

# **Project Documentation**

Project Title: video upload from one drive to youtube

Source code : [https://github.com/SANKETHSB/one\\_drive\\_upload](https://github.com/SANKETHSB/one_drive_upload)

Name : **Sanketh S B**

Email :sanketh.opqtech@gmail.com

## **Abstract**

The **Video Upload from OneDrive to YouTube** project is designed to streamline the process of transferring video files from OneDrive to YouTube. This automation tool utilizes Microsoft Graph API to interact with OneDrive, download video files, and then upload them to YouTube using the YouTube Data API.

The system authenticates users through Microsoft and Google OAuth, ensuring secure access to both platforms. It first scans a specified OneDrive folder for video files, downloads them locally, and then uploads them to YouTube with customizable metadata like title, description, and privacy settings. Additionally, the system includes error handling and logging mechanisms to monitor the process and provide feedback to users.

The project eliminates the need for manual file transfer between platforms, making it ideal for users who frequently upload content to YouTube from OneDrive, such as content creators, educators, and organizations.

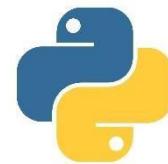
## Table of Contents

1. Pre-requisites
2. Setup Instructions
  - o Microsoft API Configuration
  - o Google API Configuration
  - o Environment Setup
3. Installing Dependencies
4. Script Breakdown
  - o Authentication Functions
  - o File Download Functionality
  - o YouTube Upload Functionality
  - o Logging and Error Handling
5. Running the Script
6. Common Issues and Troubleshooting
7. Additional Notes

## 1. Pre-requisites

Before setting up the script, ensure that the following are available:

- Python 3.x installed on your machine.



- An Azure Active Directory (AAD) application for Microsoft API access.

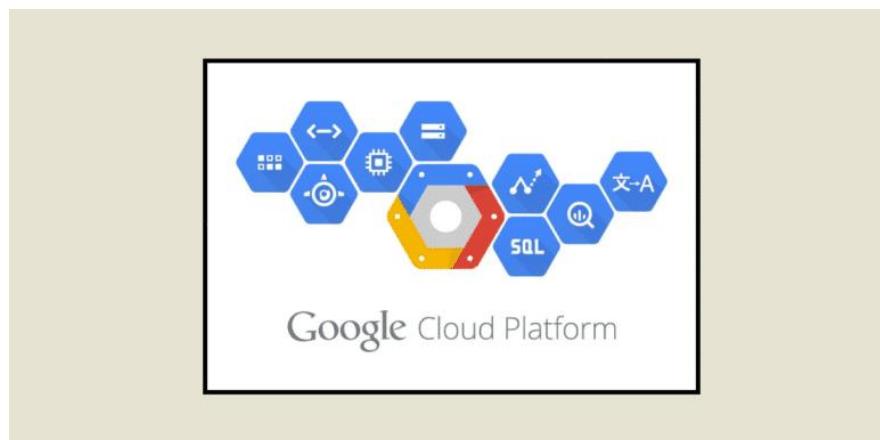


- A Google Cloud Platform (GCP) project with YouTube Data API enabled.



Google Cloud

## GOOGLE CLOUD CONSOLE



## 2. Setup Instructions

### Microsoft API Configuration

To download files from OneDrive, you need to register an application in **Azure Active Directory (AAD)** and configure the Microsoft Graph API.

#### 1. Register your application in Azure Active Directory:

- Go to [Azure Portal](#) and sign in.
- Navigate to **Azure Active Directory > App registrations > New registration**.



- Provide an application name (e.g., Video Downloader).
- Choose **Accounts in any organizational directory and personal Microsoft accounts** for the supported account type.

**Name**  
The user-facing display name for this application (this can be changed later).

**Supported account types**  
Who can use this application or access this API?  
 Accounts in this organizational directory only (Default Directory only - Single tenant)  
 Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)  
 Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)  
 Personal Microsoft accounts only  
[Help me choose...](#)

**Redirect URI (optional)**  
By proceeding, you agree to the [Microsoft Platform Policies](#)

- After registering, go to **Certificates & secrets > New client secret**. Save the **client secret** value.

