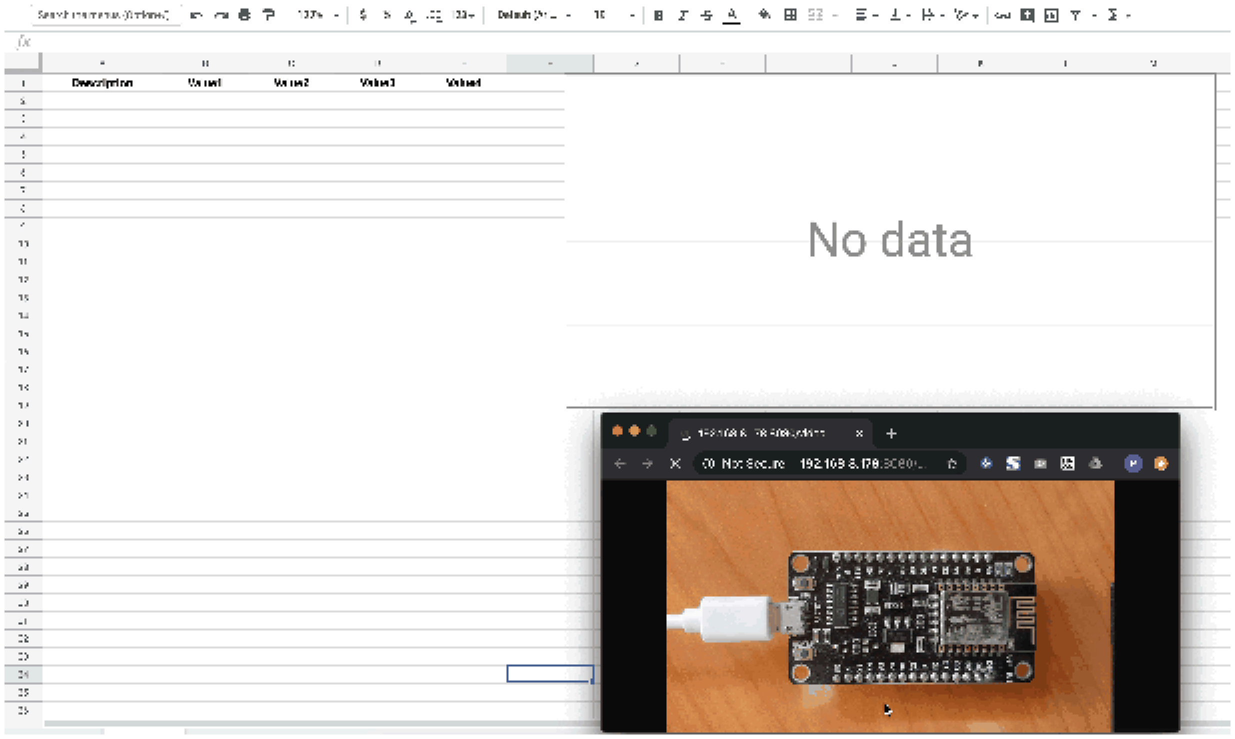
Google Oauth2 API Explained

Pumudu Ruhunage · Follow
6 min read · Apr 13, 2020

48



ESP8266 based google sheet data feeder for one of my home projects (Based on Google APIs)

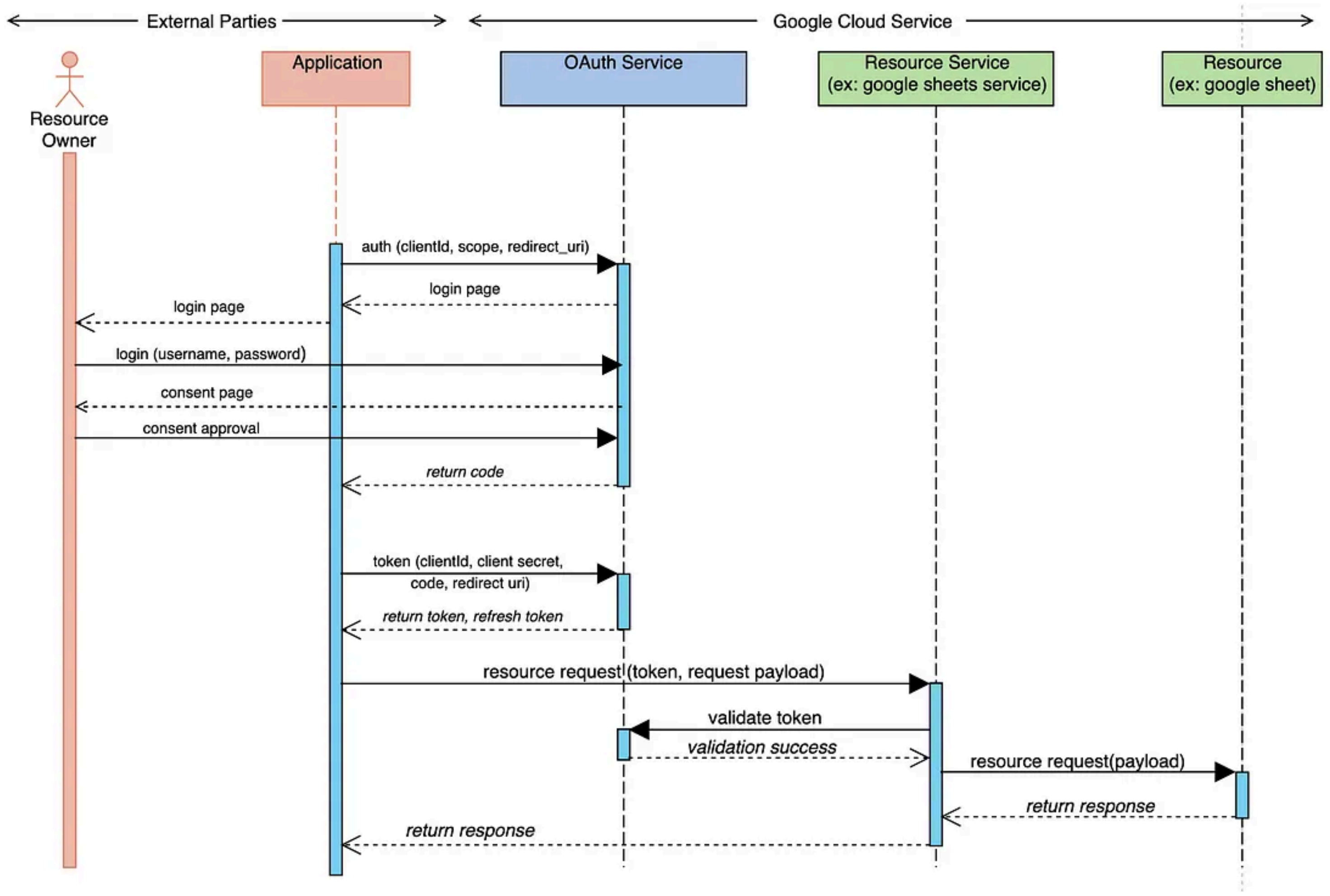
As we all know Google is a powerful cloud service provider with numerous products and services. For application developers there's even more offered by Google. It provides more than 280+ (as of march 2020) API's from maps to machine learning. This opens up various opportunities for app developers to utilize these services within their applications/projects.

Google Cloud APIs only accept requests from *registered applications*, which are uniquely identifiable applications that present a credential at the time of the request. Requests from anonymous applications are rejected.

There are several methods to authenticate for Google API platform. As for web applications most common method is to use Oauth2 client credentials. Google has several documentations explaining the Oauth2 flow with examples.

In this article i'll try to explain the Google Oauth2 flow in most simple request/response manner to understand the flow. (This comes in handy when invoke google api's for IoT use cases with minimum resource utilization)

Google OAuth Flow — In Theory


External Parties
Google Cloud Service
Resource Owner
Application
OAuth Service
Resource Service (ex: google sheets service)
Resource (ex: google sheet)
auth (clientId, scope, redirect_uri)
login page
login page
login (username, password)
consent page
consent approval
return code
token (clientId, client secret, code, redirect uri)
return token, refresh token
resource request (token, request payload)
validate token
validation success
resource request(payload)
return response
return response

Google OAuth Flow — Sequence Diagram

Above sequence diagram sums up the flow in theory. In my opinion this flow can't be any simplified with the amount of services google exposes.

