[OAuth2](https://oauth.net/2/) is a protocol that describes a **stateless** authorization (that means we don’t need to maintain sessions between clients and our server).

This protocol allows third party clients to access protected resources on behalf of the resource owner. There are four basic roles in OAuth2:

* **Resource owner** - the owner of the resource –
* **Resource server** -   the server hosting all the protected resources
* **Client** - the application accessing the resource server
* **Authorization server** - the server that handles issuing access tokens to clients. This could be the same server as the resource server

Furthermore, there are two types of tokens:

* **access token**, which usually has limited lifetime and enables the client to access protected resources by including this token in the request header
* **refresh token** with longer lifetime used to get a new access token once it expires (without the need of sending credentials to the server again)

It is important to note, that OAuth2 should be used with HTTPS because it requires the client to exchange sensitive information with the server (tokens or credentials).

Clients need to be registered with the authorization server in order to receive their client-id and client-secret which are later used when requesting the access tokens. Each token has a **scope** which is defined by the user when communicating with the authorization server (e.g. the user authorizes the client application to access certain resources on the resource server).

There are four different **grant types** defined by OAuth2. These grant types define interactions between the client and the auth/resource server. More detailed information can be found [here](https://alexbilbie.com/guide-to-oauth-2-grants/).

* **Authorisation code** - redirection-based flow for confidential client, the client communicates with the server via user-agent (web browser etc.), typical for web servers
* **Implicit** - typically used with public clients running in a web browser using a scripting language, does not contain refresh tokens, this grant does not contain authentication and relies on redirection URI specified during client registration to the auth server
* **Resource owner password credentials** - used with trusted clients (e.g. clients written by the same company that owns the auth server), user credentials are passed to the client and then to the auth server and exchanged for access and refresh tokens
* **Client credentials** - used when the client itself is the resource owner (one client does not operate with multiple users), client credentials are exchanged directly for the tokens

**Access Token And Refresh Token:**

This two types of token are provided by your authorization server. access\_token is responsible for accessing your resources from the resource server. This token usually has a little validity time. You can access your data with this token a certain time before it get’s expired. So after it expires, you need to request Authorization server for a new access\_token with your refresh token, client id, and client secret, so that you don’t need to send user credentials again and again. Refresh token has more validation time than Access Token. Typically 7-90 days, depends on you.

So we can say,

1. The responsibility of access token is to access data before it gets expired.
2. The responsibility of Refresh Token is to request for a new access token when the access token is expired.

