Youtube Data Harvesting & Warehousing

# Overview

This web application demonstrates how data could be harvested using Youtube Data v3 APIs and warehoused using MySQL database

# Workflows

## Summary

This screen helps you view basic summary of channels those are already warehoused in the MySQL DB

## Collect

This screen allows users to review the details of a channel, which includes channel, playlist, comments and video information for that channel. It also allows the user to save the channel after reviewing.

## Analyze

This workflow allows users to view 10 predefined insights based on the warehoused information.

# Modules

The solution is composed of the following modules:

1. **Web Application:** This acts as a controller that implements UI workflows for Harvesting, Warehousing and Viewing Insights. This is built using **Streamlit web application framework** and deployed using a just in time hosting for demonstration
2. **Data Access Object (YtDao):** This is a python library that is used by the web application to store and retrieve business objects. A MySQL database hosted in CCP is used for persisting objects.
3. **API Client (YtApi):** A python module that is responsible for retrieving channel, playlist, comments and video information from YouTube Data v2 APIs
4. **DB Init (YtDbInit)**: Manages MySQL cloud Instance and DB schema from a python environment such as Google Colab or Jupyter
5. **DB Connector(YtDbConnector):** Handles management of DB connections from a python application to the **‘youtube’** DB that is hosted in the cloud. We use SQL Alchemy and Google Cloud SQL to achieve this
6. A tools library for commonly used application agnostic functions

# Solution Architecture



# Configuration:

The configuration for DB connections and API connections must be present in a JSON file, with following format.

{

"db": {

"project\_id": "**<Your GCP Project ID>**",

"region":"**<GCP Region>**",

"instance\_name": "**<GCP Instance>**",

"user":"**<DB User>**",

"passwd":**"<DB Password>**",

"name":"**<DB Name>**"

},

"api":{

"key": "<API Key For YouTube Data v3 API",

"version":"v3",

"service\_name": "youtube"

}

}

# Prerequisites

1. A valid Google account
2. Obtain an API key to avail GCP Service APIs. Could be done via GCP Console
3. Ensure that following GCP services are enabled - Cloud SQL, Cloud Instance, Cloud SQL Admin & YouTube Data API v3
4. Following modules MUST be present in web application’s top folder - ytapi.py, ytdao.py, ytconnector.py and tools.py
5. Config file cfg.json MUST be present in the application root folder

# Future Enhancements:

* The styling of the presentations could be enhanced
* Visually appealing charts could be used in few places as opposed to using dataframe widgets
* YtApi could be encapsulated in a class