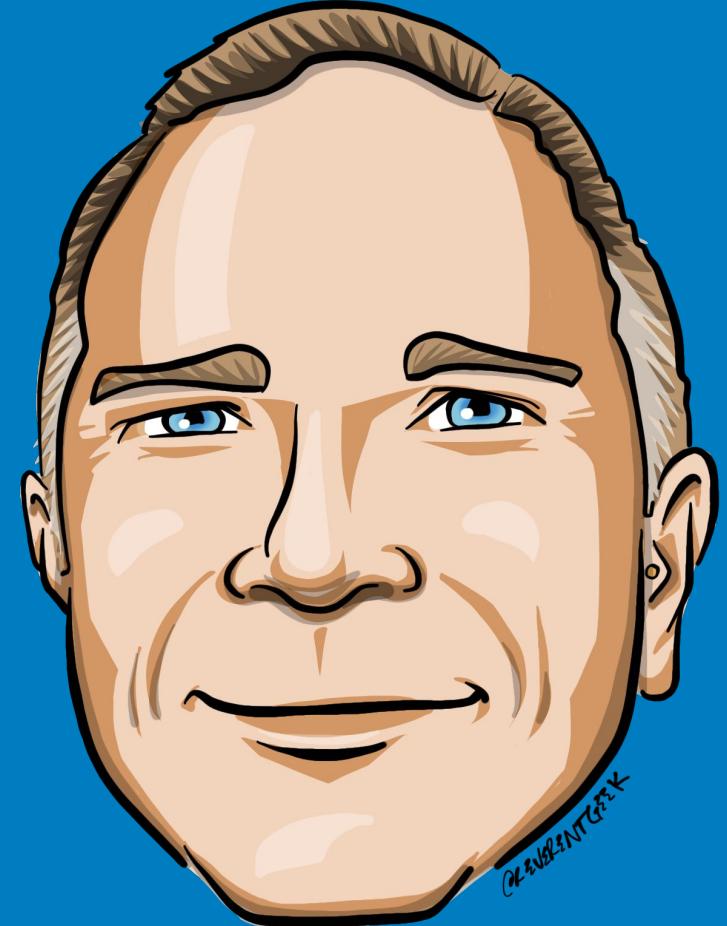


# OAuth and OpenID Connect

(IN PLAIN ENGLISH)

MICAH SILVERMAN  
@AFITNERD  
@OKTADEV



# Identity use cases (circa 2007)

- ★ Simple login – forms and cookies
- ★ Single sign-on across sites – SAML
- ★ Mobile app login – ???
- ★ Delegated authorization – ???

# Don't do it this way!

## Are your friends already on Yelp?

Many of your friends may already be here, now you can find out. Just log in and we'll display all your contacts, and you can select which ones to invite! And don't worry, we don't keep your email password or your friends' addresses. We loathe spam, too.

Your Email Service



Your Email Address

ima.testguy@gmail.com (e.g. bob@gmail.com)

Your Gmail Password

\*\*\*\*\* (The password you use to log into your Gmail email)

Skip this step

**Check Contacts**

# Don't do it this way!

Step 1  
Find Friends

Step 2  
Profile Information

Step 3  
Profile Picture

**Are your friends already on Facebook?**

Many of your friends may already be here. Searching your email account is the fastest way to find your friends on Facebook.

 Gmail

Your Email:

Email Password:

**Find Friends**

Facebook will not store your password.

---

 Yahoo! Find Friends

---

 Windows Live Hotmail Find Friends

---

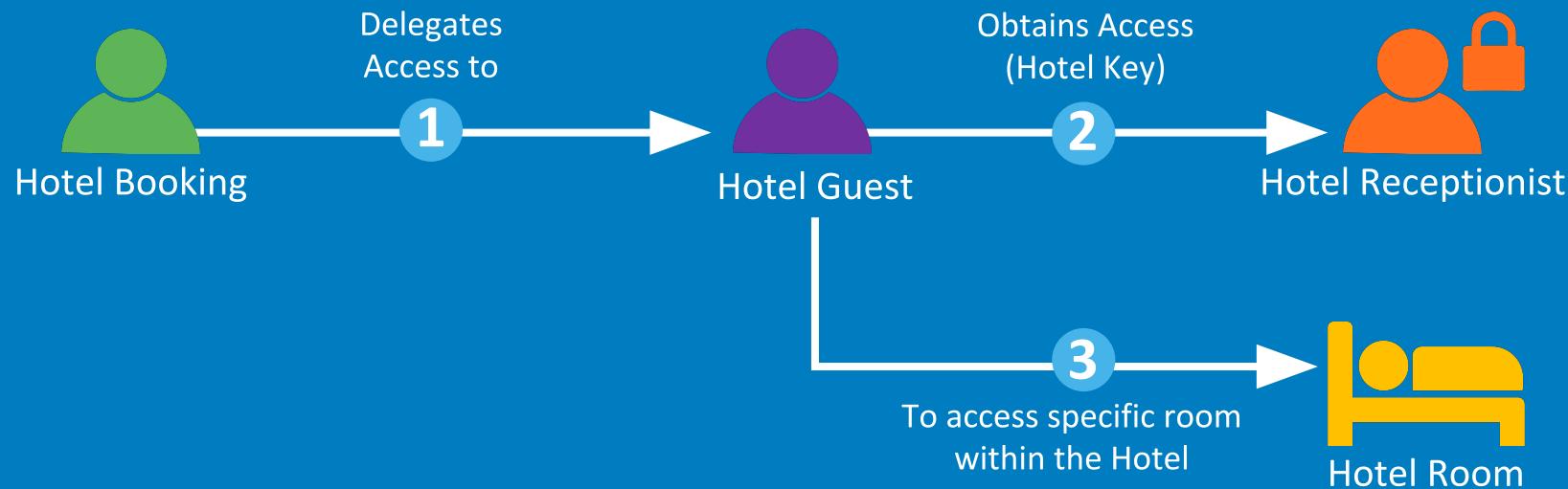
 Other Email Service Find Friends

The delegated authorization problem  
HOW CAN I LET A WEBSITE ACCESS  
MY DATA, WITHOUT GIVING IT MY  
PASSWORD?

