# Overview

The google\_fit\_agent is a Python-based application designed to interact with the Google Fit API to retrieve aggregated daily step count data for a specified time range. This document provides detailed instructions for setting up the project, authenticating with Googles OAuth 2.0, configuring the environment, and using the agent to fetch data. It is intended for developers with basic Python knowledge and access to the Google Cloud Console.

# Prerequisites

Before starting, ensure you have the following:

* A Google account for accessing the Google Cloud Console.
* Python 3.7 or higher installed.
* A text editor or IDE (e.g., VS Code, PyCharm).
* Internet access for API calls and package installation.
* Basic familiarity with Python and API authentication.

# Setup Instructions

This section guides you through configuring the Google Cloud project, enabling the Google Fit API, setting up credentials, and preparing the Python environment.

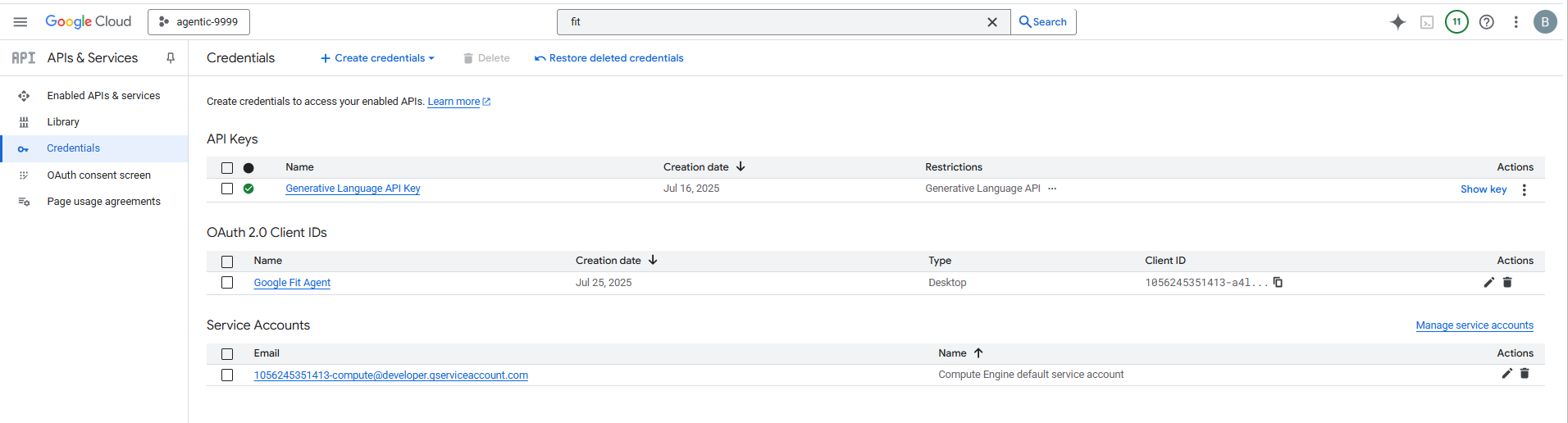
## Creating a Google Cloud Project

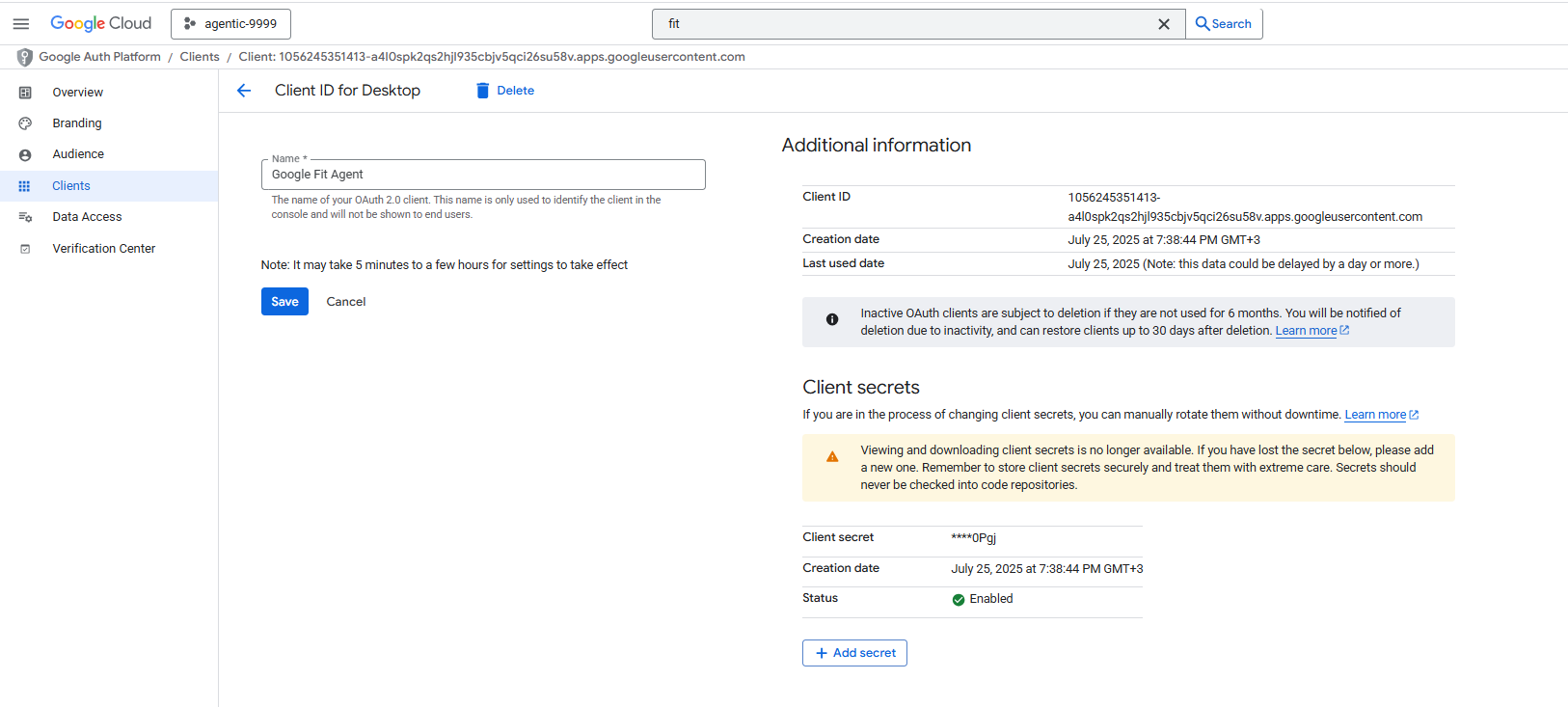
* + 1. Visit the [Google Cloud Console](https://console.cloud.google.com/).
    2. Click the project dropdown at the top and select **New Project**.
    3. Enter a project name (e.g., "Google Fit Agent") and click **Create**.

