* OpenID is an open standard and decentralized authentication protocol. Promoted by the non-profit OpenID Foundation, it allows users to be authenticated by co-operating sites (known as relying parties, or RP) using a third-party service, eliminating the need for webmasters to provide their own ad hoc login systems, and allowing users to log into multiple unrelated websites without having to have a separate identity and password for each.
* An end-user is the entity that wants to assert a particular identity. A relying party (RP) is a web site or application that wants to verify the end-user's identifier. Other terms for this party include "service provider" or the now obsolete "consumer". An identity provider, or OpenID provider (OP) is a service that specializes in registering OpenID URLs or XRIs. OpenID enables an end-user to communicate with a relying party. This communication is done through the exchange of an identifier or OpenID, which is the URL or XRI chosen by the end-user to name the end-user's identity.
* **OAuth2.0:** ([https://auth0.com/docs/protocols/protocol-oauth2#authorization-endpoint](https://auth0.com/docs/protocols/protocol-oauth2%23authorization-endpoint))  
  The [OAuth 2.0](https://tools.ietf.org/html/rfc6749) authorization framework is a protocol that allows a user to grant a third-party web site or application access to the user's protected resources, without necessarily revealing their long-term credentials or even their identity.

OAuth 2.0 uses two endpoints: the /authorize endpoint and the /oauth/token endpoint.

**An OAuth 2.0 flow has the following roles:**

* Resource Owner: Entity that can grant access to a protected resource. Typically, this is the end-user.
* Resource Server: Server hosting the protected resources. This is the API you want to access.
* Client: Application requesting access to a protected resource on behalf of the Resource Owner.
* Authorization Server: Server that authenticates the Resource Owner and issues access tokens after getting proper authorization. In this case, Auth0.
* **PKCE** (RFC 7636: Proof Key for Code Exchange)

tools.ietf.org/html/rfc7636

PKCE (RFC 7636) is an extension to the Authorization Code flow to prevent certain attacks and to be able to securely perform the OAuth exchange from public clients.

* Auth0 audience:

Auth0 supports only one resource identifier (“audience”) per access token, but an access token can carry multiple scopes (and custom claims).

* **What is OpenID Connect?** [(https://openid.net/connect/)]((https:/openid.net/connect/))

OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It allows Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.

OpenID Connect allows clients of all types, including Web-based, mobile, and JavaScript clients, to request and receive information about authenticated sessions and end-users. The specification suite is extensible, allowing participants to use optional features such as encryption of identity data, discovery of OpenID Providers, and session management, when it makes sense for them.

See https://openid.net/connect/faq/ for a set of answers to Frequently Asked Questions about OpenID Connect.

**How is OpenID Connect different than OpenID 2.0?**

OpenID Connect performs many of the same tasks as OpenID 2.0, but does so in a way that is API-friendly, and usable by native and mobile applications. OpenID Connect defines optional mechanisms for robust signing and encryption. Whereas integration of OAuth 1.0a and OpenID 2.0 required an extension, in OpenID Connect, OAuth 2.0 capabilities are integrated with the protocol itself.

* Difference Between OAUTH, OpenID and OPENID Connect in very simple term?

<https://security.stackexchange.com/questions/44611/difference-between-oauth-openid-and-openid-connect-in-very-simple-term>

* <https://developer.okta.com/blog/2019/10/21/illustrated-guide-to-oauth-and-oidc>
* Why You Should Never Use the Implicit Flow Again?

<https://developer.okta.com/blog/2019/08/22/okta-authjs-pkce>

* <https://www.pingidentity.com/en/company/blog/posts/2018/securely-using-oidc-authorization-code-flow-public-client-single-page-apps.html>
* <https://auth0.com/docs/quickstart/webapp/aspnet-core/01-login>