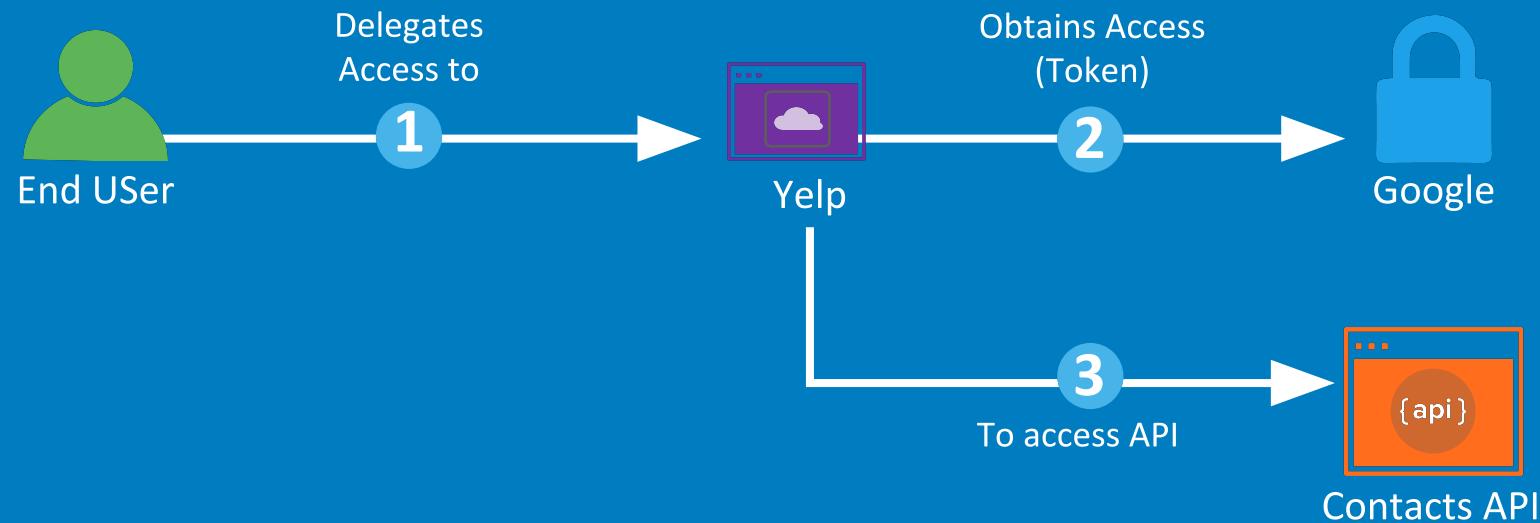
# A lot of confusion around OAuth & OIDC

- ✗ Terminology and jargon
- ✗ Incorrect advice
- ✗ To understand OpenID Connect, you need to learn first about OAuth
- ✗ Hard to find a life-like example

# Hotel Analogy



# OAuth: App Scenario



# OAuth 2.0 Terminology

Hotel	App	OAuth	Description
Hotel Booking	End User	Resource Owner	Wants a Client App to do something on their behalf
Hotel Guest	Yelp	Client Application	Needs authorization to interact with an API on behalf of a user
Hotel Receptionist	Google	Authorization Server	Grants access (in the form of tokens) to an app
Hotel Room	Contacts API	Resource Server	Has an API that an app can use if presented with a token

# Delegated authorization with OAuth 2.0



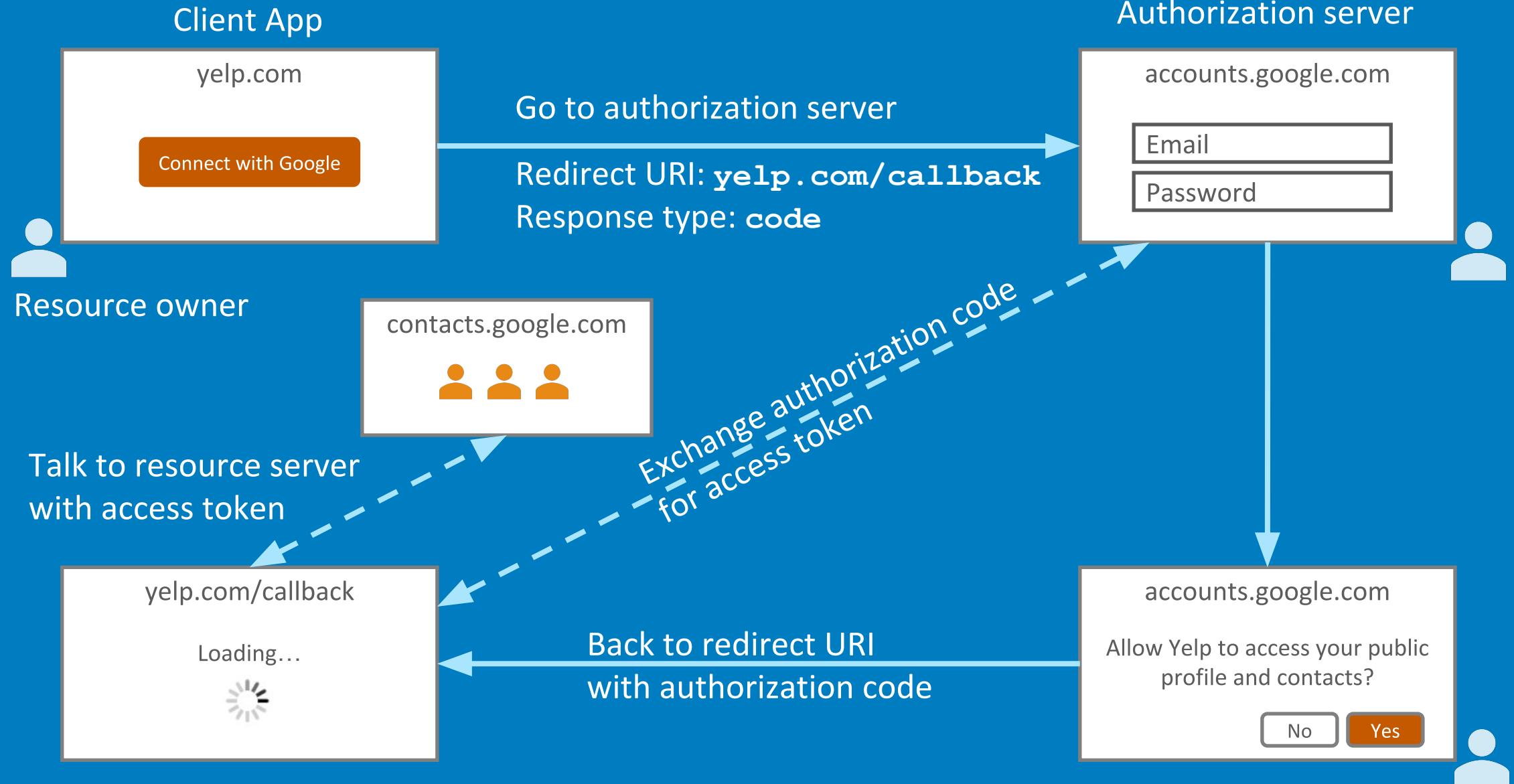
I trust Gmail and I kind of trust Yelp. I want Yelp to have access to my contacts only.

