



# OAUTH

*Maurice Tedder • Grand Circus Detroit*

# TOPICS

## **Topics covered:**

- Definition of OAuth2
- OAuth2 Use Cases
- Example OAuth2 flow using Github API

# OAUTH2

## What is OAUTH2?

- An open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications.<sup>1</sup>
- The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service.<sup>1</sup>
- OAuth 2.0 is an authorization protocol supported by most APIs

Reference

1. <http://oauth.net/>

# OAUTH2 PROVIDERS

## **A small list of some OAuth2 providers**

- Google
- Facebook
- Amazon
- Twitter
- LinkedIn
- Spotify

# OAUTH2 AUTHENTICATION

- When available, many people prefer a 3rd party sign-in as long as the permissions they're asked for are non-intrusive
  - For example, use LinkedIn's OAuth2 API to get a user's LinkedIn Headline for your Application
  - LinkedIn's API will only give that information if the owner of the information grants your application access
  - This requires that we authenticate the user to verify that they are the true owner of the information we seek
- We delegate authenticating the user to LinkedIn's OAuth2 service endpoint

# OAUTH2 ACCESS CODE AND AUTHORIZATION

- But how does our app gain access to the User's information after the owner has granted our app permission?
- The OAuth2 service will give your app an Authorization code that you can use to get an access token

# OAUTH2 ACCESS TOKEN

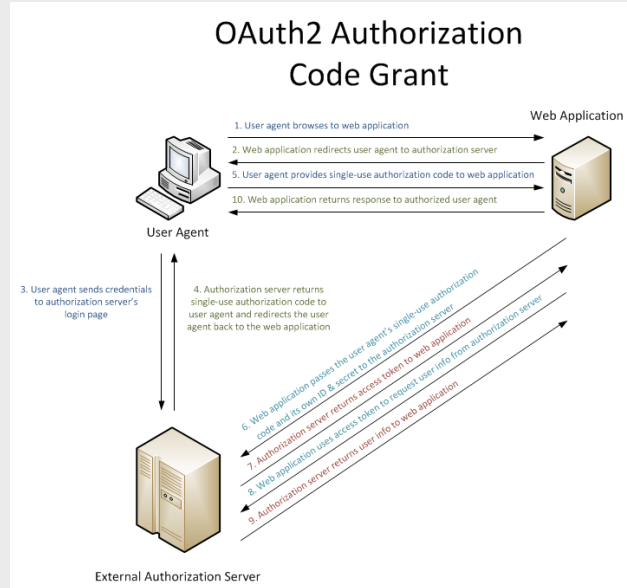
- When your application requests private data, the request must be authorized by an authenticated user who has access to that data.
- The Access Token is the key we can use to call the third party (ex. LinkedIn API) and get the information we want about the user
- The Access Token is our proof to the API that the owner of the information is authenticated and has agreed to grant access to our app to get the information we requested.

# OAUTH2 FLOW

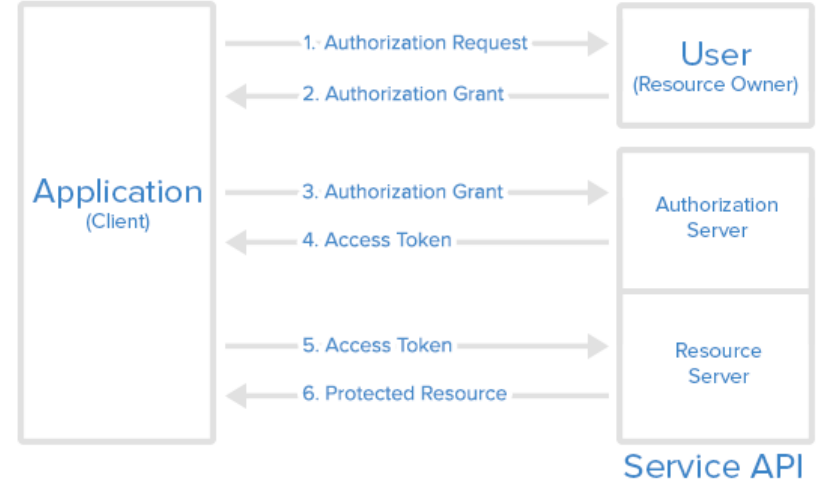
- Register your application with the API (varies by API)
  - You will get an Application ID and API Key (JavaScript, Server, Android or iOS)
- Provide a redirect URL for JavaScript calls or callback function for server calls
- Request an Authorization code by redirecting the user to the Authentication page of the OAuth2 provider
- Exchange the Authorization code for an Access Token
- Use the Access Token to make authenticated requests on the API



