Extracting YouTube Data With Python Using API

sandesh more
Intern
(App Development)

Project Definition:

The goal of this project is to extract data from YouTube using its API and Python programming language. The extracted data can be used for various purposes such as data analysis and content recommendation.

Introduction:

YouTube is making people go viral and actually YouTube itself is going viral. It is the second-largest search engine after Google.

Started in the year 2005 the platform now has over 2 billion monthly active users. YouTube has helped content creators to get exposure and also earn some revenue out of it.

In this Project , we will be using the YouTube API with the help of Python programming. By doing so we will be able to extract and scrap data from YouTube which could be used in multiple projects.

What is an API?

API is short for Application Programming Interface and it is a software interface that allows programmers to have an efficient way for client-server communication.

Developers often use APIs to build client-server applications. They may have to provide some data and then the API returns the specific information from the backend. Not only the information but multiple operations could be accompanied by APIs.

Here we will be using YouTube API which is provided officially by YouTube itself. It allows developers to retrieve various attributes related to the provided information.

Suppose you wanted to create an application where you will show the most view count of the entered channel by the user. You could take the help of the API to retrieve that information in no time.

There are various things that an API could extract and depends on the platform and owner. Let us now see what all things we could extract and scrap with YouTube API.

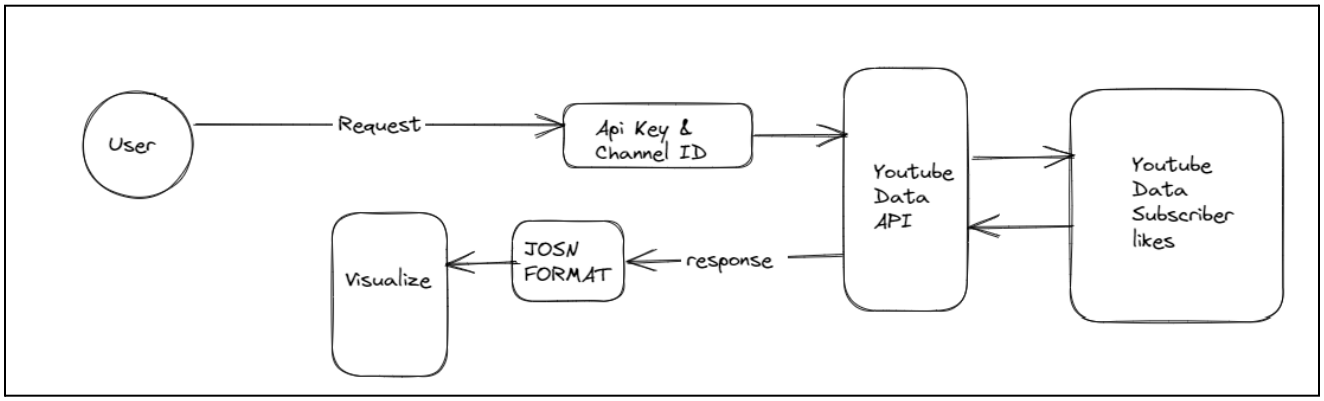
Things You Could Extract From YouTube API?

There is plenty of information and data that you could extract from this YouTube API using Python. We are mentioning some of the important attributes below:

- **Channel's Statistics**: Return important statistical information about the channels specified.
- **View count**: Get the total number of view count by that YouTube channels.
- **Total No. of Subscribers**: As the name suggests, it would fetch you the Number of Subscribers with YouTube API.
- **Snippet**: It lets you fetch multiple things from Channel's data like Description, Title, etc.

There are a lot more things that **YouTube API** lets you extract from its database.

System Architecture:



Flow chart:

Start
|
[Create API key]
|
[Enable the YouTube Data API]
|
[Authorize API access using API key or OAuth client ID]
|
[Retrieve the channel ID for the desired YouTube channel]
|
[Making a request to the YouTube Data API using the channel ID and API key]
|
[Receive a response from the API in JSON format]
|
[Parse the JSON response to extract the subscriber count and view count]
|
[convert to dataframe and save to csv format]
|
[visualize subscribers and view count using matplotlib]
|
End

Working on YouTube API with Python:

Prerequisites:

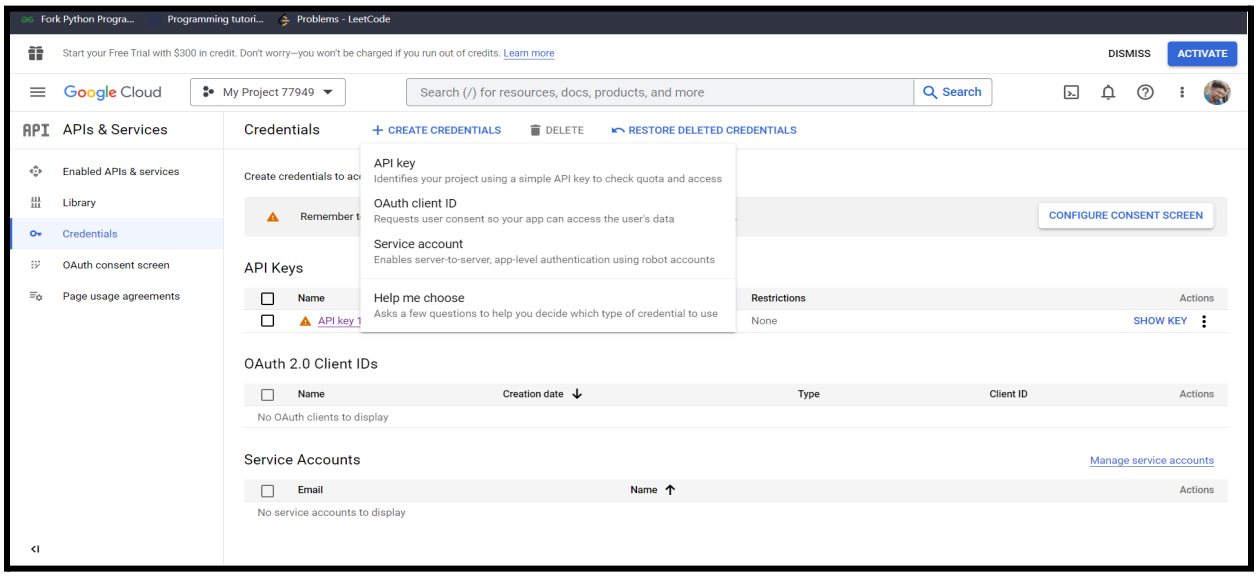
1. **For Windows**: Google API Client (pip install google-api-python-client)
2. **Framework**: Flask
3. **Library**: Pandas, matplotlib, seaborn, dotenv.

Step1:

Visit the YouTube Data API Overview page - API KEY

Step2:

1. Create a project in the Google Developers Console and obtain authorization credentials so your application can submit API requests.



2. After creating your project, make sure the YouTube Data API is one of the services that your application is registered to use:
   a. Go to the API Console and select the project that you just registered.
   b. Visit the Enabled APIs page. In the list of APIs, make sure the status is **ON** for the **YouTube Data API v3**.

Step3: **Code and Libraries operations:**

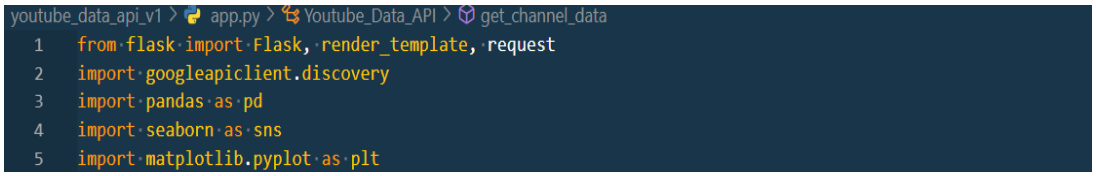
1. Installation of Google API Client

   Cmd : pip install google-api-python-client

2. Import Libraries
   a. We will need Libraries to work upon our YouTube API extraction. Import them using the code below:

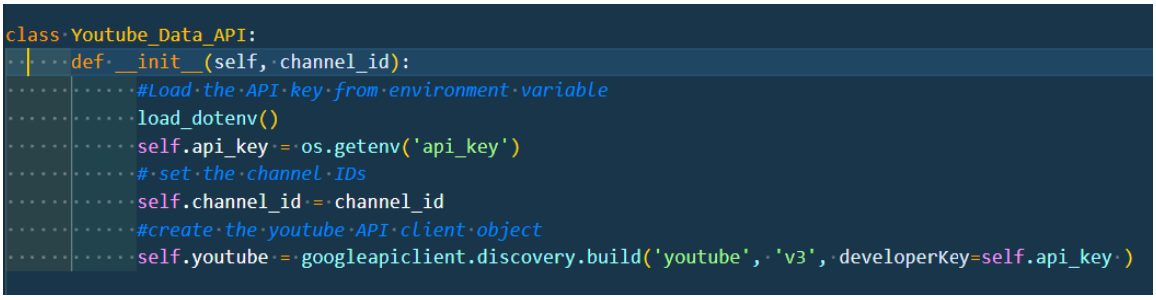
   from googleapiclient.discovery import build

   b. Basically follow the requirements.txt file to import all the libraries and modules.Here is the screenshot of importing module and libraries



