

# OAuth and OpenID Connect

(IN PLAIN ENGLISH)

NATE BARBETTINI  
@NBARBETTINI  
@OKTADEV

# A lot of confusion around OAuth.

- × Terminology and jargon
- × Incorrect advice

## Identity use cases (circa 2007)

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





# The delegated authorization problem

HOW CAN I LET A WEBSITE ACCESS MY DATA  
(WITHOUT GIVING IT MY PASSWORD)?

# 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   Hotmail      

Your Email Address  (e.g. bob@gmail.com)

Your Gmail Password  (The password you use to log into your Gmail email)

[Skip this step](#)

# 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](#)

# Delegated authorization with OAuth 2.0



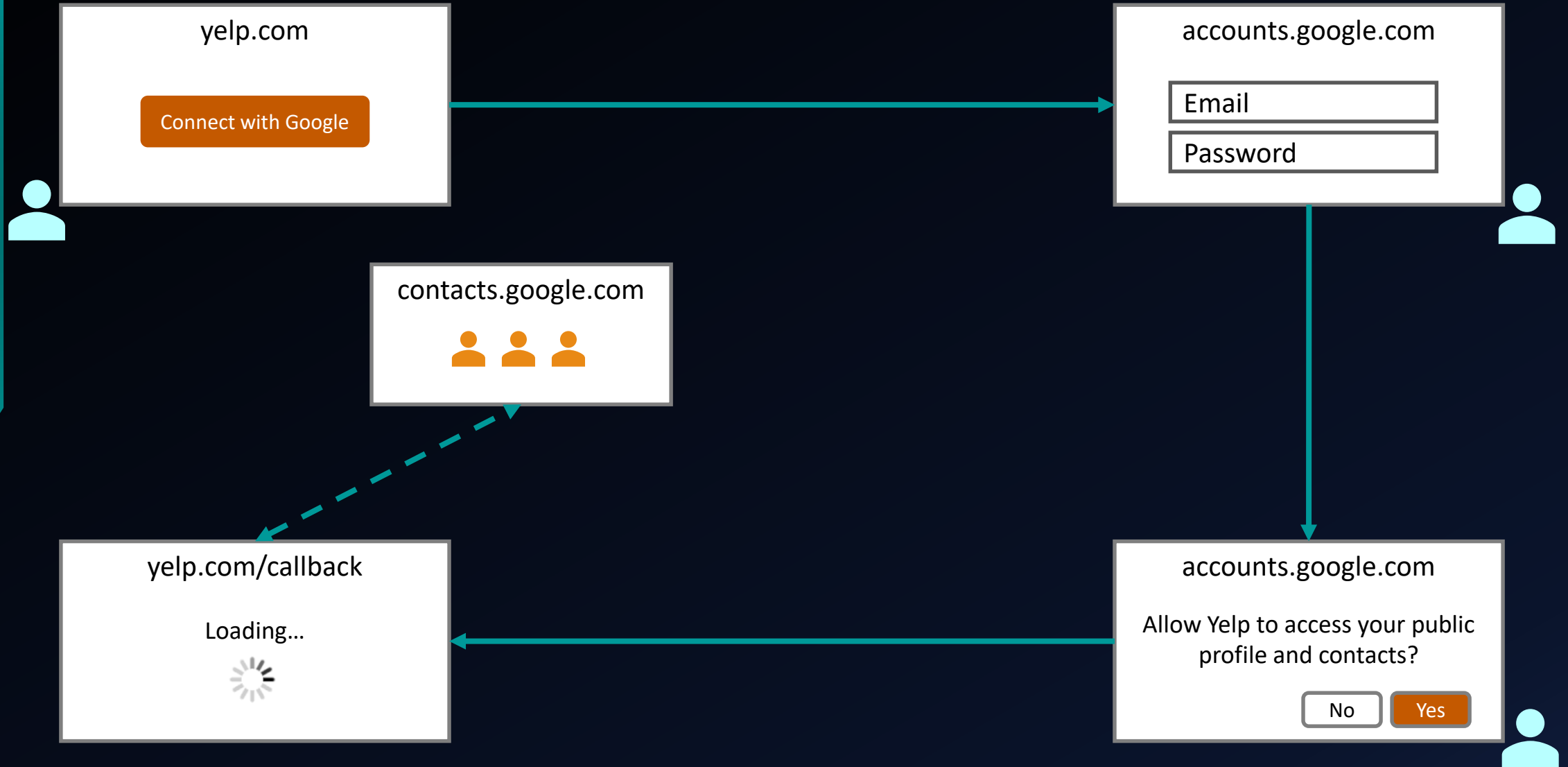
I trust Gmail and I kind of trust Yelp. I want Yelp to have access to my contacts only.