# OAUTH2 FLOW DIAGRAM



## Abstract Protocol Flow



Reference:

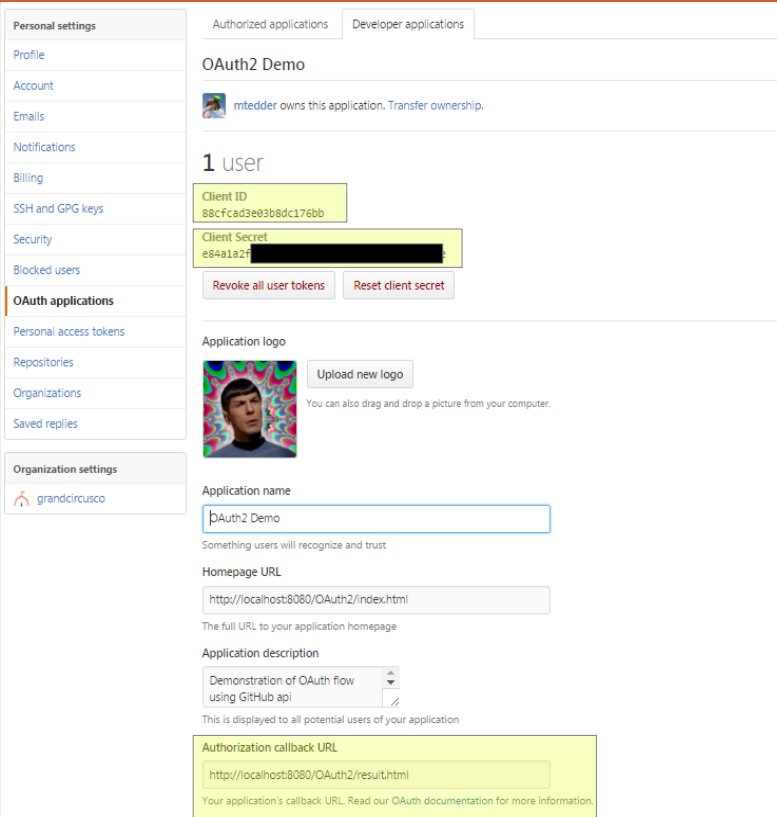
<http://i.stack.imgur.com/PBlvj.png>

[https://assets.digitalocean.com/articles/oauth/abstract\\_flow.png](https://assets.digitalocean.com/articles/oauth/abstract_flow.png)

# WEB EXAMPLE – GITHUB OAUTH2

1. Register application on GitHub to get a Client ID & Client secret
2. Redirect users to request GitHub access – (provide a value for the state/nonce and scopes you need)
3. GitHub redirects back to your site with a temporary code – (returned state/nonce should match state passed in step 2)
4. Exchange temporary code for an access token
5. Use access tokens to make calls to the GitHub Api on behalf of the user

