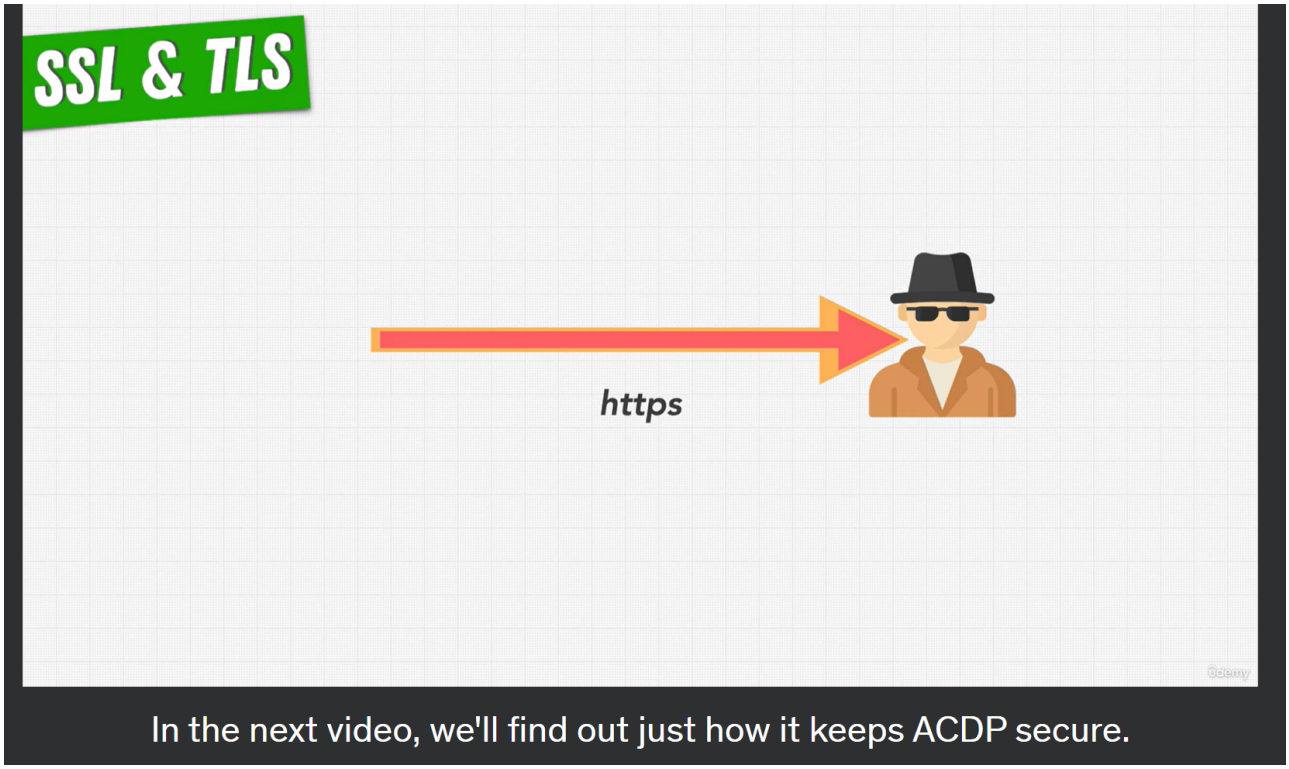
Fundamentals Node Security + Authentication

