OIDC Autherization Code Grant

ttps://accounts.google.com/o/oauth2/v2/auth

client_id=CLIENT_ID&redirect_uri=REDIRECT_URL&scope=openid &response_type=code
response_mode=form_post &nonce=qdhdhewh

Victoria

Browser

2

3. Sign in

4. Reque client

5. Authorize

6. Redirect with A code

Sign in with Google

Choose an account

to continue to Crypto Demo

Wojciech Lesniak
voitek.lesniak@gmail.com

Use another account

To continue, Google will share your name, email address, language preference, and profile picture with Crypto Demo.

English (United States) Help Privacy Terms

Crypto Portfolio

client id

secret

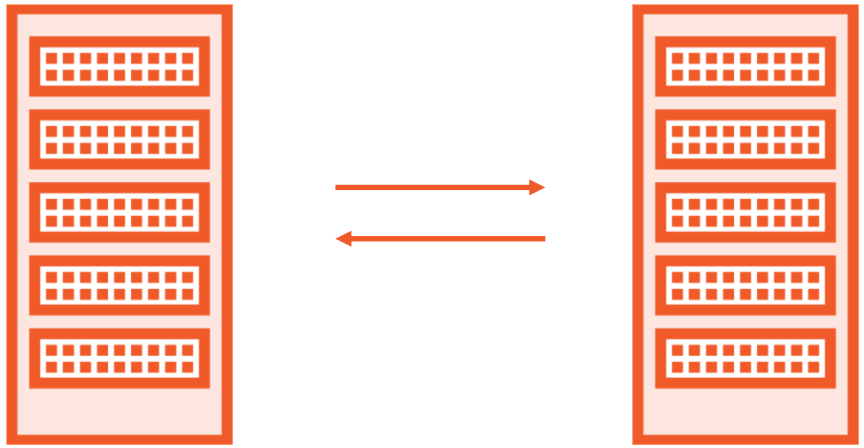
. Auth code

tokens

10. Request user details

Access token

Back Channel

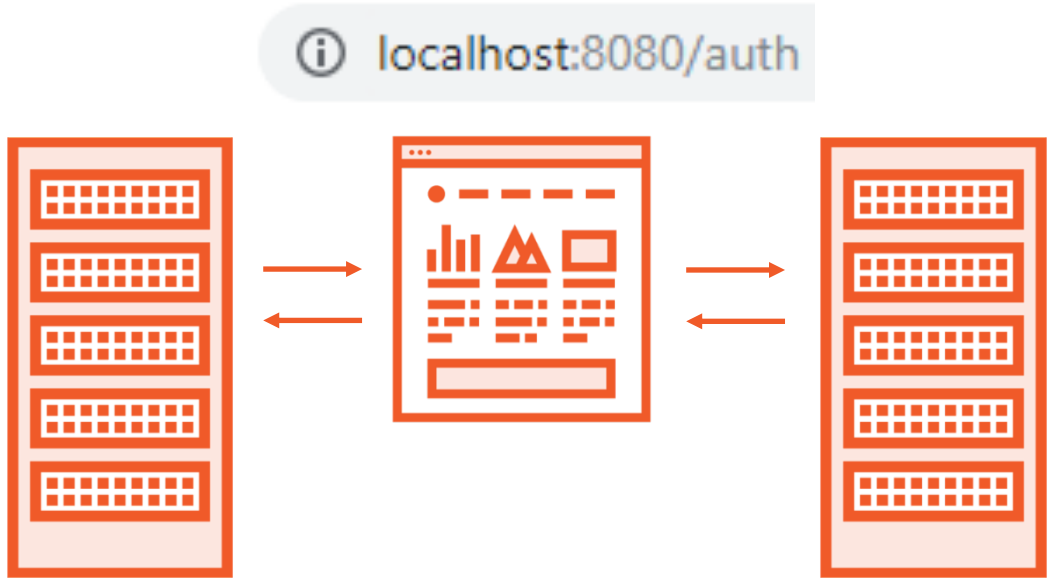


Server to server

More secure

Request cannot be tampered

Front Channel



Transmitted via the browsers address bar

Less secure

Client does not require an IP address

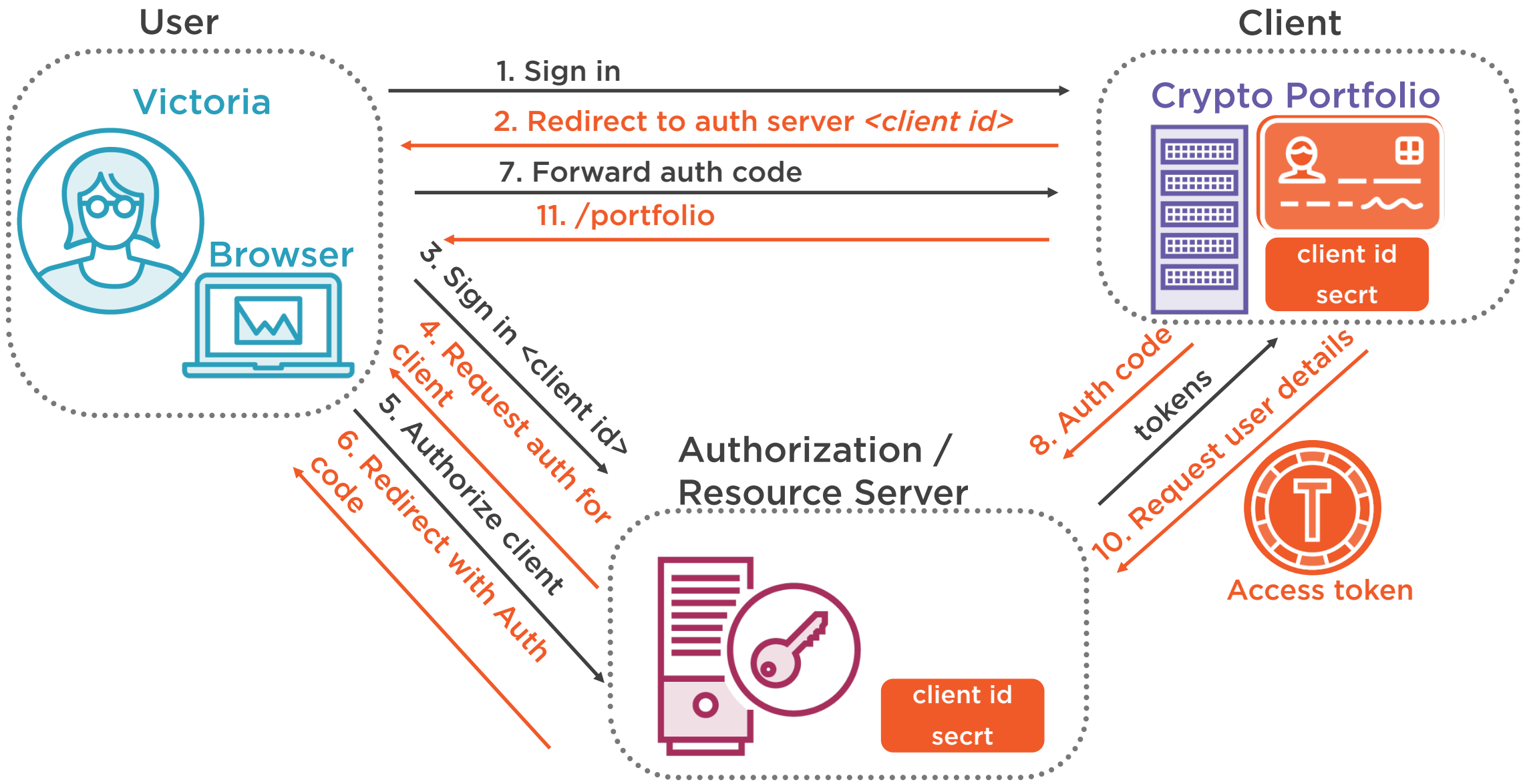
Back Channel Challenges for Public Clients