yelp.com

Connect with Google



# Delegated authorization with OAuth 2.0

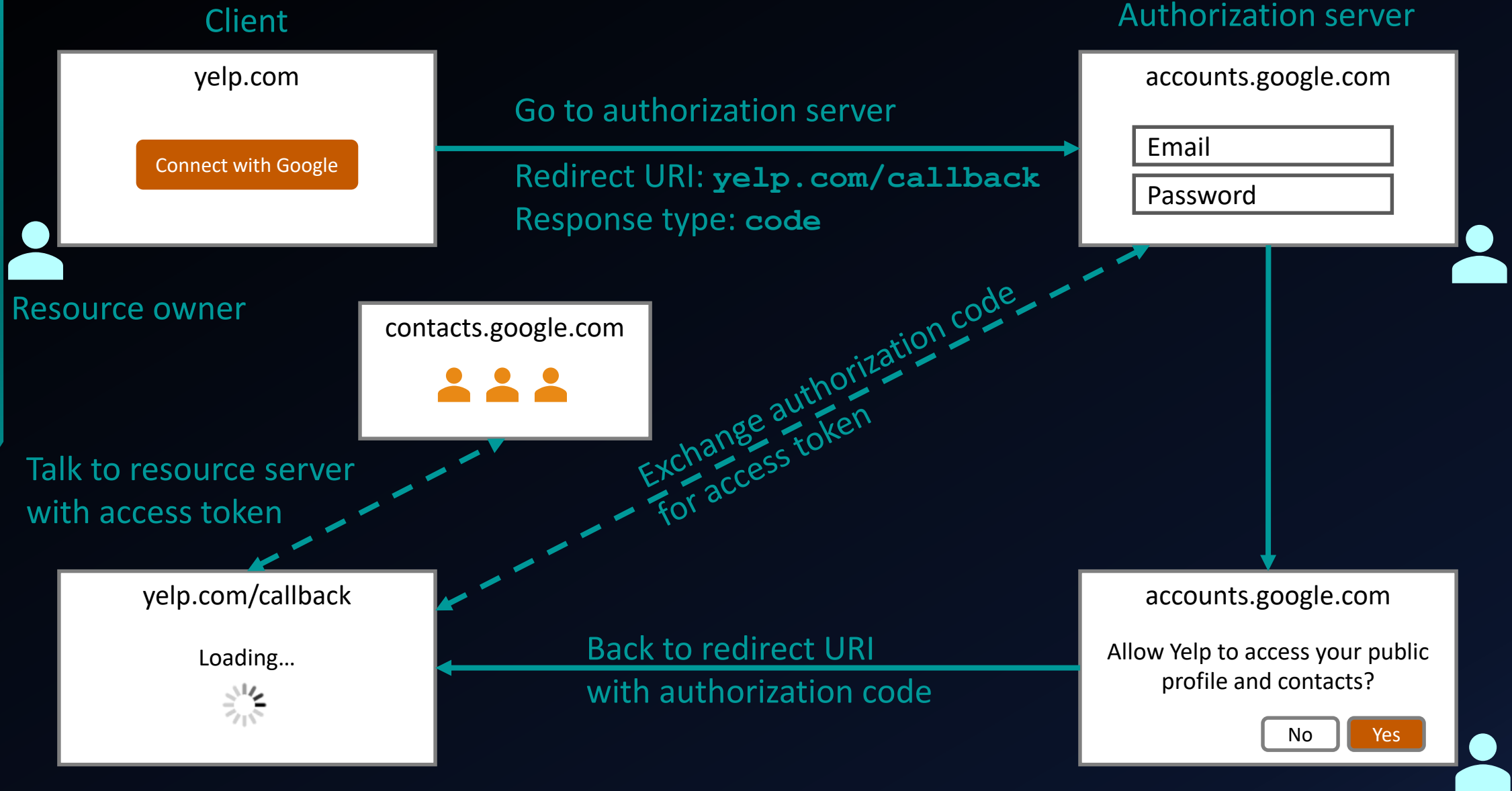




# OAuth 2.0 terminology

- Resource owner
- Client
- Authorization server
- Resource server
- Authorization grant
- Access token

