OAuth Notes and Exercises
Neo Lichaba
October 2022

OAuth and OpenID Connect

OktaDev of Nate Barbettini

Industry best practices – Oauth and OpenID Connect

Identity use cases (Pre-2010)

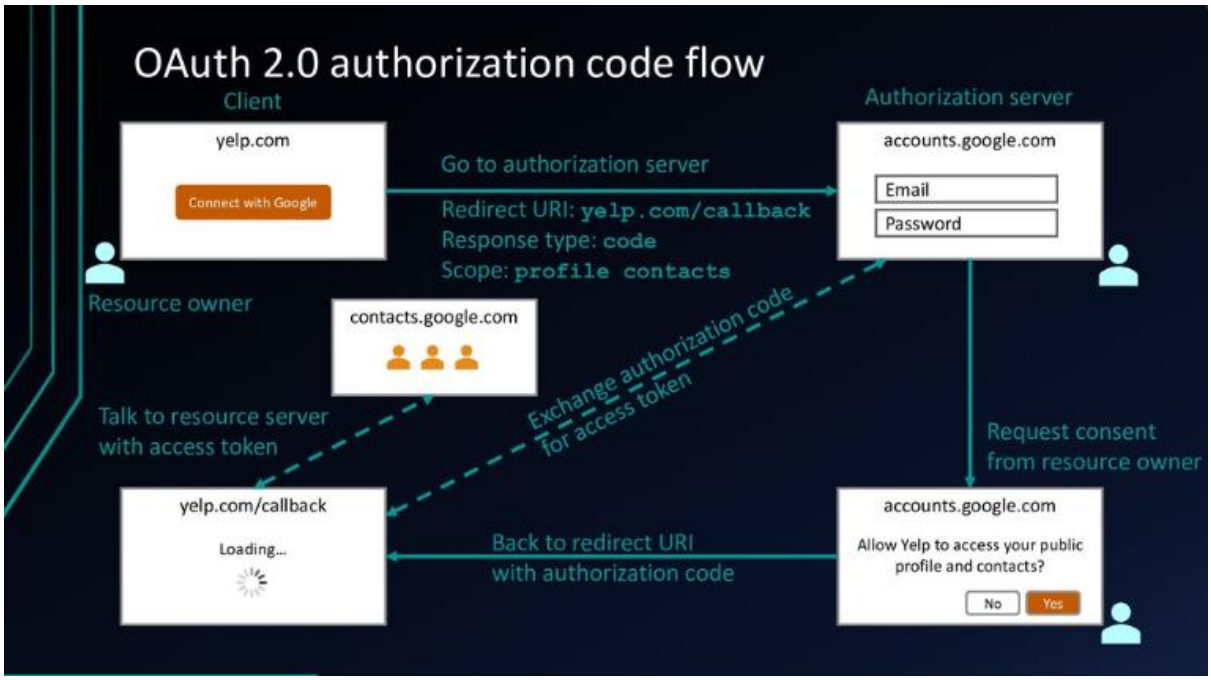
- Simple login (forms and cookies)
- Single sign-on across sites (SAML – employee accounts)
- Mobile app login (???)
- Delegated authorization (???) – Where OAuth protocol began
    - How can a website access data w/o giving password?
        - Eg. Yelp example – Wanted to obtain new users but requested user details (incl. password) to access your contacts
            - Yelp > Connect to Google Account > Prompted to login at Google domain (email and password) > Pop up to grant access to Yelp (yes/no) – Oauth flow > redirected to URI/Callback (e.g. contacts for Google) > back to redirect URI > Exchange authorization code for access token > talk to resource server with access token

OAuth 2.0 Terminology

- Resource Owner – End User
- Client – Application used to access DB (i.e. Yelp)
- Authorization Server – System used to grant access (i.e Google)
- Resource Server – API holds data client wants to obtain (i.e. Google Contacts API)
- Authorization Grant – Proves user gave consent
- Redirect URI – aka Callback/Endpoint
- Access Token – Client requirement. Key used to access data for specific requirement
- Scope – Permissions (e.g. contact.read, contact.write, contact.delete)
- Consent – List of scopes displayed to user whether app given access is allowed/not
- Back channel – highly secure channel (e.g API/HTTP request between servers – passes through SSL encryption, HTTPS etc.)
- Front channel – less secure channel (e.g browser – consists of loop holes

OAuth Notes and Exercises
Neo Lichaba
October 2022

OAuth 2.0 Authorization Code Flow (General)



Authorization Code vs. Access Token

Back channel (highly secure channel) – Backend server to another API

Token exchange happens within backend

Client secret – don't want for it to be within the browser. Could be stolen

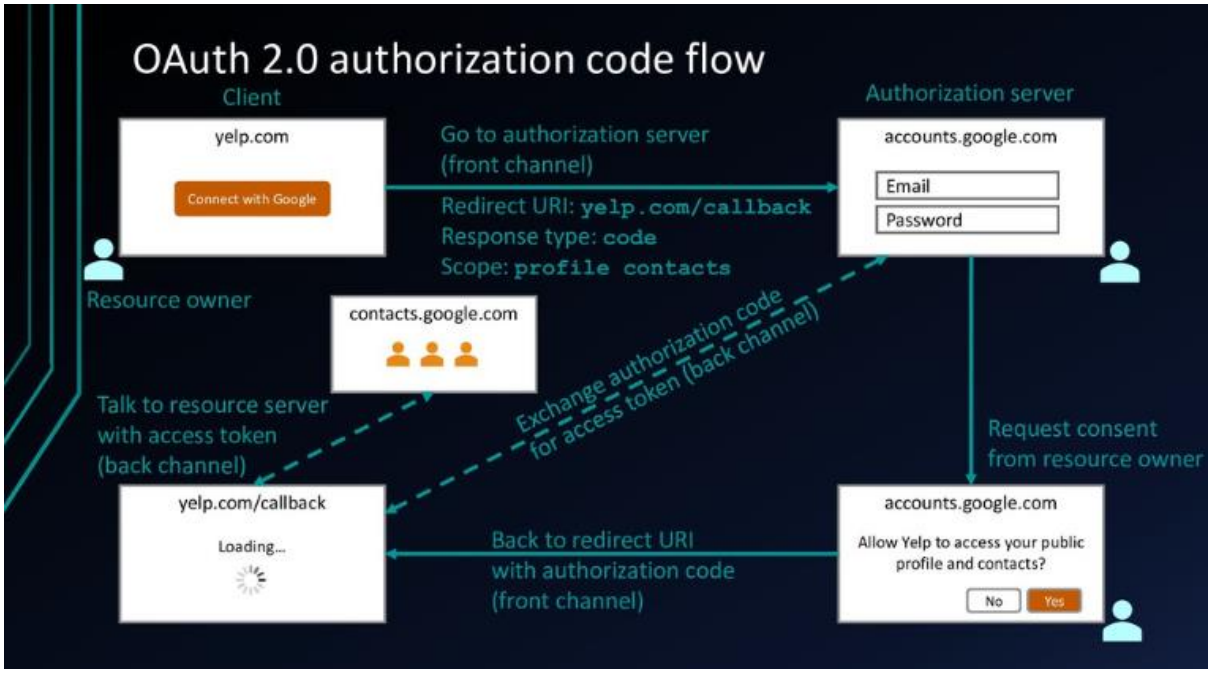
Talk to resource with access token

Front channel (less secure channel) – Browser with loop holes.

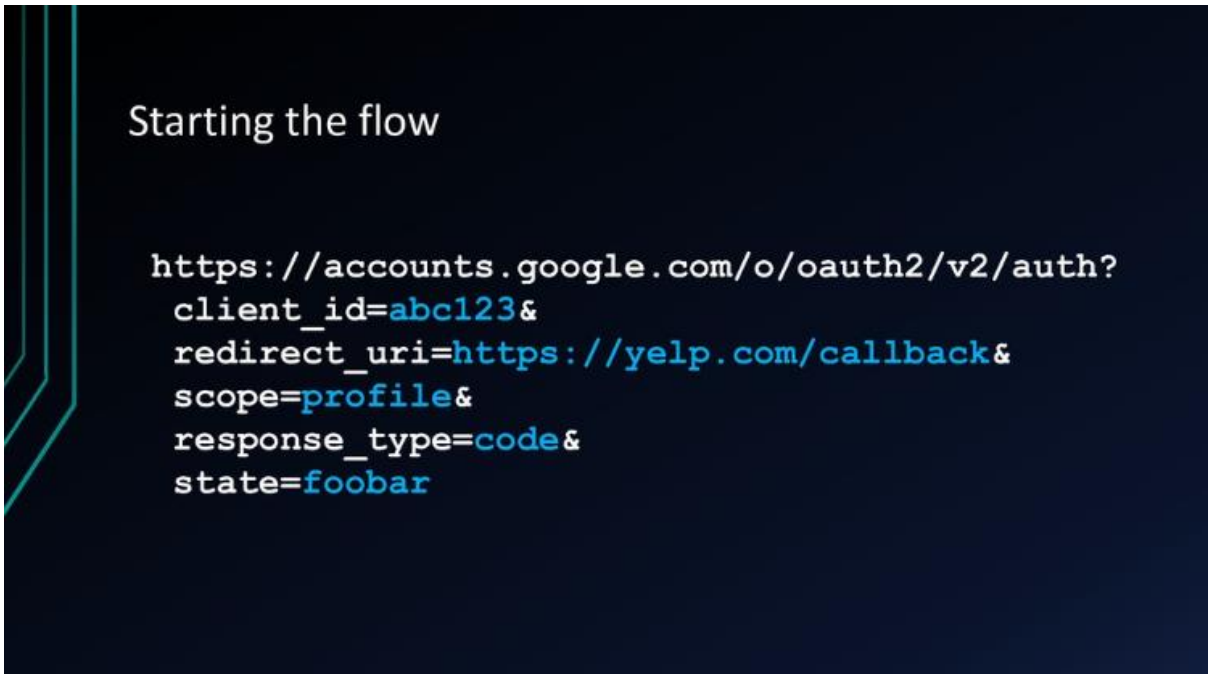
- Redirect URI, response type, scope, auth code (solid lines) – in the query parameters of address bar