Cross-origin requests not supported by older versions of browsers

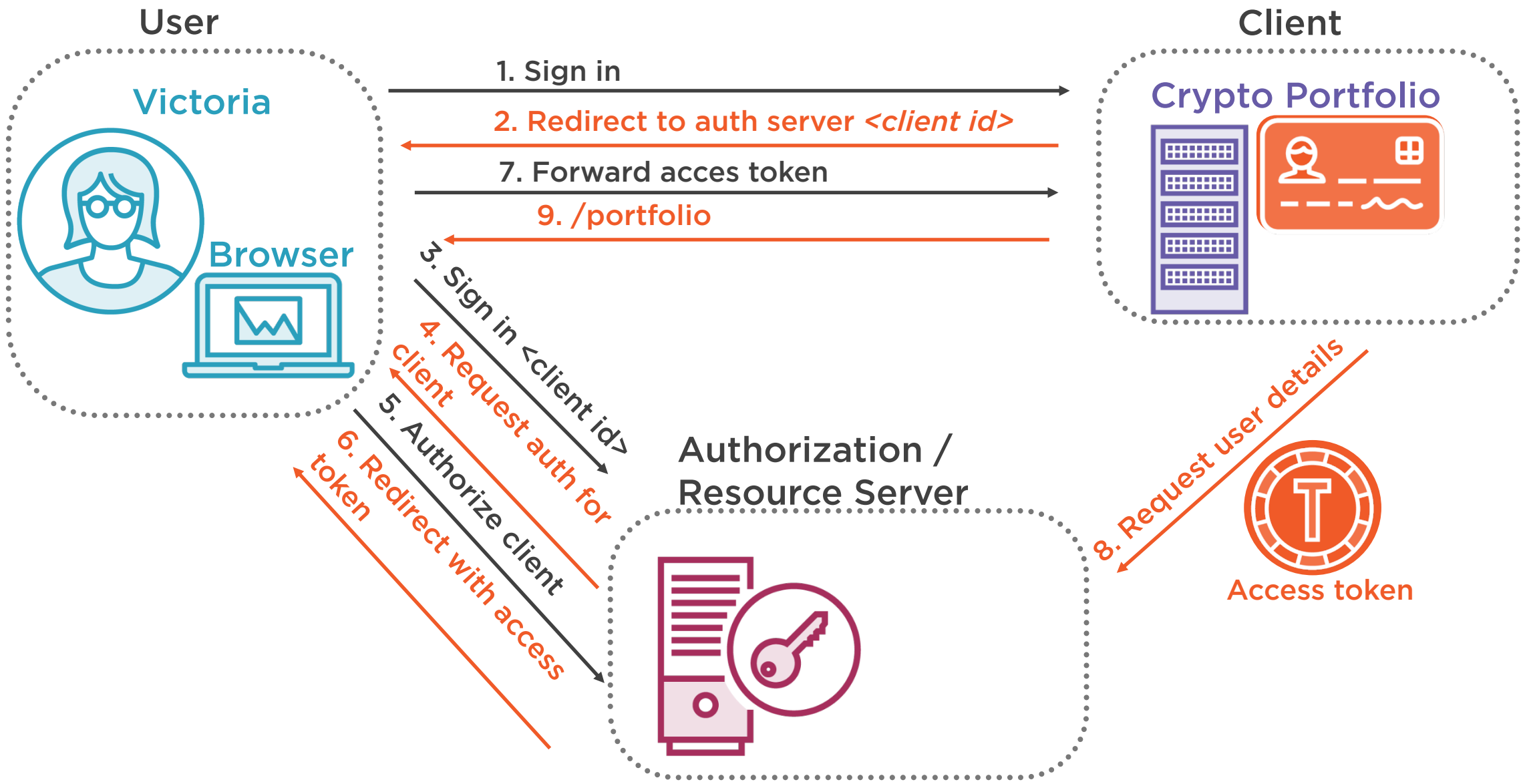
Storing the client secret

Code is running on the users device

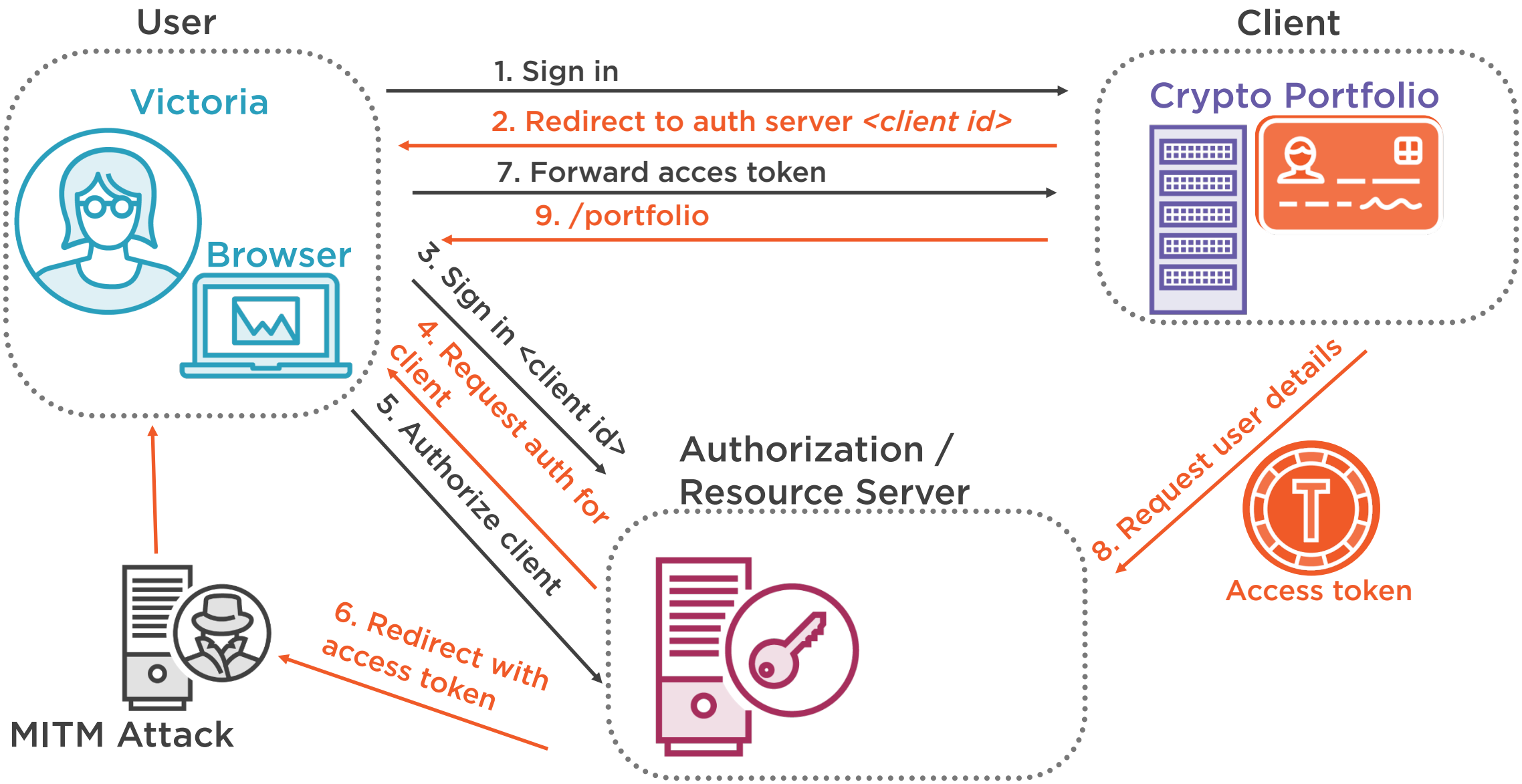
Authorization Code Grant



Implicit Grant



Implicit Grant



Oauth2 Front Channel

URLs are written to browser history

Authorization code is useless without the client secret

Allows the resource owner to be involved in the transaction, to give consent

Client does not require an IP address

Implicit Flow

IETF's OAuth working group no longer recommend the Implicit Flow

- CORS (Cross-Origin Resource Sharing is now universally adopted by Brower
- Authorization code flow with PKCE (Proof of Code Exchange) is now a viable option

