*Touching down Google Cloud Platform (GCP)  
How to export an Excel workbook to Google sheets*

Table of Contents

[Overview 2](#_Toc79581458)

[Scope 2](#_Toc79581459)

[Pre-requisites 2](#_Toc79581460)

[Setup 2](#_Toc79581461)

[Final Words 10](#_Toc79581462)

[Appendix 11](#_Toc79581463)

Overview

Google Sheets is a web-based spreadsheet application that allows you to store and organize various types of information in collaboration with other people. While Google Sheets does not offer all of Excel's advanced features, it's easy to create and edit spreadsheets ranging from the simple to the complex.

Scope  
   
This tutorial is intended to describe the steps to follow for exporting a local Ms. Excel workbook to google spreadsheet.

# Pre-requisites

The following components come into play:

* **Pygsheets:** A simple, intuitive python library to access google spreadsheets through the [Google Sheets API v4](https://developers.google.com/sheets/api/).  
  **Features:**
  + Open and create spreadsheets by **title**.
  + Add or remove permissions from you spreadsheets.
  + Simple calls to get a row, column or a defined range of values.
  + Change the formatting properties of a cell.
  + Supports named ranges & protected ranges.
  + Queue up requests in batch mode and then process them in one go.
* **Pandas:** A fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the [Python](https://www.python.org/) programming language.

The purpose of this tutorial is to develop a lightweight command line based utility, through Python based modules that in order to upload a local Ms. Excel workbook to google spreadsheet.

If this tutorial intrigues you, then grab its code from the following GitHub repository: “[https://github.com/bassemmarji/GoogleSuite/](https://github.com/bassemmarji/GoogleSuite/upload_xls.py) ”.

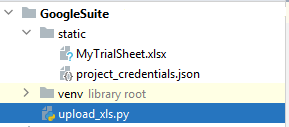
# Setup

To setup the environment, first and foremost you need python3 installed on your system. It is highly recommended to setup a virtual environment which will host the needed libraries.

1. Create a virtual environment and activate it.
2. Create a file named requirements.txt and add the following lines to it.

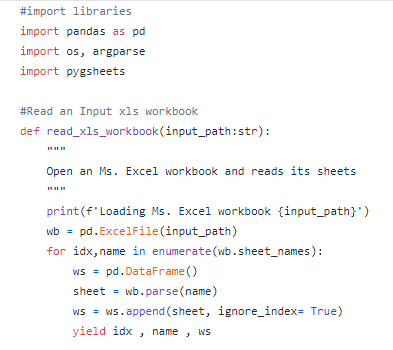
|  |
| --- |
| requirements.txt |
| pygsheets==2.0.5  xlrd==1.2.0 |

1. Now, let’s install the required libraries to the project.  
   pip install –r requirements.txt
2. Create a folder for our project called “GoogleSuite”.

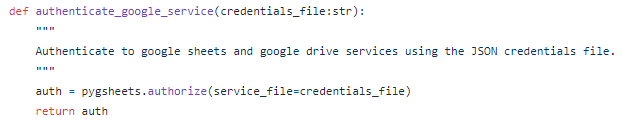
At the end, our folder structure will look like the following:  
  


1. Enable Python access to Google Sheets service by following the steps mentioned in the enclosed appendix.  
   Once you execute the outlined steps and in order to authenticate to your google service account save the service account key which you already downloaded under the project “static” folder and rename it to “project\_credentials.json”.

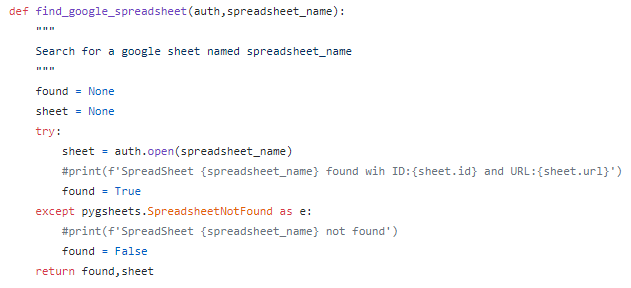
Let’s jump into coding:  
 *#upload\_xls.py*



* This generator function iterates through the worksheets of the Ms. Excel workbook and yields the index, name and content of the corresponding worksheet as a panda data frame.