Takes both into consideration so that it's highly secure

OAuth Notes and Exercises
Neo Lichaba
October 2022



Starting the Flow



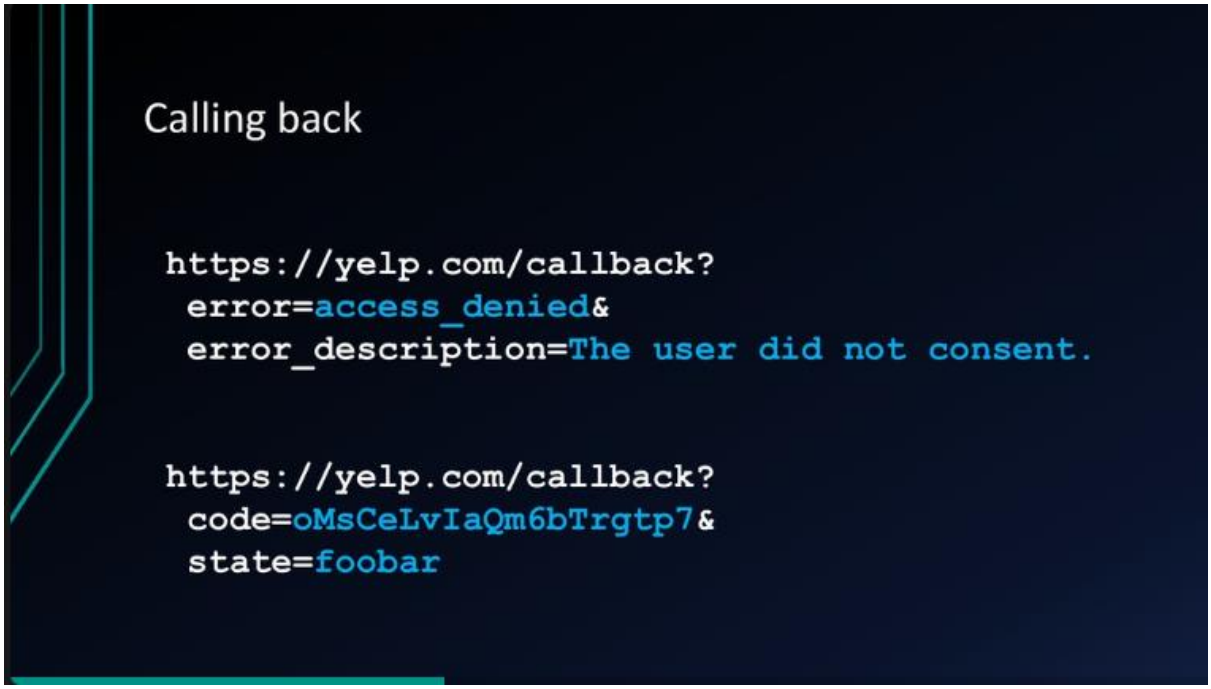
Create a client

Obtain ClientID and Client Secret

Receive authorization code

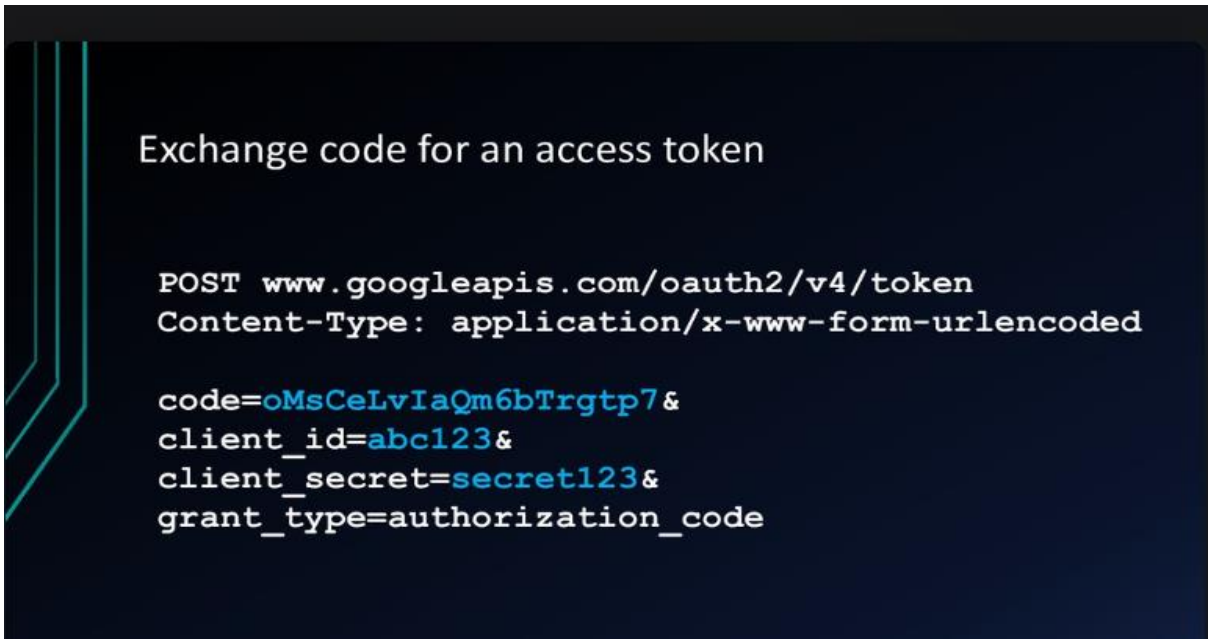
State – text value

OAuth Notes and Exercises
Neo Lichaba
October 2022



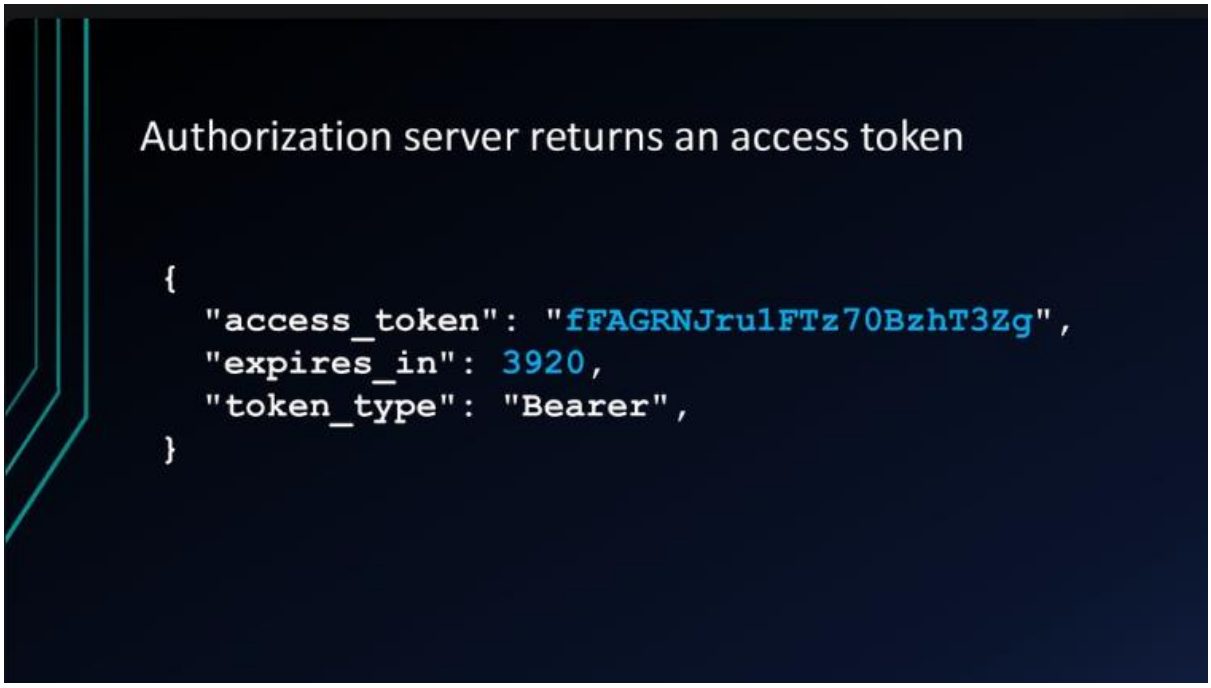
Error received if parameters not set up accordingly