Home > App registrations > video\_downloader

**video\_downloader | Certificates & secrets**

Search Got feedback? ...

Overview Quickstart Integration assistant Diagnose and solve problems

Manage **1** Branding & properties Authentication **2** Certificates & secrets

Token configuration API permissions Expose an API App roles Owners Roles and administrators Manifest Support + Troubleshooting

Add a client secret

Description Enter a description for this client secret  
Recommended: 180 days (6 months)

Expires

1 Application registration certificates, secrets and federated credentials

Certificates (0) Client secrets (1) Federated credentials (0)

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

+ New client secret

Description	Expires
Password uploaded on Fri Dec 27 2024	6/25/2025

**3**

Add Cancel

Application registration certificates, secrets and federated credentials can be found in the tabs below.

Certificates (0) **Client secrets (1)** Federated credentials (0)

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

+ New client secret

Description	Expires	Value ⓘ	Secret ID
Password uploaded on Fri Dec 27 2024	6/25/2025	S... **	525375d9-e0a9-4c68-b81b-6f69d5445...

**Copy the secret value**

- Copy and update the secret value in the .env file with a name( **CLEINT\_SECRET**)
- Under **API Permissions**, add the following Microsoft Graph permissions:
  - **Files.ReadWrite.All**
  - **User.Read**
  - **Files.ReadWrite**
  - **Files.Read**

**video\_downloader | API permissions**

Search Refresh Got feedback?

Overview Quickstart Integration assistant Diagnose and solve problems

Manage Branding & properties Authentication Certificates & secrets Token configuration

**API permissions** (2) (circled)  
Expose an API App roles Owners

**Configured permissions**

The "Admin consent required" column shows the default value for an organization. However, user consent can be customized per permission, user, or app. This column may not reflect the value in your organization, or in organizations where this app will be used. [Learn more](#)

Add a permission Grant admin consent for Default Directory

API / Permissions name	Type	Description	Admin consent req...	Status
> Microsoft Graph (6)				

**Request API permissions**

3 Delegated permissions Your application needs to access the API as the user.

Select permissions files

The "Admin consent required" column shows the default value for an organization. However, user consent can be customized per permission, user, or app. This column may not reflect the value in your organization, or in organizations where this app will be used. [Learn more](#)

Permission Admin consent required

- CrossTenantUserProfileSharing
- Files (3) (circled)
  - FileStorageContainer

Update permissions Discard

5 Files (3)

	Description	
Files.Read	Read user files	No
Files.Read.All	Read all files that user can access	No
File.Read.Selected	Read files that the user selects (preview)	No
File.ReadWrite	Have full access to user files	No
File.ReadWrite.All	Have full access to all files user can access	No

Refresh Got feedback?

Granting tenant-wide consent may revoke permissions that have already been granted tenant-wide for that application. On their own behalf aren't affected. [Learn more](#)

The "Admin consent required" column shows the default value for an organization. However, user consent can be customized per permission, user, or app. This column may not reflect the value in your organization, or in organizations where this app will be used. [Learn more](#)

**Configured permissions**

6 Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. This list includes all the permissions the application needs. [Learn more about permissions and consent](#)

Add a permission Grant admin consent for Default Directory (circled)

API / Permissions name	Type	Description	Admin con...
> Microsoft Graph (4)			
Files.Read.Selected	Delegated	Read files that the user selects (preview)	No
Files.ReadWrite	Delegated	Have full access to user files	No
Files.ReadWrite.All	Delegated	Have full access to all files user can access	No
User.Read	Delegated	Sign in and read user profile	No

7 Save and continue Cancel

- Make a note of the **Application (client) ID** and update in the .env with the name **(APPLICATION\_ID)**

The screenshot shows the Azure portal's application registration interface. On the left, there's a sidebar with options like Quickstart, Integration assistant, Manage, Authentication, Certificates & secrets, Token configuration, API permissions, Expose an API, and App roles. The main area displays the application's details under the 'Essentials' tab. The 'Display name' is 'video\_downloader'. The 'Application (client) ID' is highlighted with a red oval and has a 'Copy' button above it. Other fields shown include 'Object ID', 'Directory (tenant) ID', and 'Supported account types'. A note at the bottom states: 'Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to'.