yelp.com

Connect with Google

Live Action  
OAuth Theater!

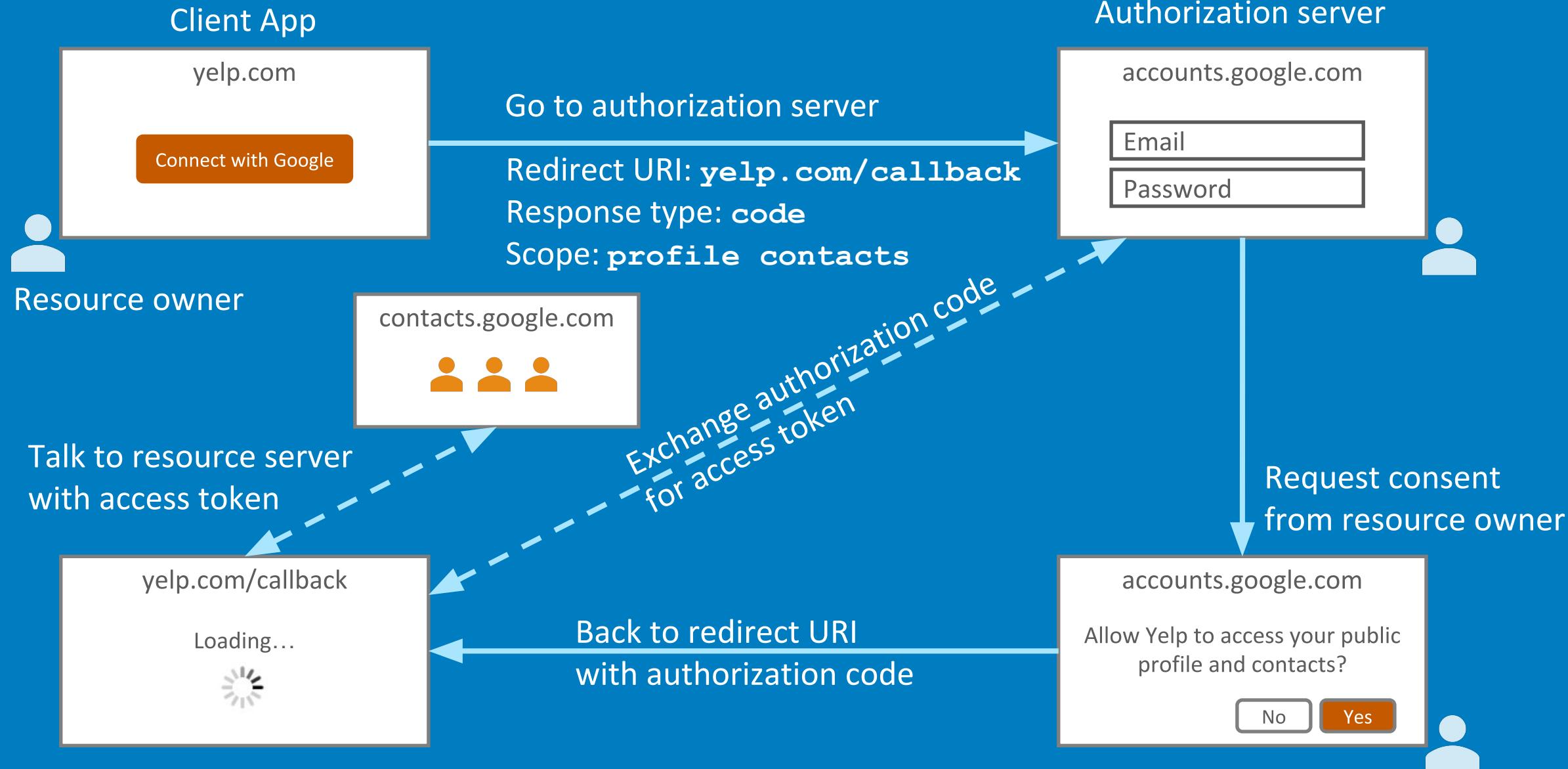
# OAuth 2.0 authorization code flow



# More OAuth 2.0 terminology

- ★ Scope
- ★ Consent

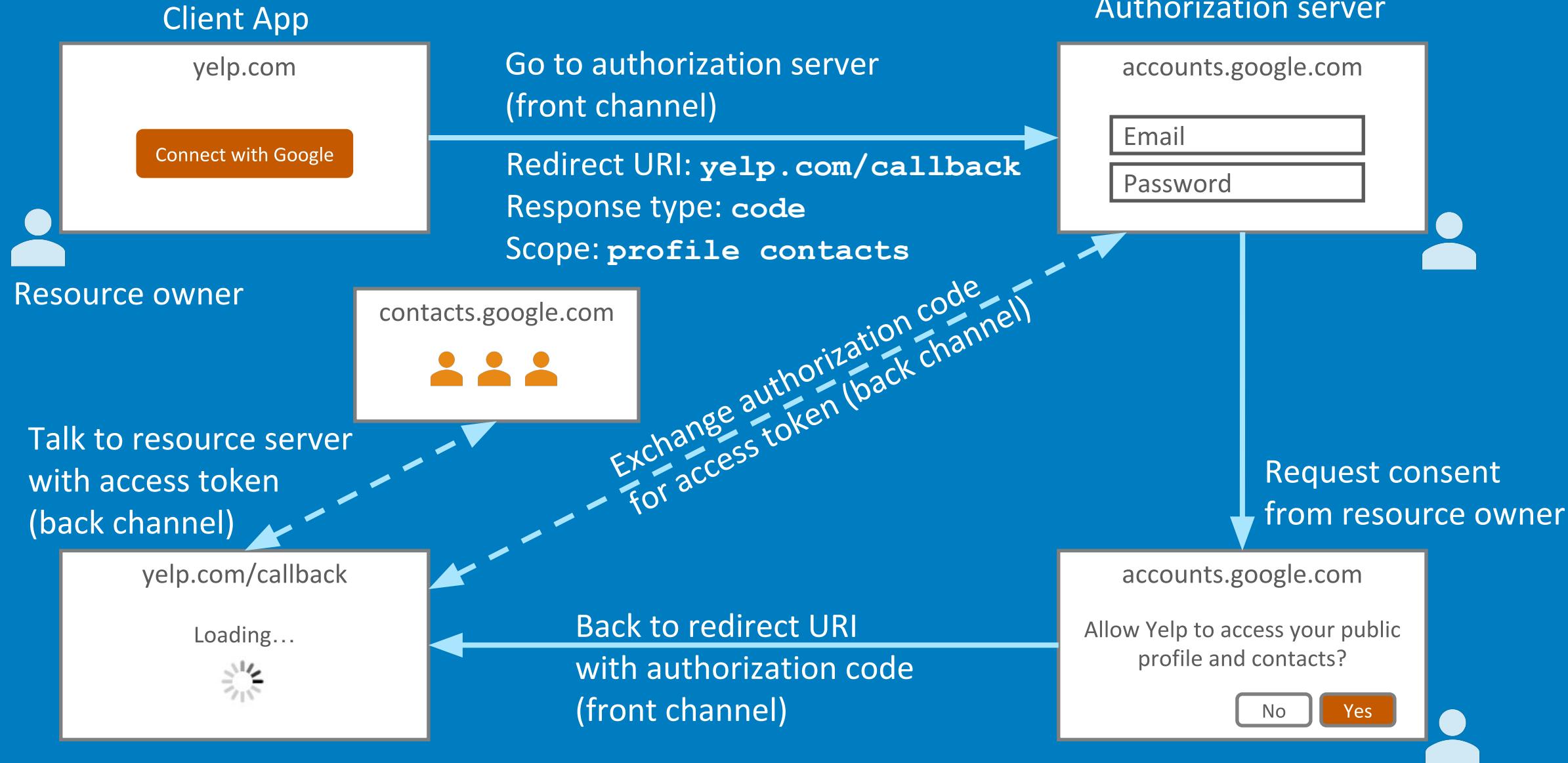
# OAuth 2.0 authorization code flow



# Even more OAuth 2.0 terminology

- ★ Back channel (highly secure channel)
- ★ Front channel (less secure channel)

# OAuth 2.0 authorization code flow



# Starting the flow

```
https://accounts.google.com/o/oauth2/v2/auth?  
client_id=abc123&  
redirect_uri=https://yelp.com/callback&  
scope=profile&  
response_type=code&  
state=foobar
```