Authorization code returned if all goes accordingly

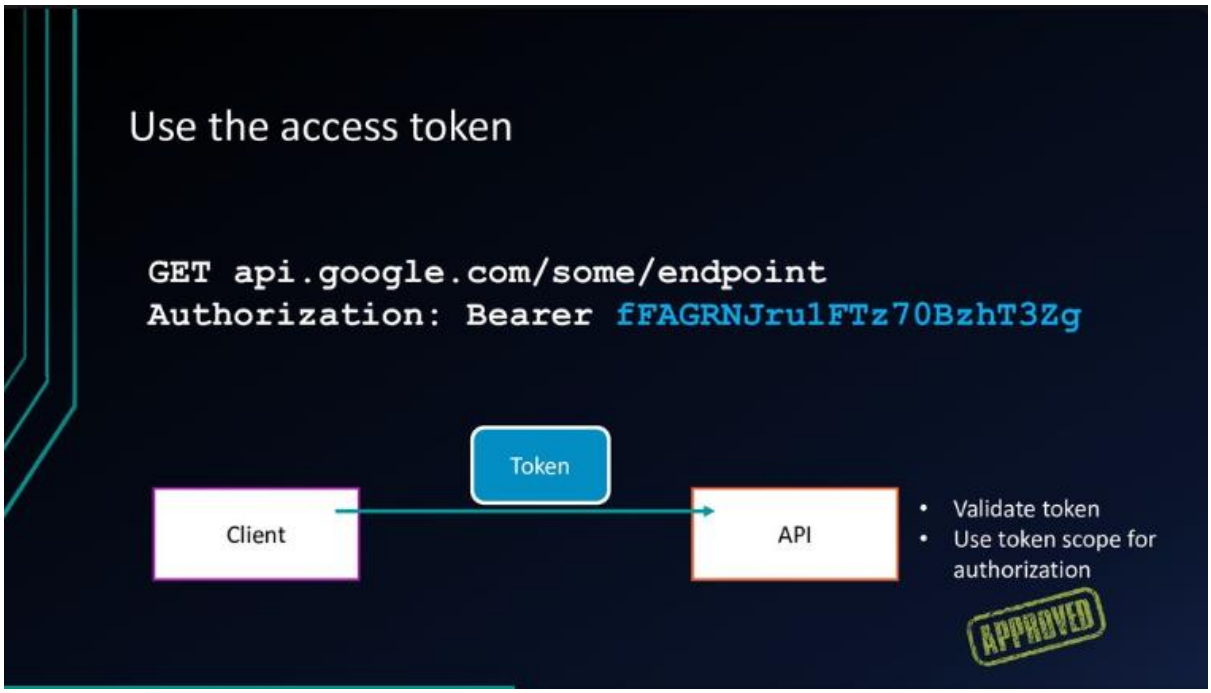


- HTTP Post to Auth server on different route, token endpoint

OAuth Notes and Exercises
Neo Lichaba
October 2022



Auth code – displays number, duration of validity



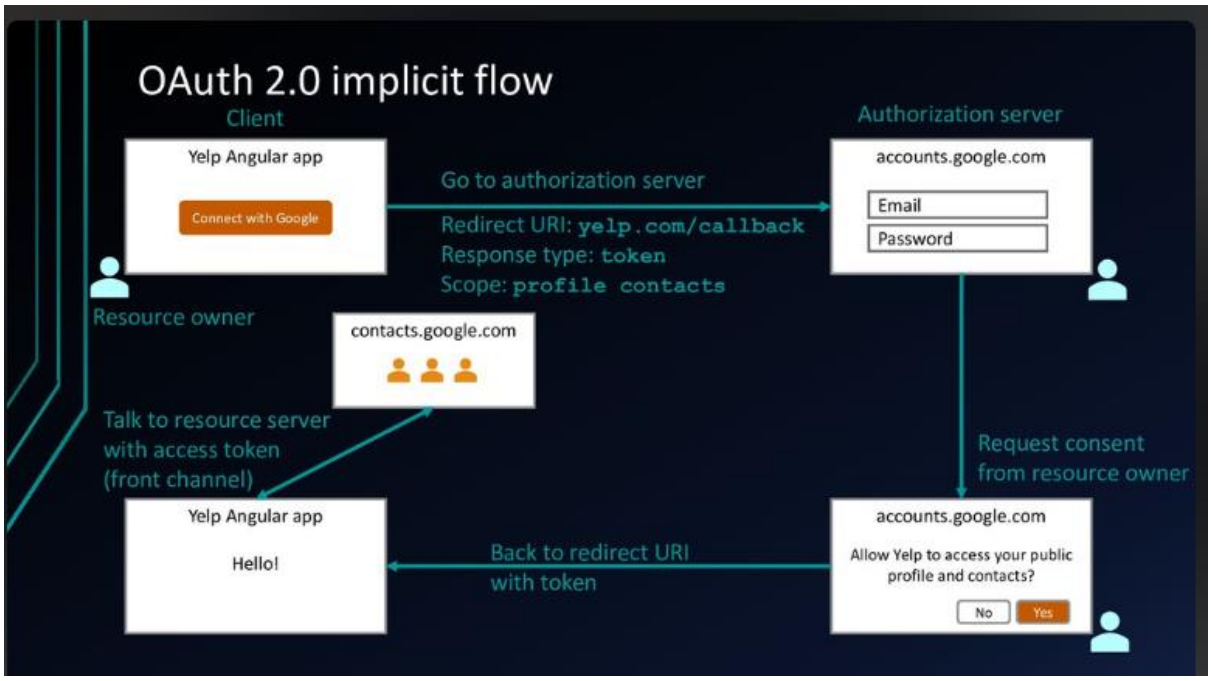
Access token used to make a request

Attach HTTP auth header to request with bearer

OAuth Notes and Exercises
Neo Lichaba
October 2022

OAuth 2.0 Flows

- Auth code (best of both front and back channel)
- Implicit (front channel only) – **single page app in Java**
    - Access token received without authorization code exchange step occurred in the background
- Resource owner password credentials (back channel only)
- Client credentials (back channel only)



Access token issued immediately. Review response type

Identity use cases (pre -2014)

Don't use OAuth for authentication due to:

- No standard way to get user's information.
- Every implementation is a little different
- No common set of scopes

OpenID is for authentication

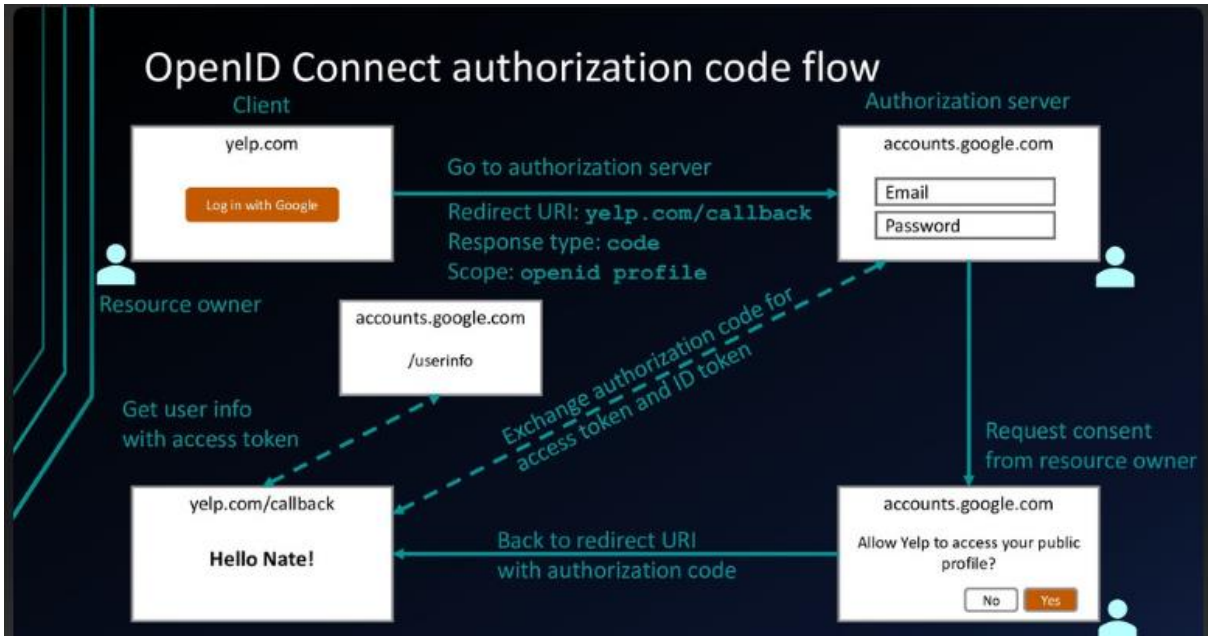
OAuth is for authorization

OAuth Notes and Exercises
Neo Lichaba
October 2022

OpenID add to OAuth:

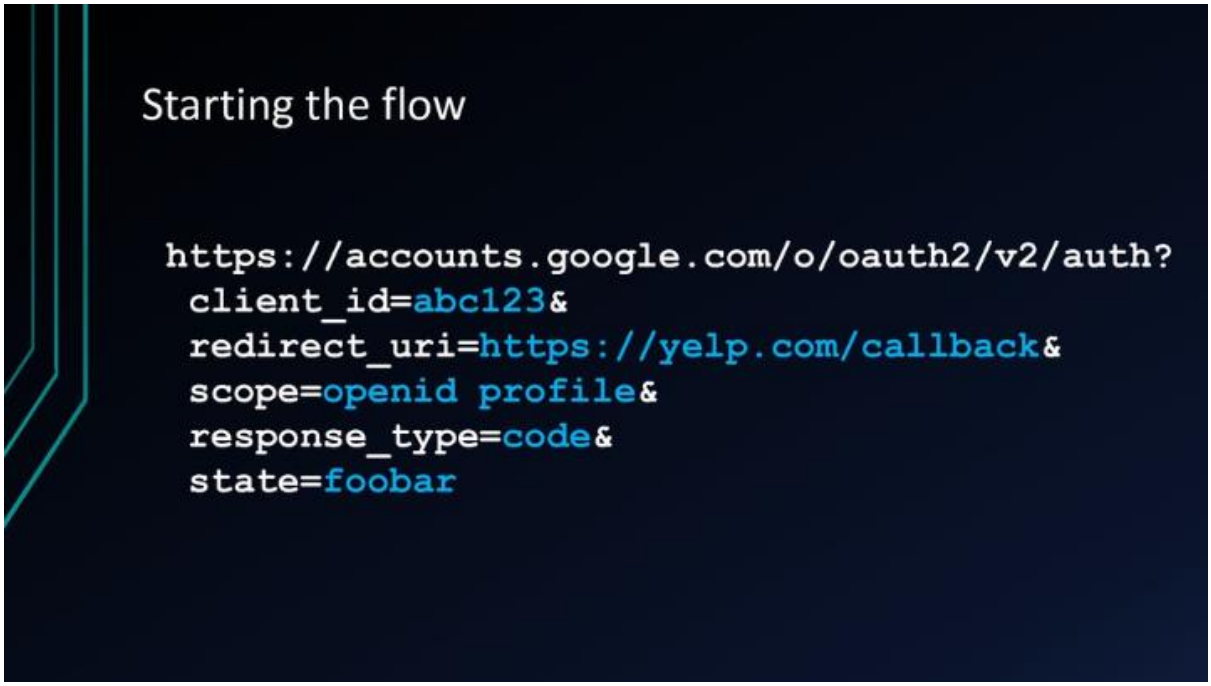
- ID Token
- UserInfo endpoint for getting more user info
- Standard set of scopes
- Standardized implementation

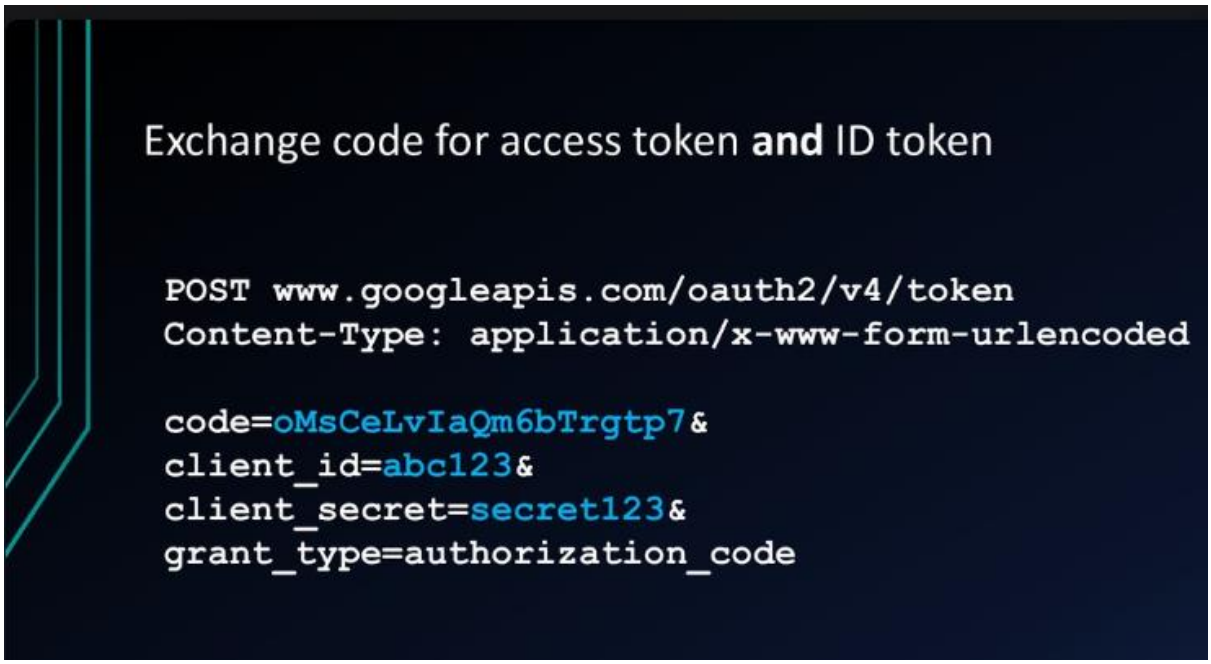
OpenID Connect authorization code flow



- Get back ID token immediately consumed and decoded
- If more info required, use same access token to obtain the info

OAuth Notes and Exercises
Neo Lichaba
October 2022





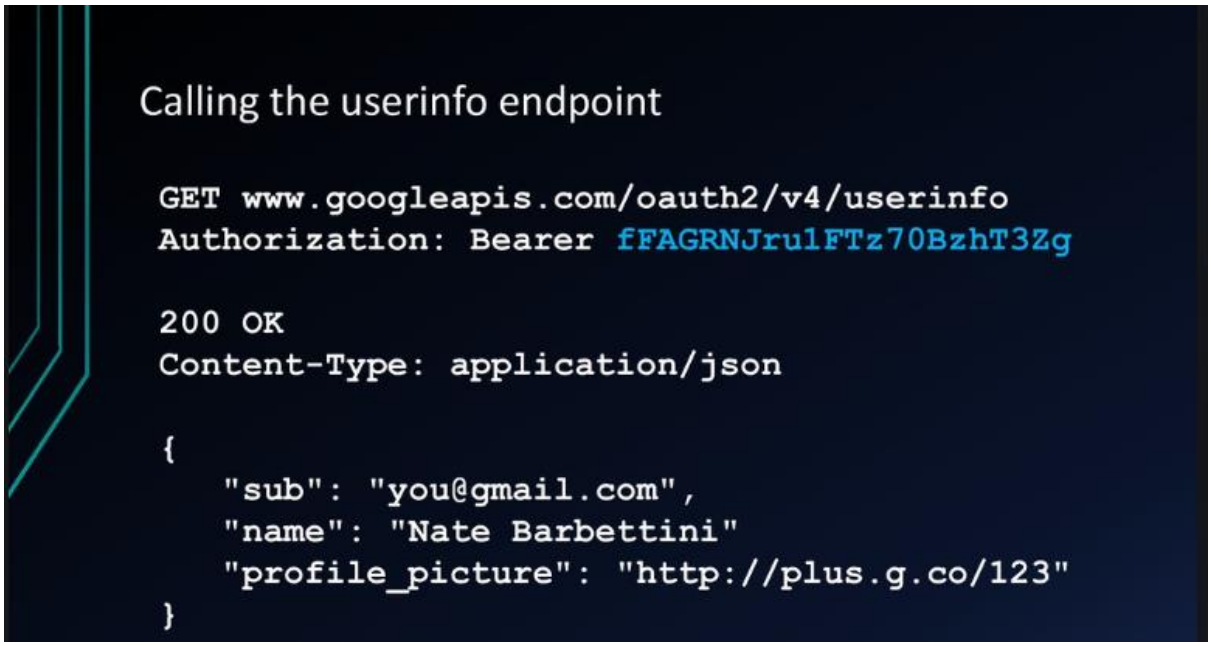
OAuth Notes and Exercises
Neo Lichaba
October 2022

JWT – Json web token

Go into a decoder, information is returned with different properties

Signature determines if ID token hasn't been modified etc. Done independently of auth server

OAuth Notes and Exercises
Neo Lichaba
October 2022



Make use of access token to call userinfo end point

Identity Cases (Today)

Simple Login (OpenID Connect – Authentication)

Single sign on across sites (OpenID Connect – Authentication)

OAuth Notes and Exercises
Neo Lichaba
October 2022

Mobile app login (OpenID Connect – Authentication)

Delegated authorization (OAuth 2.0 – Authorization)

Grant Flow to be used:

- Web application with backend server: authorization code flow
- Native mobile app (iOS and Android): authorization code flow with PKCE
- JS App with API backend: Implicit flow
- Microservices and APIs: client credentials flow