Following are the main parties involving the OAuth flow.

Application : Application is the entity or which request access to certain google resource owned by different party.

Resource Owner : Resource owner is the person/organization whom own the given google resource. This person/organization should allow access to application via login page and consent to access the resources to complete the flow.

OAuth Service : Google OAuth service have two endpoints ('/auth' and '/token').

"/auth" endpoint provides short lived "authorization code" which confirms user credentials and consent is valid for given scope.

"/token" endpoint exchange code and return bearer and refresh token. With this bearer token and refresh token, application can access the consented resource by resource owner.

Resource Service : Resource service exposes the google resources via structured, well documented set of APIs. Once a request received for this service it will verify the token and execute given request if token is valid.

Resource : This is the actual resource which exposes via APIs. For example, google sheet which creates and maintains by the resource owner is a resource.

Google OAuth Flow — In Practice

To explain the Oauth flow i'll be using google sheets api. For that, before begin we need a google sheet resource to access.

Create google sheet document and note down the sheet id and tab name. Add some dummy data between A1:D3 cells which we'll be reading via google sheets api later on.

Google sheet Id can be seen in url as below :

> *https://docs.google.com/spreadsheets/d/**190Ym4DG9s7Nk_YxA8cr1D_8VEmNpCnYsskyRgAxdadc**/edit#gid=0*

Sheet Id : **190Ym4DG9s7Nk_YxA8cr1D_8VEmNpCnYsskyRgAxdadc**

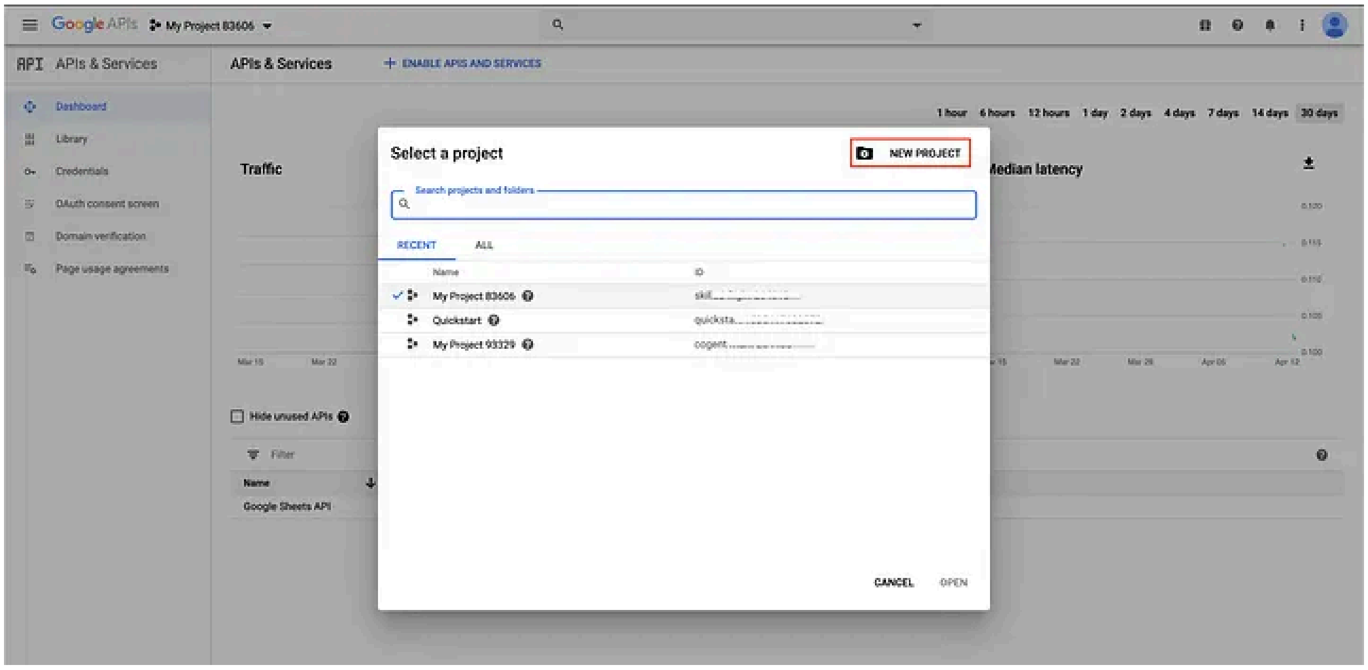
Tab Name : **Sheet1**

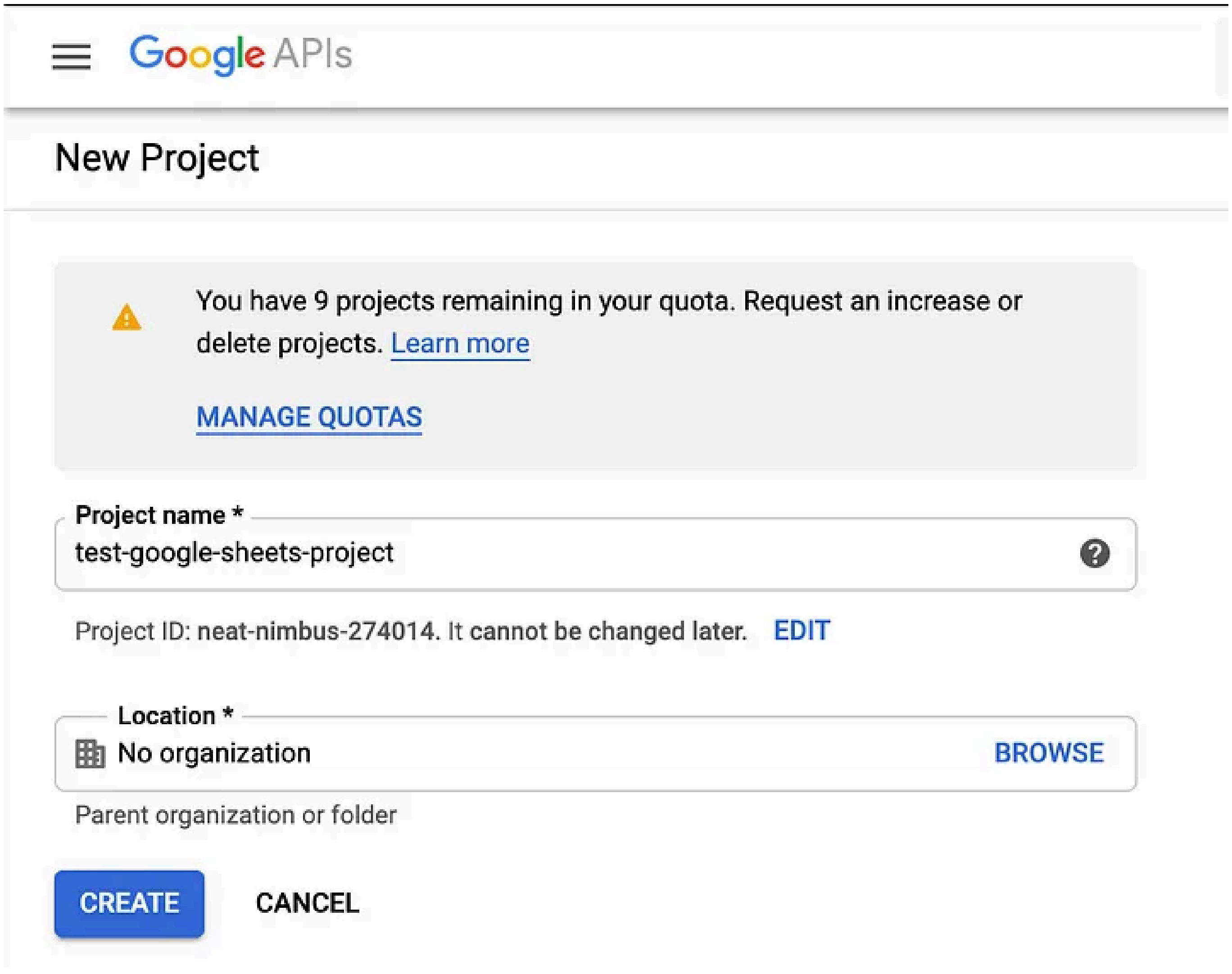
Step 1 : Create Project and Retrieve Client Credentials for Application

In-order to get the client Id and client secret for the application we need to create the application via google api console.