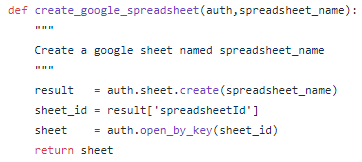
* This function will enable Python access to Google Sheets using the credentials (service account key .JSON file) downloaded from Google developer console and placed under the “static” folder.



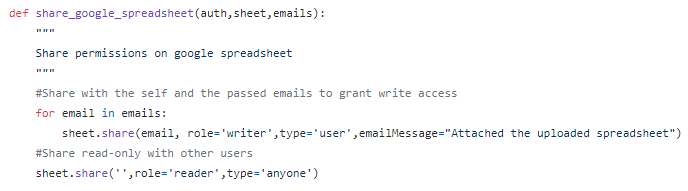
* This function searches for and opens a google spreadsheet having a title equal to the input parameter “spreadsheet\_name”.

Worth noting that there are three ways to open a spreadsheet:

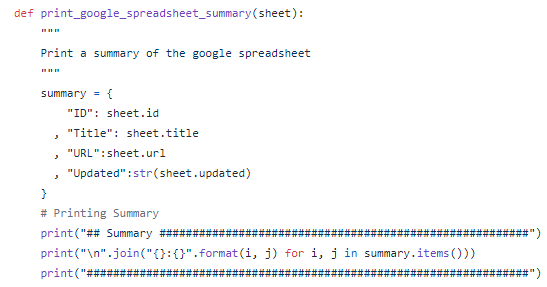
* + By its title as it appears in Google Docs.
  + By its unique ID.
  + By its exact URL.



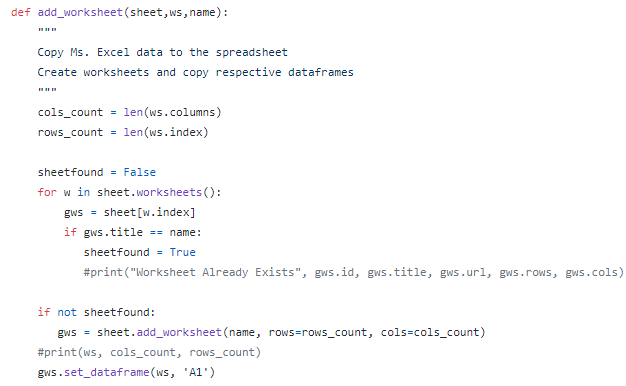
* This function creates a google spreadsheet entitled as per the input parameter “spreadsheet\_name” then it opens this spreadsheet and returns it as an object.



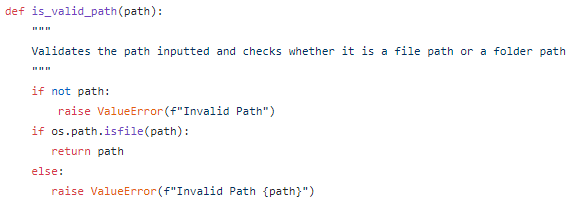
* This function shares the google spreadsheet with the list of emails specified as an argument and grants the user’s accounts of these emails write access on the latter.  
  Moreover and once this function is executed an automatic email is sent to their respective accounts to notify them about this spreadsheet.



* This function prints a summary exhibiting the ID, title/name, URL and last modified date of the spreadsheet specified as parameter.