OAuth Notes and Exercises

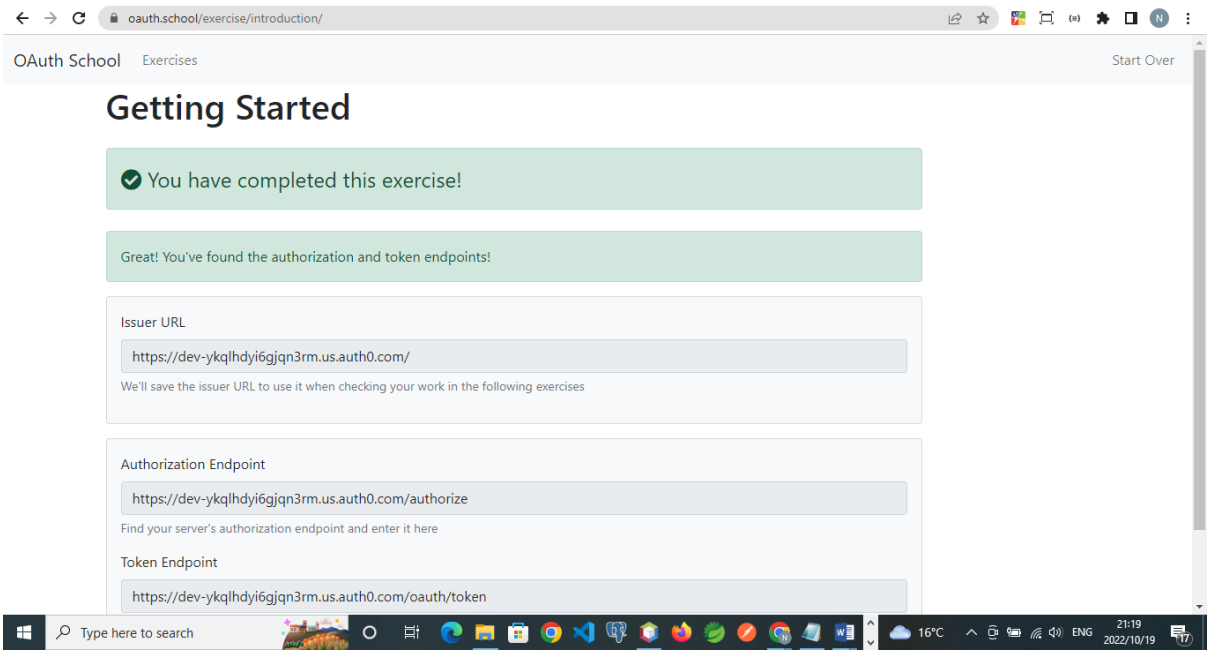
Neo Lichaba

October 2022

1. Getting Started

Lesson Outcome: To sign up for a developer account, enabling one to follow allow with exercises

Create account
Create an API Resource
API resource is protected with AuthO access tokens



Set default audience
End goal is to use the access tokens with your own API, either set a default audience on AuthO tenant or specify audience in each authorization request.
If you wanted to segment your account into multiple API resources, have each OAuth client specify audience for each request

Find Issuer URI – Identifier of your API
Endpoints – OpenID configuration URL

Find Auth and Token End Point

OAuth Notes and Exercises
Neo Lichaba
October 2022

2. OAuth for Web Applications

Lesson Outcome: Obtain an access token using the authorization code flow and PKCE as a confidential client

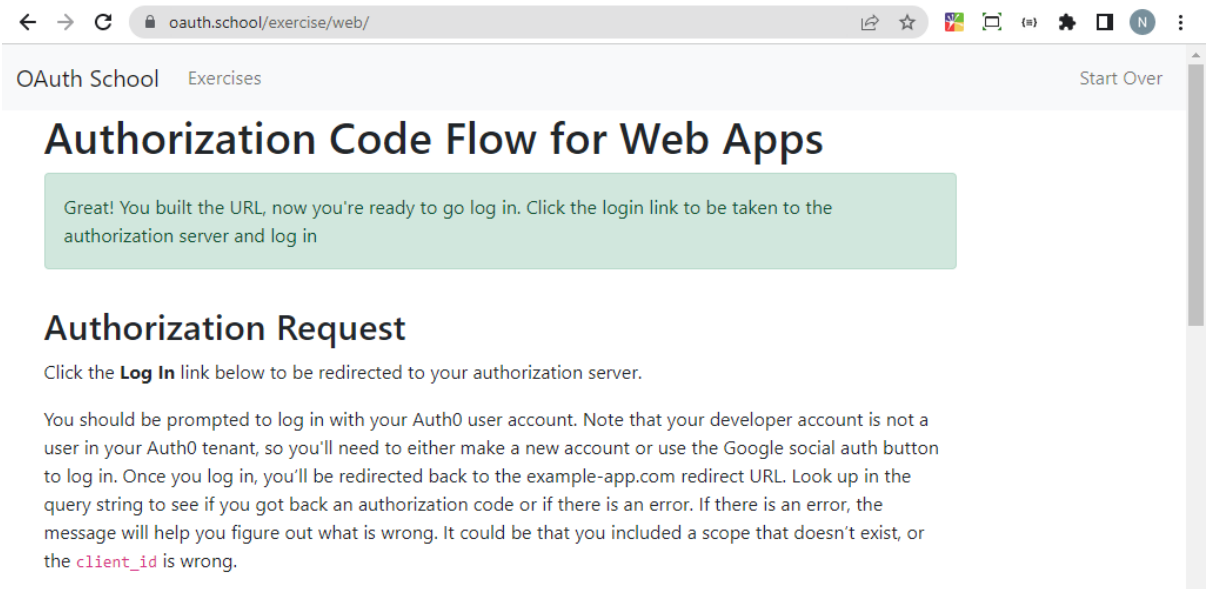
Authorization Code Flow + PKCE = OpenId Connect Flow designed to authenticate native or mobile application users

PKCE, pronounced “pixy” = Proof Key for Code Exchange

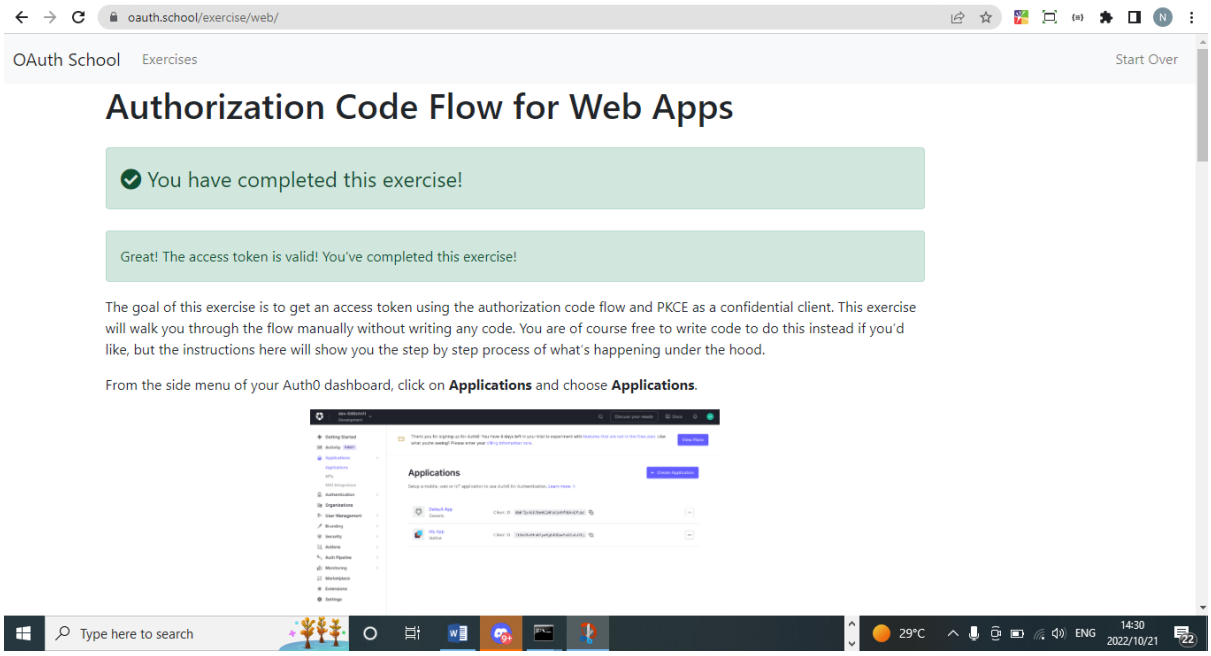
- Users aren’t required to provide a client_secret
- Reduces security risk for native apps
- Client app creates a unique string value, code verifier, hashes and encodes as a code_challlenge.
- Client app initiates Authorization Code flow, hashed code_challenge sent
- PKCE used by mobile apps