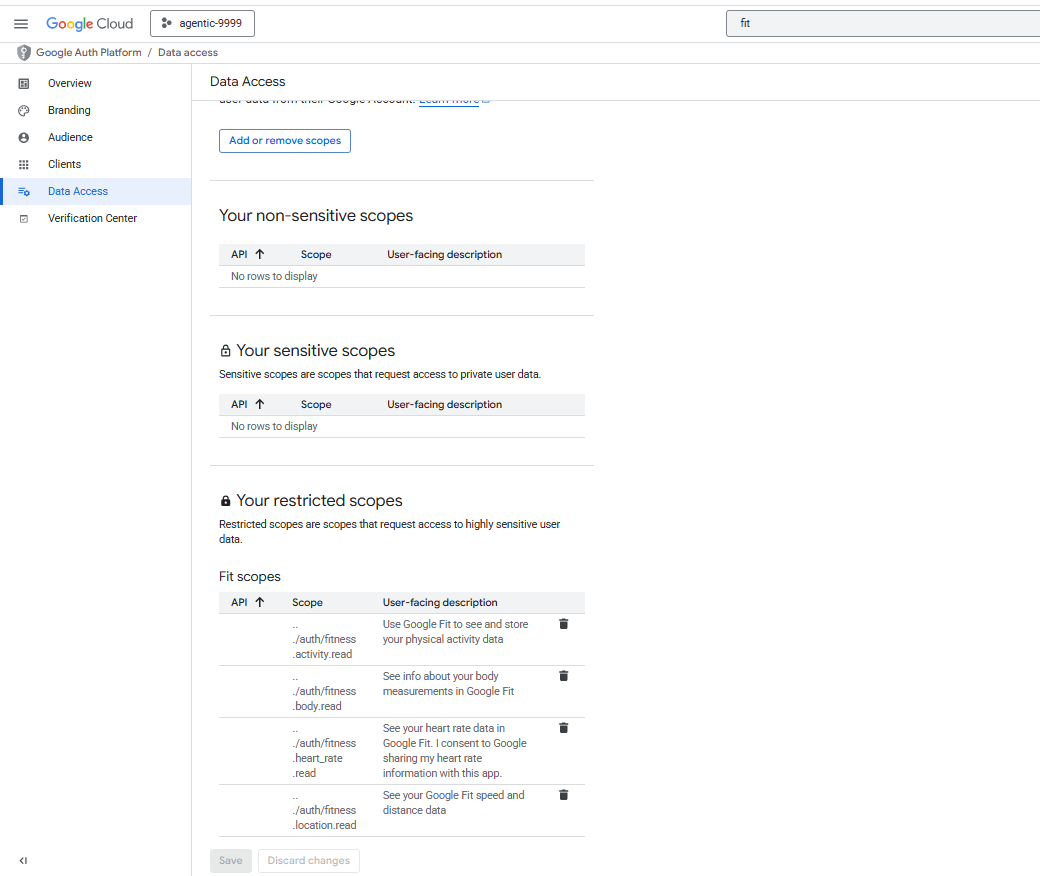
## Enabling the Google Fit API

* + 1. In the project dashboard, navigate to **APIs & Services** > **Library**.
    2. Search for **Google Fit REST API**.
    3. Click **Enable** to activate the API for your project.

## Configuring the OAuth Consent Screen

* + 1. Go to **APIs & Services** > **OAuth consent screen**.
    2. Select **External** as the user type and click **Create**.
    3. Complete the required fields:
       - **App name**: e.g., "Google Fit Agent"  
           
         



* + - * **User support email**: Your Gmail address
      * **Developer contact information**: Your Gmail address
    1. Add the following scopes:
       - [https://www.googleapis.com/auth/fitness.activity.read](http://www.googleapis.com/auth/fitness.activity.read)
       - [https://www.googleapis.com/auth/fitness.activity.write](http://www.googleapis.com/auth/fitness.activity.write)
       - [https://www.googleapis.com/auth/fitness.location.read](http://www.googleapis.com/auth/fitness.location.read)
       - [https://www.googleapis.com/auth/fitness.location.write](http://www.googleapis.com/auth/fitness.location.write)
    2. Add your Gmail address as a test user.
    3. Save and proceed through the remaining steps.  
         
         
       

## Creating OAuth 2.0 Credentials

* + 1. Navigate to **APIs & Services** > **Credentials**.
    2. Click **Create Credentials** > **OAuth client ID**.
    3. Select **Desktop App** or **Web Application**.
    4. If choosing **Web Application**, set the redirect URI to [http://localhost:8090.](http://localhost:8090/)
    5. Click **Create** and record the **Client ID** and **Client Secret**, or download the JSON file.

## Setting Up the Python Environment

## 

* + 1. Create a project directory (e.g., google\_fit\_agent).
    2. Install required dependencies by creating a requirements.txt file with the follow- ing content:

google-auth

google-auth-oauthlib google-auth-httplib2 requests

python-dotenv google-adk pandas

Install the dependencies:

pip install -r requirements.txt

* + 1. Create a .env file in the project root with your credentials:

# OAuth 2.0 Credentials for Google Fit API CLIENT\_ID=Your\_Client\_ID CLIENT\_SECRET=Your\_Client\_Secret [REDIRECT\_URI=http://localhost:8090](http://localhost:8090/)

[SCOPES=https://www.googleapis.com/auth/fitness.activity.read](http://www.googleapis.com/auth/fitness.activity.read) https://www.googlea