# Calling back

`https://yelp.com/callback?`  
`error=access_denied&`  
`error_description=The user did not consent.`

`https://yelp.com/callback?`  
`code=oMsCeLvIaQm6bTrgtp7&`  
`state=foobar`

# Exchange code for an access token

POST www.googleapis.com/oauth2/v4/token

Content-Type: application/x-www-form-urlencoded

code=oMsCeLvIaQm6bTrgtp7&  
client\_id=abc123&  
client\_secret=secret123&  
grant\_type=authorization\_code

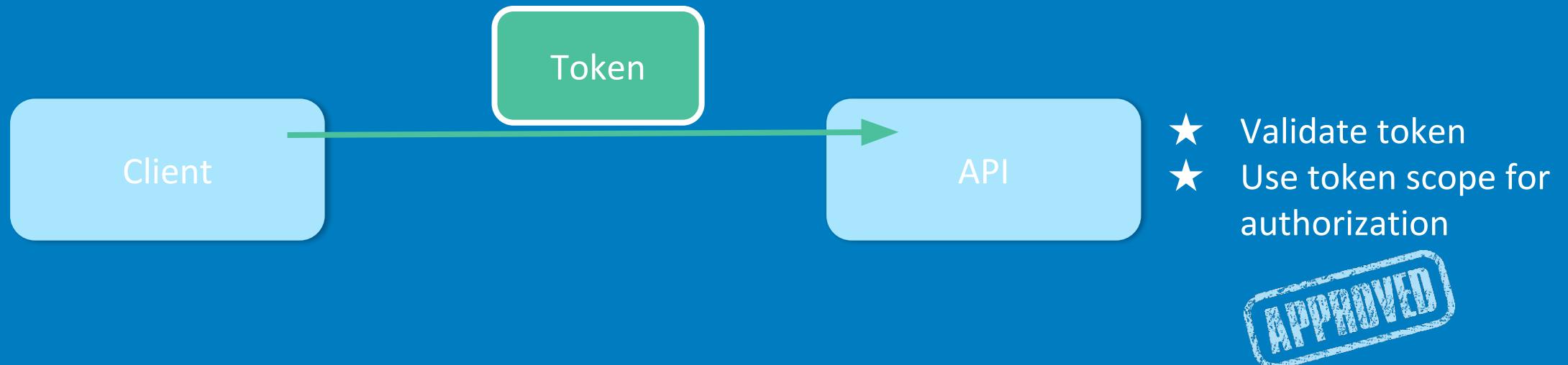
# Authorization server returns an access token

```
{  
  "access_token": "fFAGRNJru1FTz70BzhT3Zg",  
  "expires_in": 3920,  
  "token_type": "Bearer",  
}
```