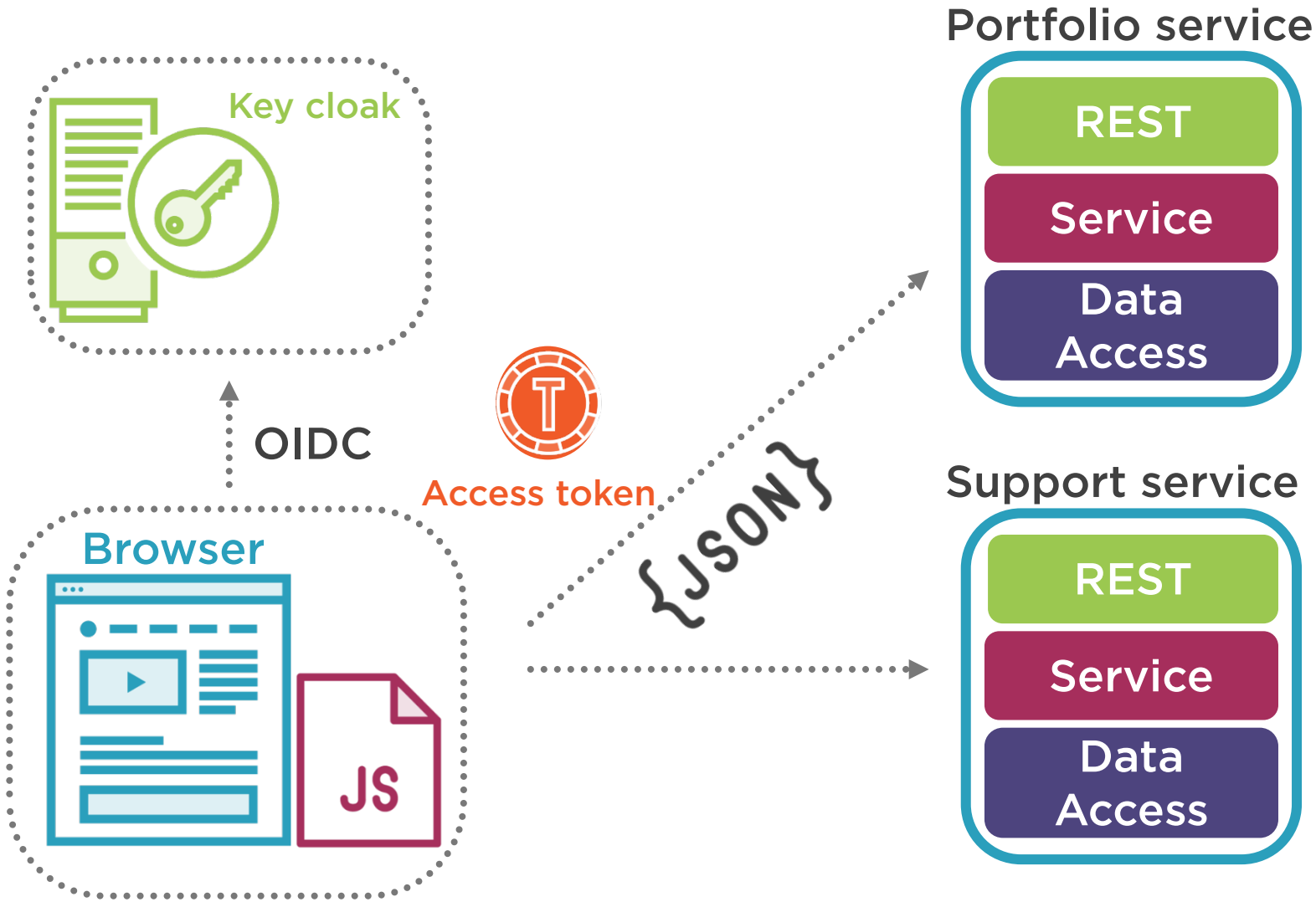
Storing the Access Token in the Browser

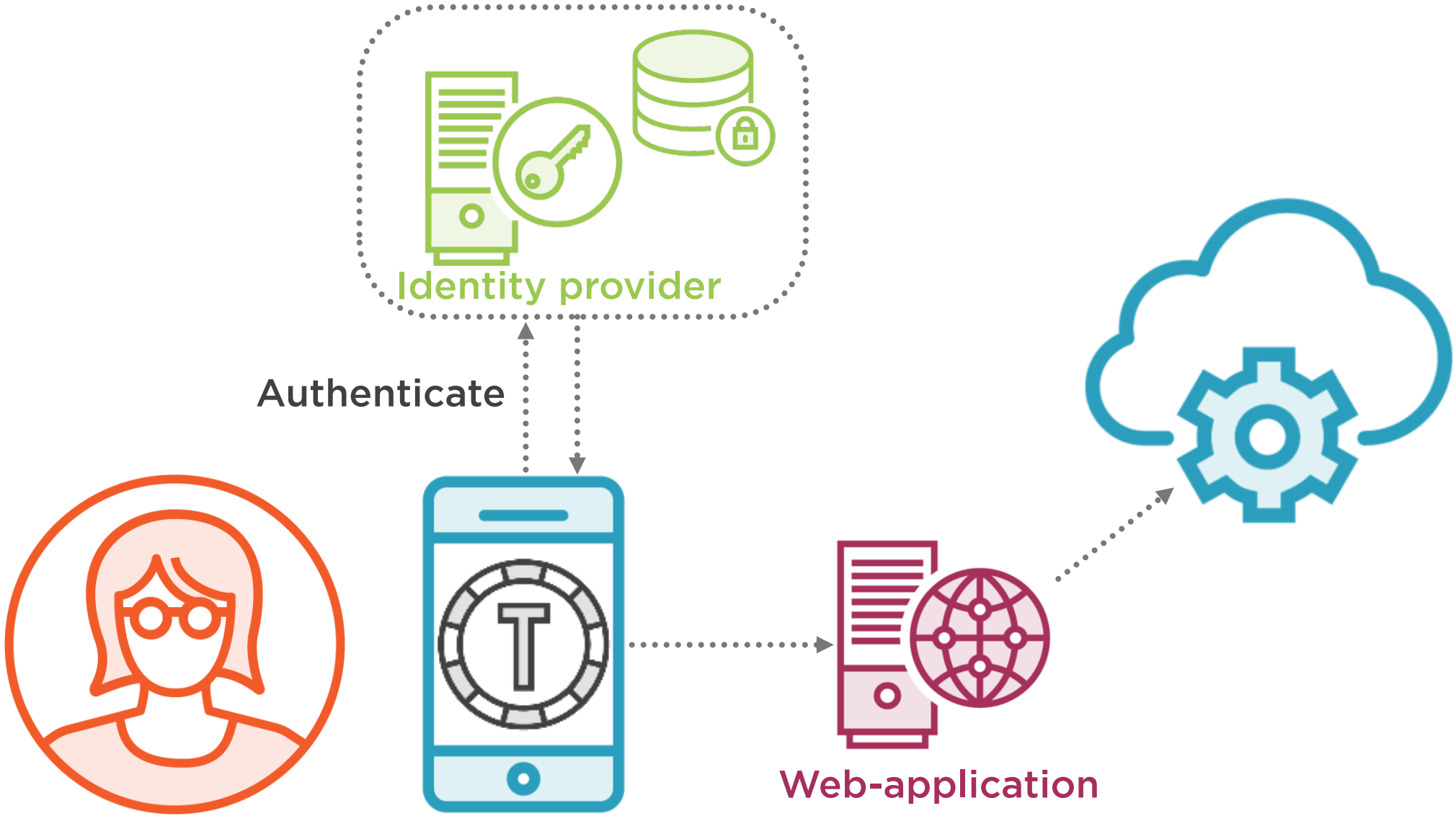
Session or Cookie

Browser local storage

Browser session storage