## 2. Create the onedrive\_token.json file:

- This file will store the access token and refresh token. The script will create it after the first authentication.

## Google API Configuration

To upload videos to YouTube, you must set up OAuth 2.0 credentials on **Google Cloud Console**.

### 1. Create a project in Google Cloud Console:

- Go to Google Cloud Console.
- Create a new project and enable the **YouTube Data API v3** for the project.

The diagram illustrates the process of setting up a Google Cloud project and enabling the YouTube Data API v3. It consists of six panels connected by arrows. Panel 1 shows the Google Cloud Welcome screen with 'My First Project' selected. Panel 2 shows the 'Select a project' dialog with the 'NEW PROJECT' button highlighted. Panel 3 shows the 'Create New Project' dialog with 'Project name' set to 'My Project 9235', 'Location' set to 'No organization', and the 'CREATE' button highlighted. Panel 4 shows the Google Cloud Welcome screen again, this time with the 'APIs & Services' section highlighted. Panel 5 shows the 'APIs & Services' library page with the 'Library' tab selected. Panel 6 shows the 'YouTube Data API v3' product page with the 'ENABLE' button highlighted.

## 2. Generate OAuth 2.0 credentials:

- In the **oAuth consent screen**, click on **External** > fill the app name (**Video\_uploader**) > select the user support email and fill the developer contact information (use the same email id) > click on save and continue > click on the add or remove scope > search for Scope : <https://www.googleapis.com/auth/youtube.upload> > click on save and continue > add the test user > Click on save

The screenshot shows the Google Cloud API & Services interface. On the left, there's a sidebar with options like Enabled APIs and services, Library, Credentials (which is selected), and OAuth consent screen. The main area is titled 'OAuth consent screen' with a sub-section 'User Type'. It shows two options: 'Internal' (disabled) and 'External' (selected). Below 'External', there's a note about testing mode and adding test users. At the bottom of this section is a large blue 'CREATE' button, which is circled in red. A callout bubble points to this button with the text 'Click on create and fill the details and save'.

- In the **Credentials** section, click **Create credentials** > **OAuth 2.0 Client IDs**
- Choose **Desktop App** as the application type.

The screenshot shows the Google Cloud API & Services interface. On the left, there's a sidebar with options like Enabled APIs and services, Library (with a count of 1), Credentials (selected and highlighted with a red circle), and OAuth consent screen. The main area is titled 'Credentials' and shows three sections: 'API key' (with a note about quota and access), 'API keys' (with a note about requesting user consent), and 'OAuth 2.0 Client ID' (with a note about service accounts). At the top right of the 'Credentials' section is a blue '+ CREATE CREDENTIALS' button, which is circled in red. A callout bubble points to this button with the number '3'.

API keys

Name	Creation date	Restrictions	Actions
No API keys to display			

OAuth 2.0 Client IDs

Name	Creation date	Type	Client ID	Actions
Desktop client 1	28 Dec 2024	Desktop	698925252306-1skk...	

Service Accounts

Email	Name	Actions
No service accounts to display		

- Download the **client\_secrets.json** file (if the file name is downloaded with a another name rename to client\_secrets.json) and save it in your project directory.

### 3. Generate the youtube\_token.json file:

- After the first authentication, the script will generate the youtube\_token.json file, which will store the access token and refresh token for future API calls.

### 4. Environment Setup

Create a `.env` file in the project root directory to store sensitive data such as your API keys and secrets.