Code Challenge creation

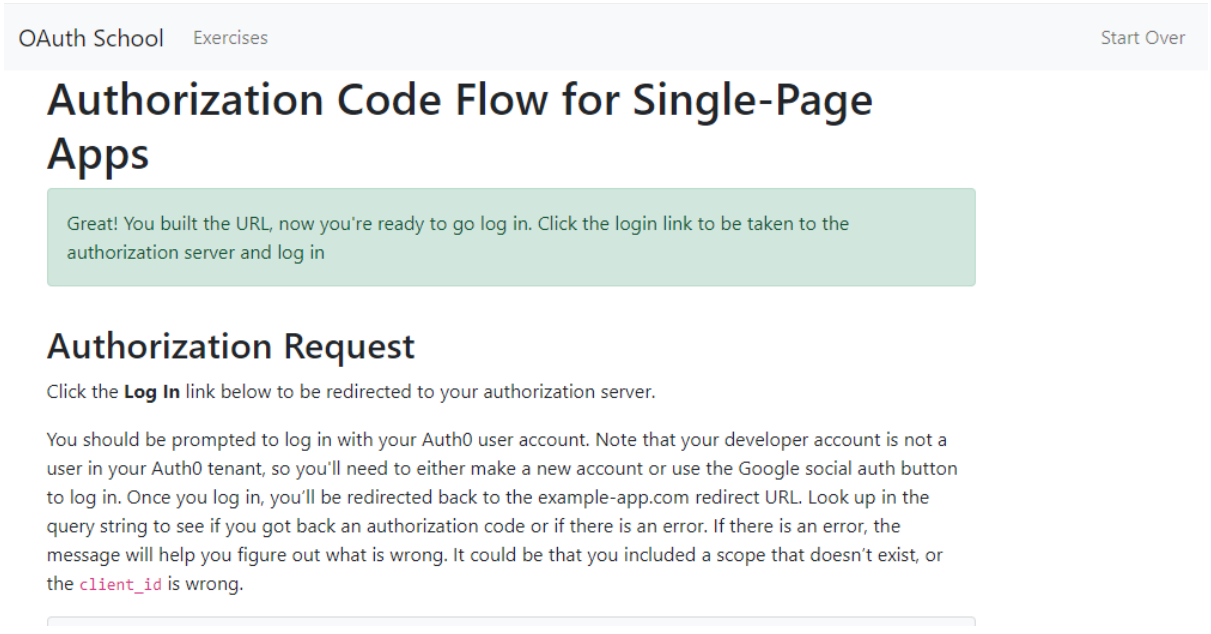
1. Unique string created = code_verifier
2. Base64-URL encoded SHA256 hash of random string generated. Generated from random string



OAuth Notes and Exercises
Neo Lichaba
October 2022



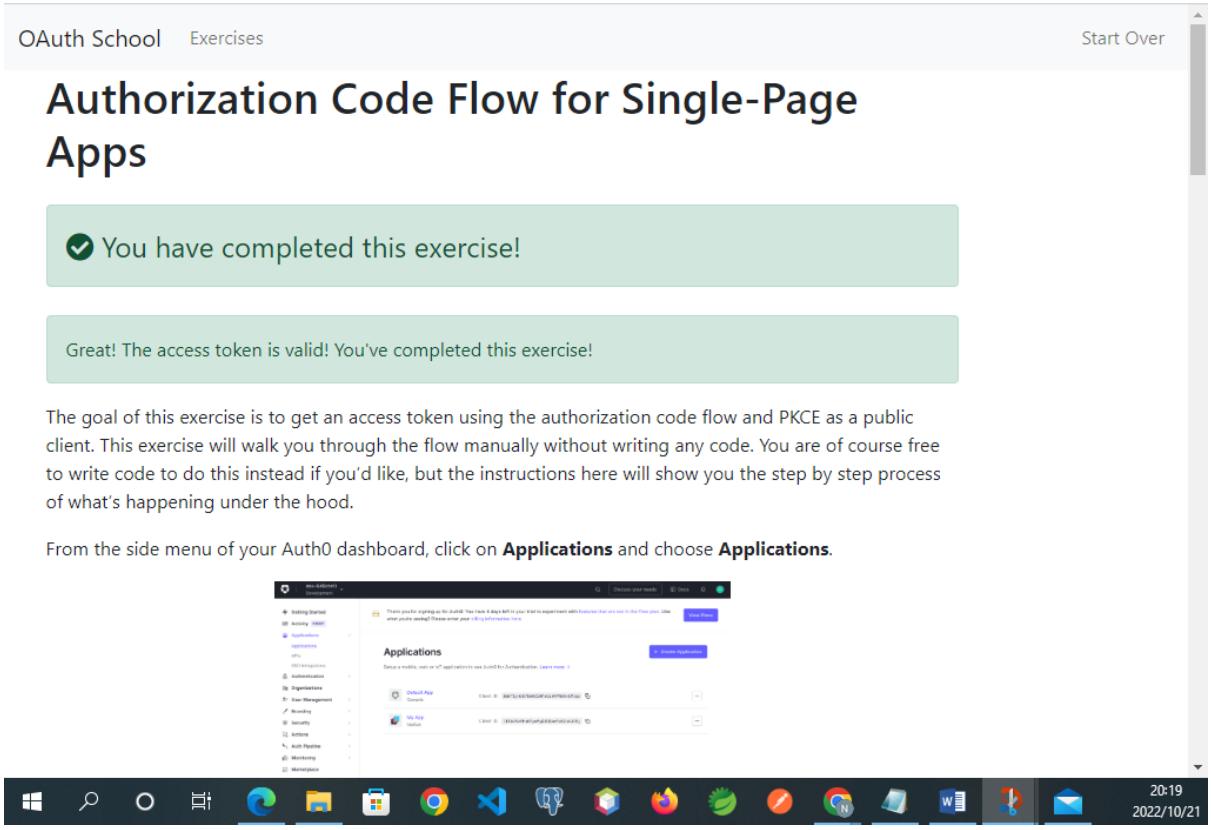
3. Authorization Code Flow for Single-Page Apps



Access token for Single Page apps generated

Get an access token using authorization code flow and PKCE as a public client

OAuth Notes and Exercises
Neo Lichaba
October 2022



4. OAuth for Machine-to-Machine Applications

Client credentials flow for service apps

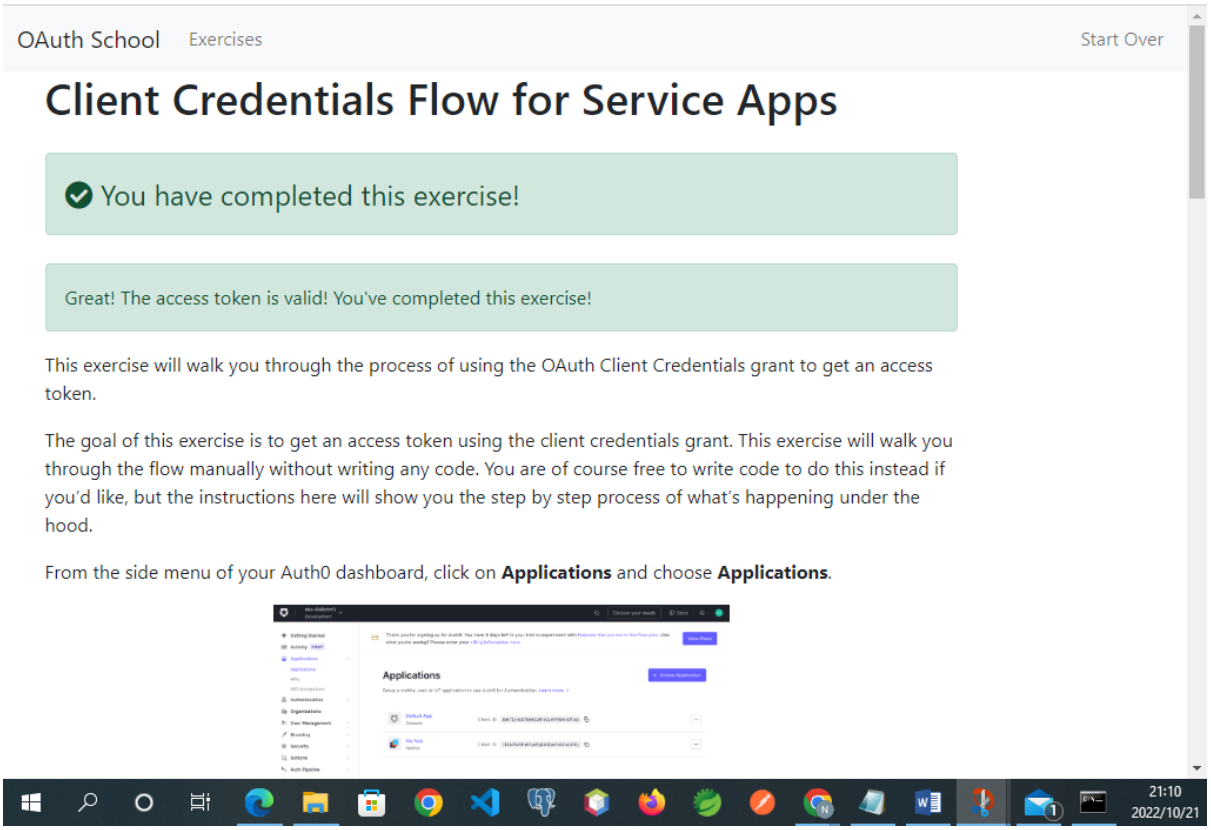
In creating the application, authorize link to API

A Machine to Machine Application represents a program that interacts with an API where there is no user involved. An example would be a server script that would be granted access to consume a Zip Codes API. It's a machine to machine interaction.