1. tls last version of the ssl

2. wrapping request tls encrypting



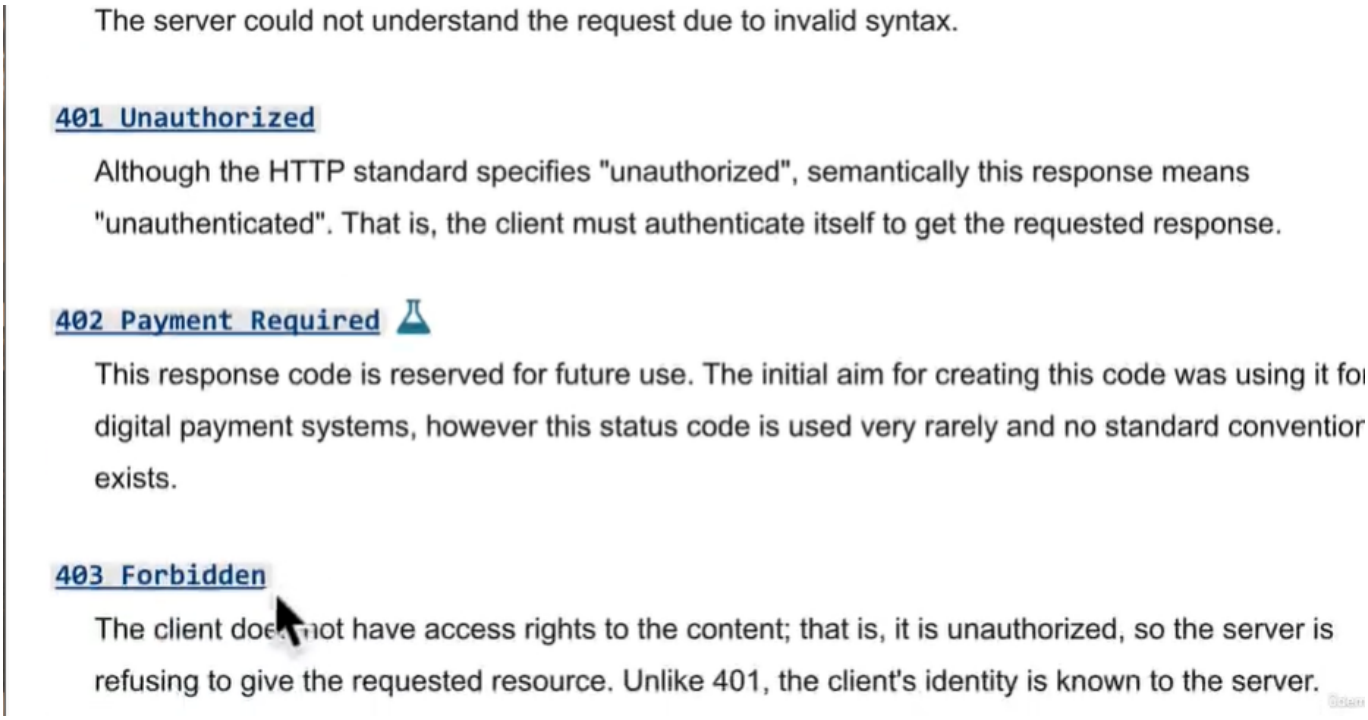
3. If need created ssl or tls certificate

Authentication

when need now who this user

Authorization

checks whether that user has permission to access a specific resource once they've been authenticated access control