1. ogin to google account and open google api dashboard.

2. Create new project if not already created, with appropriate name.





You have 9 projects remaining in your quota. Request an increase or delete projects. Learn more

MANAGE QUOTAS

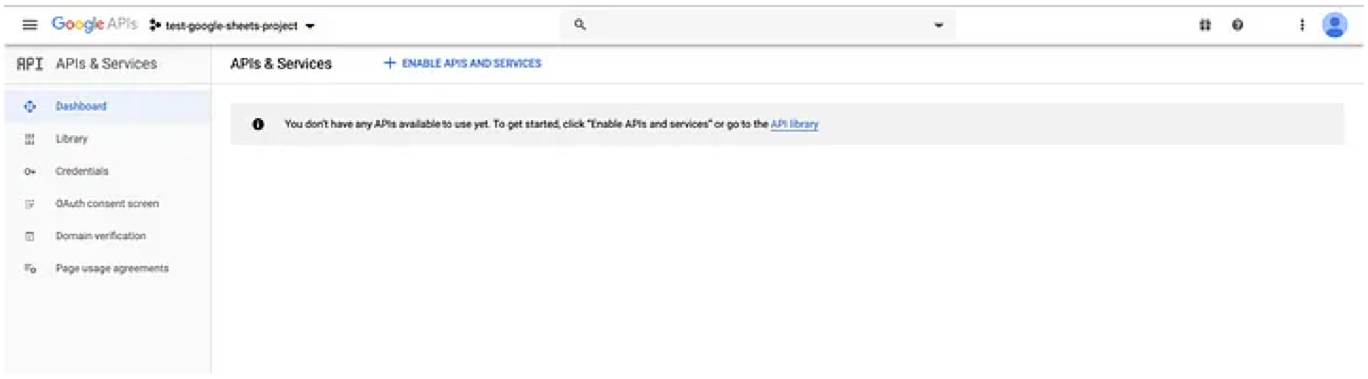
Project name *
test-google-sheets-project

Project ID: neat-nimbus-274014. It cannot be changed later. EDIT

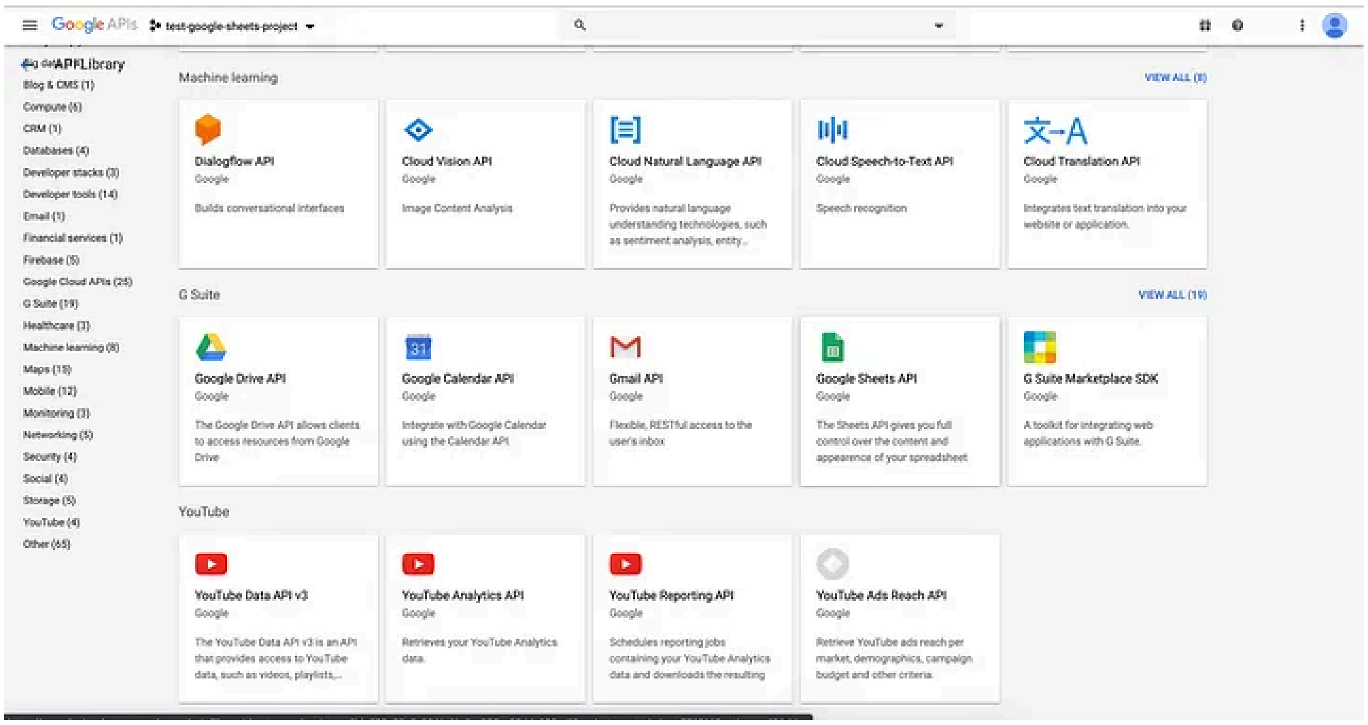
Location *
No organization BROWSE

Parent organization or folder

3. Enable API service(s) for the given Project. (in this example we'll be using google sheets api)



You don't have any APIs available to use yet. To get started, click "Enable APIs and services" or go to the API library



Dialogflow API
Google
Builds conversational interfaces

Cloud Vision API
Google
Image Content Analysis

Cloud Natural Language API
Google
Provides natural language understanding technologies, such as sentiment analysis, entity…

Cloud Speech-to-Text API
Google
Speech recognition

Cloud Translation API
Google
Integrates text translation into your website or application.

Google Drive API
Google
The Google Drive API allows clients to access resources from Google Drive

Google Calendar API
Google
Integrate with Google Calendar using the Calendar API.

Gmail API
Google
Flexible, RESTful access to the user's inbox

Google Sheets API
Google
The Sheets API gives you full control over the content and appearence of your spreadsheet

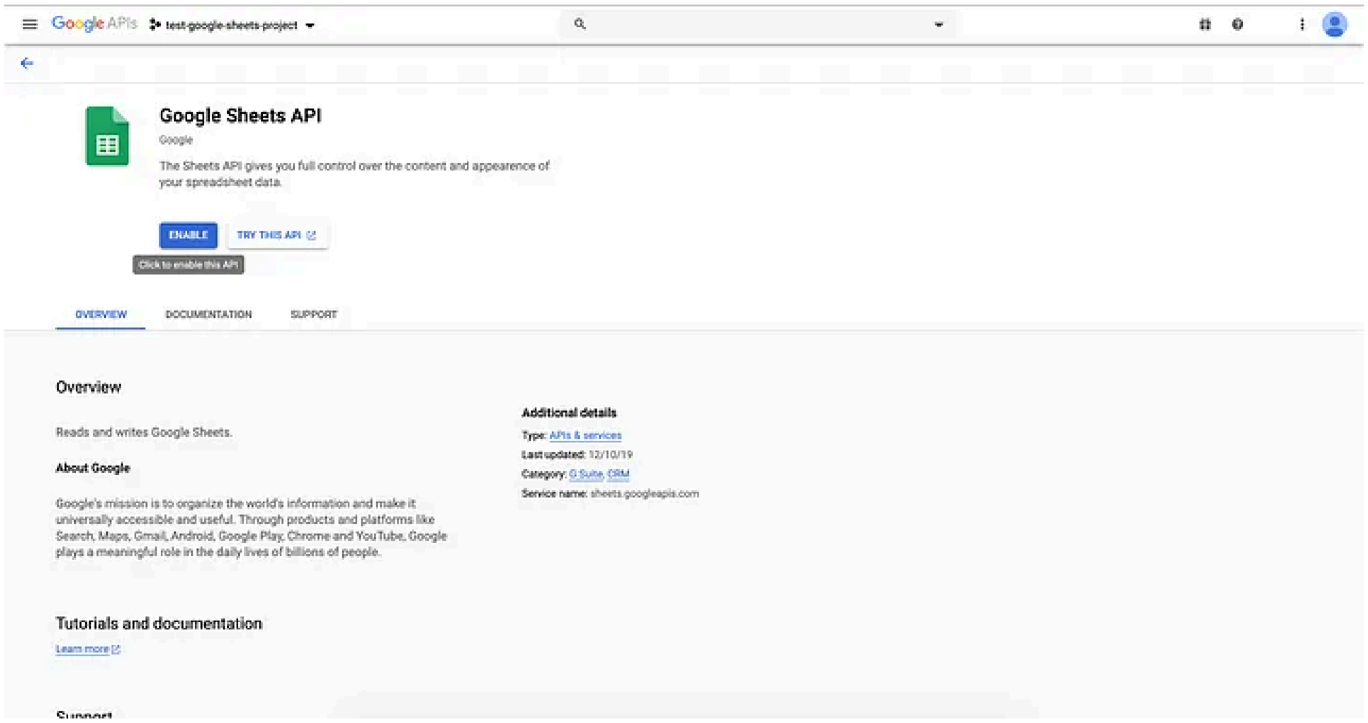
G Suite Marketplace SDK
Google
A toolkit for integrating web applications with G Suite.