# Google AI API Key for the LLM GOOGLE\_API\_KEY=Your\_Google\_API\_Key

## Authentication

* + 1. Create an auth\_helper.py file with the following content to handle OAuth au- thentication:

import os import json import pathlib

from dotenv import load\_dotenv

from google\_auth\_oauthlib.flow import InstalledAppFlow

# Load environment variables from .env file load\_dotenv()

# We will create a temporary secrets file for the library to use CLIENT\_SECRETS\_FILE = "client\_secrets.json"

SCOPES = os.getenv("SCOPES", "").split()

# Create the client\_secrets.json structure from your .env file client\_config = {

"installed": {

"client\_id": os.getenv("CLIENT\_ID"), "client\_secret": os.getenv("CLIENT\_SECRET"), "redirect\_uris": [os.getenv("REDIRECT\_URI")],

"auth\_uri": "https://accounts.google.com/o/oauth2/auth", "token\_uri": "https://oauth2.googleapis.com/token"

}

}

with open(CLIENT\_SECRETS\_FILE, "w") as f: json.dump(client\_config, f)

# Use the standard library function to create a flow from the file

flow = InstalledAppFlow.from\_client\_secrets\_file(CLIENT\_SECRETS\_FILE, scopes=SCO

# Use the console flow - it’s much easier than running a local server.

# It will print a URL, you paste it in your browser, and paste the code back. creds = flow.run\_local\_server(port=8090)

# Save the credentials to "token.json" (no dot at the beginning)

token\_path = pathlib.Path("token.json") with open(token\_path, "w") as token\_file:

token\_file.write(creds.to\_json())

print(f"\n Credentials successfully saved to {token\_path}")

# Clean up the temporary secrets file os.remove(CLIENT\_SECRETS\_FILE)

* + 1. Run the authentication script:

python auth\_helper.py

* + 1. Follow the printed URL, authenticate in your browser, and paste the authorization code into the terminal. This generates a token.json file in the 5\_google\_fit\_agent directory.

## Agent Implementation

Create an agent.py file to define the Google Fit agent:

import os import json import requests

from google.adk.agents import Agent from dotenv import load\_dotenv

load\_dotenv()

def get\_access\_token() -> str | None:

"""Reads the access token from the token.json file.""" try:

with open("5\_google\_fit\_agent/token.json", "r") as f: creds = json.load(f)

return creds.get("token") except FileNotFoundError:

print("Error: 5\_google\_fit\_agent/token.json not found. Please run auth\_helper return None

except (KeyError, json.JSONDecodeError): print("Error: Invalid token.json file.") return None

def fetch\_google\_fit\_steps(inputs: dict) -> dict: """

Fetches aggregated daily step count data from Google Fit for a given time range. The ’inputs’ dictionary must contain ’startTimeMillis’ and ’endTimeMillis’.

"""

access\_token = get\_access\_token() if not access\_token:

return {"error": "Could not retrieve access token. Run authentication first."

start\_time = inputs.get("startTimeMillis") end\_time = inputs.get("endTimeMillis")

if not start\_time or not end\_time:

return {"error": "Missing ’startTimeMillis’ or ’endTimeMillis’ in inputs."}

headers = {

"Authorization": f"Bearer {access\_token}", "Content-Type": "application/json"

}

body = {

"aggregateBy": [{

"dataTypeName": "com.google.step\_count.delta",

"dataSourceId": "derived:com.google.step\_count.delta:com.google.android.g

}],

"bucketByTime": { "durationMillis": 86400000 }, "startTimeMillis": start\_time,

"endTimeMillis": end\_time

}

response = requests.post( "https://www.googleapis.com/fitness/v1/users/me/dataset:aggregate", headers=headers,

data=json.dumps(body)

)

if response.status\_code != 200:

return {"error": f"API request failed with status {response.status\_code}", "d return {"steps\_data": response.json()}

agent = Agent( name="google\_fit\_agent",

description="An agent to interact with the Google Fit API.", model="gemini-1.5-pro-latest", tools=[fetch\_google\_fit\_steps],

)

# Usage

To use the google\_fit\_agent:

1. Verify that the token.json file exists in the 5\_google\_fit\_agent directory.
2. Import and initialize the agent in a Python script:

from agent import agent

1. Call the fetch\_google\_fit\_steps function with appropriate inputs:

inputs = {

"startTimeMillis": "1696118400000", # e.g., Oct 1, 2023

"endTimeMillis": "1696204800000" # e.g., Oct 2, 2023

}

result = agent.tools[0](inputs) print(result)