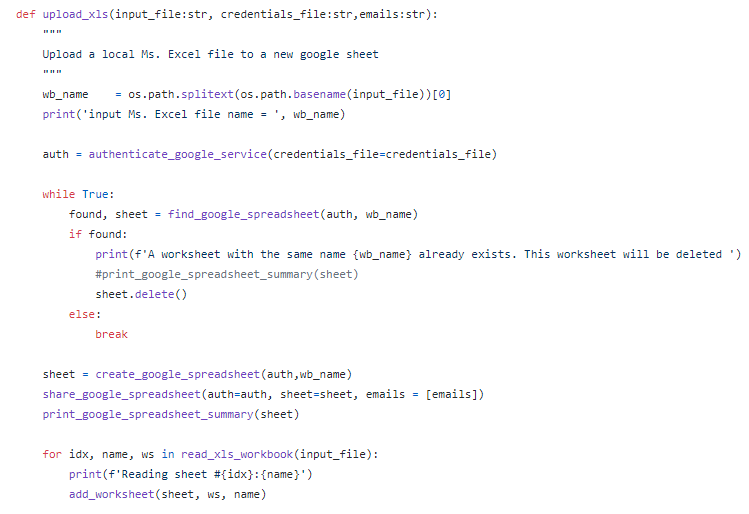
* This function creates a worksheet within the google spreadsheet, set its title as per the parameter specified and copies the inputted data frame to it.



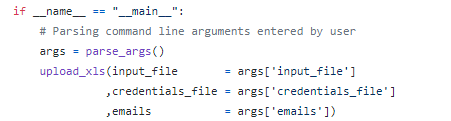
* This function validates a path inputted as a parameter and ensures that it is a file path.



* This function defines and sets the appropriate constraints for the command line arguments to be specified by the user when running this utility.
* I will describe hereafter the defined arguments:
  + Input file: A required parameter to input the path of the Ms. Excel file to be processed, this parameter is associated with the function ‘is\_valid\_path’ previously defined.
  + Credentials file: refer to the path of the respective service account key file.
  + Emails: the list of emails to share with the google spreadsheet to be generated.



* This function constitutes the core of our utility; It performs the following:
  + Authenticates Python to the google service account.
  + Deletes existing google spreadsheets having the title/name equal to the Ms. Excel workbook to be exported.  
    Worth noting that Google sheets allows you to create multiple spreadsheets having the same name but assigns for each one a different ID.
  + Creates a google spreadsheet having a title/name equal to the Ms. Excel workbook to be exported.
  + Shares this spreadsheet with the list of specified emails.
  + Iterates through the chosen Ms. Excel workbook and copies its respective worksheets to the google spreadsheet.