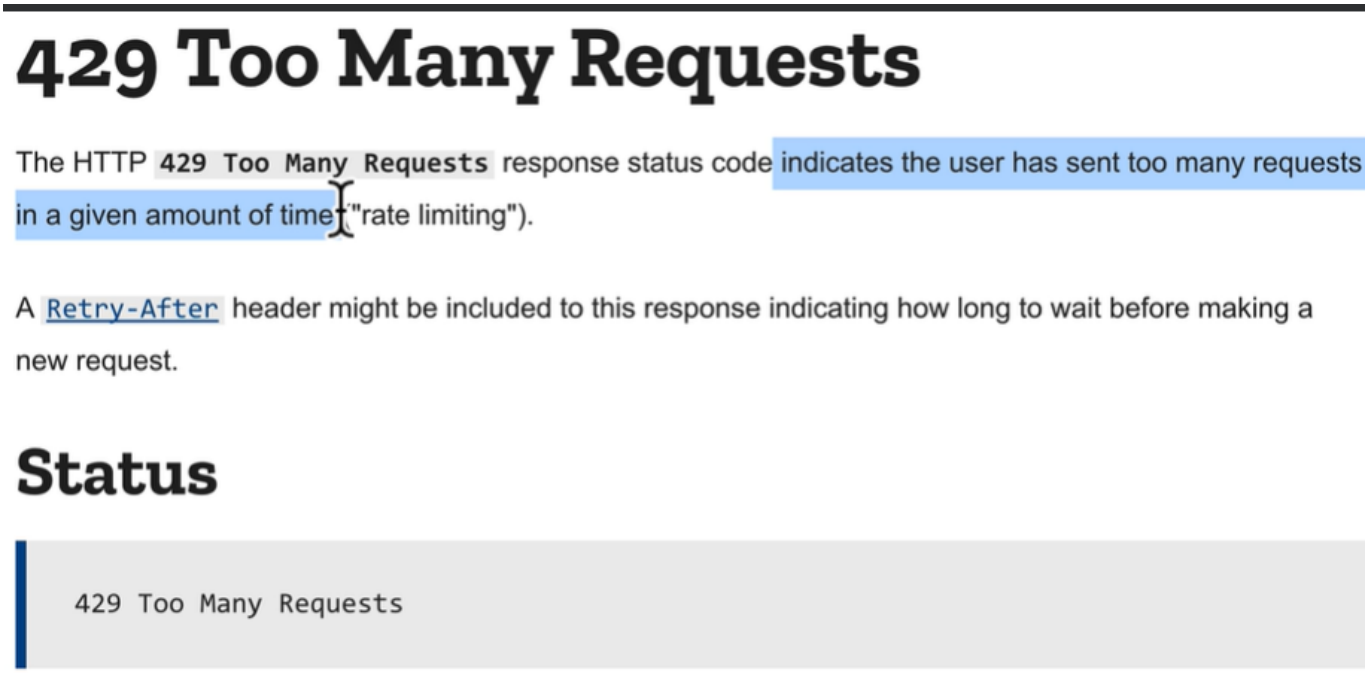
404 permission 401 authenticate

Api key

It's a string

passing as either a query parameter or as a header in http request

---



goole maps

Adding the API key to your request

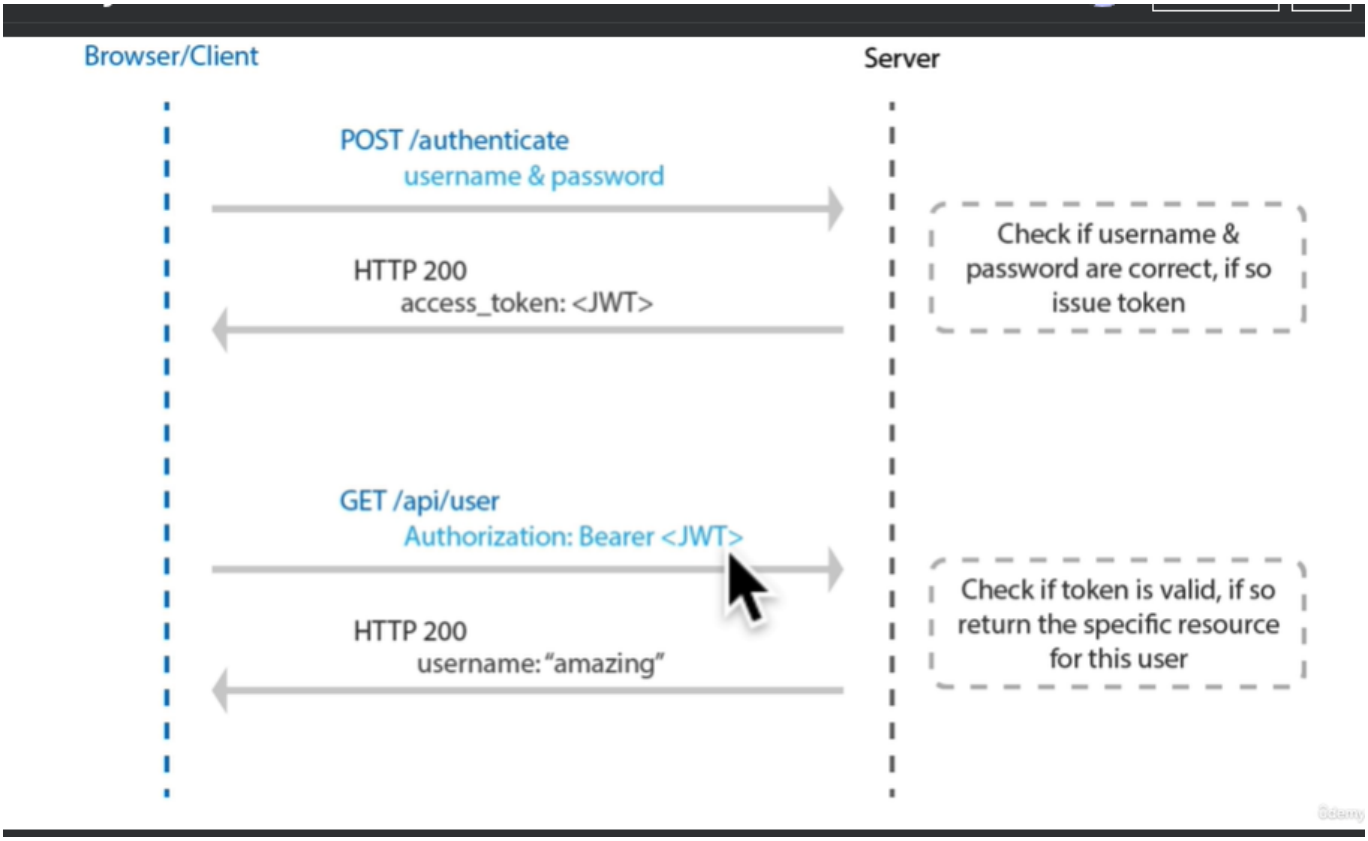
You must include an API key with every Maps JavaScript API request. In the following example, replace `YOUR_API_KEY` with your API key.

```
<script async defer src="https://maps.googleapis.com/maps/api/?key=YOUR_API_KEY&callback=initM
type="text/javascript"></script>
```

