# How to set up the APIs for this program

## The Big Idea: Getting a Library Card for Your Robot

## An introduction for people who are new to API automation

Imagine each of these platforms (Meta, Google, LinkedIn) is a giant, super-secure library full of books. The "books" are the data you want—clicks, impressions, cost, etc.

You can't just walk into this library and take books. You need a special library card that proves who you are and what you're allowed to see.

Your Python script is like a helpful robot you've built. To get the data for you, this robot needs its own copy of your library card to show the librarian.

An **API (Application Programming Interface)** is just a fancy name for the librarian. It's the set of rules for how your robot can ask for books (data) and get a response.

Our goal is to get a special "library card" (API credentials) from each of these three "libraries" and give them to our Python robot so it can fetch the data automatically.

## Before You Start: The Prep Work

Before we visit each library, we need three things ready.

**1. Your Secret Wallet (.env file):**  
Your robot needs a safe place to store its library cards. We don't want to write the secret codes directly in our Python script where others might see them. The .env file is like a secret wallet. Make sure it's in your project folder and looks something like this:

Generated code

# .env file

# Meta (Facebook/Instagram) Credentials

META\_APP\_ID="YOUR\_META\_APP\_ID\_GOES\_HERE"

META\_APP\_SECRET="YOUR\_META\_APP\_SECRET\_GOES\_HERE"

META\_ACCESS\_TOKEN="YOUR\_META\_ACCESS\_TOKEN\_GOES\_HERE"

META\_AD\_ACCOUNT\_ID="act\_YOUR\_AD\_ACCOUNT\_ID\_GOES\_HERE"

# Google Ads Credentials

GOOGLE\_CLIENT\_ID="YOUR\_GOOGLE\_CLIENT\_ID\_GOES\_HERE"

GOOGLE\_CLIENT\_SECRET="YOUR\_GOOGLE\_CLIENT\_SECRET\_GOES\_HERE"

GOOGLE\_REFRESH\_TOKEN="YOUR\_GOOGLE\_REFRESH\_TOKEN\_GOES\_HERE"

GOOGLE\_DEVELOPER\_TOKEN="YOUR\_GOOGLE\_DEVELOPER\_TOKEN\_GOES\_HERE"

GOOGLE\_LOGIN\_CUSTOMER\_ID="YOUR\_LOGIN\_CUSTOMER\_ID\_GOES\_HERE"

# LinkedIn Credentials

LINKEDIN\_CLIENT\_ID="YOUR\_LINKEDIN\_CLIENT\_ID\_GOES\_HERE"

LINKEDIN\_CLIENT\_SECRET="YOUR\_LINKEDIN\_CLIENT\_SECRET\_GOES\_HERE"

LINKEDIN\_ACCESS\_TOKEN="YOUR\_LINKEDIN\_ACCESS\_TOKEN\_GOES\_HERE"

LINKEDIN\_AD\_ACCOUNT\_ID="urn:li:sponsoredAccount:YOUR\_AD\_ACCOUNT\_ID\_GOES\_HERE"

Use code [with caution](https://support.google.com/legal/answer/13505487).

2. The Right Toolkits (Python Libraries):  
Your robot needs special toolkits to talk to each librarian (API). We need to install them. Open your terminal or command prompt and run these commands:

Generated bash

pip install facebook\_business

pip install google-ads

pip install requests-oauthlib

Use code [with caution](https://support.google.com/legal/answer/13505487).Bash

(For LinkedIn, the official library is a bit complex, so we'll often use a standard request library, which is simpler to start with).

**3. The Right Permissions (Admin Access):**  
You must be an **Admin** on the client's Meta Business Account, Google Ads Account, and LinkedIn Ad Account. If you are not, you won't be able to create a library card or check out any books.

## Part 1: Connecting to the Meta Library (Facebook & Instagram Ads)

**Goal:** Get the Meta App ID, App Secret, and a long-lasting Access Token.

#### Step 1: Register Your Robot at the Library

Go to the [Meta for Developers](https://www.google.com/url?sa=E&q=https%3A%2F%2Fdevelopers.facebook.com%2F) website and log in with your Facebook account.

Click "My Apps" in the top right.

Click the green "Create App" button.

It will ask what you want the app to do. Choose "Business" and click Next.

Give your app a name (e.g., "Client Reporting App") and provide your contact email. Click "Create App."

### Step 2: Get Your Library Card (App ID & Secret)

On the left-hand menu of your new app, go to App Settings -> Basic.

You will see your App ID and App Secret right there.

Copy these two values and paste them into your .env file.

### Step 3: Get Your Robot a Special Key (Access Token)

This is the trickiest part. You need a special, long-lasting key so your robot doesn't have to ask for permission every single time.

On the left menu, click "Tools" and then select "Graph API Explorer."