Example `.env` file:

```
APPLICATION_ID=your-application-id-here
CLIENT_SECRET=your-client-secret-here
```

## 3. Installing Dependencies

The script requires several Python packages for functionality. Install them using pip:

```
pip install msal httpx google-auth google-auth-oauthlib google-api-python-client python-dotenv
```

## 4. Script Breakdown

### Authentication Functions

- Microsoft Authentication: The script uses the `msal` library to authenticate with the Microsoft Graph API. The authentication flow follows OAuth2, and the token is stored in the `onedrive_token.json` file. If the token is expired or absent, the script requests a new token.

Python `def get_access_token(application_id, client_secret, scopes):`

```
# Authentication flow for MS Graph API
```

- YouTube Authentication: The `google-auth` and `google-auth-oauthlib` libraries handle the OAuth2 flow for authenticating with the YouTube API. Tokens are stored in the `youtube_token.json` file. If the token expires, it will be refreshed.

---

```
def authenticate_youtube():
```

```
    # Handles OAuth2 authentication for YouTube
```

## File Download Functionality

The download\_file function retrieves files from OneDrive. It sends a request to the Microsoft Graph API and checks for a 302 redirect response. If a file is found, it is downloaded to the specified directory.

```
def download_file(headers, file_id, file_name):
```

```
    # Download video file from OneDrive
```

## YouTube Upload Functionality

After downloading the videos from OneDrive, the script uploads them to YouTube using the upload\_video function. It uses the YouTube Data API v3 and uploads the file with a title, description, and category.

```
def upload_video(youtube, video_file, title, description, category='22', privacy='private'):
```

```
    # Upload video to YouTube
```

## Logging and Error Handling

The script includes logging of actions (such as video uploads and file downloads) with timestamps for each step. The logs are written to the console.

```
import time
```

```
def log_action(message):
```

```
    timestamp = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())
```

```
    print(f"[{timestamp}] {message}")
```

## 5. Running the Script

After setting up the dependencies and configurations, you can run the script using the following command:

```
python app.py
```

The script performs the following steps:

1. Authenticates with Microsoft Graph API to fetch files from OneDrive.
2. Downloads video files from OneDrive to the downloads folder.
3. Uploads videos to YouTube.
4. Cleans up the downloads folder after uploading.

## 6. Common Issues and Troubleshooting

### Microsoft Graph Authentication Fails

- Ensure that the Application ID and Client Secret in the .env file are correct.
- Make sure that the required permissions (Files.ReadWrite.All, User.Read) are granted in Azure AD.

### YouTube Upload Fails

- Ensure that the OAuth2 credentials (client secrets) are correctly configured.
- Make sure that the YouTube Data API v3 is enabled in Google Cloud Console.

### Missing downloads Folder

- If the downloads folder does not exist, the script will automatically create it, but ensure that the script has proper permissions to write files.

## 7. Additional Notes

- Token Expiry: The script handles token expiry by checking if the access token is valid. If expired, it will automatically refresh the token.
- Error Handling: The script will print error messages if any part of the process fails (e.g., API call failures, file not found).
- Logs: Every action (e.g., file download, video upload) is logged with a timestamp to help track the progress.

## Conclusion

This script provides a comprehensive solution for automating the process of downloading video files from OneDrive and uploading them to YouTube. It leverages the Microsoft Graph API for secure file access from OneDrive and the YouTube Data API for video uploads, ensuring seamless integration between the two platforms. With built-in authentication flows for both APIs, token management, and error handling, the script minimizes manual intervention.

By following the detailed setup instructions and ensuring proper API configurations and dependencies, you can easily configure and run the script. It also logs every action with timestamps, making it easier to track progress and debug issues if they arise.

This solution can be particularly useful for users who need to manage and upload content from OneDrive to YouTube regularly, automating a tedious process and ensuring both security and efficiency throughout.

You can access the full source code and instructions for setting up this script on GitHub:

[GitHub Repository - https://github.com/SANKETHSB/one\\_drive\\_upload](https://github.com/SANKETHSB/one_drive_upload)

Or

GitHub link - [https://github.com/SANKETHSB/one\\_drive\\_upload](https://github.com/SANKETHSB/one_drive_upload)