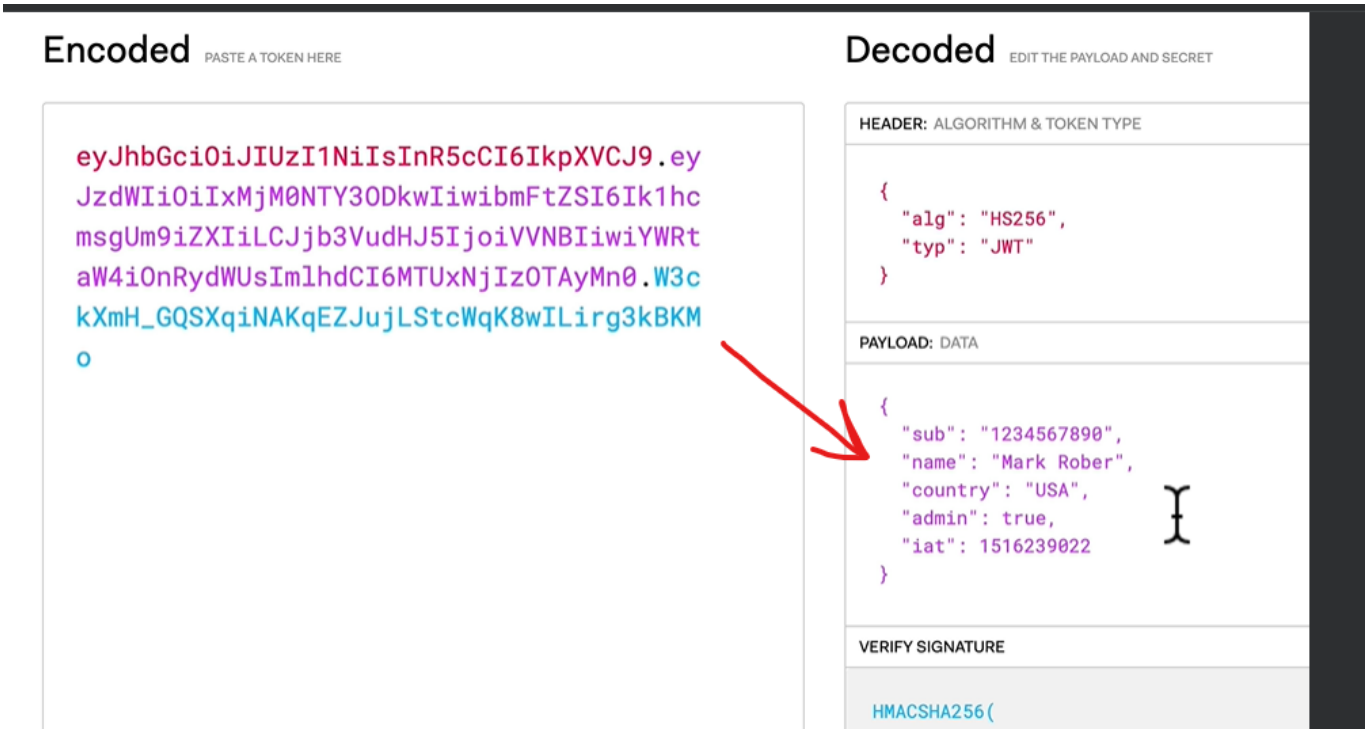
HTTPS is required for requests that use an API key, and recommended for requests that use a client ID. HTTPS is also required for applications that include sensitive user data - such as a user's location - in requests.