# Use the access token

GET api.google.com/some/endpoint

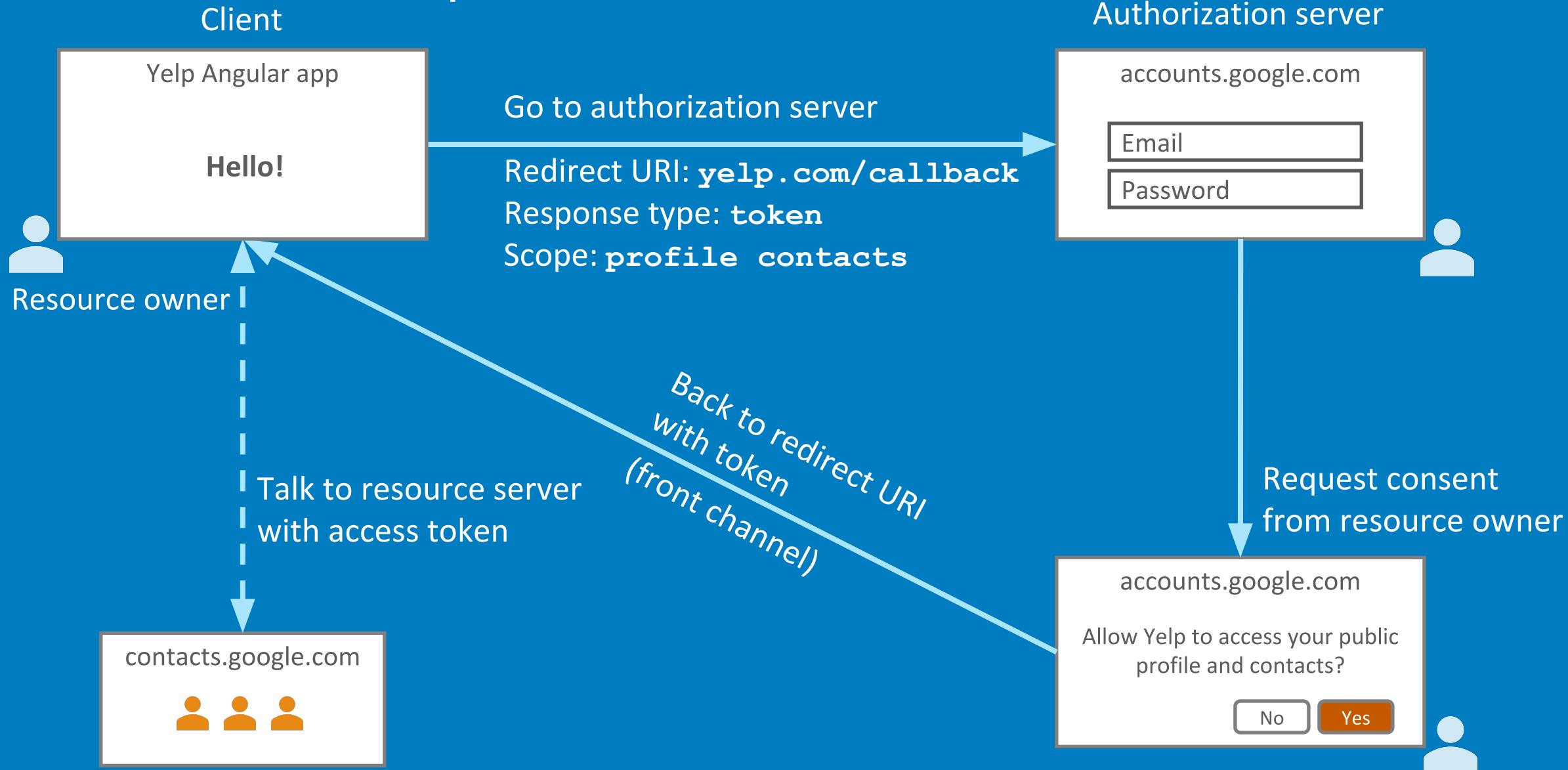
Authorization: Bearer fFAGRNJru1FTz70BzhT3Zg



# OAuth 2.0 flows

- ★ Authorization code (front channel + back channel)
- ★ Authorization code w/ PKCE (front channel + back channel)\*
- ★ Implicit (front channel only)
  
- ★ Resource owner password (back channel only)
- ★ Client credentials (back channel only)

# OAuth 2.0 implicit flow



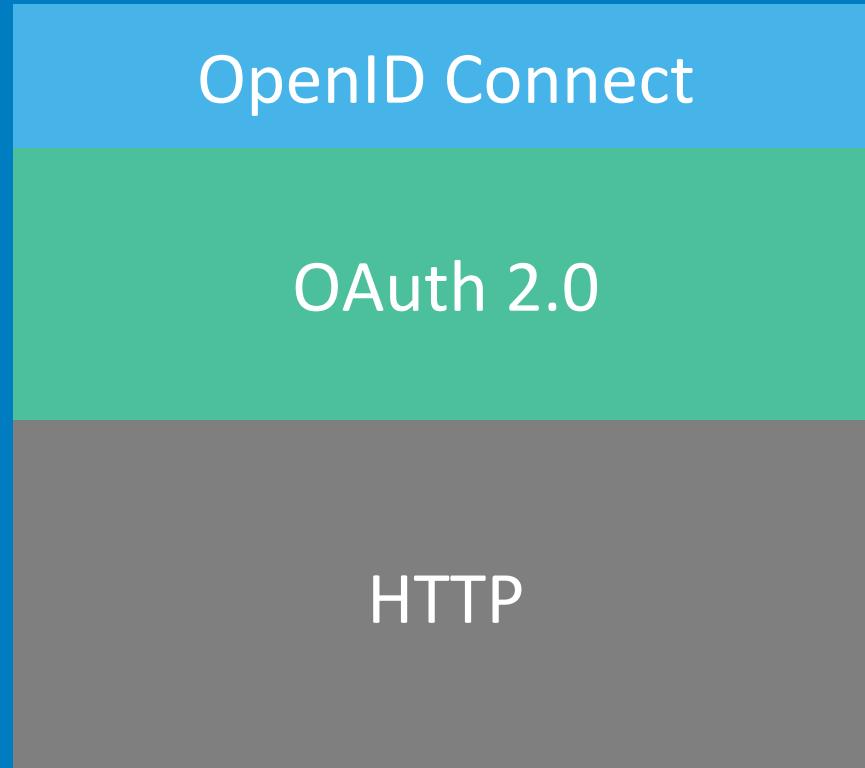
# Identity use cases (circa 2012)