YouTube Data API v3
Google
The YouTube Data API v3 is an API that provides access to YouTube data, such as videos, playlists,…

YouTube Analytics API
Google
Retrieves your YouTube Analytics data.

YouTube Reporting API
Google
Schedules reporting jobs containing your YouTube Analytics data and downloads the resulting

YouTube Ads Reach API
Google
Retrieve YouTube ads reach per market, demographics, campaign budget and other criteria.



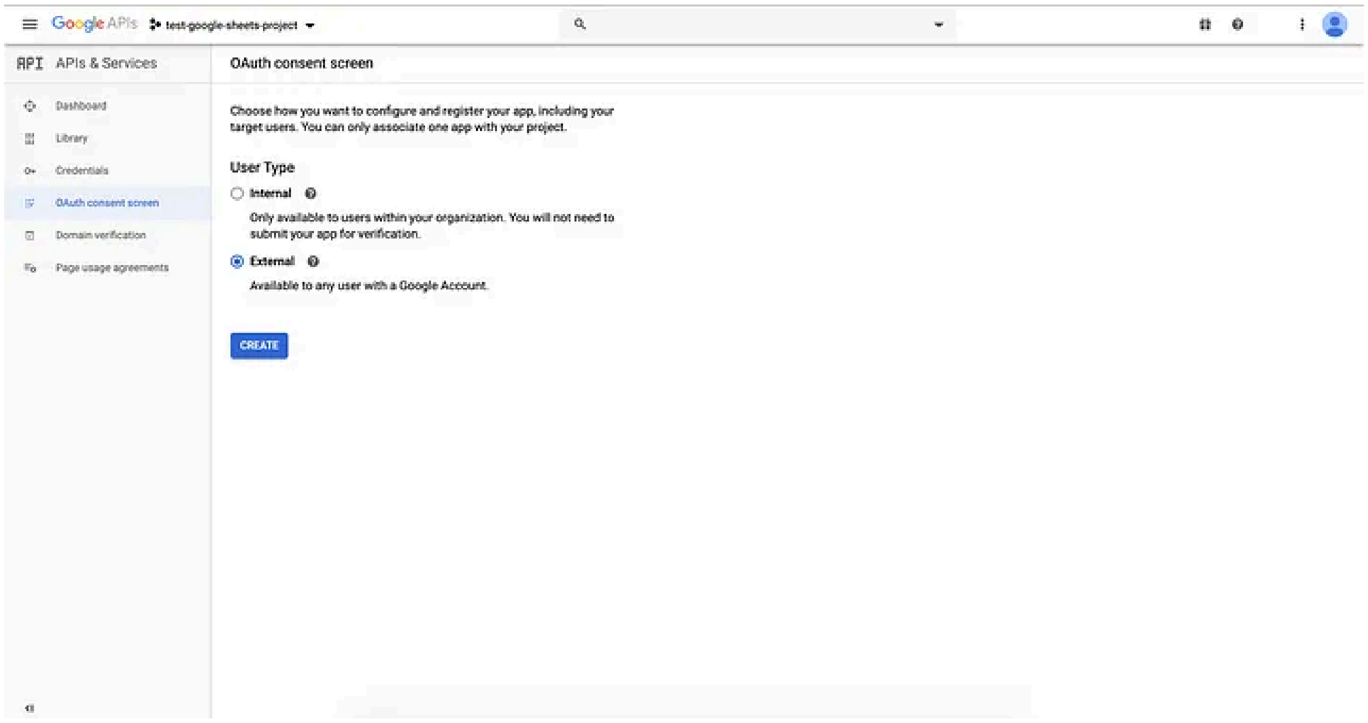
The Sheets API gives you full control over the content and appearence of your spreadsheet data.

Reads and writes Google Sheets.

Google's mission is to organize the world's information and make it universally accessible and useful. Through products and platforms like Search, Maps, Gmail, Android, Google Play, Chrome and YouTube, Google plays a meaningful role in the daily lives of billions of people.

Type: APIs & services
Last updated: 12/10/19
Category: G Suite, CRM
Service name: sheets.googleapis.com

4. Create Oauth consent screen under "OAuth consent screen" tab.



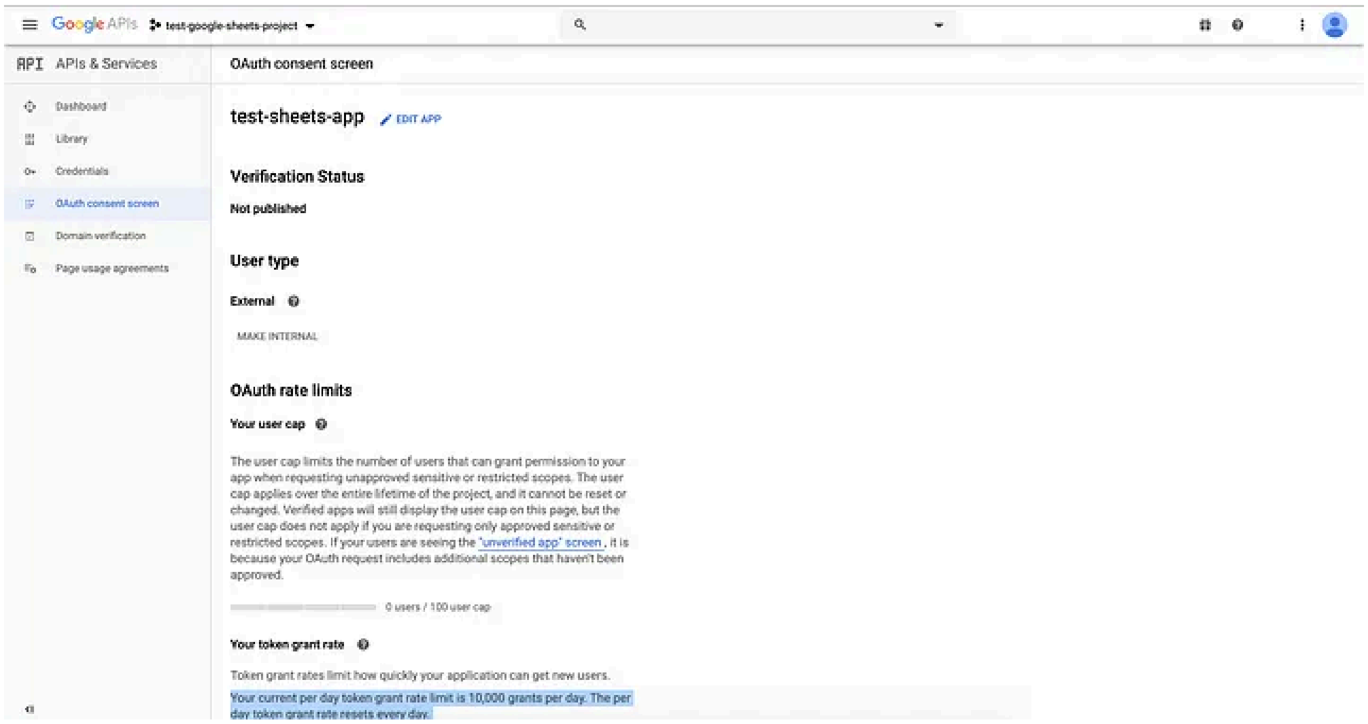
Choose how you want to configure and register your app, including your target users. You can only associate one app with your project.