Since this is a machine-to-machine flow, there is no user involved in the flow so there is no browser involved either

Application can make a direct request to the authorization server's token endpoint to get an access token.

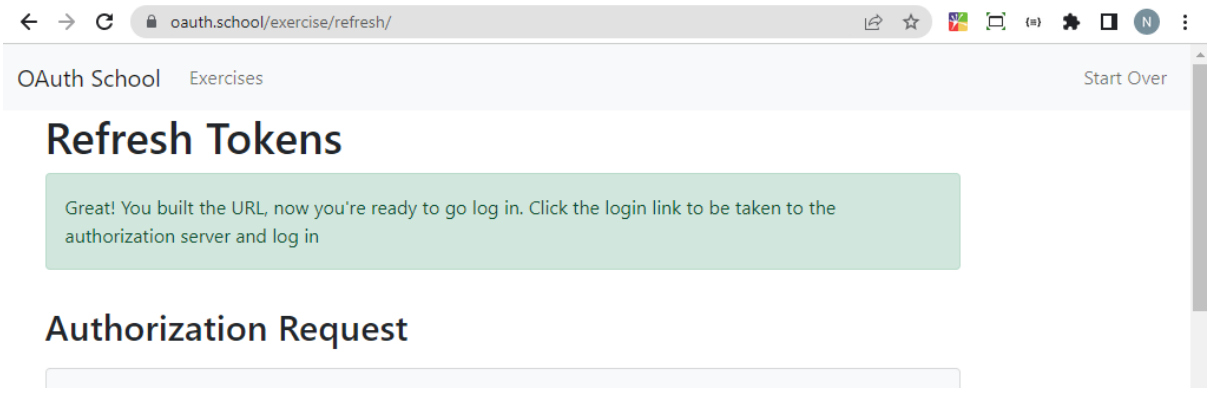
OAuth Notes and Exercises
Neo Lichaba
October 2022



5. **Refresh Tokens**

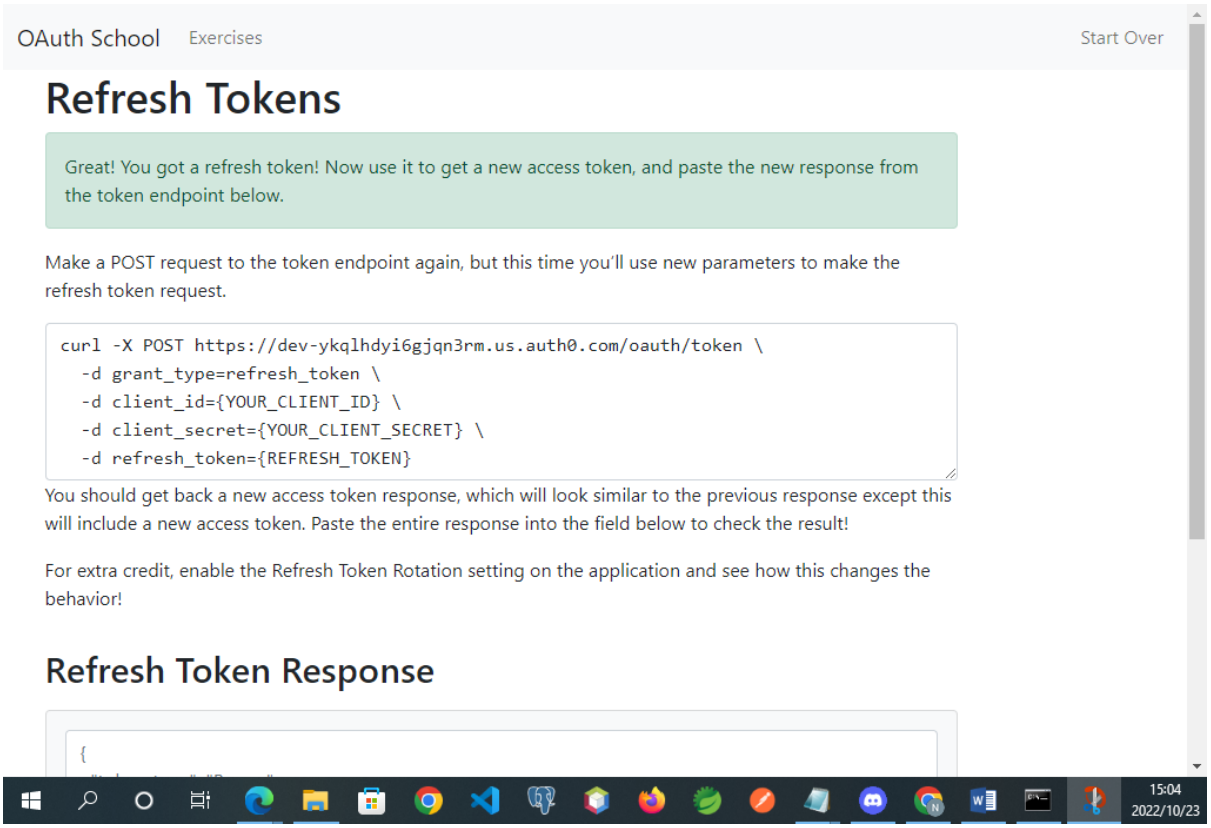
In this exercise you'll learn how to obtain a refresh token and use it to get new access tokens

- Enable the Allow Offline Access Toggle
- Able to request refresh tokens for API
- Scope – "offline_access" added to request



Received an error when creating authorization code. Went into advanced settings of application > Grant Types > Grants (Implicit, Auth Code, Refresh Token, Client Credentials)

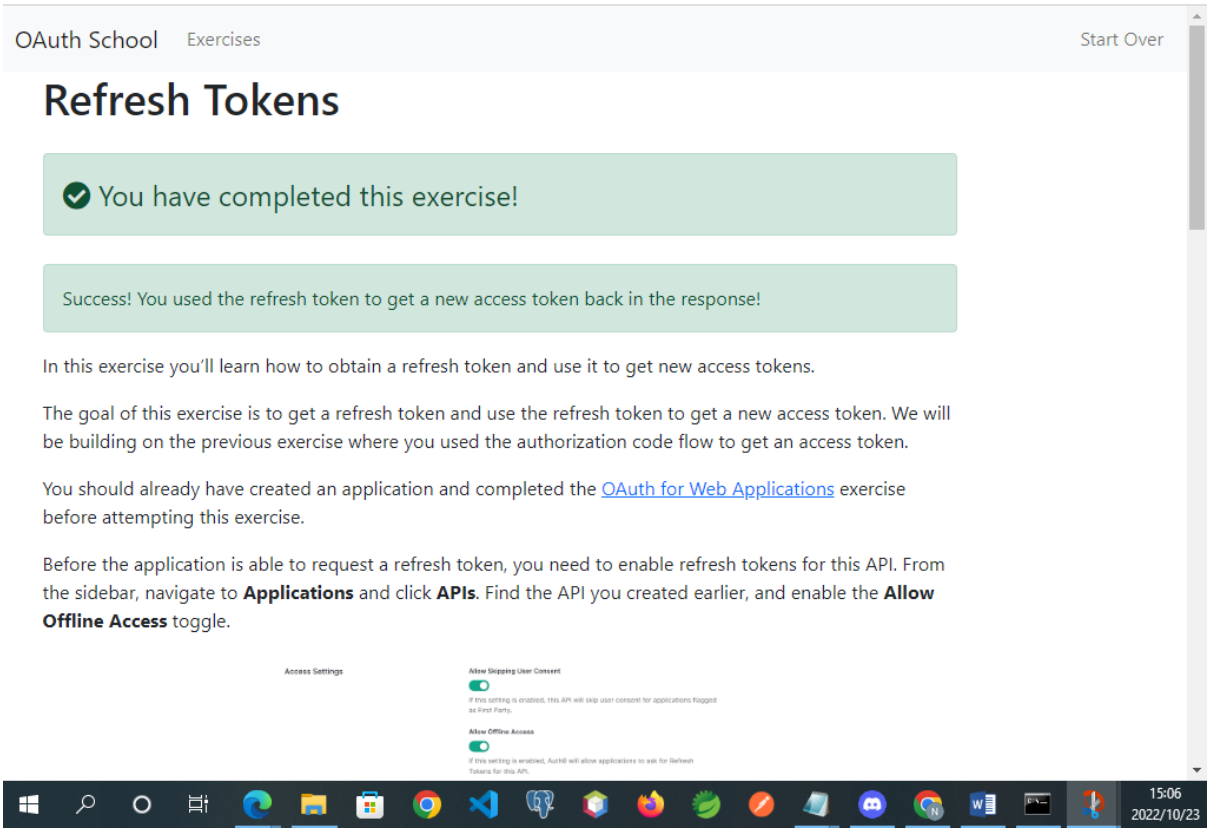
OAuth Notes and Exercises
Neo Lichaba
October 2022