New Architecture



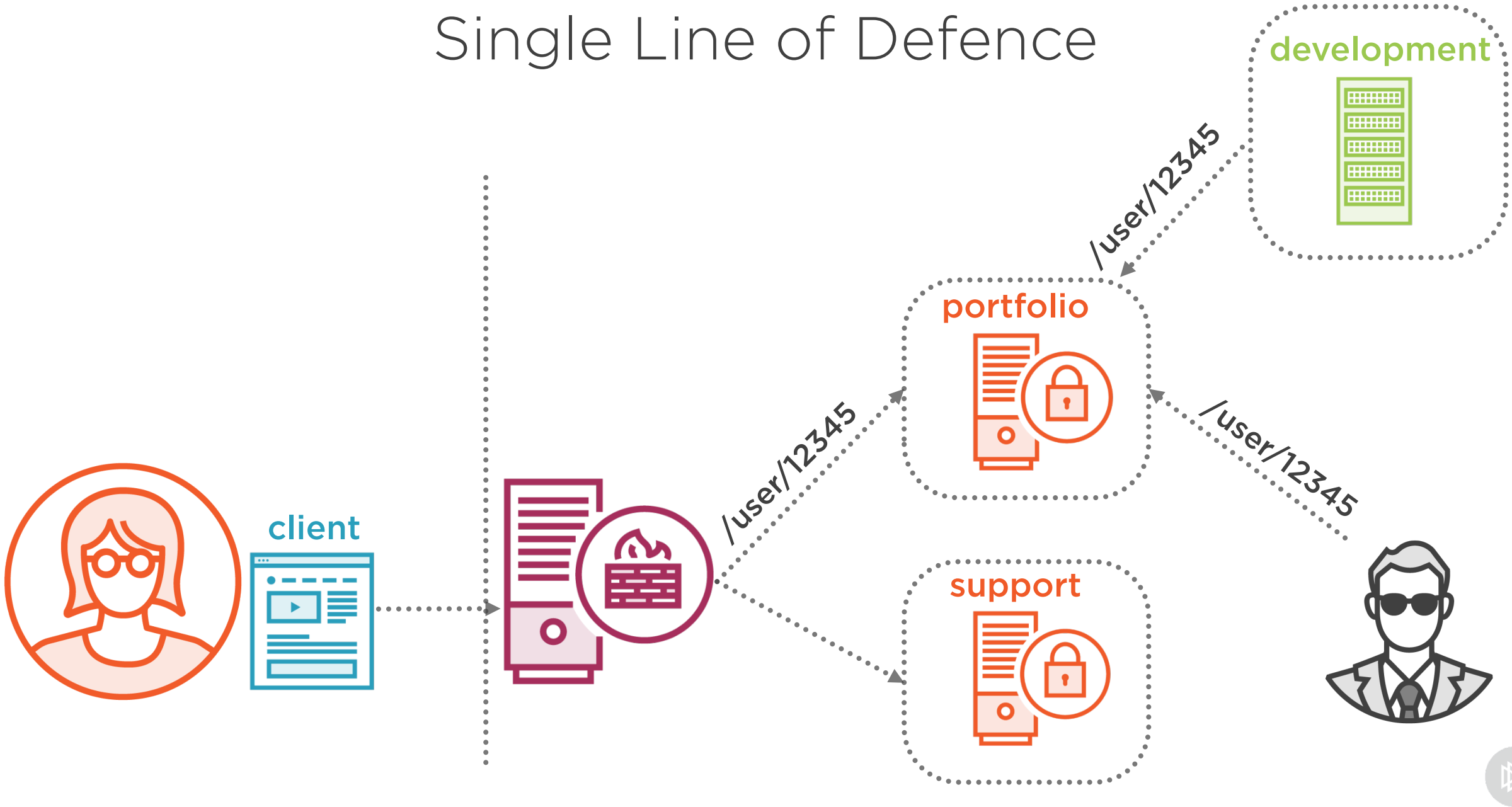

Identity provider
Authenticate
Web-application

The remainder of this module will be using a React version of the web-application