# GITHUB OAUTH2 – REGISTER APPLICATION



The screenshot shows the GitHub 'Developer applications' settings page. The left sidebar contains navigation links: Personal settings, Profile, Account, Emails, Notifications, Billing, SSH and GPG keys, Security, Blocked users, OAuth applications (highlighted), Personal access tokens, Repositories, Organizations, and Saved replies. Under 'Personal settings', there are links for Organization settings and grandcircuso. The main content area is titled 'Authorized applications' and 'Developer applications'. It shows an 'OAuth2 Demo' application owned by 'mtedder'. It lists '1 user' and displays the 'Client ID' (88cFcad3e83b8dc176bb) and 'Client Secret' (e8481a27). Below these are buttons for 'Revoke all user tokens' and 'Reset client secret'. The 'Application logo' section shows a placeholder image of a person's face and a button to 'Upload new logo'. The 'Application name' field contains 'OAuth2 Demo'. The 'Homepage URL' field contains 'http://localhost:8080/OAuth2/index.html'. The 'Application description' field contains 'Demonstration of OAuth flow using GitHub api'. The 'Authorization callback URL' field contains 'http://localhost:8080/OAuth2/result.html'.

Personal settings

Profile

Account

Emails

Notifications

Billing

SSH and GPG keys

Security

Blocked users

**OAuth applications**

Personal access tokens

Repositories

Organizations

Saved replies

Organization settings

grandcircuso

Authorized applications

Developer applications

OAuth2 Demo

mtedder owns this application. [Transfer ownership.](#)

1 user

Client ID  
88cFcad3e83b8dc176bb

Client Secret  
e8481a27

[Revoke all user tokens](#) [Reset client secret](#)

Application logo

Upload new logo

You can also drag and drop a picture from your computer.

Application name  
OAuth2 Demo

Something users will recognize and trust

Homepage URL  
http://localhost:8080/OAuth2/index.html

The full URL to your application homepage

Application description  
Demonstration of OAuth flow using GitHub api

This is displayed to all potential users of your application

Authorization callback URL  
http://localhost:8080/OAuth2/result.html

Your application's callback URL. Read our OAuth documentation for more information.

Register application on GitHub to get a Client ID & Client secret

- Go to Settings → Developer settings in your Github account
- Provide a callback URL

# GITHUB OAUTH2 – REDIRECT TO GITHUB AUTHORIZATION API REST SERVICE

- 1) Redirect users to request GitHub access – (provide a value for the state and scopes you need)

```
https://github.com/login/oauth/authorize  
?client_id=88cfcad3e03b8dc176bb&  
scope=user&state=12345&  
redirect_uri=http://localhost:8080/0Auth2/result.html
```

# GITHUB OAUTH2 – RECEIVE ACCESS CODE RESPONSE

2) GitHub redirects back to your site with a temporary code and your state value as parameters to your redirect URL – (returned state should match state passed in step 2)

```
url:http://localhost:8080/Auth2/result.html?  
code=840ddd039d280c591175&  
state=12345
```

# GITHUB OAUTH2 – REQUEST ACCESS TOKEN

3) Exchange temporary code for an access token

```
https://github.com/login/oauth/access_token?  
client_id=88cfcad3e03b8dc176bb&  
client_secret=e84a1a2fd82219fe3e6bdf05374586af2f4eb8ce&  
redirect_uri=http://localhost:8080/OAuth2/result.html&  
code=840ddd039d280c591175
```

4) The following JSON will be returned to your redirect uri with the access token

```
{"access_token":  
"8c33a9686c6aaf031eecdc54d2b5ca1d0ed54716",  
"token_type": "bearer", "scope": "user"}
```

# GITHUB OAUTH2 – USE ACCESS TOKENS TO MAKE CALLS TO THE GITHUB API

5) Use access tokens to make calls to the GitHub Api on behalf of the

```
https://api.github.com/user?  
access_token=8c33a9686c6aaf031eecdc54d2b5ca1d0ed54716
```

6) The following JSON will be returned to your redirect uri with the user data

```
{ "login": "superfrog",  
  "id": 2585215,  
  "avatar_url": "https://avatars.githubusercontent.com/u/2585215?v=3",  
  "gravatar_id": "",  
  "url": "https://api.github.com/users/superfrog",  
  "html_url": "https://github.com/superfrog",  
  "followers_url": "https://api.github.com/users/superfrog/followers",  
  ...
```

# RECAP

## **What you should know at this point:**

- The definition of OAuth2
- How OAuth2 is used for authorization
- Basic OAuth2 flow
- How to implement Github OAuth2 API to get user Github data