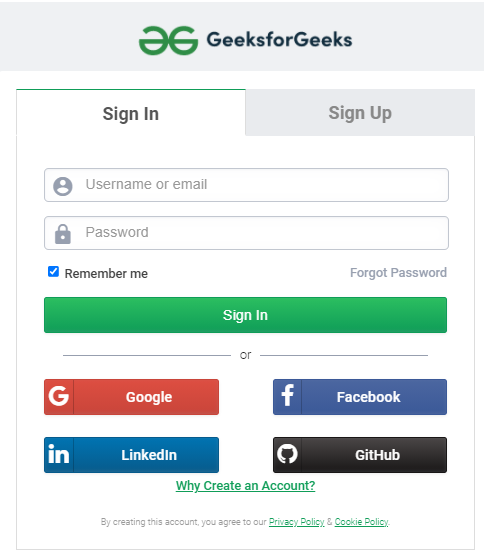
OAuth2.0 is an Open industry-standard authorization protocol that allows a third party to gain limited access to another HTTP service, such as Google, Facebook, and GitHub, on behalf of a user, once the user grants permission to access their credentials.

Most websites require you to complete a registration process before you can access their content. It is likely that you have come across some buttons for logging in with *Google, Facebook*, or another service.



***GeeksforGeeks Registration Page***

Clicking those buttons will get you access to these third-party services without entering any credentials. I’m sure you’re wondering how this happens. ***OAuth****brings this to light.*

Let’s have a quick refresher on **Authentication** and **Authorization** before we dive into OAuth. Authorization refers to the process by which an administrator grants access to authenticated users, whereas authentication verifies that the user is who they claim to be.

Consider the G*eeksforGeeks* website as an example.

As a reader, you can read blogs without authenticating, but to add comments, you must register. Once you’ve signed up, you can access the free courses, improve articles and contribute. As a contributor, you have the right to edit your articles.

*Let us now discuss OAuth.*

**OAuth is an open-standard authorization framework that enables third-party applications to gain limited access to user’s data**.

**Essentially**, **OAuth is about delegated access.**

**Delegation** is a process in which an owner authorizes a service provider to perform certain tasks on the owner’s behalf. Here the task is to provide limited access to another party.

Let’s take two real-life examples;

House owners often approach real estate agents to sell their house. The house owner authorizes the real estate agent by giving him/her the key. Upon the owner’s consent, the agents show the buyers the property. The buyer is welcome to view the property, but they are not permitted to occupy it. In this scenario, the buyer has limited access, and the access is limited by the real estate agent who is acting on the owner’s behalf.

A  classic example of valet parking is often retold to understand this concept. In this case, the car owner has access to both the car and the valet. To have his car parked for him, the car owner gives the valet key to the attendant. The valet key starts the car and opens the driver’s side door but prevents the valet from accessing valuables in the trunk or glove box.

Thus, the Valet key has delegated the task of limiting the access of the valet.

**What is the point of OAuth?**

**OAuth allows granular access levels**. Rather than entrusting our entire protected data to a third party, we would prefer to share just the necessary data with them. Thus, we need a trusted intermediary that would grant limited access(known as scope) to the editor without revealing the user’s credentials once the user has granted permission.(known as consent).

*Here’s an example of an application for editing photos.*

You go to a photo editing app to resize an image. They ask you to upload the image you want to edit from your Google Drive account. The third party only needs access to the single photo you need to edit. Oauth will ensure that the photo editor gets just that.

Let’s take another example, you would like to share your edited picture with your friend, but they must have the same editing software. The editing software cannot request your Google account credentials; instead, it redirects you to your account. If you choose to invite your friend through that app, the app will request access to your Google address book to send the invitation.

* **Read/write only** -A third party can only read your data, not modify it. In some instances, it can also request content modifications on your account. For example, you can cross-post a picture from your Instagram account to your Facebook account.
* **Revoke Access –**You can deauthorize Instagram’s access to your Facebook wall so it can no longer post on your wall.