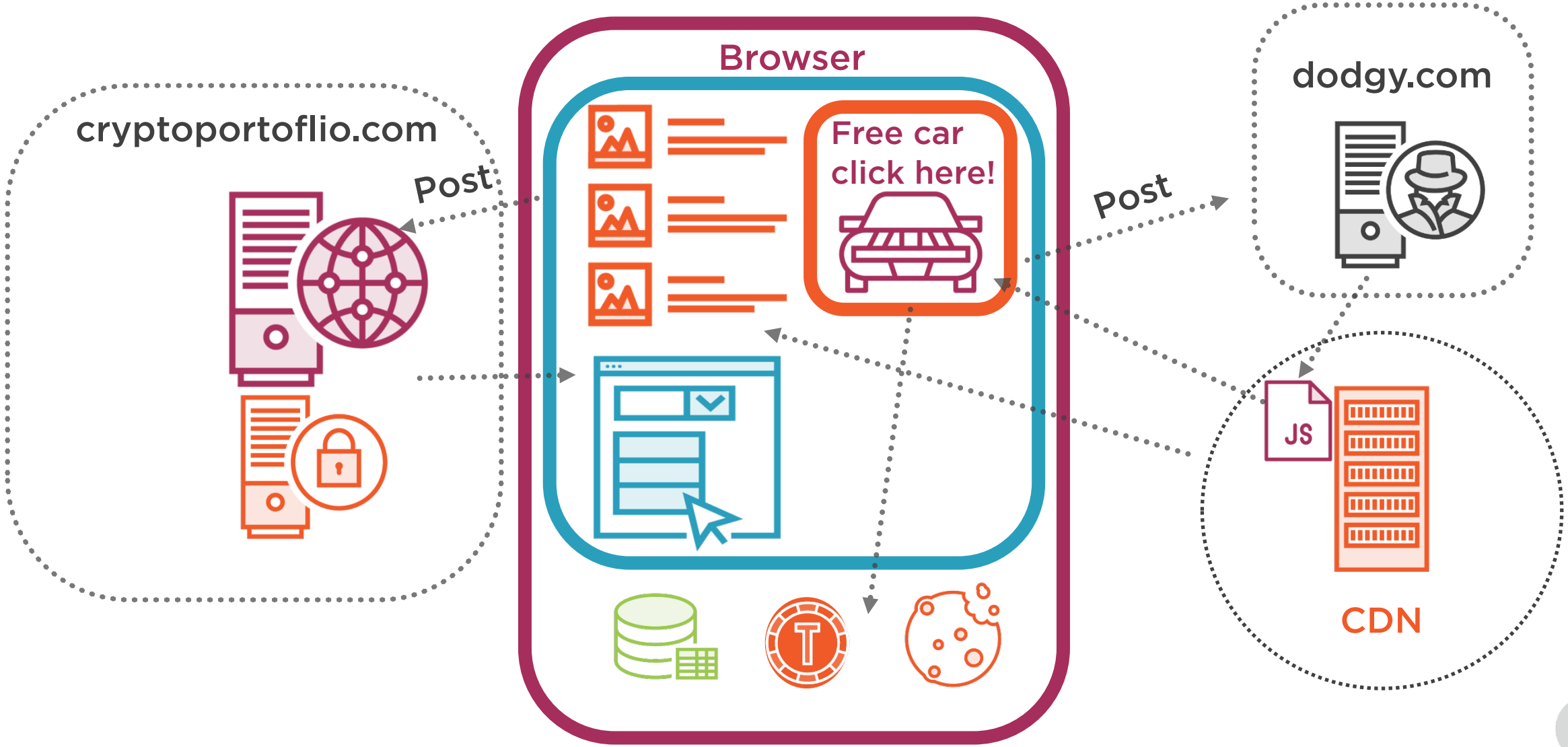
Resource Server

- **Extract the JWT token from the request**
- **Validate the JWT access token**
  - Signature
  - Expiry
  - Issuer and audience

Single Line of Defence

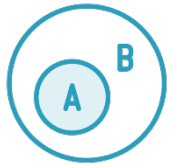


Cross-Origin Resource Sharing (CORS)



CORS is Triggered

A different domain (e.g. crypto.com calls api.com)



A different sub domain (e.g. site at crypto.com calls api.crypto.com)

A different port (e.g. site at crypto.com calls crypto.com:8081)

A different protocol (e.g. http://crypto.com calls https://crypto.com)

