https://tools.ietf.org/html/rfc6749#section-1.2

Authorization Grant vs Access token

Authorization Grant:

- The authorization grant represent a **confirmation** that the resource owner **has authorized the client** to access protected resources.
- There is different grant types.
- Github uses **Authorization code** (request token).

Access Tokens:

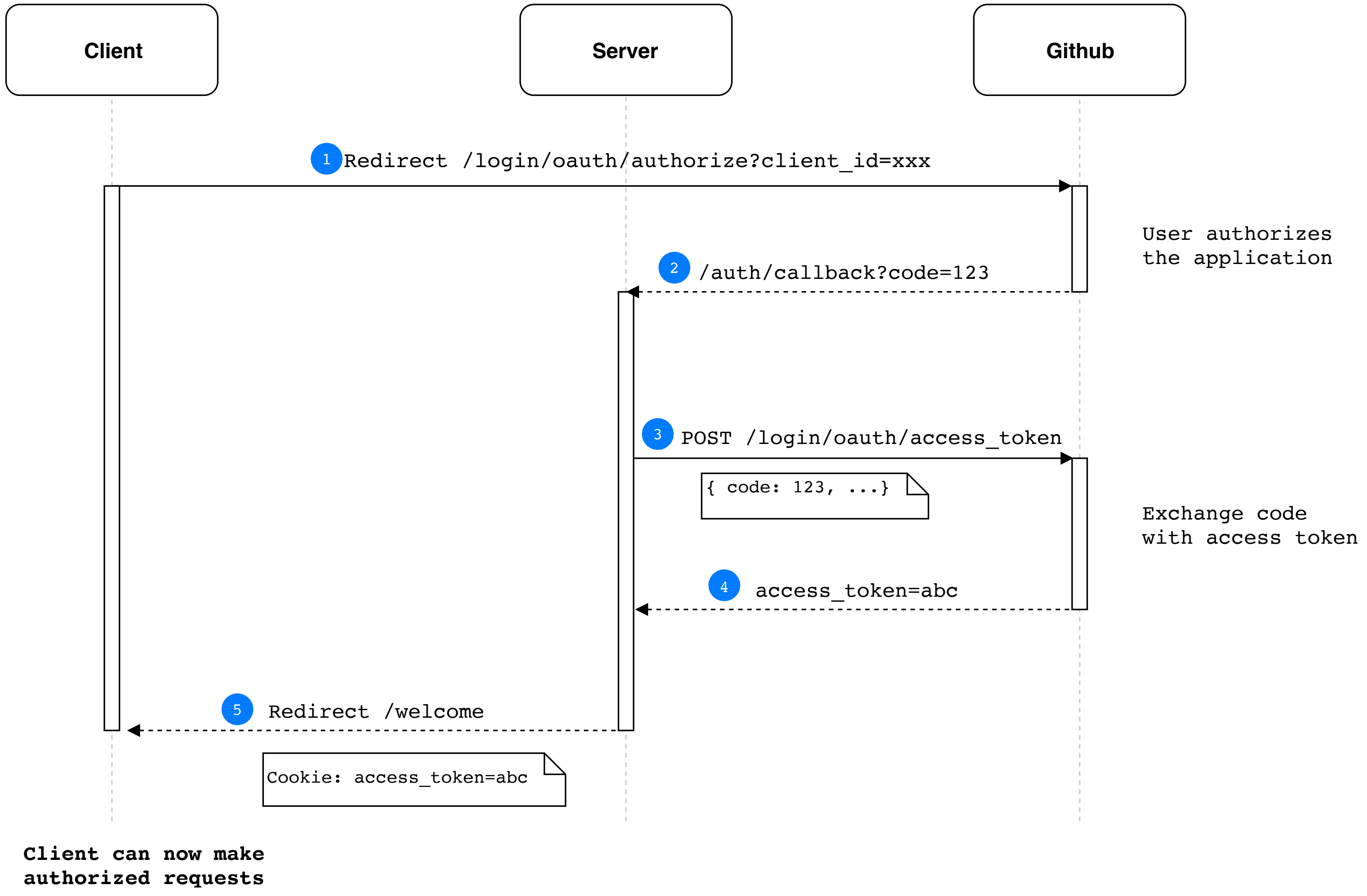
- are **credentials** used to access protected resources
- replaces (`username` and `password`) for single token string
- contains all necessary information to **identify a user**
- have an **expiry** time
- have a **scope**

Github OAuth in practice ?