( ) Internal
Only available to users within your organization. You will not need to submit your app for verification.

(●) External
Available to any user with a Google Account.

Since I just need to access my own test google sheet, i'll be selecting external unverified consent screen.

On create, just provide the name of the application(any name) and click on save without any app verification.



OAuth consent screen

test-sheets-app

Verification Status

Not published

User type

External

OAuth rate limits

Your user cap

The user cap limits the number of users that can grant permission to your app when requesting unapproved sensitive or restricted scopes. The user cap applies over the entire lifetime of the project, and it cannot be reset or changed. Verified apps will still display the user cap on this page, but the user cap does not apply if you are requesting only approved sensitive or restricted scopes. If your users are seeing the 'unverified app' screen, it is because your OAuth request includes additional scopes that haven't been approved.

0 users / 100 user cap

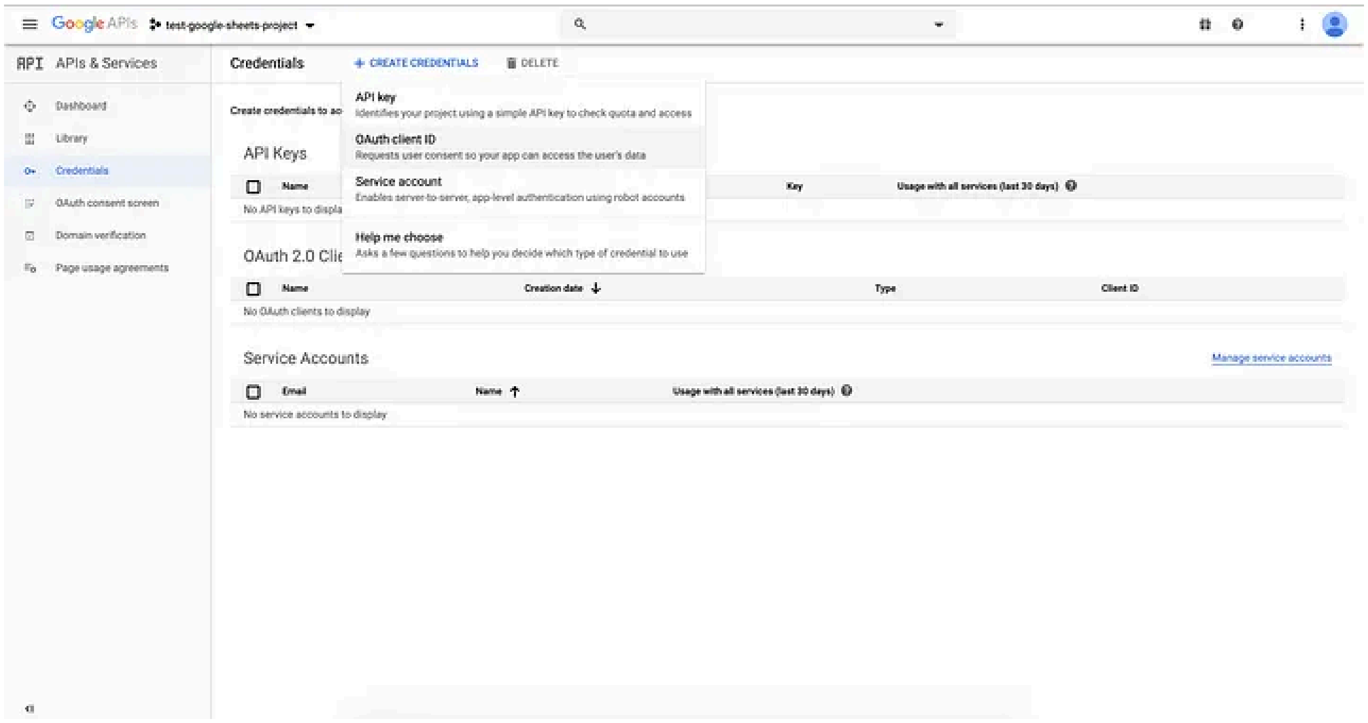
Your token grant rate

Token grant rates limit how quickly your application can get new users.

Your current per day token grant rate limit is 10,000 grants per day. The per day token grant rate resets every day.

Unverified applications are limited to 10,000 users per day, but it’s more than sufficient for this OAuth test.

In-order to authenticate with google we need client credentials. For that, OAuth client ID needs to be created as shown below.

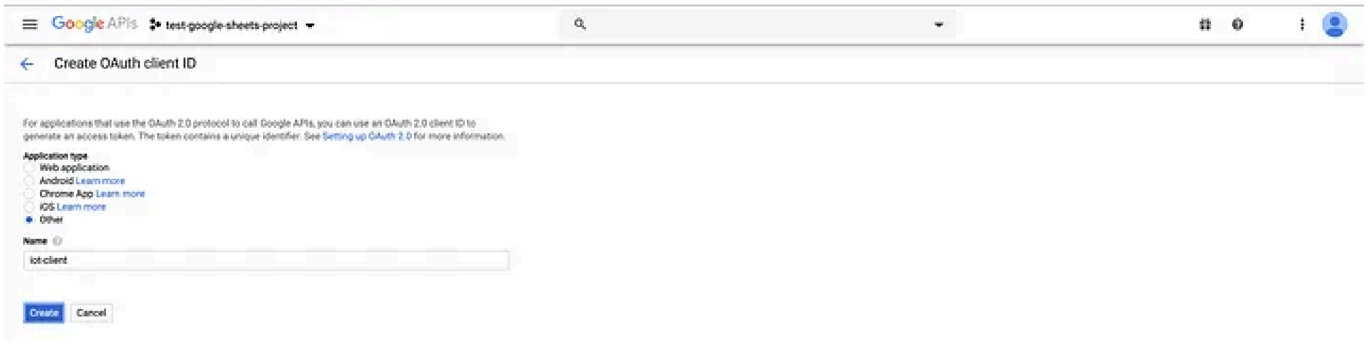


API key
Identifies your project using a simple API key to check quota and access

OAuth client ID
Requests user consent so your app can access the user's data

Service account
Enables server-to-server, app-level authentication using robot accounts

Help me choose
Asks a few questions to help you decide which type of credential to use



For applications that use the OAuth 2.0 protocol to call Google APIs, you can use an OAuth 2.0 client ID to generate an access token. The token contains a unique identifier. See Setting up OAuth 2.0 for more information.

Application type
- Web application
- Android Learn more
- Chrome App Learn more
- iOS Learn more
- [✓ Other]

Name
iot-client

Based on the requirement we can create several types of client credentials, For my usecase, I created basic credentials with ‘other’ option.

Once created, this will provide clientId and secret for the given application. Note down these **clientId** and **client secret** values for next steps of the Oauth flow.