CORS

For simple requests: GET, POST or HEAD

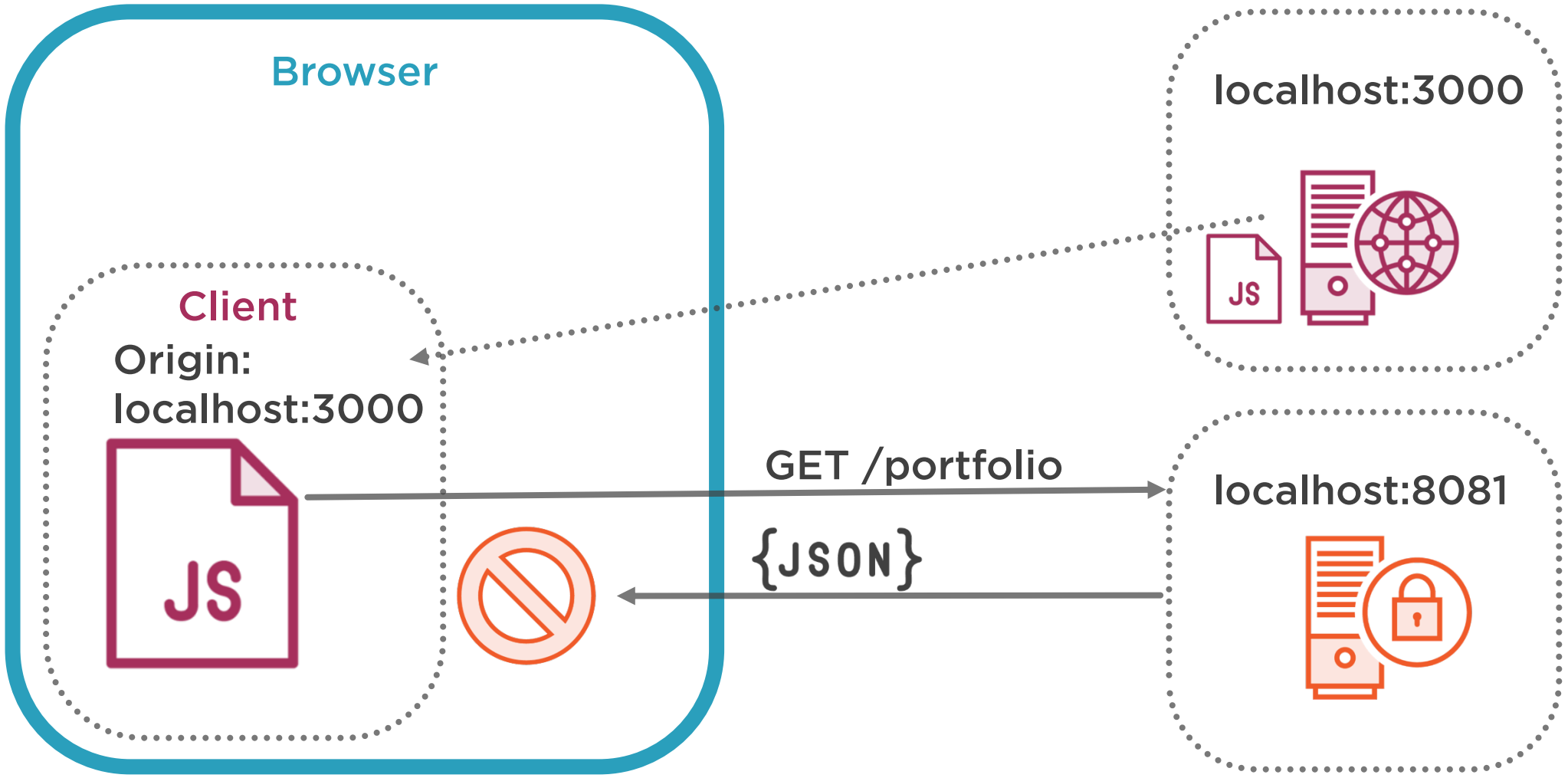
With standard headers:

- Accept, Accept-Language, Content-Language, Content-Type, Save-Data etc

And standard content types:

- application/x-www-form-urlencoded
- multipart/form-data
- text/plain

Cross Origin Simple Requests



Access Control Headers

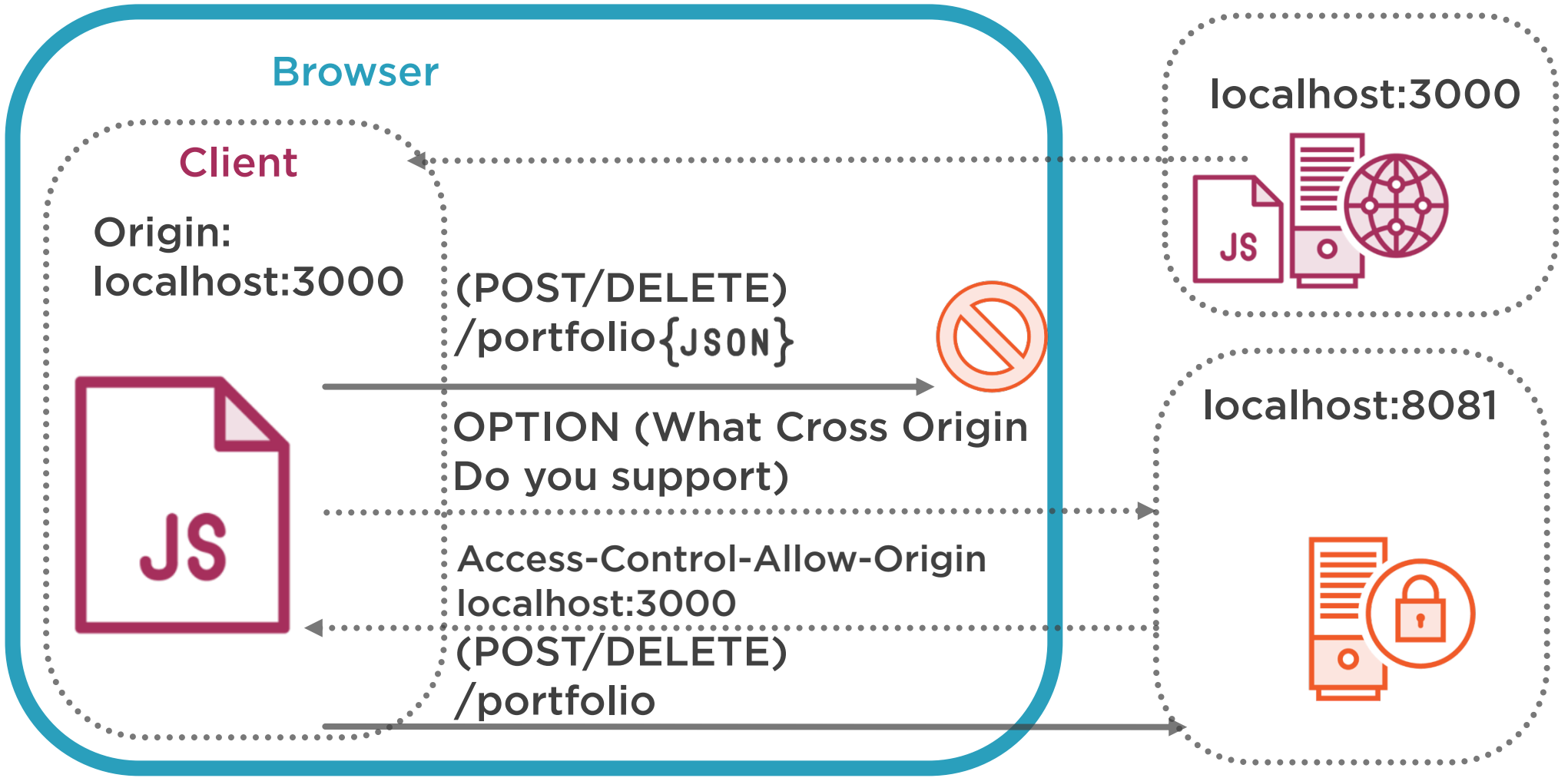
Access-Control-Allow-Origin

Access-Control-Allow-Headers

Access-Control-Allow-Methods

Access-Control-Expose-Headers

Cross Origin Simple Requests



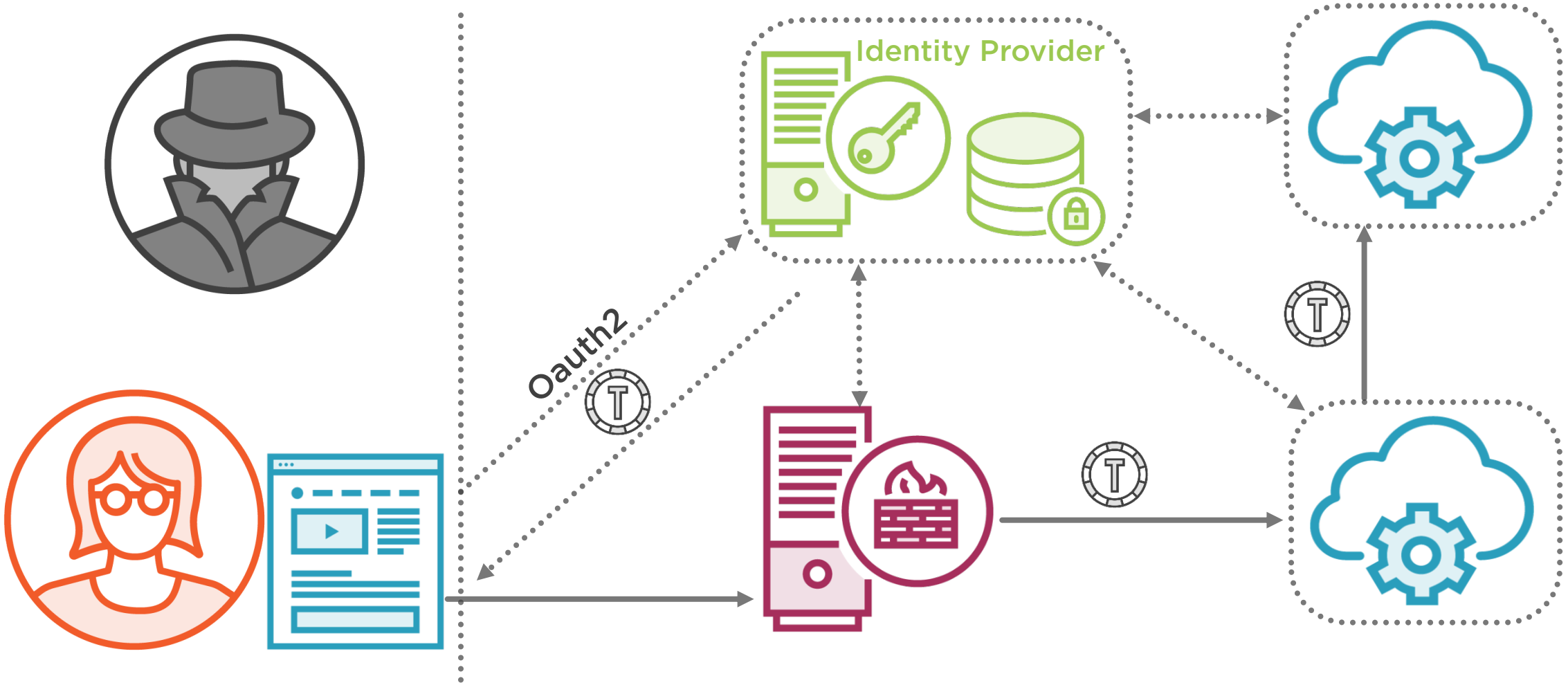
Wrap-Up

What you have learned so far:

- Oauth2 and OIDC for basic MVC and SPA web application configuration.

What's next:

- Oauth2 for machine-machine interactions.
- Microservices security patterns.


Identity Provider
Oauth2