- ★ Simple login – OAuth 2.0 Authentication
- ★ Single sign-on across sites – OAuth 2.0 Authentication
- ★ Mobile app login – OAuth 2.0 Authentication
- ★ Delegated authorization – OAuth 2.0 Authorization

# Problems with OAuth 2.0 for authentication

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

# OAuth 2.0 and OpenID Connect



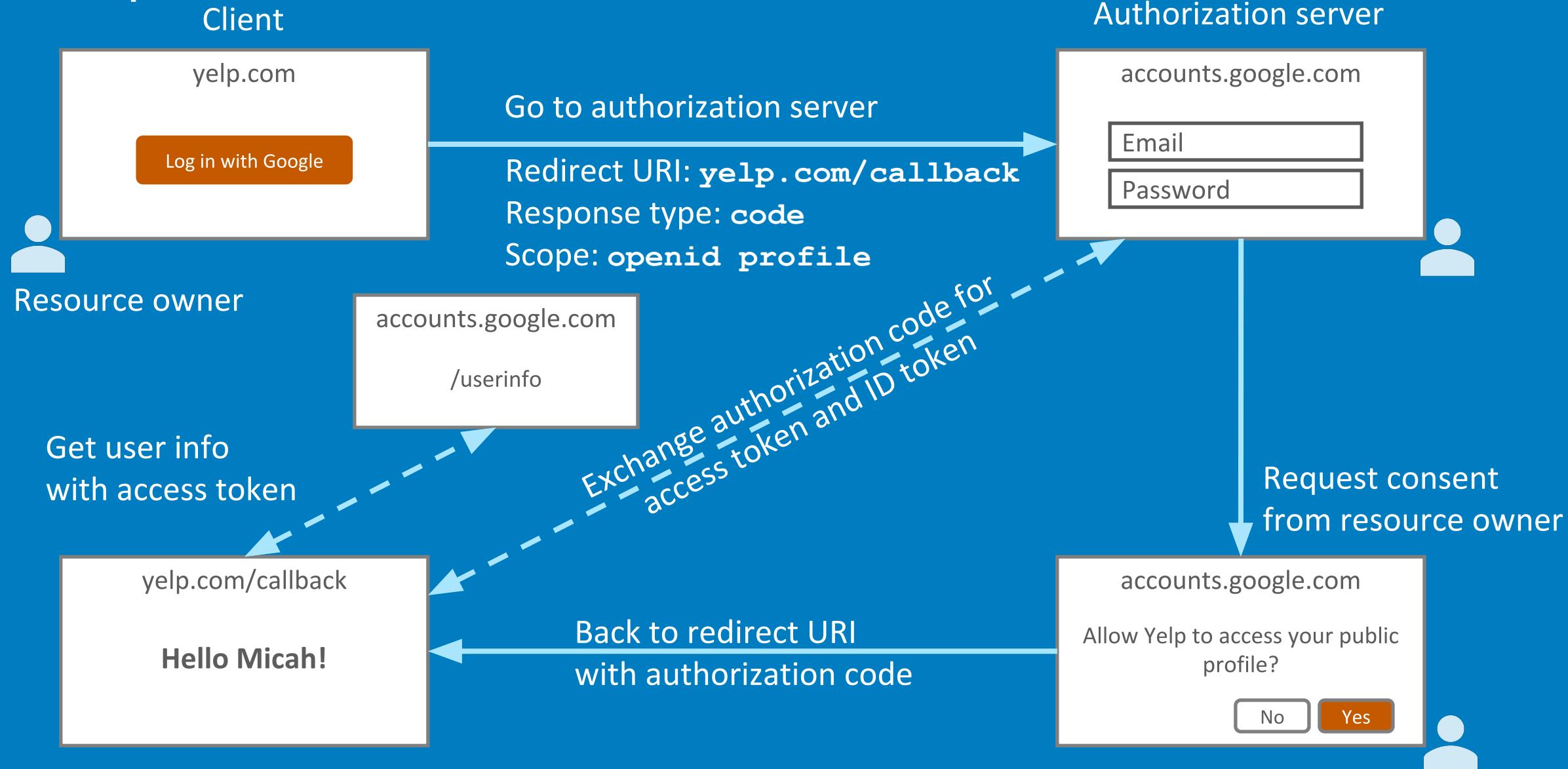
OpenID Connect is for  
**authentication**

OAuth 2.0 is for  
**authorization**

# What OpenID Connect adds

- ★ ID token
- ★ /userinfo endpoint for getting more user information
- ★ Standard set of scopes
- ★ Standardized implementation

# OpenID Connect authorization code flow



# Getting the discovery document

```
GET https://accounts.google.com/.well-known/openid-configuration
```

```
{
  "issuer": "https://accounts.google.com",
  "authorization_endpoint": "https://accounts.google.com/o/oauth2/v2/auth",
  "token_endpoint": "https://www.googleapis.com/oauth2/v4/token",
  "userinfo_endpoint": "https://www.googleapis.com/oauth2/v3 userinfo",
  "response_types_supported": [
    "code",
    "token",
    "id_token",
    "code token",
  ],
  ...
}
```

# Starting the flow

```
https://accounts.google.com/o/oauth2/v2/auth?  
client_id=abc123&  
redirect_uri=https://yelp.com/callback&  
scope=openid profile&  
response_type=code&  
state=foobar
```

# Calling back

`https://yelp.com/callback?`  
`error=access_denied&`  
`error_description=The user did not consent.`

`https://yelp.com/callback?`  
`code=oMsCeLvIaQm6bTrgtp7 &`  
`state=foobar`

# Exchange code for access token **and** ID token

POST `www.googleapis.com/oauth2/v4/token`

`Content-Type: application/x-www-form-urlencoded`

`code=oMsCeLvIaQm6bTrgtp7&`  
`client_id=abc123&`  
`client_secret=secret123&`  
`grant_type=authorization_code`