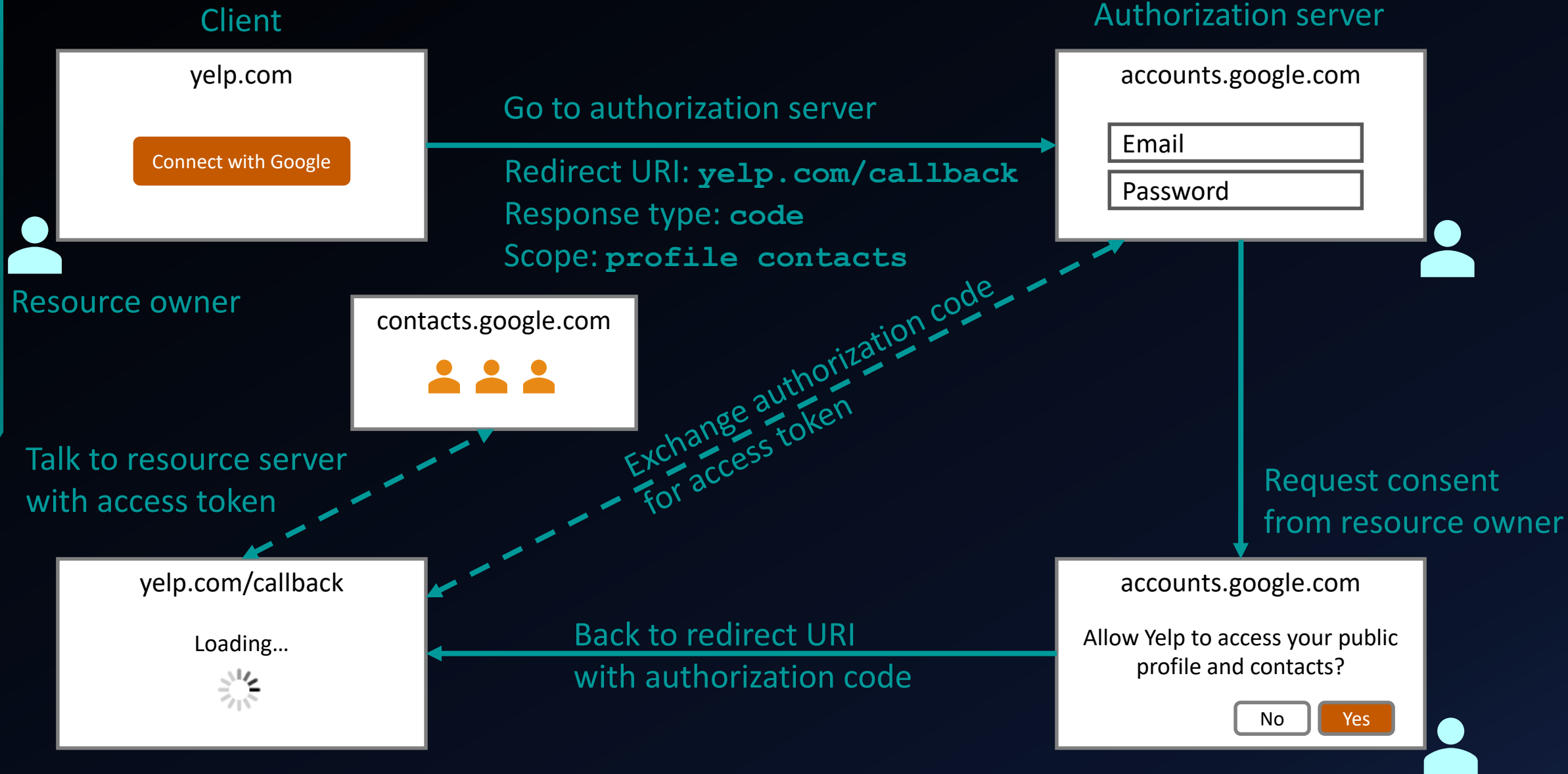
# OAuth 2.0 authorization code flow



# More OAuth 2.0 terminology

- Scope
- Consent

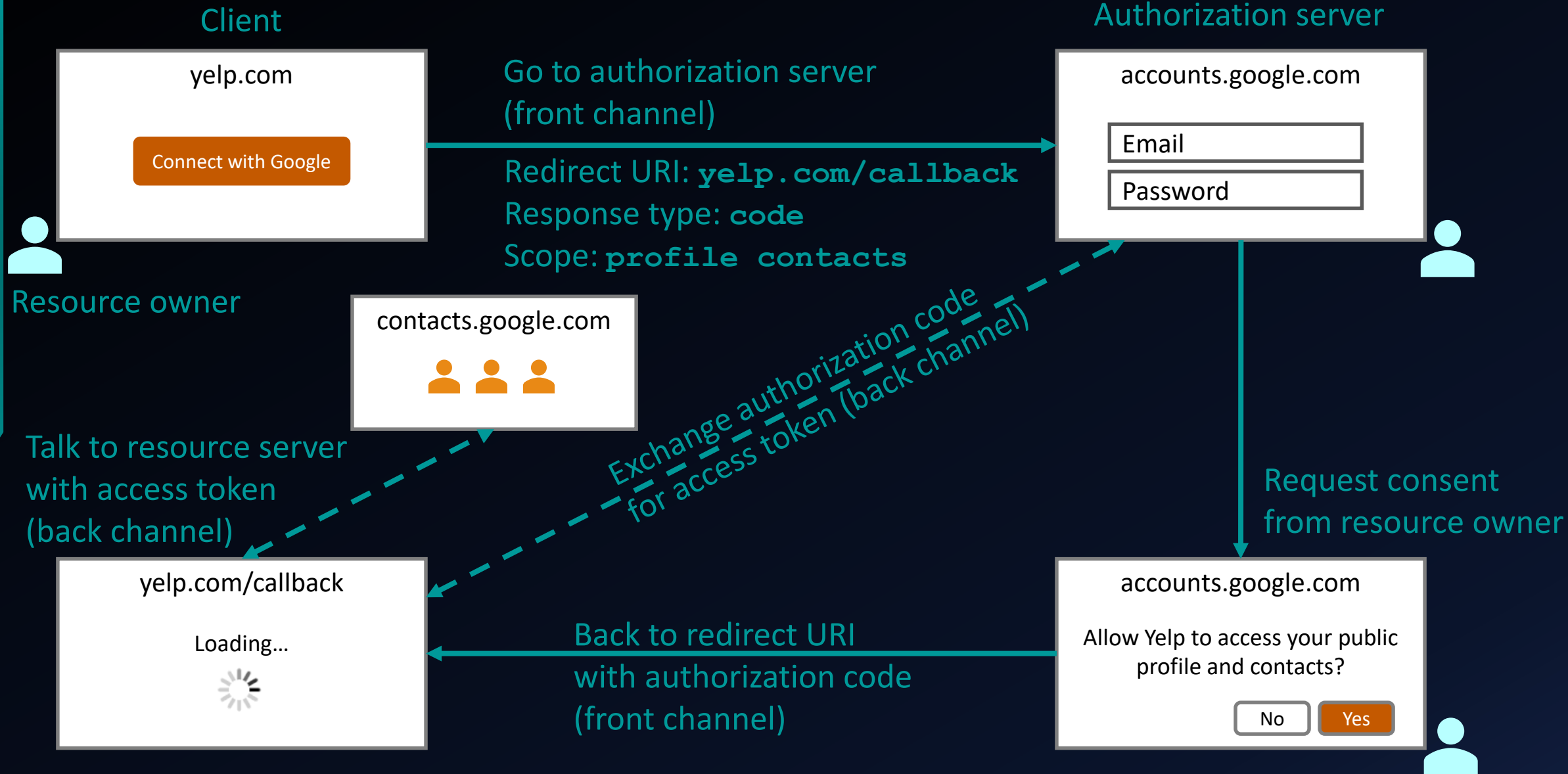
# OAuth 2.0 authorization code flow



## 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
```



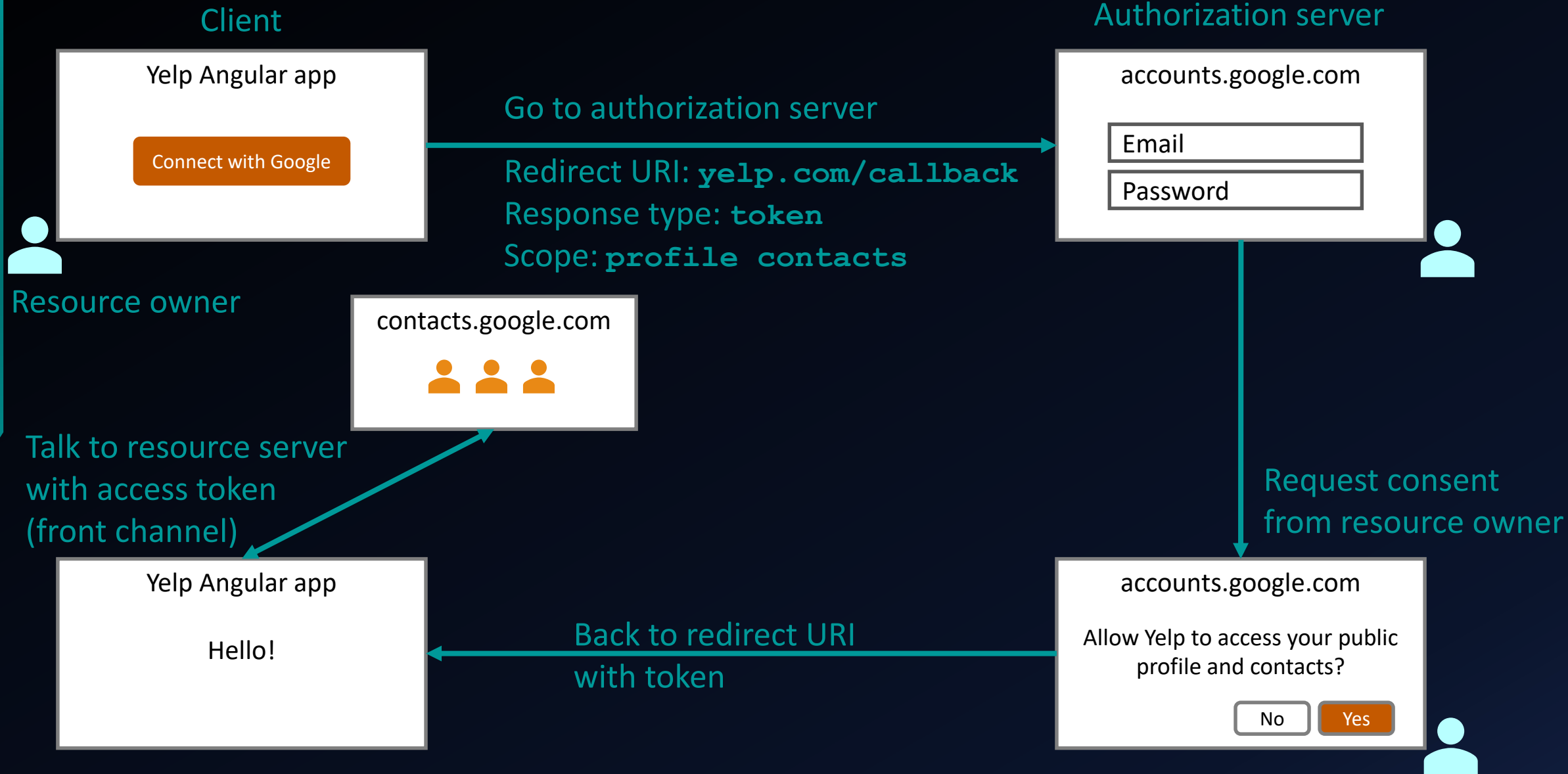
- Validate token
- Use token scope for authorization