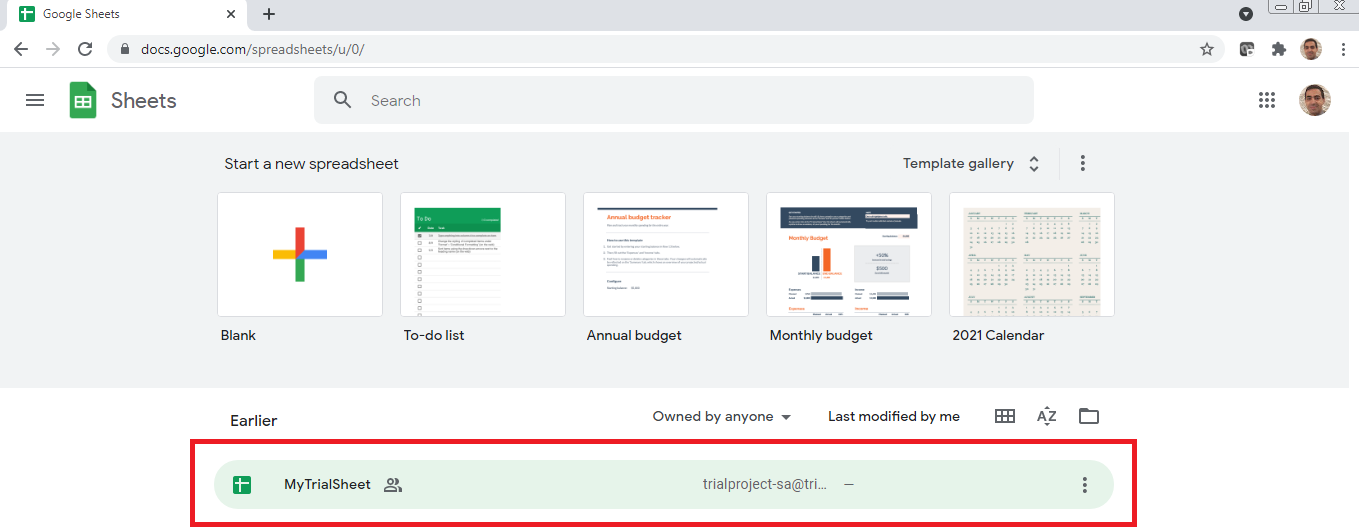
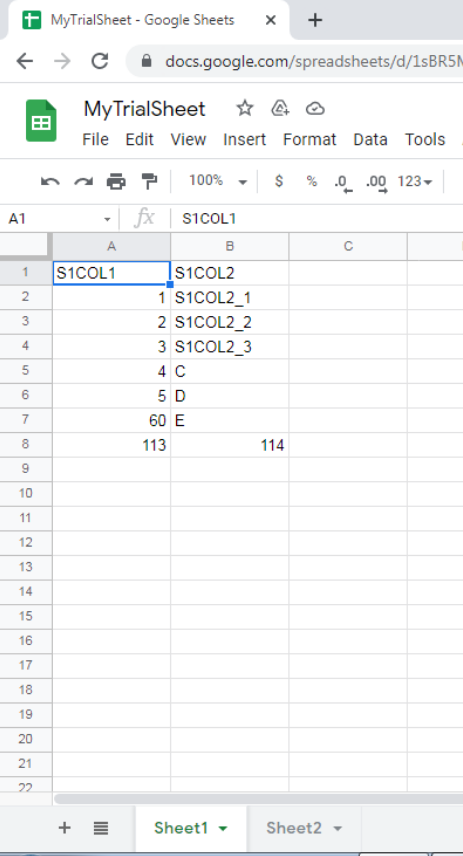


* The above represents the main function of our program.

Let’s test our program:  
  
Kindly proceed as per the following steps:

1. Open up a terminal window and type the following in it:

**update\_xls** **–i** ".\static\MyTrialSheet.xlsx"

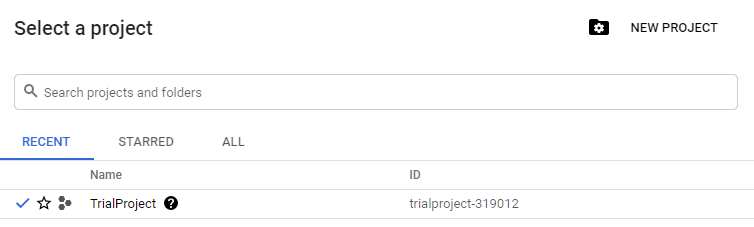
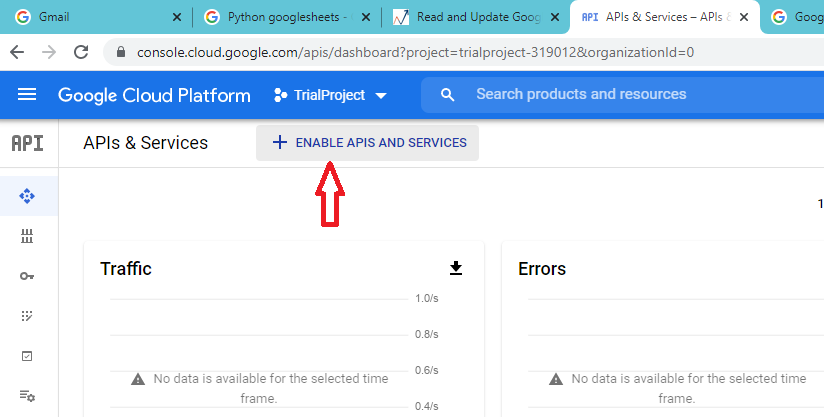
1. Log into the respective google account and check the generated spreadsheet:  
     