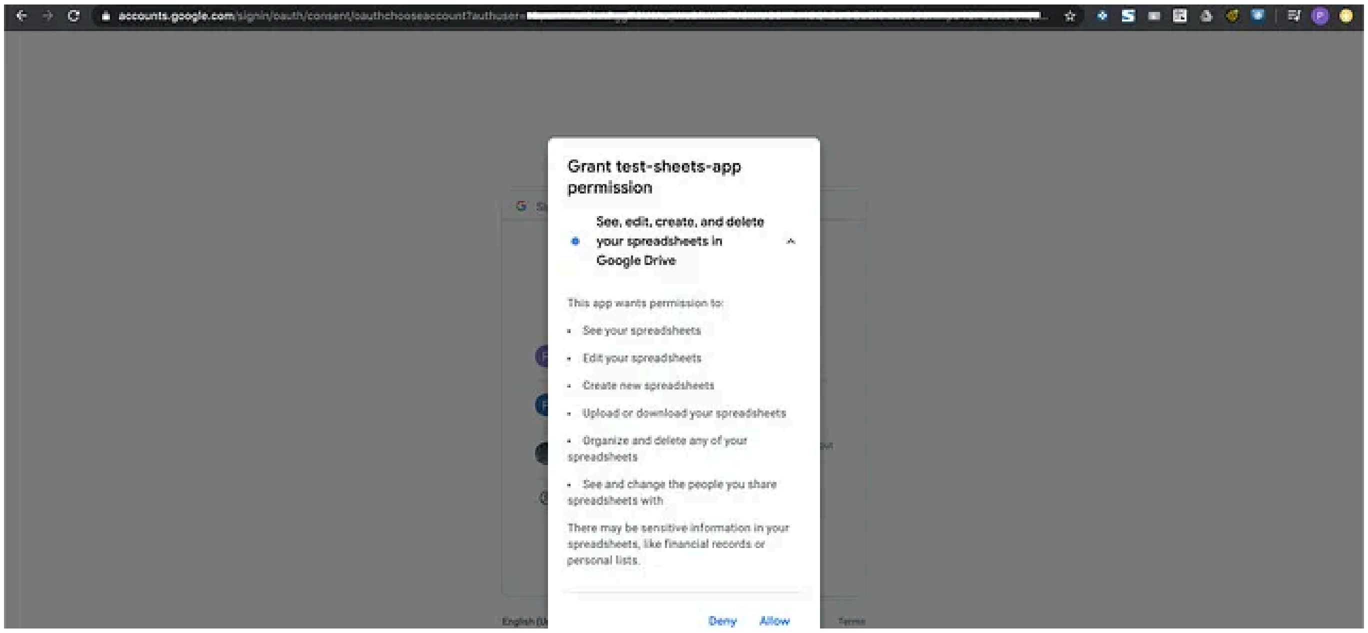
Step 2 : Get the authorization code

Open a browser tab and enter following url with query parameters.

scope : https%3A//www.googleapis.com/auth/spreadsheets
access_type: offline
response_type: code
redirect_uri: http%3A//localhost:8000
client_id: 985957988504-sgp8fm02ob4keorekf2t544u765961l1.apps.googleusercontent.com

```
https://accounts.google.com/o/oauth2/v2/auth?
scope=https%3A//www.googleapis.com/auth/spreadsheets&access_type=offl
ine&response_type=code&redirect_uri=http%3A//localhost:8000&client_id
=985957988504-
sgp8fm02ob4keorekf2t544u765961l1.apps.googleusercontent.com
```

If all goes well this should redirect to a google login screen and consent screens.


Grant test-sheets-app permission

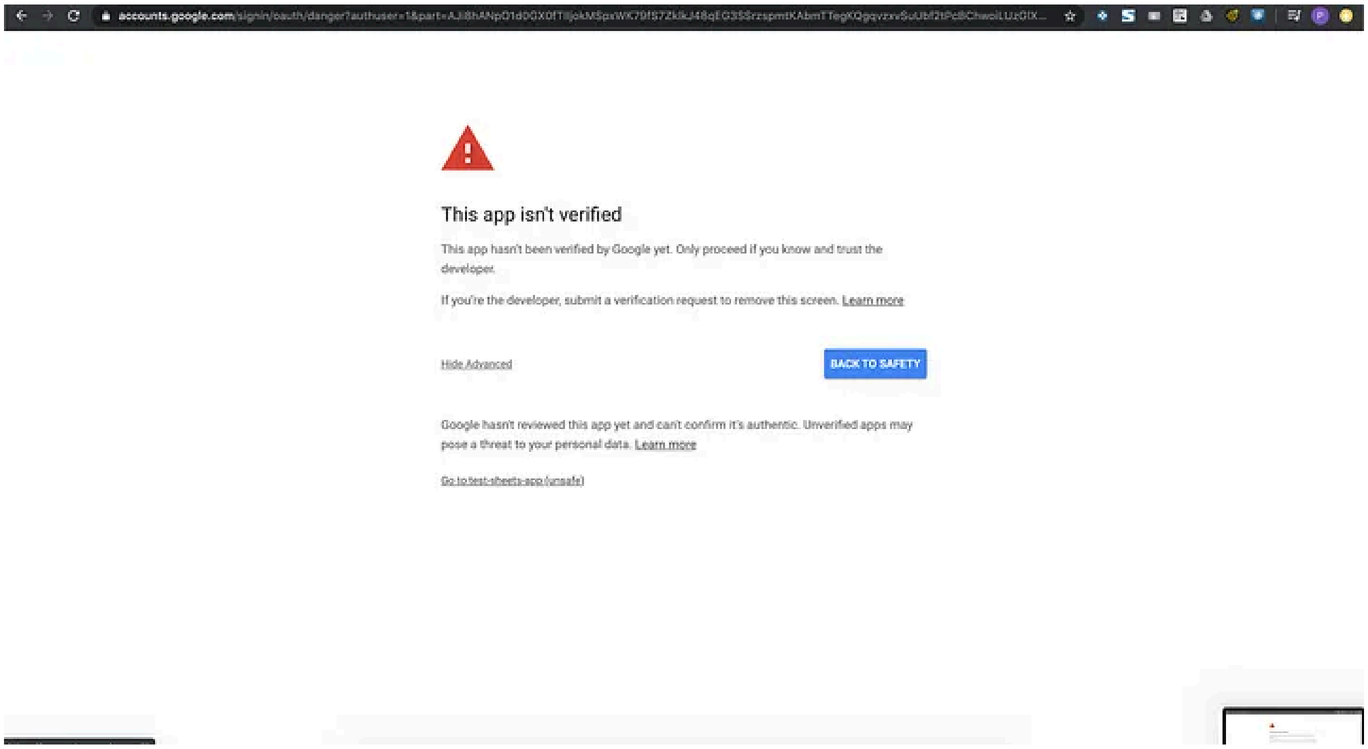
See, edit, create, and delete your spreadsheets in Google Drive

This app wants permission to:

- See your spreadsheets
- Edit your spreadsheets
- Create new spreadsheets
- Upload or download your spreadsheets
- Organize and delete any of your spreadsheets
- See and change the people you share spreadsheets with

There may be sensitive information in your spreadsheets, like financial records or personal lists.

Since we have used unverified consent screen it will throw a warning as well highlighting this app is not yet verified.


This app isn't verified

This app hasn't been verified by Google yet. Only proceed if you know and trust the developer.

If you're the developer, submit a verification request to remove this screen. Learn more

Google hasn't reviewed this app yet and can't confirm it's authentic. Unverified apps may pose a threat to your personal data. Learn more

Go to test-sheets-app (unsafe)

Once consent is granted, page will then get redirected to above given redirect_uri with the code.

localhost:8000/?code=4/ygHeyskWdKW58DPow-a41rn3IZqXB2y16ga3UBBmFI5BijDPmMfogUuA1CXbEtJJpT_G1tnhC9hbMwMR45GT_vw&scope=https://www.googleapis.com

```
http://localhost:8000/?code=4/ygHeyskWdKW58DPow-
a41rn3IZqXB2y16ga3UBBmFI5BijDPmMfogUuA1CXbEtJJpT_G1tnhC9hbMwMR45GT_vw
&scope=https://www.googleapis.com/auth/spreadsheets
```

Authorization code : 4/ygHeyskWdKW58DPow-a41rn3IZqXB2y16ga3UBBmFI5BijDPmMfogUuA1CXbEtJJpT_G1tnhC9hbMwMR45GT_vw

Note down the **code** returned with the redirect url.

Step 3 : Exchange authorization code with bearer token and refresh token

Once we have the authorization code, it's time to retrieve bearer token from token endpoint.