JSON Web token

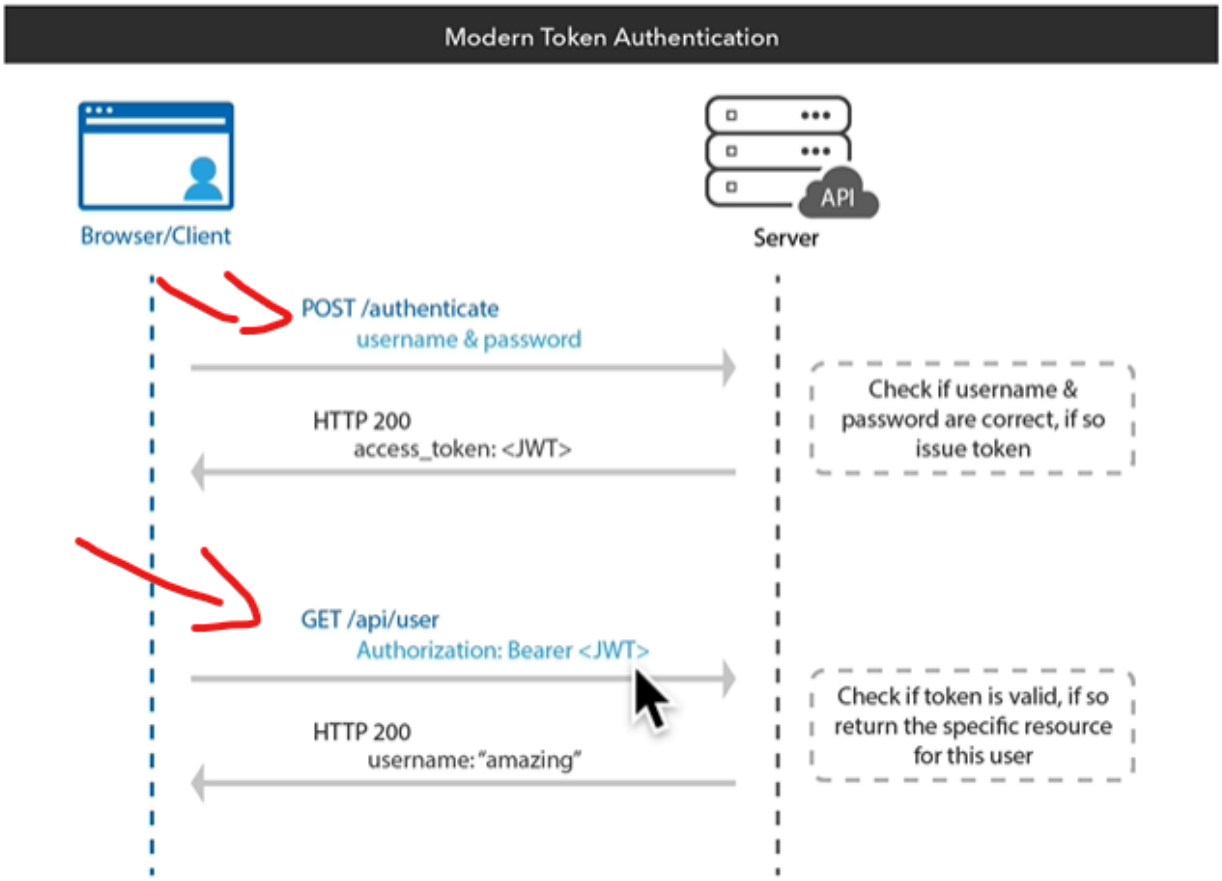
two types JWT || Opaque tokes



JWT encoded information



split to three part header -> alg (HS256) and type payload -> signature ->

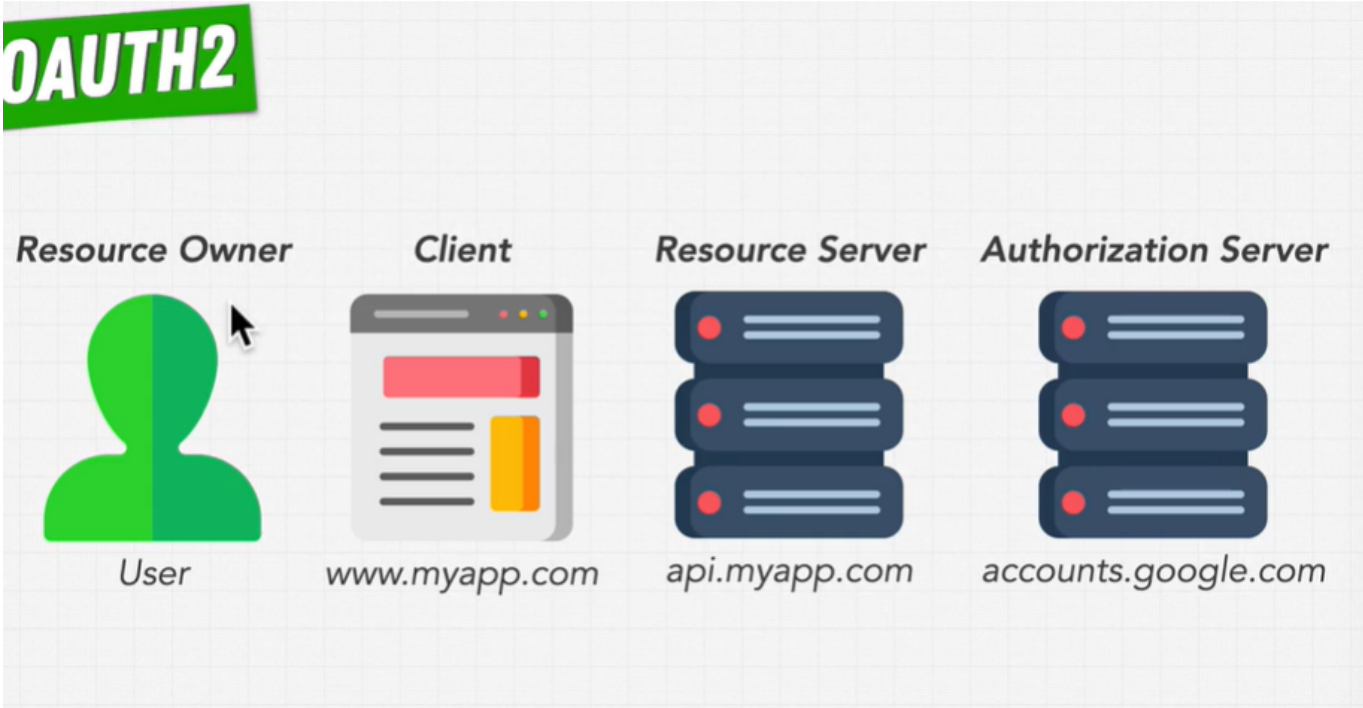


---

data that's constantly being passed around from client to the server.