# Authorization server returns access and ID tokens

```
{  
  "access_token": "fFAGRNJru1FTz70BzhT3Zg",  
  "id_token": "eyJraB03ds3F..."  
  "expires_in": 3920,  
  "token_type": "Bearer",  
}
```

# ID token (JWT)

eyJhbGciOiJSUzI1NiIsImtpZCI6IkRNA3Itd0JqRU1EYnh0Y25xaVJISVhuYUxu  
bWI3UUUpfWF9rWmJyaEtBMGMifQ

Header

.

eyJzdWIiOiIwMHU5bzFuawtqdk9CZzVabzBoNyIsInZlciI6MSwiaXNzIjoiaHR0  
cHM6Ly9kZXQtMzQxNjA3Lm9rdGFwcmV2aWV3LmNvbS9vYXV0aDIvYXVzOW84d3Zr  
aG9ja3c5VEwwaDciLCJhdWQiOijswFN1bkx4eFBpOGtRVmpKRTVzNCIsImlhCI6  
MTUwOTA00Tg50CwiZXhwIjoxNTA5MDUzNDk4LCJqdGkiOijJRC5oa2RXSXNBSXZT  
bnBGYVFHTVRYUGNVSmhhMkgwS2c5Ykl3ZEVvVm1ZZHN3IiwiYW1yIjpbImtiYSIs  
Im1mYSIsInB3ZCJdLCJpZHAiOiIwMG85bzFuawprawpLeGNpbjBoNyIsIm5vbml  
IjoidwpwMmFzeHlqN2UiLCJhdXRoX3RpbWUiOjE1MDkwNDk3MT19

Payload  
(claims)

.

dv4Ek8B4BDee1PcQT\_4zm7kxDEY1sRIGbLoNtlodZcSzHz-XU5GkKy16sAVmdXOI  
PU1AIrJAhNfQWQ-\_XZLBVPjETiZE8CgNg5uqNmeXMUnYnQmvN5owlXUZ8Gcub-GA  
bJ8-NQuyBmyec1j3gmGzX3wemke8NkuI6SX2L4Wj1PyvkknBtbjfiF9ud1-ERKbo  
baFbnjDFOFTzvL6g34SpMmZWy6uc\_Hs--n4IC-ex-\_Ps3FcMwRggCW\_-7o2FpH6r  
JTOGPZYr0x44n3ZwAu2dGm6axtPI-sqU8b6sw7DaHpogD\_hxsXgMIOzOBMbYsQEi  
czogn71ZFz\_107FiW4dH6g

Signature



# The ID token (JWT)

(Header)

.

{

```
"iss": "https://accounts.google.com",
"sub": "micah.silverman@okta.com",
"name": "Micah Silverman"
"aud": "s6BhdRkqt3",
"exp": 1311281970,
"iat": 1311280970,
"auth_time": 1311280969,
```

}

.

(Signature)

# Calling the userinfo endpoint

```
GET www.googleapis.com/oauth2/v4/userinfo  
Authorization: Bearer fFAGRNJru1FTz70BzhT3Zg
```

200 OK

Content-Type: application/json

```
{  
  "sub": "micah.silverman@okta.com",  
  "name": "Micah Silverman"  
  "profile_picture": "http://plus.g.co/123"  
}
```

# Identity use cases (today)

- ★ Simple login – OpenID Connect      Authentication
- ★ Single sign-on across sites – OpenID Connect      Authentication
- ★ Mobile app login – OpenID Connect      Authentication
- ★ Delegated authorization – OAuth 2.0      Authorization

# OAuth and OpenID Connect

## **Use OAuth 2.0 for:**

Granting access to your API

Getting access to user data in  
other systems

**(Authorization)**

## **Use OpenID Connect for:**

Logging the user in

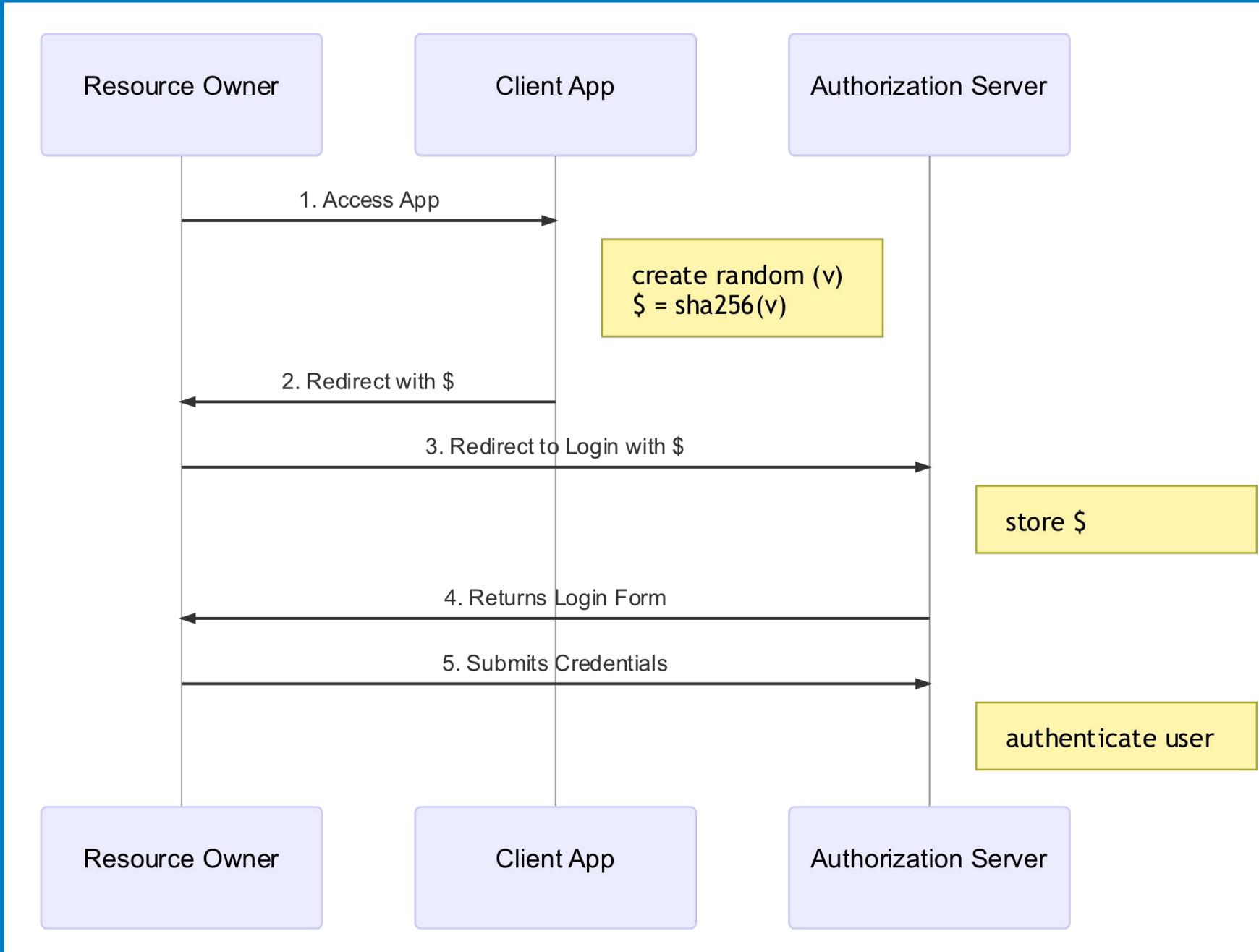
Making your accounts  
available in other systems