Following POST request is required to retrieve tokens from authorization code.

```
POST https://oauth2.googleapis.com/token

redirect_uri='http://localhost:8000'

code=4/ygHeyskWdKW58DPow-
a41rn3IZqXB2y16ga3UBBmFI5BijDPmMfogUuA1CXbEtJJpT_G1tnhC9hbMwMR45GT_vw
```

```
client_id=985957988504-
sgp8fm02ob4keorekf2t544u765961l1.apps.googleusercontent.com

client_secret=ZMdFZHB8LbJllKMxWUGrKTjh

grant_type=authorization_code
```

I'll be using curl commands to invoke OAuth requests as below. This can be done via postman, python scripts etc.

Curl :

```
curl --data redirect_uri='http://localhost:8000' --data
code=4/ygHeyskWdKW58DPow-
a41rn3IZqXB2y16ga3UBBmFI5BijDPmMfogUuA1CXbEtJJpT_G1tnhC9hbMwMR45GT_vw
--data client_id=985957988504-
sgp8fm02ob4keorekf2t544u765961l1.apps.googleusercontent.com --data
client_secret=ZMdFZHB8LbJllKMxWUGrKTjh --data
grant_type=authorization_code https://oauth2.googleapis.com/token
```

Response :

```
{
  "access_token":
"ya29.a0Ae4lvC20haaIxuYpJBx4_DS3fKicdT6OTWJYILJNrjALVu9AQPM9VwuRgiEgi
pdPvnaExdG7q8CIMDuU6TDPMAceZYde0IIa1zogXuy9v8NH2ts79Nl82t0A4RzyoyG2Ih
fX8wx0Fa4AxBU6iuTGtmO9ugNic66aIWU",
  "expires_in": 3599,
  "refresh_token": "1//0g6GkDsirYq1uCgYIARAAGBASNwF-
L9IrhQJOfpESn6FUTbUgO_S9ZAMP-rnd154A5CIAMJSjZL6XYbqH286y3qMA-
yxSVE9tOHc",
  "scope": "https://www.googleapis.com/auth/spreadsheets",
  "token_type": "Bearer"
}
```

With bearer token, finally we can invoke google sheets api resources.

Step 4 : Invoke resource service

Once we have the bearer token it needs to be sent as a header value with each and every api request. Google sheet api's are well documented with samples for reference.

Following is a simple read operation which reads from given cell rage. Depending on the operation it can be GET, POST,PUT, DELETE rest call.

Add 'sheet id' and the 'sheet tab name' from the sample google sheet which created early on.

```
GET
https://sheets.googleapis.com/v4/spreadsheets/190Ym4DG9s7Nk_YxA8cr1D_8VEmNpCnYsskyRgAxdadc/values/Sheet1!A1:D3?majorDimension=COLUMNS

Header : "Authorization: Bearer
ya29.a0Ae4lvC20haaIxuYpJBx4_DS3fKicdT6OTWJYILJNrjALVu9AQPM9VwuRgiEgipdPvnaExdG7q8CIMDuU6TDPMAceZYde0IIa1zogXuy9v8NH2ts79Nl82t0A4RzyoyG2IhfX8wx0Fa4AxBU6iuTGtmO9ugNic66aIWU"
```

Curl :

```
curl -H "Authorization: Bearer
ya29.a0Ae4lvC20haaIxuYpJBx4_DS3fKicdT6OTWJYILJNrjALVu9AQPM9VwuRgiEgipdPvnaExdG7q8CIMDuU6TDPMAceZYde0IIa1zogXuy9v8NH2ts79Nl82t0A4RzyoyG2IhfX8wx0Fa4AxBU6iuTGtmO9ugNic66aIWU"
'https://sheets.googleapis.com/v4/spreadsheets/190Ym4DG9s7Nk_YxA8cr1D_8VEmNpCnYsskyRgAxdadc/values/Sheet1!A1:D3?majorDimension=COLUMNS'
```

Response :

```
{
  "range": "Sheet1!A1:D3",
  "majorDimension": "COLUMNS",
  "values": [
    [
      "Description",
      "Test Iteration 221",
      "Test Iteration 222"
    ],
    [
      "Value1",
      "939",
      "781"
    ],
    [
      "Value2",
      "421",
      "242"
    ],
    [
      "Value3",
      "1725",
      "1137"
    ]
  ]
}
```

Step 5 : Refresh token (not shown in the sequence diagram)

Bearer token is short lived and requires to regenerate once expired. For this we can use the returned refresh token with token endpoint.

```
POST https://oauth2.googleapis.com/token

client_id : 985957988504-
sgp8fm02ob4keorekf2t544u765961l1.apps.googleusercontent.com

client_secret : ZMdFZHB8LbJllKMxWUGrKTjh

refresh_token : 1//0g6GkDsirYq1uCgYIARAAGBASNwF-
L9IrhQJOfpESn6FUTbUgO_S9ZAMP-rnd154A5CIAMJSjZL6XYbqH286y3qMA-
yxSVE9tOHc
```

```
grant_type : refresh_token
```

Curl :

```
curl --data client_id=985957988504-
sgp8fm02ob4keorekf2t544u765961l1.apps.googleusercontent.com --data
client_secret=ZMdFZHB8LbJllKMxWUGrKTjh --data
refresh_token=1//0g6GkDsirYq1uCgYIARAAGBASNwF-
L9IrhQJOfpESn6FUTbUgO_S9ZAMP-rnd154A5CIAMJSjZL6XYbqH286y3qMA-
yxSVE9tOHc --data grant_type=refresh_token
https://oauth2.googleapis.com/token
```

Response :

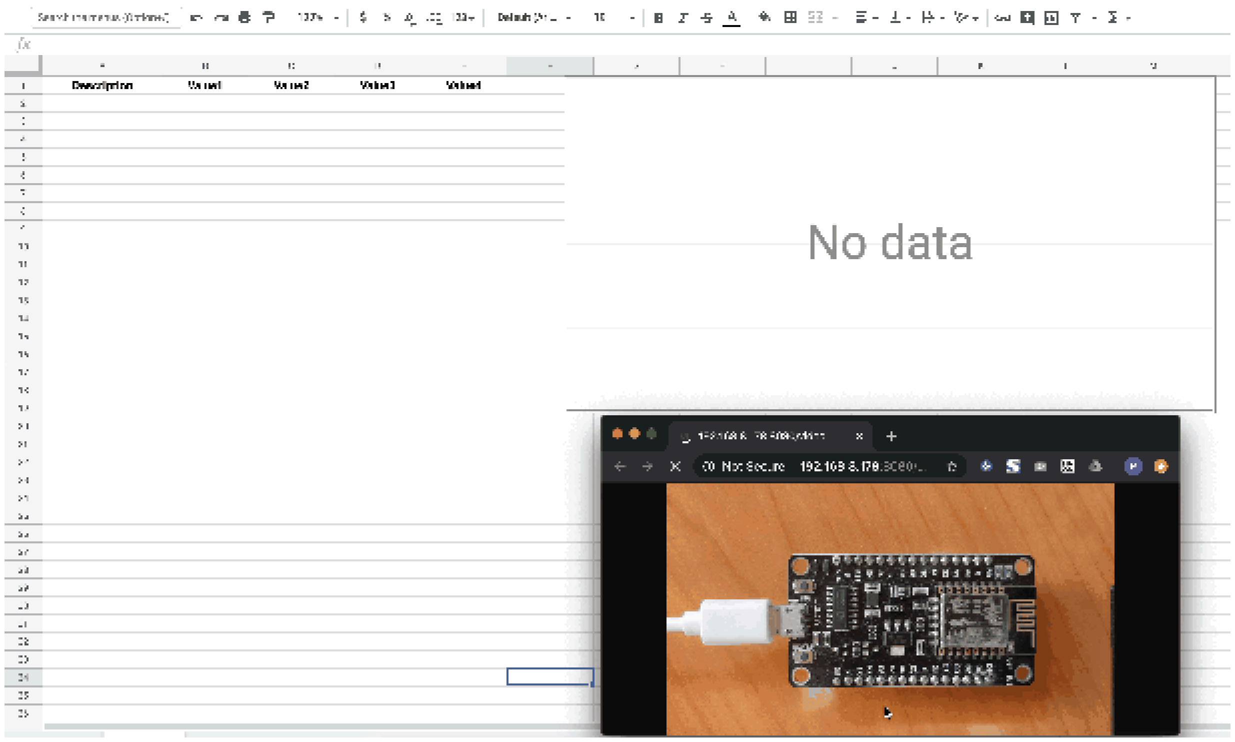
```
{
  "access_token": "ya29.a0Ae4lvC0_TklRtYGFPJ-GR-
9BrOLkMrpF8zw1QGlWtk26ccoZ2ibfbbcObOtvTZ12adFGjMjDO3CNYmrdRaCrC6nzs0s
QpzhbbKL8FiBbOMGkedcQ_o7-m6YBcZ5kAIZecpup5kWR9RarlJAYUSh6gsom75xtaoR-
cK0",
  "expires_in": 3599,
  "scope": "https://www.googleapis.com/auth/spreadsheets",
  "token_type": "Bearer"
}
```

Update: These oauth credentials along with the project has been deleted before publishing this article. Therefore, none of the codes, credentials mentioned are valid at this point.