## OAuth 2.0 flows

- Authorization code (front channel + back channel)
- Implicit (front channel only)
- Resource owner password credentials (back channel only)
- Client credentials (back channel only)

# OAuth 2.0 implicit flow



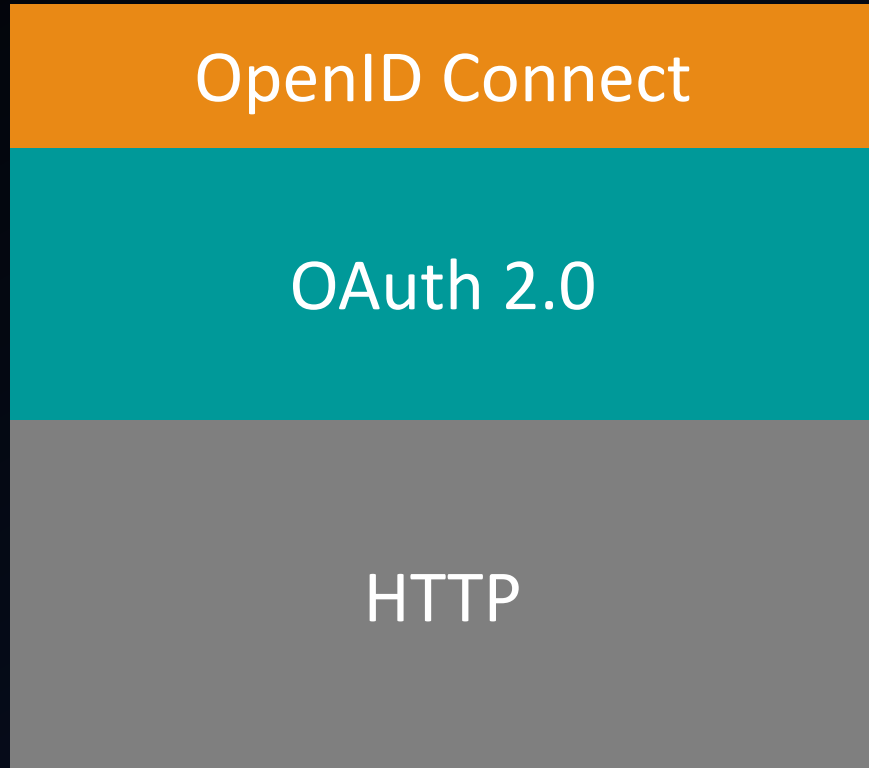
## Identity use cases (circa 2012)

- 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