3. Creating Object:
   a. Create the object by using below code format:

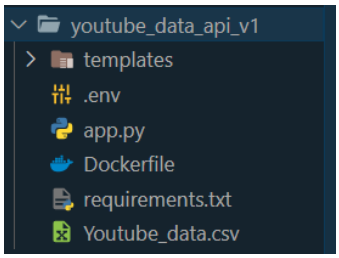
self.youtube = googleapiclient.discovery.build('youtube', 'v3', developerKey=self.api_key)



4. Getting Statistics from YouTube API

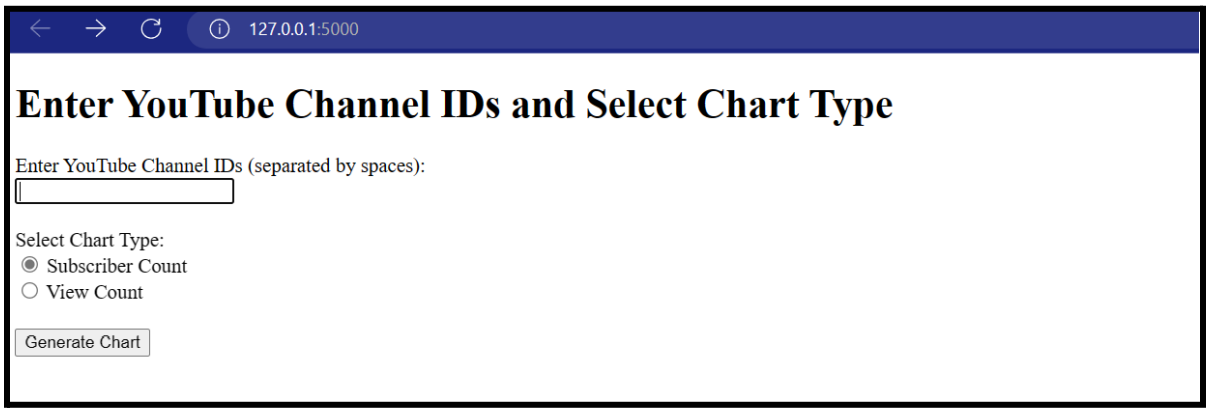
```python
def get_channel_data(self):
    #Make the Request to youtube API and get response as channel details
    all_data = []
    self.request = self.youtube.channels().list(
    part='snippet,contentDetails,statistics',
    id=','.join(self.channel_id)
    )
    #Response in json format
    response = self.request.execute()
```

5. Once we will get all the statistics then we can convert into **Dataframe** using pandas and pandas methods also we can save as **csv file**. and this data plot using matplotlib and seaborn library