Refresh Token received

Posted another request using cURL, with the refresh token received

OAuth Notes and Exercises
Neo Lichaba
October 2022



6. OpenID Connect

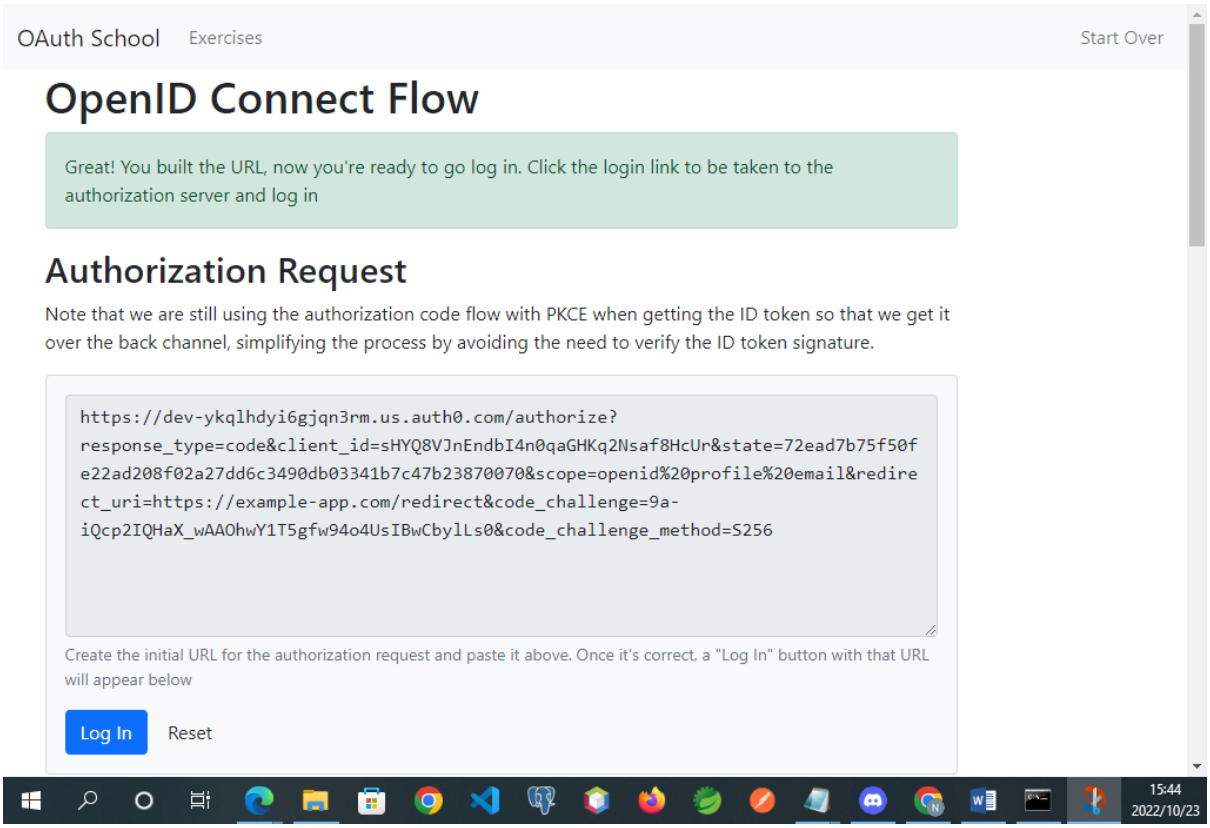
- Request an OpenID Connect ID token and extract the user's information from it using a JWT decoder.
- We will be building on the previous exercise where you used the authorization code flow to get an access token.
- To get an ID token, you need to add the Openid scope to the authorization request. You can also add the profile and email scopes to get more information about the user.

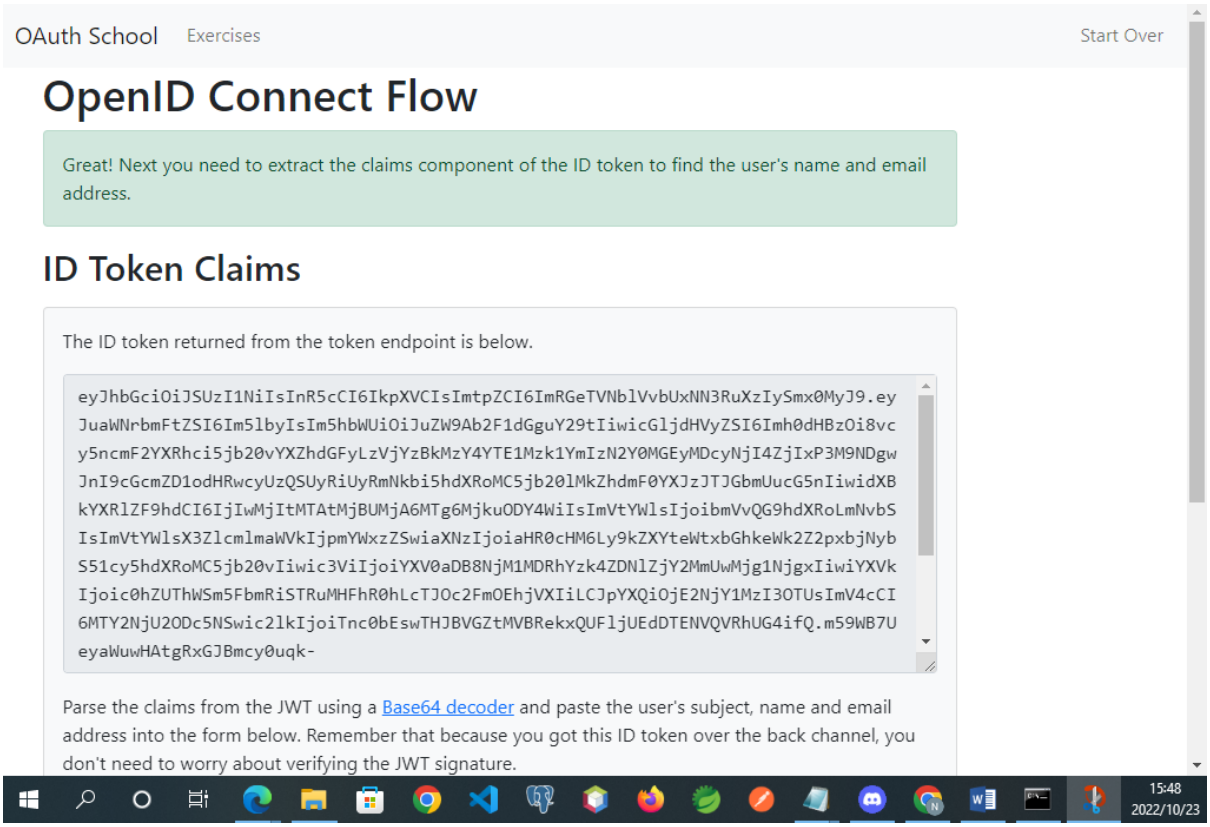
When adding in scopes into authorization request, include "%20" between different scopes. All space delimited

OAuth Notes and Exercises

Neo Lichaba

October 2022

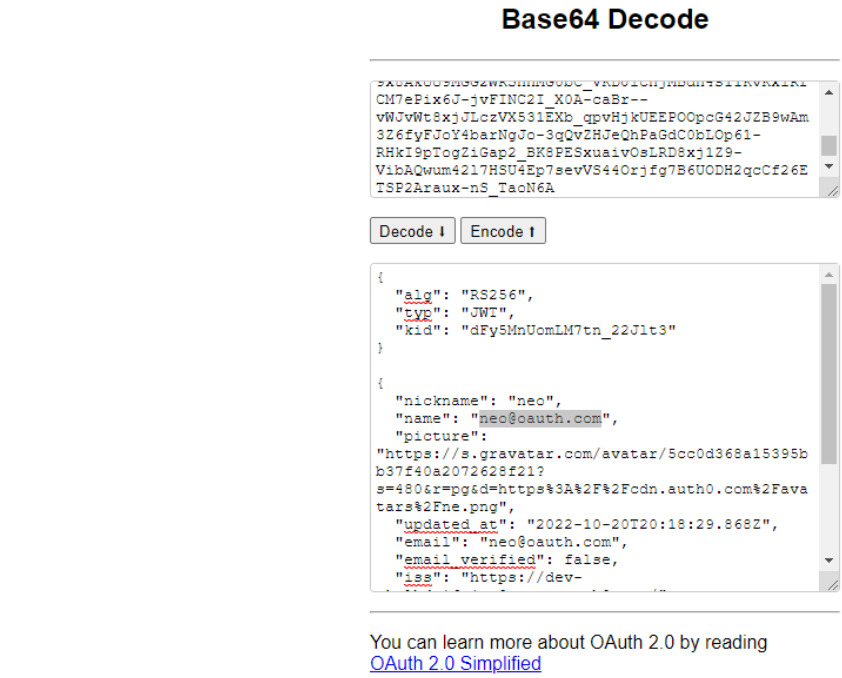




OAuth Notes and Exercises
Neo Lichaba
October 2022

Next Steps: Extract claims component of ID token to find user's name and email address

Decode ID Token



OAuth Notes and Exercises
Neo Lichaba
October 2022