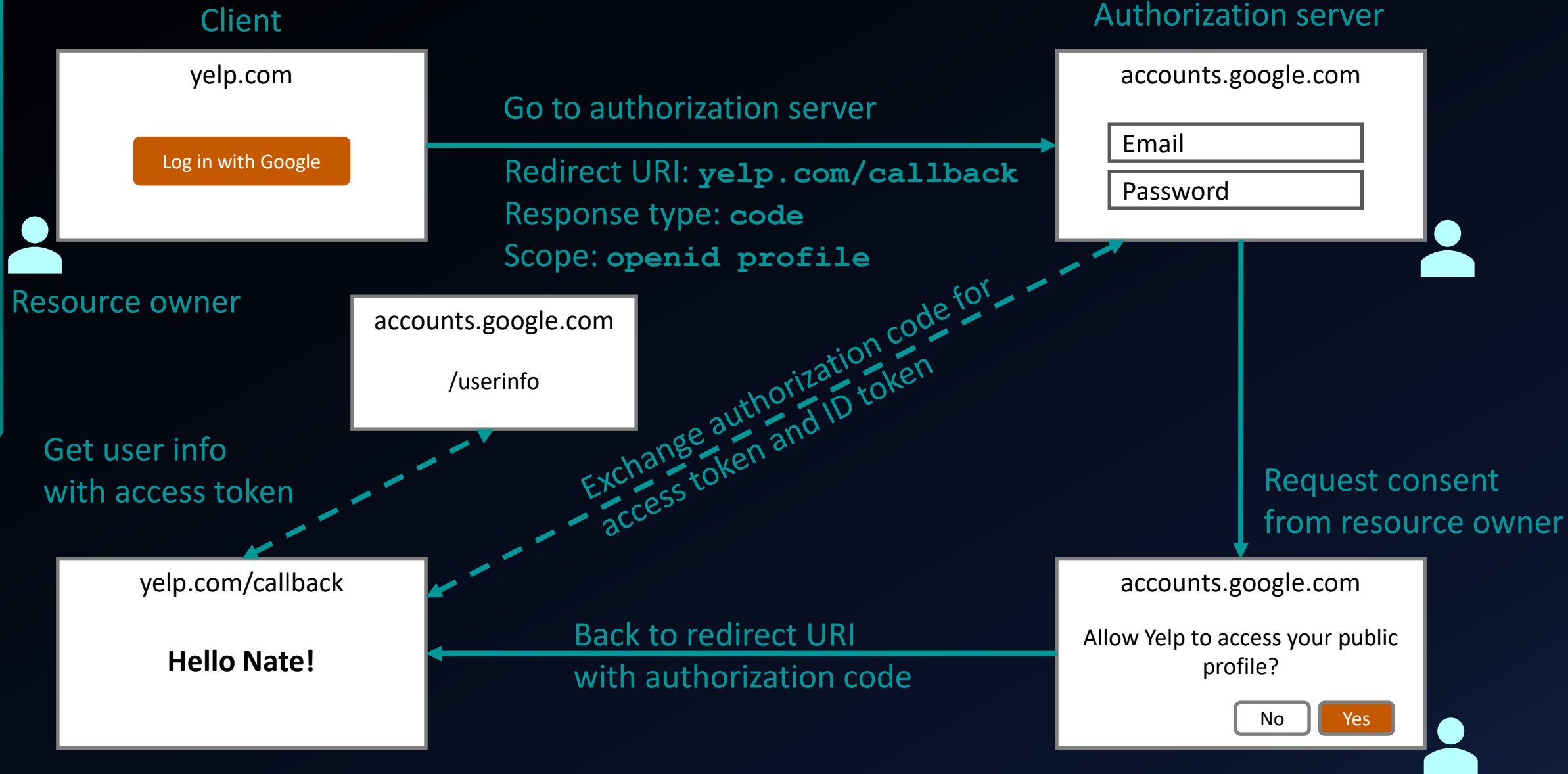
- OpenID Connect is for **authentication**
- OAuth 2.0 is for **authorization**



# What OpenID Connect adds

- ID token
- UserInfo endpoint for getting more user information
- Standard set of scopes
- Standardized implementation

# OpenID Connect authorization code flow



## 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
```