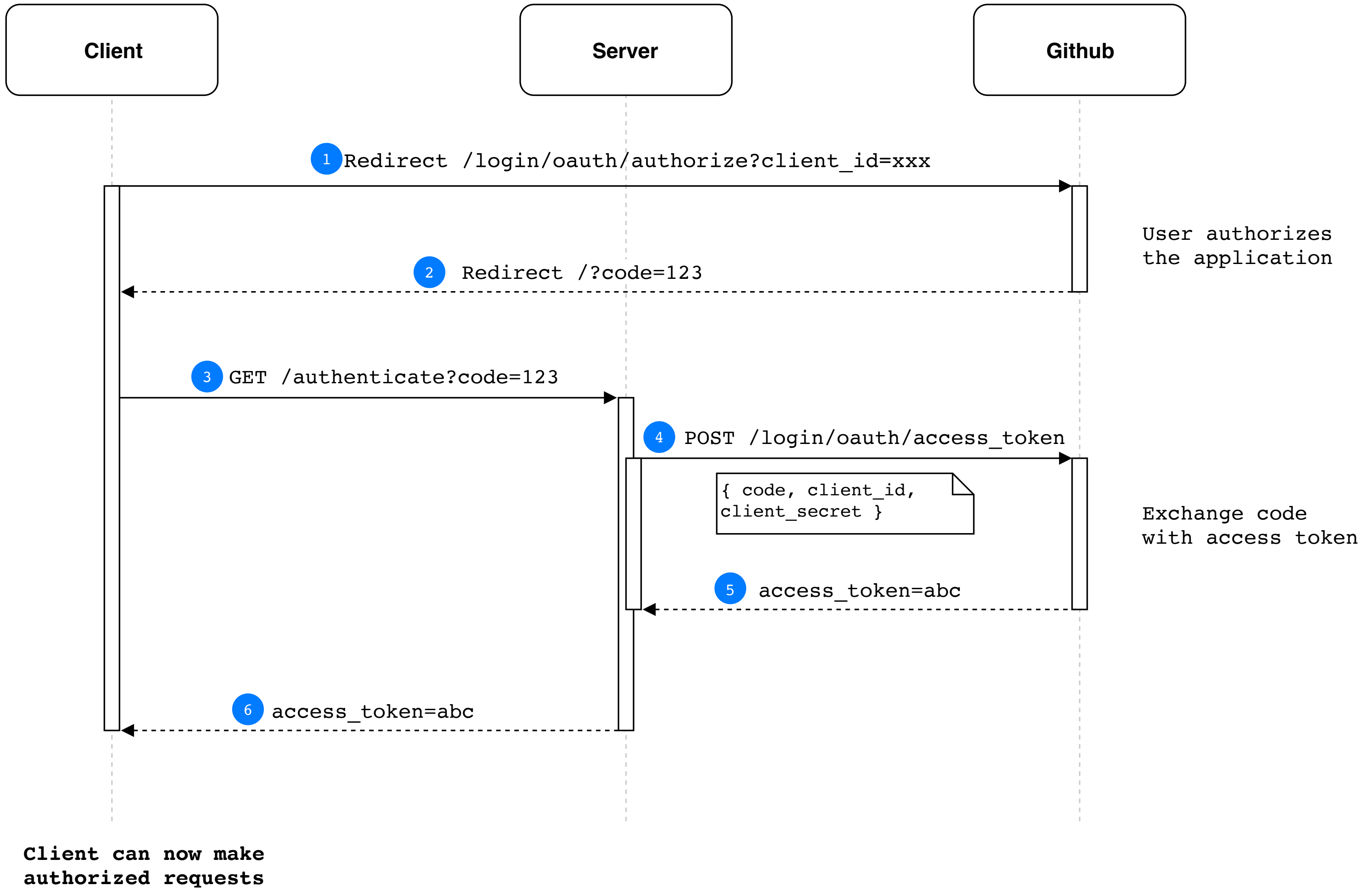
- You need to create a Github App or Github OAuth App.
- You need a **client app** that can redirect users to Github authorization page and receive an **authorization grant**
- You need a **server** to store secret information (`client_secret`) and exchange authorization grants with request **access tokens**.
- At the end, you want an access token to make **authenticated requests**.

Follow the official guide: web application flow

OAuth flow in practice (1)


Client
Server
Github
1 Redirect /login/oauth/authorize?client_id=xxx
User authorizes the application
2 /auth/callback?code=123
3 POST /login/oauth/access_token
{ code: 123, ...}
Exchange code with access token
4 access_token=abc
5 Redirect /welcome
Cookie: access_token=abc
Client can now make authorized requests

OAuth flow in practice (2)


Client
Server
Github
1 Redirect /login/oauth/authorize?client_id=xxx
User authorizes the application
2 Redirect /?code=123
3 GET /authenticate?code=123
4 POST /login/oauth/access_token
{ code, client_id, client_secret }
Exchange code with access token
5 access_token=abc
6 access_token=abc
Client can now make authorized requests

How many unit tests should I write?

> You should instead ask : **What** should I test with unit tests?

- Test the common case of everything you can
  - This will serve as **documentation** for you and others
  - You will be informed about any **breaking changes**
- Test the **edge cases** of a few unusually **complex code** that you think will probably have errors
- Whenever you find a **bug**, write a test case to cover it before fixing it
- Add edge-case tests to less critical code whenever someone has time to kill
- Favor tests with few or **single logical assertion**
- Following those advices leads to more **robust** and **modular** codebase

How to prepare for the test ?

You should expect theoretical questions on **chapters 1 - 5**, except for *04 - Towards deployment* > *Writing next generation JavaScript / Babel*

Be sure to ...

- practice with asynchronous programming — callbacks and promises
- understand how to structure data efficiently in MongoDB
- be able to describe all tools we saw, how they work and how they complement with each other
- understand javascript language core features and its relationship with engines
- Finish and understand all aspect of the project 1.
- ~~read all references and links~~