On the right side of the screen, find the "Facebook App" dropdown and select the app you just created.

Under "User or Page," select "Get User Access Token."

A popup will ask for permissions. You MUST select ads\_read and read\_insights. Grant the permissions.

You now have a temporary key! Click the blue "i" icon next to the token and click "Open in Access Token Tool."

On the new screen, click the "Extend Access Token" button at the bottom. This turns your short-term key into a long-term one (it lasts about 60 days).

Copy this new, long key and paste it into the META\_ACCESS\_TOKEN field in your .env file.

### Step 4: Update Your Python Robot (data\_loader.py)

Now, replace the stub function with the real code.

Generated python

# In modules/data\_loader.py

import os

import pandas as pd

from facebook\_business.api import FacebookAdsApi

from facebook\_business.adobjects.adaccount import AdAccount

def load\_meta\_ads\_data():

"""

Connects to the Meta Ads API and fetches real ad performance data.

"""

# Initialize the "librarian" with your library card

FacebookAdsApi.init(

app\_id=os.getenv("META\_APP\_ID"),

app\_secret=os.getenv("META\_APP\_SECRET"),

access\_token=os.getenv("META\_ACCESS\_TOKEN")

)

# Tell the robot which ad account to look at

account = AdAccount(os.getenv("META\_AD\_ACCOUNT\_ID"))

# Ask for the "books" (data) you want

fields = [

'ad\_name',

'impressions',

'clicks',

'conversions', # Note: This might need custom conversion setup

'spend',

'revenue', # Note: Revenue tracking must be configured in Meta

]

params = {

'date\_preset': 'last\_30d', # Get data for the last 30 days

'level': 'ad',

}

# The robot gets the data

ads\_data = account.get\_insights(fields=fields, params=params)

# Clean it up for our program

data\_list = []

for ad in ads\_data:

data\_list.append({

'ad\_name': ad.get('ad\_name'),

'impressions': int(ad.get('impressions', 0)),

'clicks': int(ad.get('clicks', 0)),

'conversions': int(ad.get('conversions', 0)),

'cost': float(ad.get('spend', 0.0)),

'revenue': float(ad.get('revenue', 0.0))

})

return pd.DataFrame(data\_list)

## Part 2: Connecting to the Google Ads Library

Goal: Get a Google Client ID, Client Secret, Refresh Token, and Developer Token. This library is the most strict!

### Step 1: Get the Special "Developer Token"

Log into your client's Google Ads account.

Go to Tools and Settings -> API Center.

If you see an "Apply for API access" form, fill it out. You usually get "Basic Access" approved instantly.

You will see a Developer Token. Copy it and paste it into your .env file.

### Step 2: Register Your Robot (Create a Google Cloud Project)

Go to the [Google Cloud Console](https://www.google.com/url?sa=E&q=https%3A%2F%2Fconsole.cloud.google.com%2F).

Create a new project (e.g., "Client Ads Reporting").

In your new project, go to APIs & Services -> Enabled APIs & Services.

Click "+ ENABLE APIS AND SERVICES" and search for "Google Ads API." Enable it.

### Step 3: Get Your Library Card (Client ID & Secret)

1. In the Google Cloud Console, go to APIs & Services -> Credentials.
2. Click "+ CREATE CREDENTIALS" and choose "OAuth client ID."
3. Configure the consent screen if it asks. Just fill in your app name and email.
4. For the Application type, choose "Desktop app."
5. Click Create. It will show you your Client ID and Client Secret.
6. Copy these two values and paste them into your .env file.

### Step 4: Get Your Robot a PERMANENT Key (Refresh Token)

Google's long-term key is called a Refresh Token. Getting one is a bit tricky, but there are helper scripts.

1. Download the official [google-ads-python library examples](https://www.google.com/url?sa=E&q=https%3A%2F%2Fgithub.com%2Fgoogleads%2Fgoogle-ads-python).
2. Find the file called authenticate\_in\_desktop\_application.py.
3. You will need to edit this file to include your Client ID and Client Secret.
4. Run this script from your terminal: python authenticate\_in\_desktop\_application.py
5. It will open a browser window asking you to log in with your Google account and grant permission for your app to access Google Ads data. You must grant permission.
6. After you approve it, go back to your terminal. The script will print out your Refresh Token. It's a long string of characters.
7. Copy this token and paste it into the GOOGLE\_REFRESH\_TOKEN field in your .env file.

### Step 5: Update Your Python Robot (data\_loader.py)

This one is more complex, so here's a conceptual guide. (The full code would involve writing a GAQL query, which is like Google's version of SQL).

Generated python

# In modules/data\_loader.py (conceptual example)

import os

import pandas as pd

from google.ads.googleads.client import GoogleAdsClient

def load\_ga4\_data(): # This would now be load\_google\_ads\_data()

"""

Connects to the Google Ads API and fetches real ad performance data.