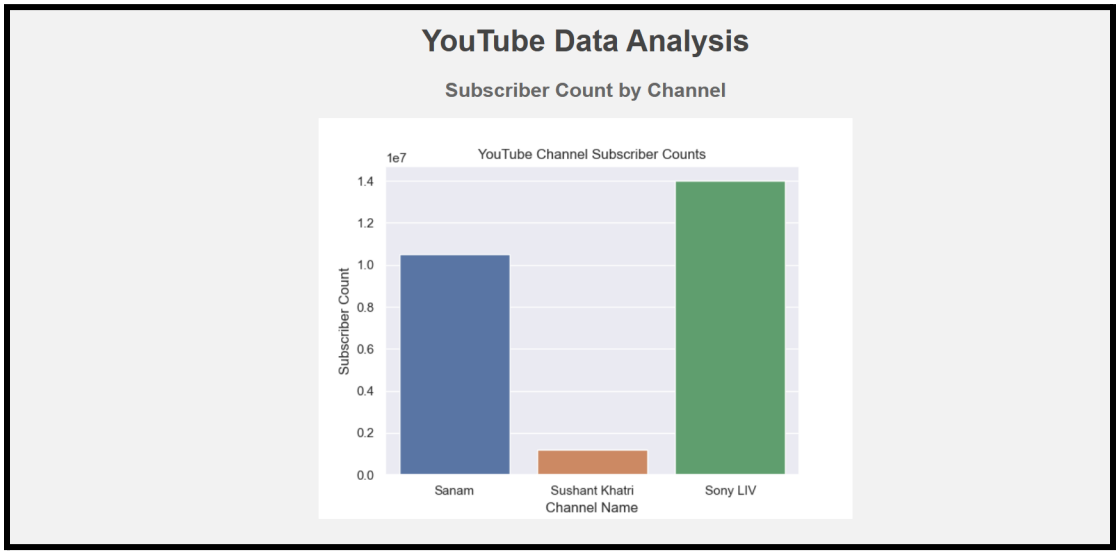


Youtube_Data.csv file

6. With the help of flask framework access Youtube data api through web browsers.
    01. Give the input as Youtube channel ID's for eg.
        a. UCOQNJjhXwvAScuELTT_i7cQ
        b. UCGiJnK-nza-ANW1LNc0JxSA
        c. UCnrG75VRwdlp2wtwfpOCBRQ



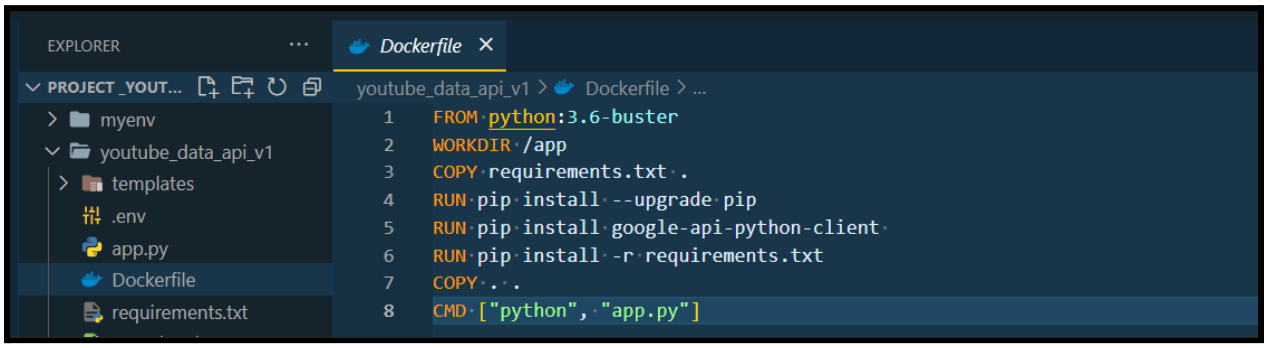
02. Then choose the chart type and get the output as plot bar of youtube channel



Creating the Docker image and containerizing the code:

Docker containerization makes it easier to deploy the code on different platforms and environments. The container can be run on any system that has Docker installed, without worrying about dependencies or compatibility issues.

1. To create a Docker image, we first need to create a **Dockerfile**, which specifies the base image, dependencies, and configurations required for our application.



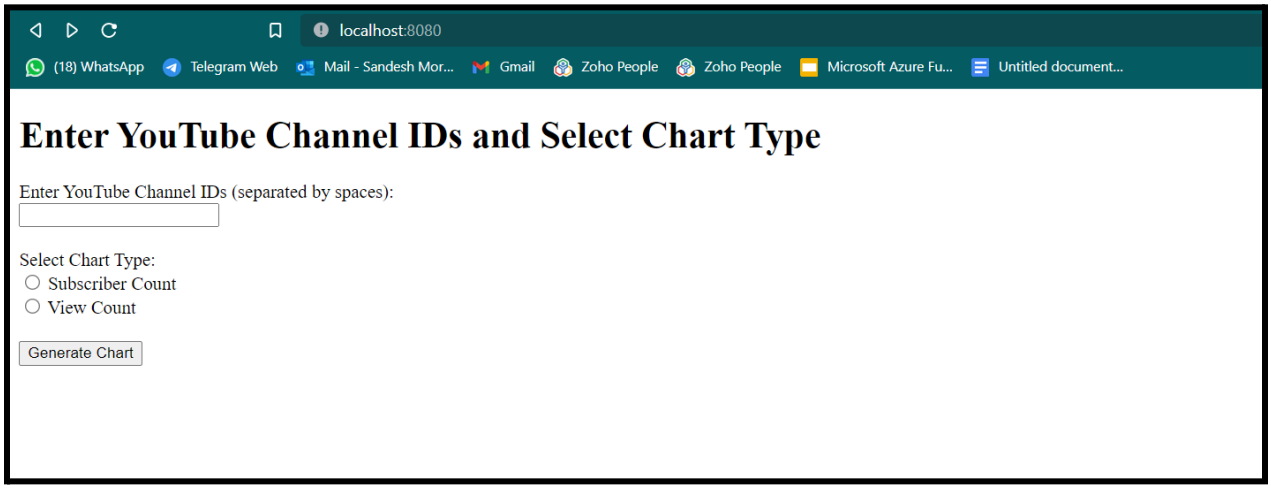
2. Once we have the Dockerfile, we can use the **docker build** command to build the Docker image. This command reads the Dockerfile and creates an image based on its instructions.

```
python_app_devlopment_basics\project _Youtube_data_api> docker build -t sandesh821/youtube_data_api .
```