# 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)

eyJhbGciOiJSUzI1NiIsImtpZCI6Ikp0RU1EYnh0Y25xaVJISVhuYUxubWI3UUpfWF9rWmJyaEtBMGMifQ

Header

eyJzdWIiOiIwMHU5bzFuaWtqdG9CZzVabzBoNyIsInZlciI6MSwiaXNzIjoiaHR0cHM6Ly9kZXYtMzQxNjA3Lm9rdGFwcmV2aWV3LmNvbS9vYXV0aDIvYXVzOw84d3ZraG9ja3c5VEwwaDciLCJhdWQiOiJscWFNlbnkx4eFBpOGtRVmpKRTVzNCIsIm1hdCI6MTUwOTA0Tg5OCwiZXhwIjojXNTA5MDUzNDk4LCJqdGkiOiJJRC5oa2RXSXNBSXZTbnBGYVFHTVRYUGNVSmhhMkgwS2c5Yk13ZEVvVm1ZZHN3IiwiaWF0IjoiIm1mYSIsInB3ZCJdLCJpZHAiOiIwMG85bzFuaWprYWwLeGNpbjBoNyIsIm5vbmNIjoidWpwMmFzeHlqN2UiLCJhdXR0X3RpbWUiOiJlMDkwNDk3MT19

Payload  
(claims)

dv4Ek8B4BDee1PcQT\_4zm7kxDEY1sRIGbLoNt1odZcSzHz-XU5GkKy16sAVmdXOIPU1AIrJAhNfQWQ-\_XZLBVPjETiZE8CgNg5uqNmeXMUnYnQmvN5oW1XUZ8Gcub-GAbJ8-NQuyBmyec1j3gmGzX3wemke8NkuI6SX2L4Wj1PyvkknBtbjfiF9ud1-ERKbobaFbnjDF0FTzvL6g34SpMmZWy6uc\_Hs--n4IC-ex-\_Ps3FcMwRggCW\_-7o2FpH6rJT0GPZYrOx44n3ZwAu2dGm6axtPI-sqU8b6sw7DaHpogD\_hxsXgMIOzOBMbYsQEiczoGn71ZFz\_107FiW4dH6g