"""

# Create the configuration dictionary for the library card

credentials = {

"developer\_token": os.getenv("GOOGLE\_DEVELOPER\_TOKEN"),

"client\_id": os.getenv("GOOGLE\_CLIENT\_ID"),

"client\_secret": os.getenv("GOOGLE\_CLIENT\_SECRET"),

"refresh\_token": os.getenv("GOOGLE\_REFRESH\_TOKEN"),

"login\_customer\_id": os.getenv("GOOGLE\_LOGIN\_CUSTOMER\_ID"), # The client's Google Ads ID (without dashes)

"use\_proto\_plus": True

}

# Initialize the librarian

google\_ads\_client = GoogleAdsClient.load\_from\_dict(credentials)

ga\_service = google\_ads\_client.get\_service("GoogleAdsService")

# This is like writing a question in Google's special language (GAQL)

query = """

SELECT

ad\_group\_ad.ad.name,

metrics.impressions,

metrics.clicks,

metrics.conversions,

metrics.cost\_micros,

metrics.conversions\_value

FROM ad\_group\_ad

WHERE segments.date DURING LAST\_30\_DAYS

"""

# Send the robot to ask the question

response = ga\_service.search\_stream(

customer\_id=os.getenv("GOOGLE\_LOGIN\_CUSTOMER\_ID"),

query=query

)

# Clean up the data

data\_list = []

for batch in response:

for row in batch.results:

data\_list.append({

'ad\_name': row.ad\_group\_ad.ad.name,

'impressions': row.metrics.impressions,

'clicks': row.metrics.clicks,

'conversions': row.metrics.conversions,

'cost': row.metrics.cost\_micros / 1000000, # Cost is in micro-dollars

'revenue': row.metrics.conversions\_value

})

return pd.DataFrame(data\_list)

## Part 3: Connecting to the LinkedIn Library

This process is very similar to Meta's.

### Step 1: Register Your Robot

1. Go to the [LinkedIn Developer Portal](https://www.google.com/url?sa=E&q=https%3A%2F%2Fwww.linkedin.com%2Fdevelopers%2F) and log in.
2. Click "Create app."
3. Fill in the app name, link it to your company's LinkedIn Page, and upload a logo. Agree to the terms and create the app.

### Step 2: Get Your Library Card (Client ID & Secret)

1. In your new app's dashboard, go to the "Auth" tab.
2. You will see your Client ID and Client Secret.
3. Copy these and paste them into your .env file.

### Step 3: Get Your Robot a Special Key (Access Token)

1. Still on the "Auth" tab, scroll down to the "OAuth 2.0" section.
2. You will need to add a "Redirect URL." For testing, you can just put https://www.google.com.
3. Go to the "Products" tab on the left. Find the product called "Advertising API" and request access. This might take a day or two for LinkedIn to approve.
4. Once approved, you can generate an access token. The easiest way is to follow LinkedIn's guide to generate one manually using your browser. It involves crafting a special URL, pasting it into your browser, authorizing the app, and then copying the code from the URL you are redirected to. You then exchange that code for an access token. (This is complex, so following their guide step-by-step is key).
5. Once you have the token, paste it into your .env file.

### Step 4: Update Your Python Robot (data\_loader.py)

LinkedIn doesn't have a great official Python library like the others, so we use a standard tool to make requests.

Generated python

# In modules/data\_loader.py

import os

import pandas as pd

import requests # Make sure you run: pip install requests

def load\_linkedin\_campaign\_data():

"""

Connects to the LinkedIn Ads API and fetches real ad performance data.

"""

access\_token = os.getenv("LINKEDIN\_ACCESS\_TOKEN")

ad\_account\_id = os.getenv("LINKEDIN\_AD\_ACCOUNT\_ID")

# This is the "address" of the librarian for ad stats

url = f"https://api.linkedin.com/v2/adAnalyticsV2?q=analytics&pivot=CREATIVE&dateRange.start.day=1&dateRange.start.month=1&dateRange.start.year=2024&timeGranularity=DAILY&accounts={ad\_account\_id}"

# The robot needs to show its key in the "header"

headers = {

'Authorization': f'Bearer {access\_token}',

'X-Restli-Protocol-Version': '2.0.0'

}

response = requests.get(url, headers=headers)

if response.status\_code == 200:

# The data from LinkedIn needs a lot of cleaning

# This is a simplified example

data = response.json().get('elements', [])

# ... complex data cleaning would go here ...

return pd.DataFrame(data)

else:

print(f"Failed to get LinkedIn data: {response.text}")

return pd.DataFrame() # Return empty if it fails

Printed 7/13/2025  
  
N.B. Depending on how much time has passed, it may be a good idea to use the latest Google AI model to review these instructions, grounding it with Google Search to make sure you have the most updated information.