     
   

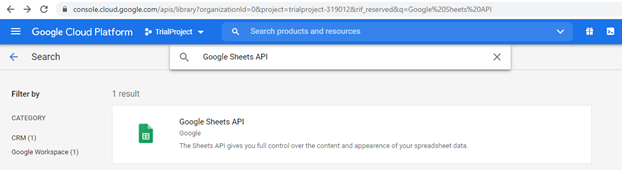
Final Words  
  
With automation, you can export data to Google Sheets from Ms. Excel and many other sources with just a few clicks and benefit from its outstanding capabilities.  
  
Hope you enjoyed this article.

|  |  |
| --- | --- |
|  | ***Bassem Marji*** *is a project implementation manager at BLOM Bank with a proven track record of success.  He managed the implementation of over 50 projects and propelled the digital transformation of mission critical applications. He spends his free time discovering the latest technology trends in the IT field.* |

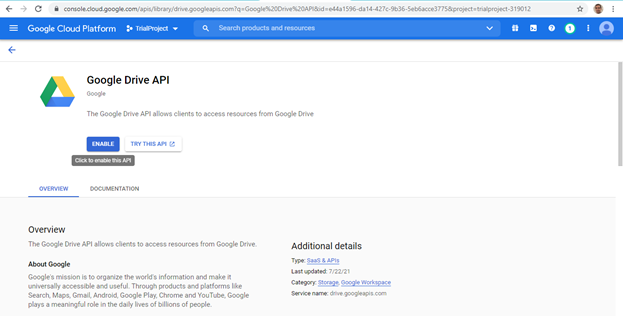
Appendix

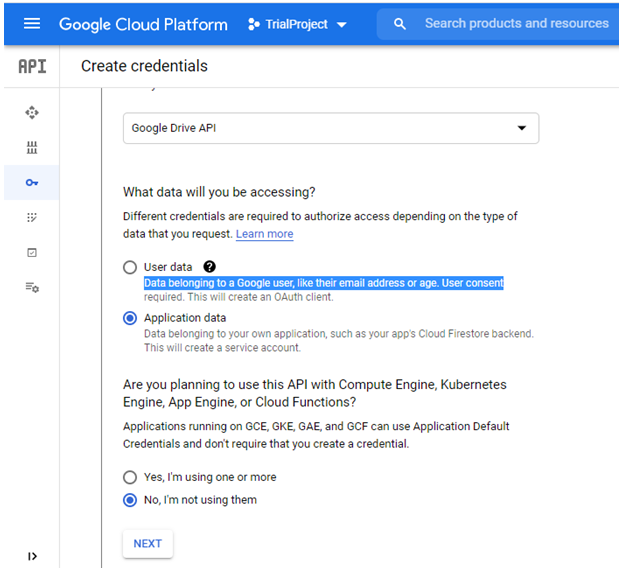
*Creating a Google service account:*  
A Google service account is a special kind of account used by an application or a virtual machine (VM) instance (not a person) in order to make authorized API calls to Google Cloud Services.  
  
Before proceeding with the next steps, please ensure that you have a Google account.