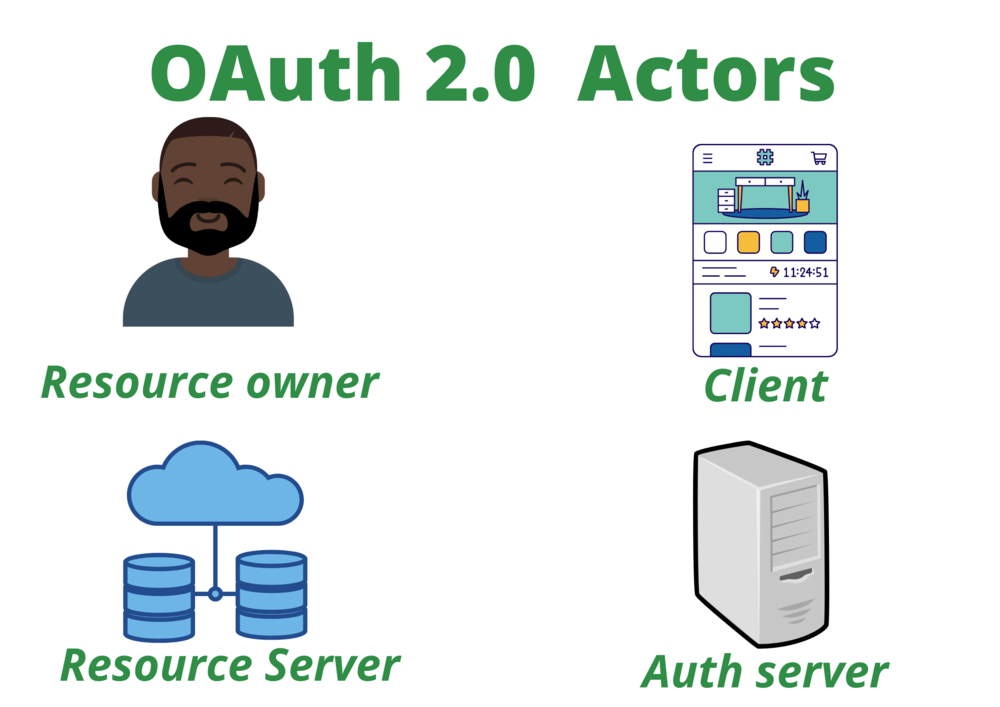
*Before we get into how OAuth works, we’ll discuss the****central components of OAuth****for more clarity.*

The elements of OAuth are listed below:

1. **Actors**
2. **Scopes and Consent**
3. **Tokens**
4. **Flows**

**Actors:**

 OAuth Interactions have the following Actors:



*OAuth2.0 Actors*

* **Resources** are protected data that require OAuth to access them.
* **Resource Owner**: Owns the data in the resource server. An entity capable of granting access to protected data. For example, a user Google Drive account.
* **Resource Server**: The API which stores the data. For example, Google Photos or Google Drive.
* **Client**: It is a third-party application that wants to access your data, for example, a photo editor application.

There seems to be an interaction between two services for accessing resources, but the issue is who is responsible for the security. The resource server, in this case, Google Drive, is responsible for ensuring the required authentication.

**OAuth is coupled with the Resource Server**. Google implements OAuth to validate the authorization of whoever accesses the resource.

* **Authorization Server**: OAuth’s main engine that creates access tokens.

**Scope and Consent:**

The scopes define the specific actions that apps can perform on behalf of the user. They are the bundles of permissions asked for by the client when requesting a token.

For example, we can share our LinkedIn posts on Twitter via LinkedIn itself. Given that it has write-only access, it cannot access other pieces of information, such as our conversations.

On the Consent screen, a user learns who is attempting to access their data and what kind of data they want to access, and the user must express their consent to allow third-party access to the requested data. You grant access to your IDE, such as CodingSandbox, when you link your GitHub account to it or import an existing repository. The Github account you are using will send you an email confirming this.