What’s Next ?

Possibilities are endless with google resources. GCP provide large number of free/almost free APIs which enables developers to build crazy new features and experiments.

For example, i developed IoT device which feeds data directly to google sheet using these REST APIs for one of my home projects.



ESP8266 uploads data directly to google sheets

It's a story for another article. :)

Gcp API Oauth2 IoT Google

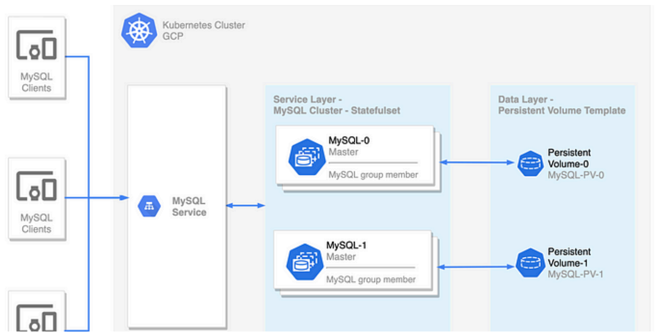
Written by Pumudu Ruhunage

Follow

31 Followers

Integration/DevOps Consultant

More from Pumudu Ruhunage



Pumudu Ruhunage

MySQL Group Replication in Kubernetes

How to configure mysql group replication with Kubernetes

6 min read · Nov 6, 2021

58 1

Raphaël Huchet in HackerNoon.com

How to query JSONB, beginner sheet cheat

Let's say we have to query a user table with a metadata JSONB column on a PostgreSQL...

3 min read · Mar 30, 2017

3K 26

Dmitry Nozhenko in HackerNoon.com

5 Ways to animate a React app in 2019.

Animation in React app is a popular topic and there are many ways to create different type...

10 min read · Jan 28, 2019

11K 31

Pumudu Ruhunage in HackerNoon.com

Open WiFi Credential Harvesting - IoT Edition

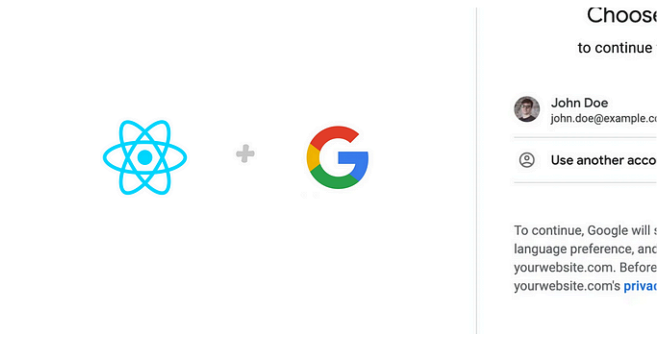
Free internet—open wifi connections, sounds tempting but is it actually free? May be you...

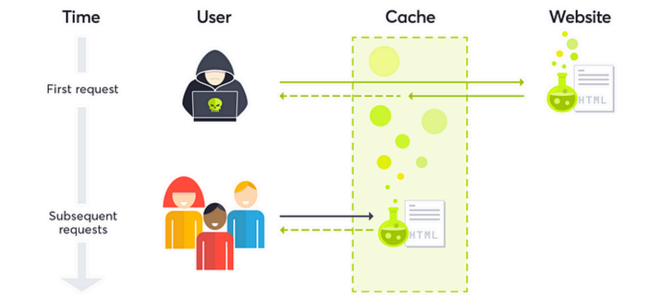
7 min read · Mar 20, 2019

26

See all from Pumudu Ruhunage

Recommended from Medium





Muhammed Sahad

React JS : A Step-by-Step Guide to Google Authentication

🚀 Welcome to 'React JS: A Step-by-Step Guide to Google Authentication.' In this…

7 min read · Jan 1, 2024

83 2

Ams._.Ghimire

Web Cache Poisoning

In this blog we will be learning about Cache, caching headers and the basics of Web Cac…

5 min read · Feb 28, 2024

9

Lists

Coding & Development

11 stories · 588 saves

Company Offsite Reading List

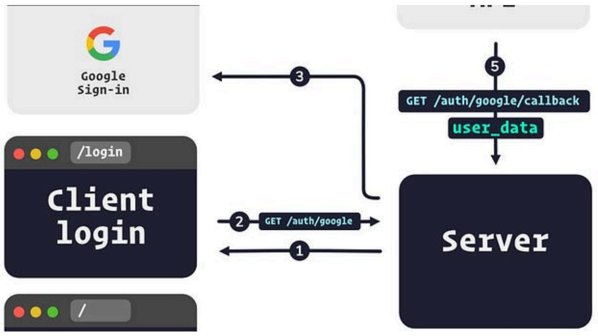
8 stories · 121 saves

Leadership upgrades

7 stories · 75 saves

Stories to Help You Grow as a Designer

11 stories · 703 saves

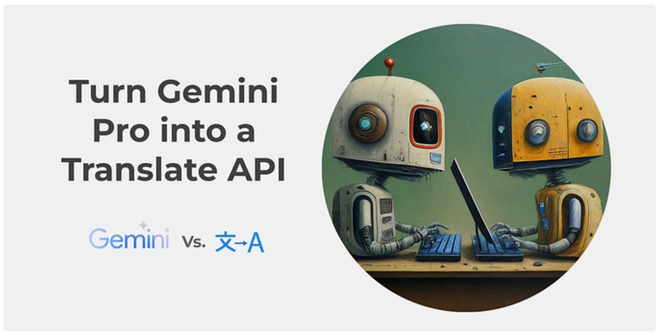


Venkata Naveen Varma V

Google Login (Authentication using Oauth2, Passport, NodeJS)

We go through how to login/authenticate users using there google account with the…

2 min read · Nov 22, 2023



Stéphane Giron

Turn Gemini Pro to a Translate API, can Gemini compet with Google…

Use Gemini Pro, new Google Generative AI model to create a Translation API.

3 min read · Jan 16, 2024

63

13 1

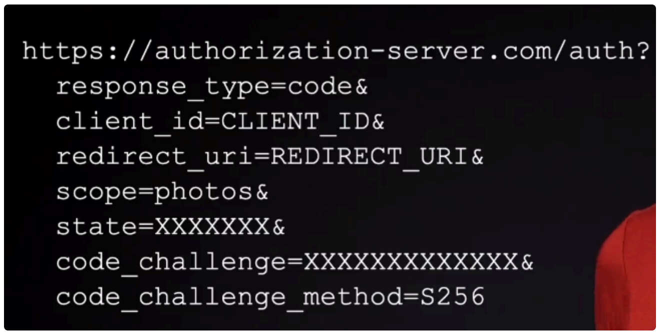
Syed Muhammed Hassan Ali in Stackademic

Use Google OAuth In NextJS And NestJS

A Step-by-Step Guide to Integrate OAuth2.0 Authentication

7 min read · Jan 1, 2024

93 2



```
https://authorization-server.com/auth?
  response_type=code&
  client_id=CLIENT_ID&
  redirect_uri=REDIRECT_URI&
  scope=photos&
  state=XXXXXXX&
  code_challenge=XXXXXXXXXXXXX&
  code_challenge_method=S256
```

Piyali Das

OAuth 2.0

OAuth 2.0 is the industry-standard protocol for authorization. OAuth 2.0 focuses on clien…

32 min read · Dec 29, 2023

3

See more recommendations