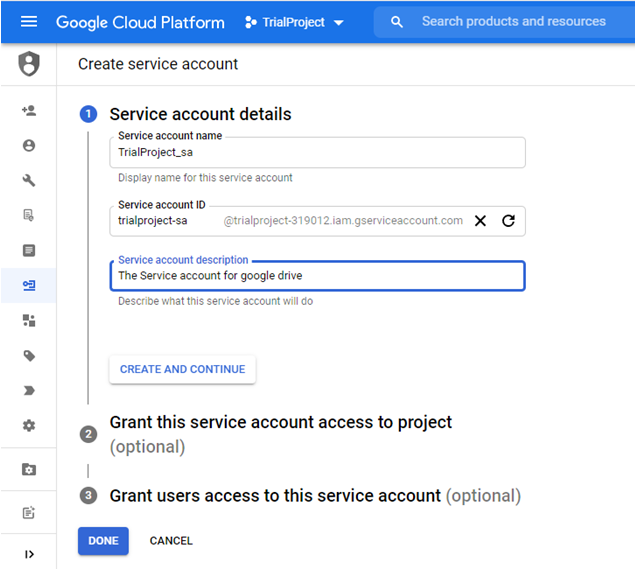
* + - 1. Head over to Google Developer console “<https://console.developers.google.com>” and either select an existing project or create a new project:  
         For the purpose of this tutorial I created a new project named “TrialProject”.  
         