Signature



# The ID token (JWT)

(Header)

```
.  
{  
  "iss": "https://accounts.google.com",  
  "sub": "you@gmail.com",  
  "name": "Nate Barbettini"  
  "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": "you@gmail.com",  
  "name": "Nate Barbettini"  
  "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:

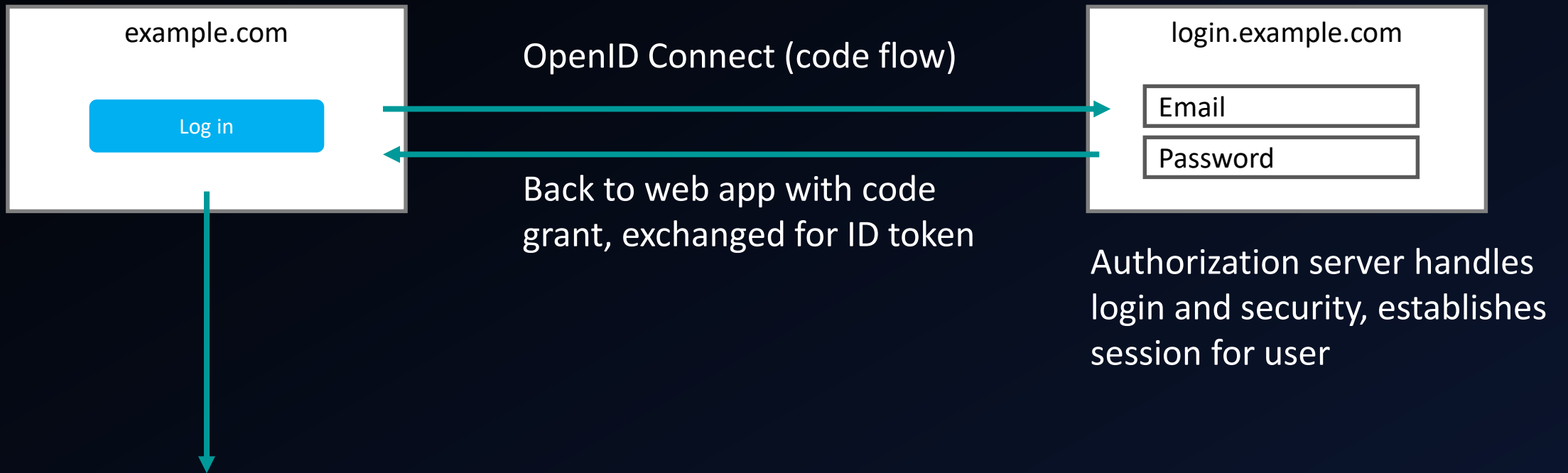
- Logging the user in
- Making your accounts available in other systems

(Authentication)

## Which flow (grant type) do I use?

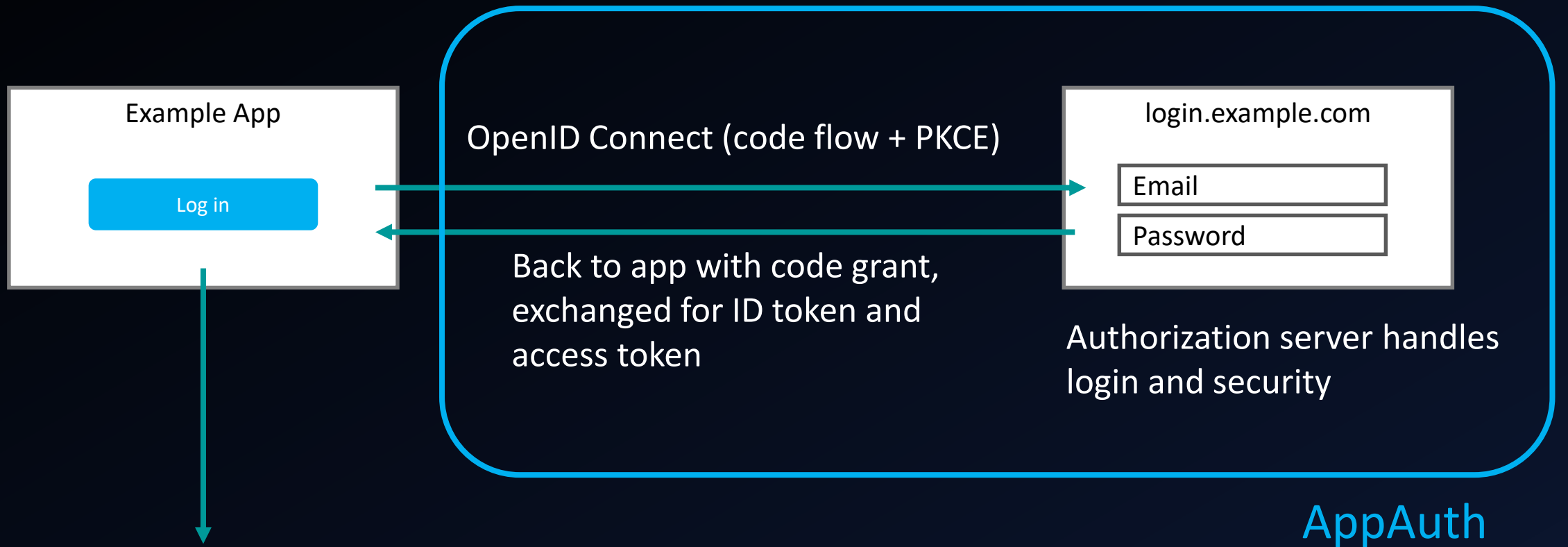
- Web application w/ server backend: **authorization code flow**
- Native mobile app: **authorization code flow with PKCE**
- JavaScript app (SPA) w/ API backend: **implicit flow**
- Microservices and APIs: **client credentials flow**

# Example: web application with server backend



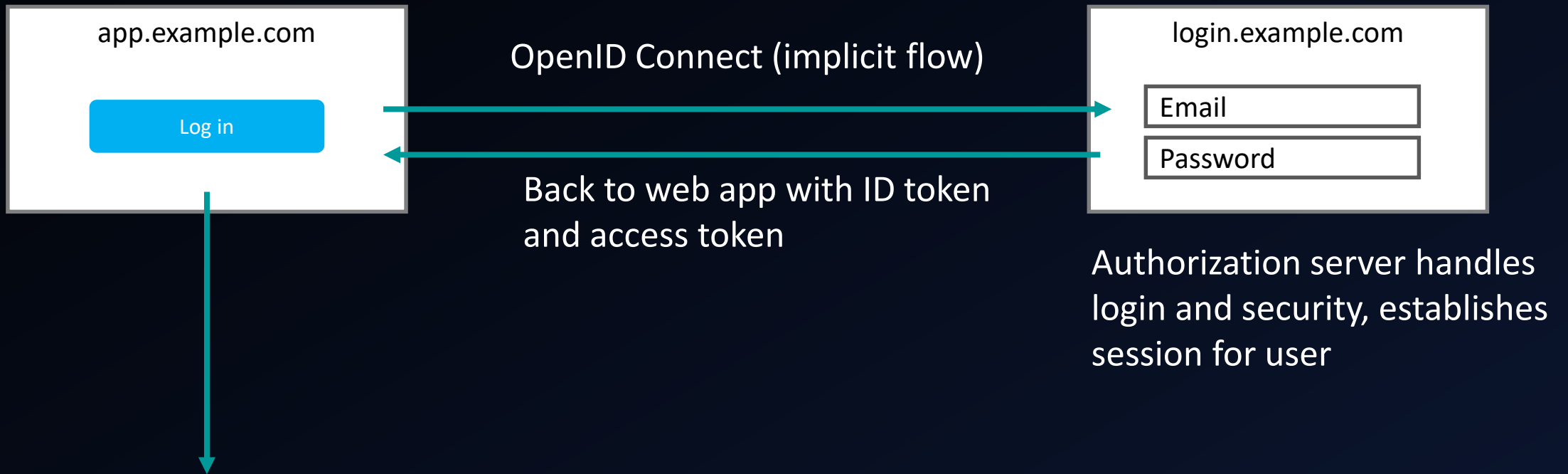
Set-Cookie: sessionid=f00b4r; Max-Age: 86400;

# Example: native mobile app



- Store tokens in protected device storage
- Use ID token to know who the user is
- Attach access token to outgoing API requests

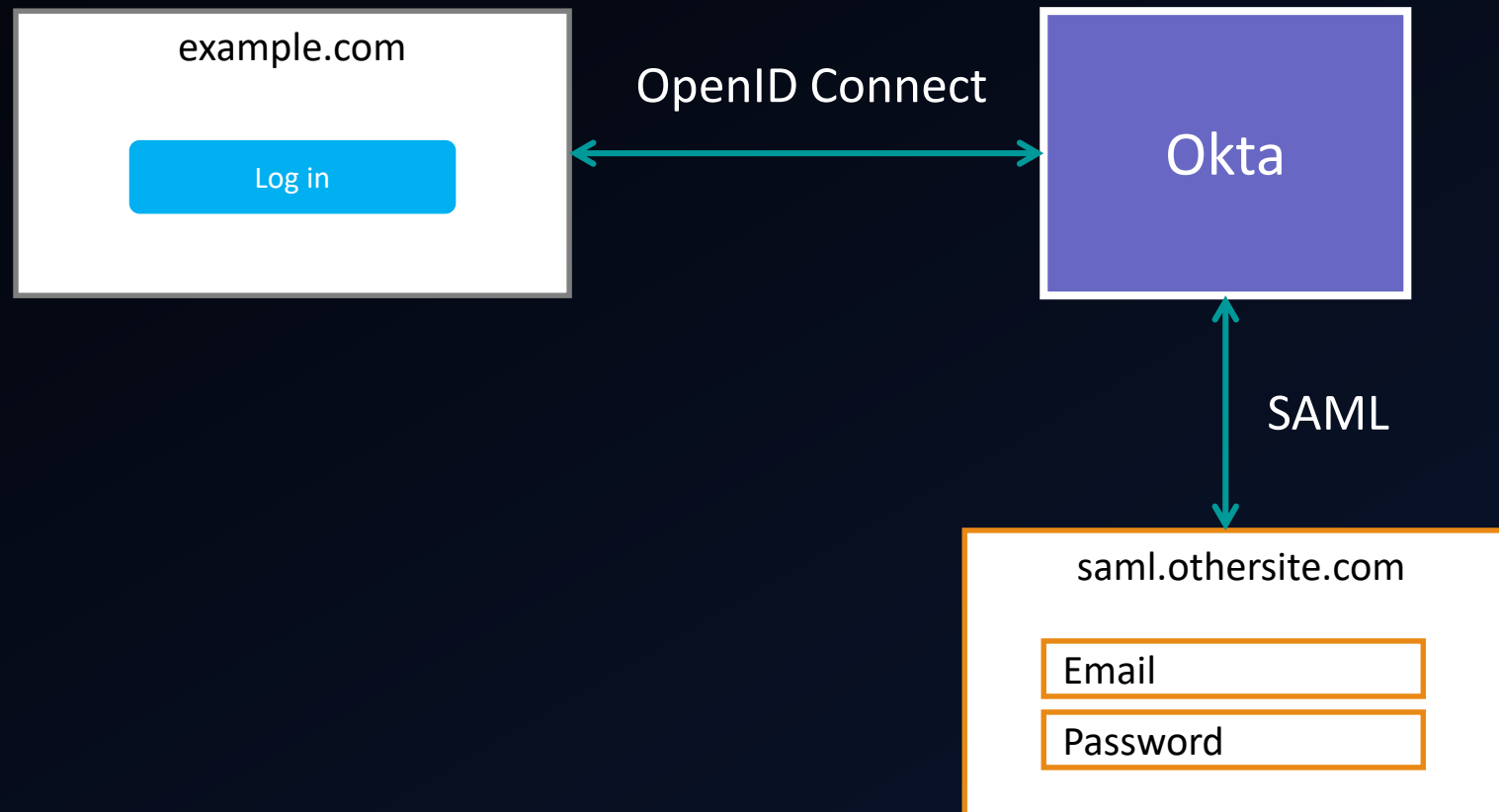
# Example: SPA with API backend



Authorization server handles login and security, establishes session for user

Store tokens locally with JavaScript  
Use ID token to know who the user is  
Attach access token to outgoing API requests

# Example: SSO with 3<sup>rd</sup>-party services



# Token validation

- The fast way: local validation
  - Check expiration timestamp
  - Validate cryptographic signature
- The strong way: introspection



# Revocation

Token issued and  
used for API calls

Device  
compromised!

What happens?

12PM

1PM

2PM

```
POST /oauth2/default/v1/revoke
```

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

```
token=fFAGRNJru1FTz70BzhT3Zg  
&token_type_hint=access_token  
&client_id=...
```

# Keeping the user signed in

For both local validation and introspection, the token is invalid once it expires, so:

- If there's a user at the keyboard, just redirect through the authorization server again.
- If there's no user (automated tasks), request a refresh token (offline scope).

# Thanks y'all!

Nate Barbettini  
[@nbarbettini](#)

[oauth.com](#)  
[@oktadev](#)

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