3. After the image is built, we can use the **docker run** command to create a container based on the image. This command starts the container and runs the application inside it.

```
\project _Youtube_data_api> docker run -d -p 8080:5000 sandesh821/youtube_youtube_api
```

4. Access docker container on web browser
   Type this url: http://localhost:8080/

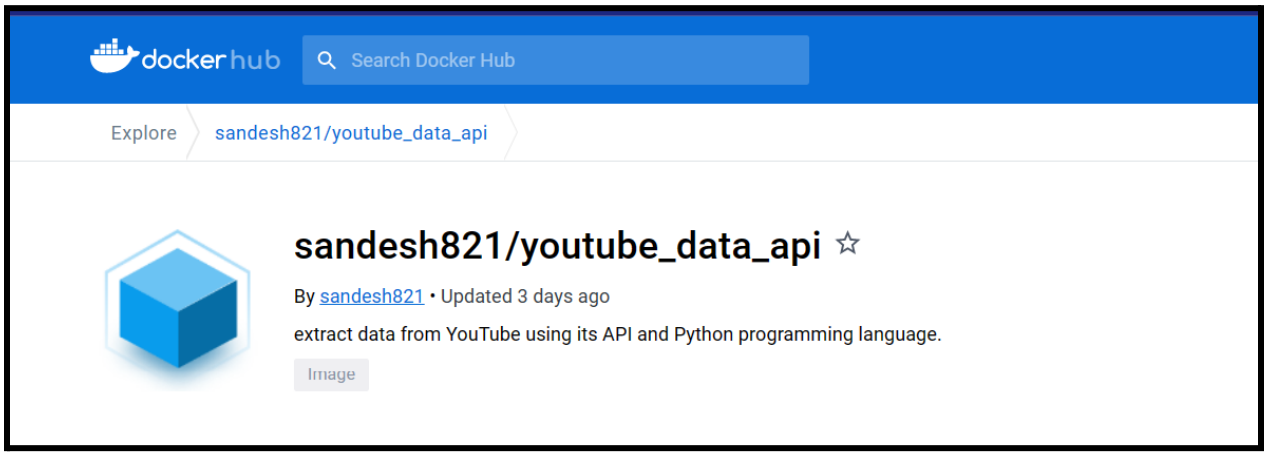


5. push a Docker image to Docker Hub:
   a. First, make sure you have a Docker Hub account. If you don't have one, create a free account at https://hub.docker.com/.
   b. Login to Docker Hub using the **docker login** command. You will be prompted for your Docker Hub username and password.
   c. Push the Docker image to Docker Hub using the docker push command. Use the same

```
project _Youtube_data_api> docker push sandesh821/youtube_data_api
```

Once the Docker image is pushed to **Docker Hub**, you can access it from anywhere with an internet connection. Other users can also pull the image and use it to run containers on their own systems.

Using the below link access the docker image.
Docker Image link: [sandesh821/youtube_data_api - Docker Image | Docker Hub]



Overall, containerizing the code with Docker provides a convenient and consistent way to package and deploy the application. The Docker image can be easily shared and deployed on any system with Docker installed, making it a popular choice for modern application development and deployment.

Conclusion

To sum up, extracting YouTube data with Python using API is a powerful way to gather insights on youtube channels and their data . By leveraging Python libraries like google-auth, google-api-python-client, and pandas, we can retrieve and analyze data on channels statistics, comments, and channels view count. While there are restrictions and limitations to the YouTube API, this method can be a valuable tool for content creators and marketers looking to optimize their video content and SEO strategies.

References:

1. YouTube Developer Documentation:
   https://developers.google.com/youtube/documentation
   https://developers.google.com/youtube/v3

2. Stackoverflow for identify and overcome form the errors
   https://stackoverflow.com/questions/44270023/accessing-public-data-via-the-youtube-data-api-without-authentication