The OAUTH 2.0 authentication standard

resource owner -> user client -> my website resource server -> your backend in your application
Authorization server -> goole service

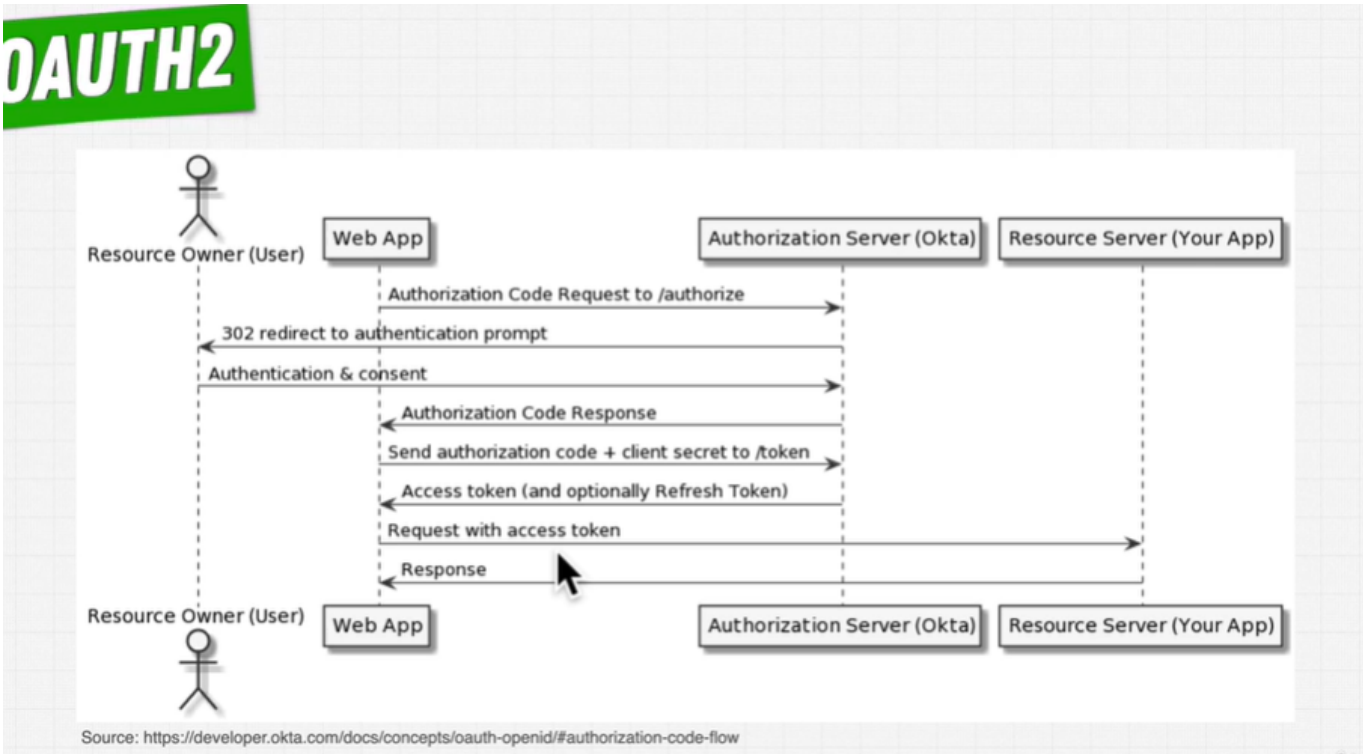


1. has difference flows

The table shows you which OAuth 2.0 flow to use for the type of application that you are building.

Type of Application	OAuth 2.0 flow
Server-side (AKA Web)	Authorization Code flow
Single-Page Application	Authorization Code flow with PKCE or Implicit flow when the SPA that you are building runs in older browsers that don't support Web Crypto for PKCE
Native	Authorization Code flow with PKCE
Trusted	Resource Owner Password flow

Server-side (AKA web)


OAUTH2
Resource Owner (User)
Web App
Authorization Server (Okta)
Resource Server (Your App)
Authorization Code Request to /authorize
302 redirect to authentication prompt
Authentication & consent
Authorization Code Response
Send authorization code + client secret to /token
Access token (and optionally Refresh Token)
Request with access token
Response
Resource Owner (User)
Web App
Authorization Server (Okta)
Resource Server (Your App)
Source: https://developer.okta.com/docs/concepts/oauth-openid/#authorization-code-flow