**(Authentication)**

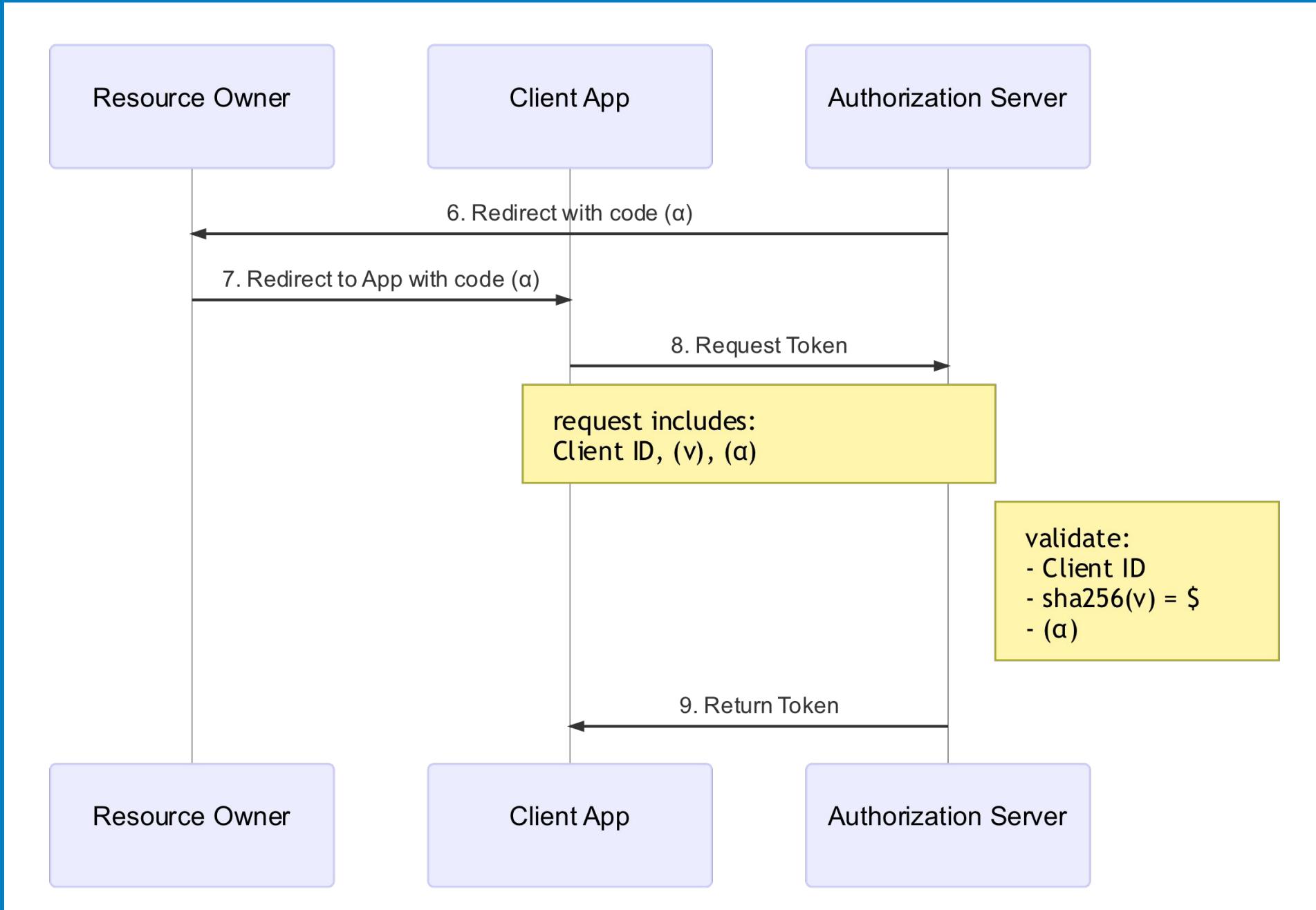
# Which flow (grant type) do I use?

- ★ Web application w/ server backend: **authorization code flow**
- ★ Native or mobile app: **authorization code with PKCE flow**
- ★ JavaScript app (SPA): **authorization code with PKCE flow**
- ★ Microservices and APIs: **client credentials flow**

# Authorization Code with PKCE Flow (Part 1)



# Authorization Code with PKCE Flow (Part 2)



# Token validation

**The fast way:** local validation

- Validate cryptographic signature

- Check expiration timestamp

**The strong way:** introspection

Micah Silverman  
@afitnerd

oauth.com  
@oktadev

Free hosted authorization server:  
[developer.okta.com](https://developer.okta.com)