1. The function returns a dictionary containing the step count data or an error mes- sage.

# Project Structure

The project directory should be organized as follows:

google\_fit\_agent/

.env requirements.txt auth\_helper.py agent.py 5\_google\_fit\_agent/

token.json

* .env: Stores OAuth credentials and Google API key.
* requirements.txt: Lists Python dependencies.
* auth\_helper.py: Manages OAuth 2.0 authentication and generates token.json.
* agent.py: Defines the google\_fit\_agent and its tools.
* token.json: Stores OAuth tokens (generated by auth\_helper.py).

# Troubleshooting

* **Token file not found**: Run auth\_helper.py to generate token.json.
* **API request failure**: Ensure token.json is valid and scopes match those in the OAuth consent screen.
* **Missing dependencies**: Re-run pip install -r requirements.txt.
* **Invalid credentials**: Verify CLIENT\_ID, CLIENT\_SECRET, and REDIRECT\_URI in

.env match the Google Cloud Console settings.

# Security Notes

* Store .env and token.json securely and exclude them from version control.
* Use the **External** user type for testing to limit access to authorized test users.
* Regularly review and rotate credentials in the Google Cloud Console.

**================ Note ================**

If you are running on google cloud shell you have to follow these steps   
  
  
Of course. This is a classic issue when running an OAuth 2.0 flow designed for a local desktop inside a remote environment like Google Cloud Shell.

**The Core Problem**

Your Python script (auth\_helper.py) is using flow.run\_local\_server(). This function does two things:

1. It generates an authorization URL for you to open in your browser.
2. It starts a **temporary web server on localhost:8090** *inside the Cloud Shell virtual machine*.

The problem is the localhost. When you authenticate in your browser, Google redirects your browser to http://localhost:8090. Your browser tries to connect to localhost, which means **your own computer**, not the Cloud Shell VM. Since you don't have a server running on port 8090 on your own computer, you get the ERR\_CONNECTION\_REFUSED error.

Here are two ways to fix this. The first solution is the recommended way for working within Cloud Shell.

**Solution 1: Use Cloud Shell's Web Preview (Recommended)**

Cloud Shell has a built-in feature to securely expose web ports from your VM to you.

**Step 1: Get Your Cloud Shell Preview URL**

1. In your Cloud Shell terminal, click the **Web Preview** button at the top-right.
2. Select **"Preview on port 8090"**.

A new browser tab will open with a URL that looks something like this:  
https://8090-a1b2c3d4-e5f6-....-....cloudshell.dev

**This is your new public redirect URI.** Copy this full URL.

**Step 2: Update Your OAuth Client in Google Cloud Console**

1. Go back to the Google Cloud Console page from your first screenshot (the "Client ID for Desktop" page).
2. Under the "Client secrets" section, you'll find the "Authorized redirect URIs" list (you may need to scroll down or edit the main client settings).
3. Click **+ ADD URI**.
4. Paste the Cloud Shell URL you copied in Step 1.
5. Click **Save**.

Your settings should now look something like this:

**Step 3: Update Your .env File**

Change the REDIRECT\_URI in your .env file to match the new Cloud Shell URL.

Generated bash

# .env file

CLIENT\_ID=1056245351413-a1b2c3d4....

CLIENT\_SECRET=GOCSPX-....

# Change this line!

REDIRECT\_URI=https://8090-a1b2c3d4-e5f6-....-....cloudshell.dev

SCOPES=https://www.googleapis.com/auth/fitness.activity.read ...

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

**Step 4: Run the Script Again**

Now, run your python script:  
python3 auth\_helper.py

| **Use Case** | **Description** |
| --- | --- |
| 🔢 Steps History | “How many steps did I walk last week?” |
| 🔥 Calorie Burn Trend | “Did I burn more calories this month than last?” |
| 💓 Heart Rate Alerts | “Alert me if my heart rate is too high at rest.” |
| 🧭 Distance Tracker | “Track how far I walk daily and weekly.” |

$ adk web

$ adk run