      2. Click on “ENABLE APIS AND SERVICES”:  
           
         
      3. Search for and enable “Google Sheets API”:

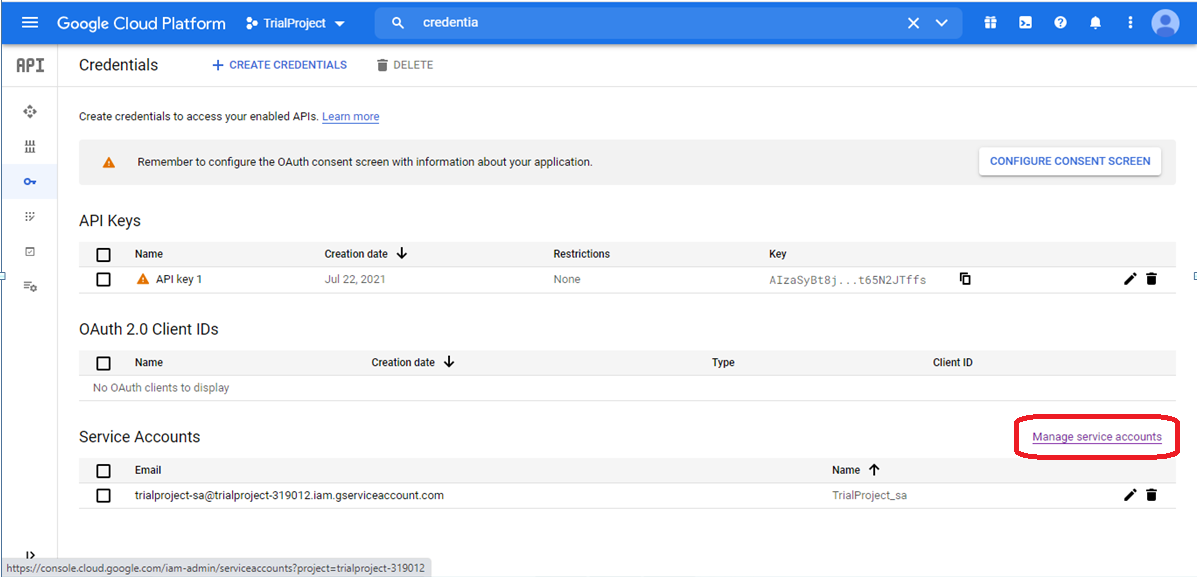
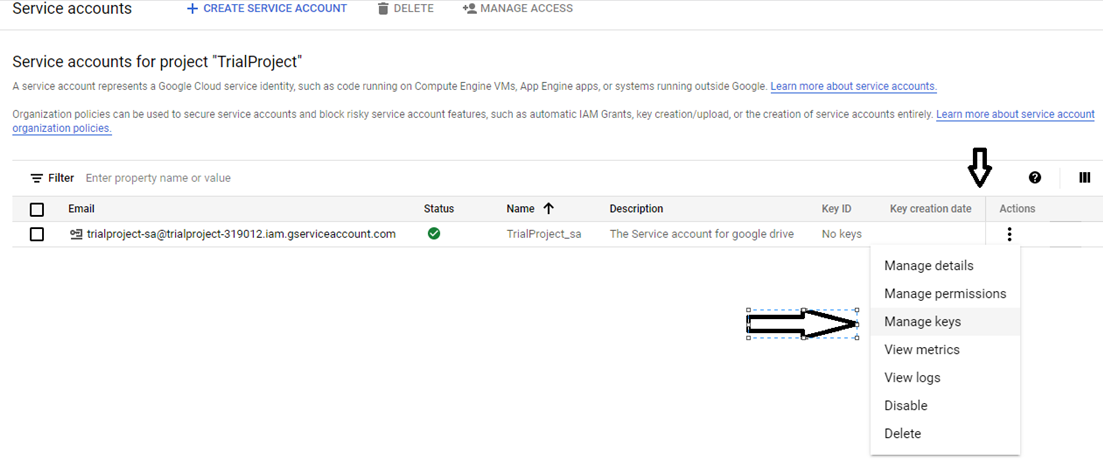
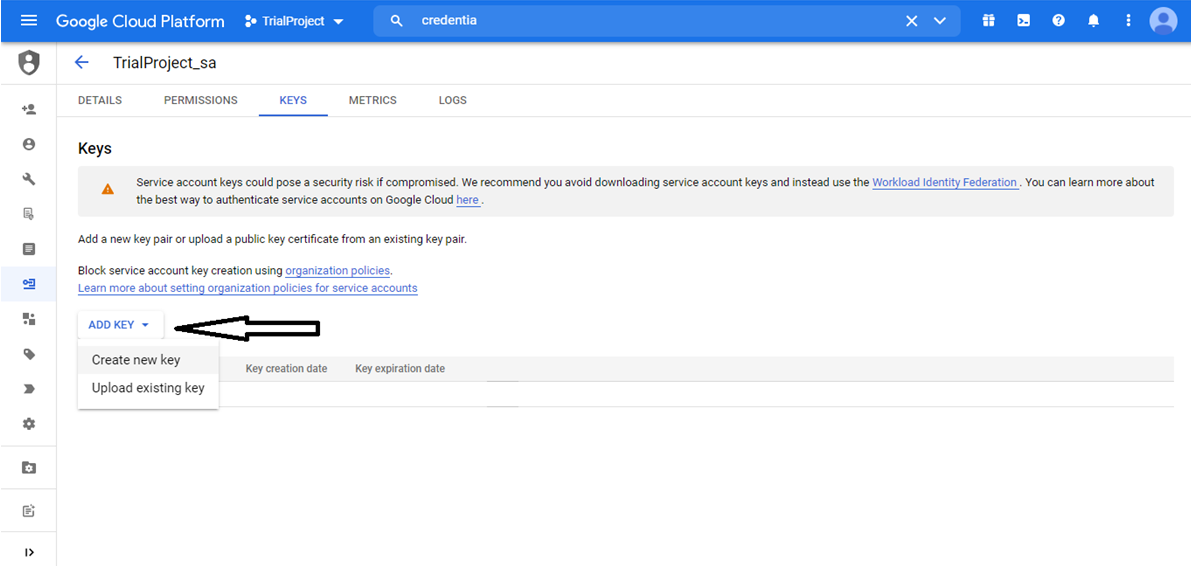


* + - 1. Search for and enable “Google Drive API”:



* + - 1. Once the needed APIs are enabled in the corresponding project then you have to create credentials for the service account. Click on the Create Credentials button to continue:  
         
      2. Create a service account:



* + - 1. Select “Service Accounts” and press on Manage Service Accounts:  
           
         
      2. Press on Manage Keys:  
           
         
      3. Create new key   
           
         
      4. Create a private key file (.Json) for the corresponding project.  
           
         