**[Intro to OAuth 2](https://substack.com/app-link/post?publication_id=2148111&post_id=150824035&utm_source=post-email-title&utm_campaign=email-post-title&isFreemail=true&r=3ufkcp&token=eyJ1c2VyX2lkIjoyMzI1MTMyMjUsInBvc3RfaWQiOjE1MDgyNDAzNSwiaWF0IjoxNzMwMTkxMzM5LCJleHAiOjE3MzI3ODMzMzksImlzcyI6InB1Yi0yMTQ4MTExIiwic3ViIjoicG9zdC1yZWFjdGlvbiJ9.RGhRkq8DOzd7cZ7z3YfpWEhzQTgTZ-zFYoL2uIVXI0s" \t "_blank)**

The OAuth process flow can be confusing to understandwith multiple moving parts and a plethora of terminologies.

Let’s try to make things clearer in this OAuth 2 introduction.

So - what is OAuth?

**OAuth stands for Open Authorization. It’s an authorization framework that lets applications get limited access to user accounts.**

This helps you use OAuth for building your application’s authentication flow.

For example, consider that you are developing a web application and want to implement a login functionality for the users. Instead of rolling your own auth server and managing passwords, you can use something like Google or Github for authentication.

**So - how does it work?**

**OAuth Terminologies**

First, we need to understand the main players involved in the OAuth process. The diagram below shows the key terminologies.

|  |  |  |
| --- | --- | --- |
|  | [A screenshot of a computer  Description automatically generated](https://substack.com/redirect/dfe472bc-81ab-4978-904c-9aa3ed96d865?j=eyJ1IjoiM3Vma2NwIn0.fhp5drYhYSKsls0kZ1EggdcK2U18KkmUpipcx9q_KAQ) |  |

[You can play around with the diagram on Eraser.io](https://substack.com/redirect/5e664609-53b1-4d6d-b98e-84cf298d3ccd?j=eyJ1IjoiM3Vma2NwIn0.fhp5drYhYSKsls0kZ1EggdcK2U18KkmUpipcx9q_KAQ)

**1 - Resource Owner**

This is the user who is going to log in to your application.

The user may have a Google account and this makes them the owner of their account information.

**2 - Client**

The client is the web application you’re building.

This application wants to access the user’s account information (such as name) and authenticate the user as well without worrying about passwords.

In other words, the client is okay if someone like Google or GitHub authenticates the user.

**3 - Resource Server**

This is the server that hosts the account information of users in a protected manner.

For our example, this will be a server owned by Google storing account information.

**4 - Authorization Server**

The authorization server’s job is to verify the identity of the user and issue an access token to the application.

Again, this server belongs to Google.

**The OAuth 2 Process Flow**

With clarity on the various OAuth roles, time to understand basic or abstract OAuth flow.

**STEP 1**

To use OAuth with your web application, you must register your application with the OAuth provider (such as Google).

This is done via a registration form on Google’s developer portal.

You provide information such as:

* Application Name
* Website
* Callback URL

After registration, you typically get two things:

* Client ID
* Client Secret

Both of these values are used to authenticate your web application to Google.

**STEP 2**

The user wants to access your web application.

And you want to authenticate the user.

To facilitate this, the web application provides an authorization code link to the user.

**STEP 3**

When the user clicks the link, Google will ask the user to log in to authenticate their identity.

Once successful, the user will be prompted by Google to authorize or deny the web application access to their account information.

This is done by showing a consent screen where the user can confirm what kind of access is being asked for.

**STEP 4**

Once the user authorizes on the consent screen, Google redirects the user to the callback URL provided by the web application in STEP 1.

While redirecting, it also sends along an authorization code.

**STEP 5**

The web application uses the Authorization Code to get an access token from the Authorization Server

The server checks if the authorization is valid and sends a response containing the access token and a refresh token (optional).

With this, the application is fully authorized.

It uses the access token to fetch the user’s account details. The same token can be used multiple times until it expires.

|  |  |  |
| --- | --- | --- |
|  | [A diagram of a process flow  Description automatically generated](https://substack.com/redirect/5e53b25f-2ce8-4032-93c8-3f1ac21da65d?j=eyJ1IjoiM3Vma2NwIn0.fhp5drYhYSKsls0kZ1EggdcK2U18KkmUpipcx9q_KAQ) |  |

[You can play around with the diagram on Eraser.io](https://substack.com/redirect/5e664609-53b1-4d6d-b98e-84cf298d3ccd?j=eyJ1IjoiM3Vma2NwIn0.fhp5drYhYSKsls0kZ1EggdcK2U18KkmUpipcx9q_KAQ)

Note that there are more applications and flows supported by the OAuth standard. However, the one we discussed is the most used for authentication purposes

**So - have you used OAuth in your project? What was your experience like?**