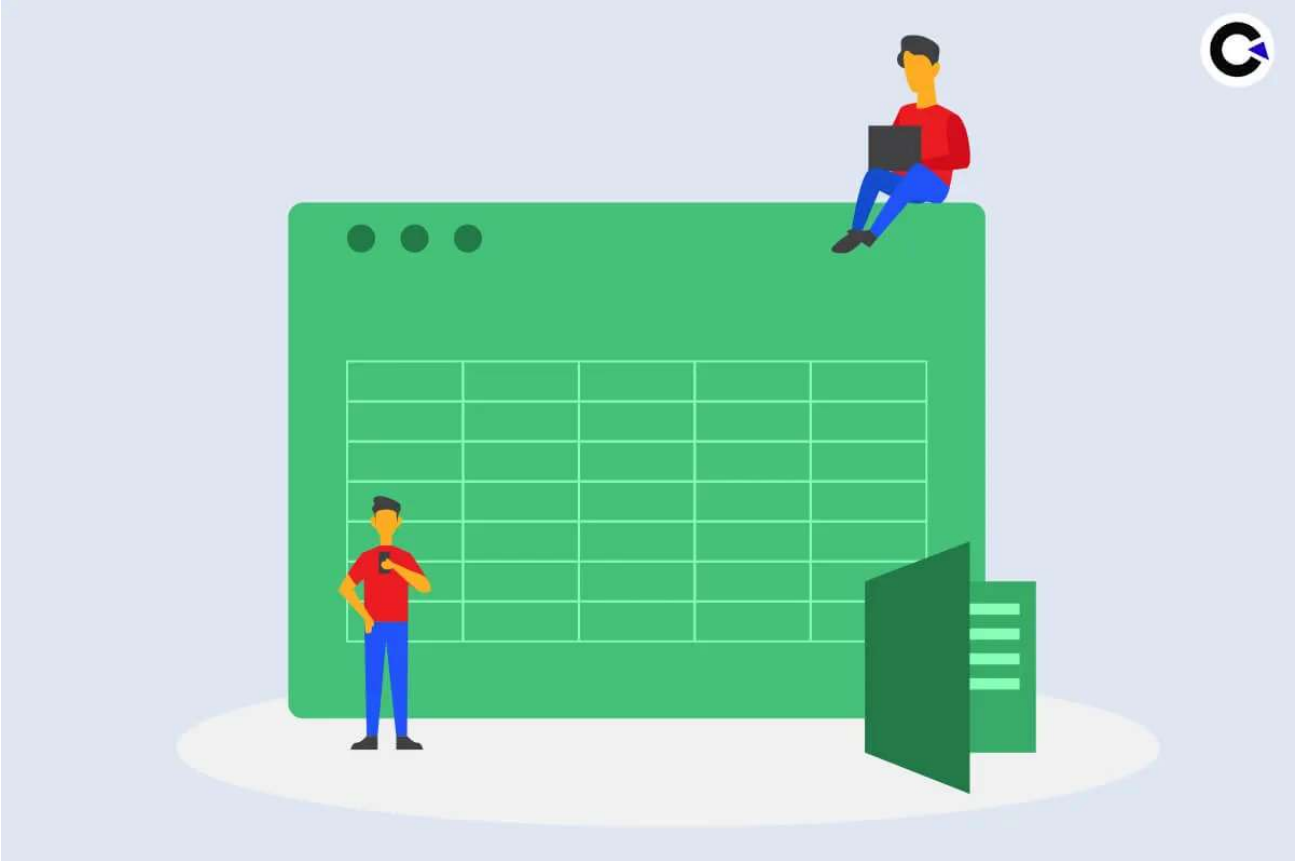
gspread Python Tutorial for Google Sheets Automation

Learn the full setup process & find out how to open a spreadsheet, read & insert data into it and much more from our gspread Python Tutorial

Hannah Rivera
Posted On 23/02/2023



Listen to this blog 0:00 / 0:00

On average, more than 2 Billion people use Google Sheets on a monthly basis. It is a powerful tool for managing and analyzing data. It is a popular choice among individuals as it is very user-friendly in nature. Additionally, it is also widely used by professionals and businesses. With continuous & extensive usage, few tasks could end up becoming repetitive and time-consuming. But with the help of automation, you will be able to overcome this major issue and streamline your workflow and perform tasks more efficiently. Being an **automation testing company**, we have written this gspread Python tutorial to help you automate google sheets using Python. So let's get started.

There are actually several libraries such as pygsheets, gdata, and gspread that one can use to automate Google Sheets using Python. And we have picked gspread for this particular tutorial. We'll explore the basic setup & configuration required to get started, explore some common use cases and then dive into how we can a

Google Sheet Automation Set up

Before we explore how to automate Google Sheets using Python in our gspread Python Tutorial, we'll

Install a Python library

We have chosen gspread for this tutorial as it is one of the most popular and widely used libraries. So the first step in our gspread Python Tutorial is to install the gspread library that is needed to achieve Google Sheets automation using Python.

You can install gspread using the below command,

Command:

```
pip install gspread
```

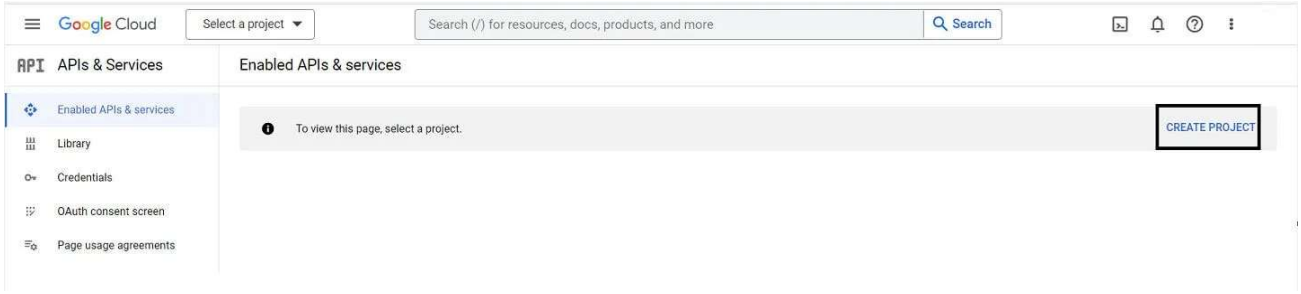
Output:

```
pip install gspread
Collecting gspread
  Downloading gspread-5.7.2-py3-none-any.whl (40 kB)
     ---------------------------------------- 40.5/40.5 kB 644.2 kB/s eta 0:00:00
Requirement already satisfied: google-auth-oauthlib>=0.4.1 in c:\users\admin\appdata\local\program
ckages (from gspread) (0.8.0)
Requirement already satisfied: google-auth>=1.12.0 in c:\users\admin\appdata\local\programs\python
```

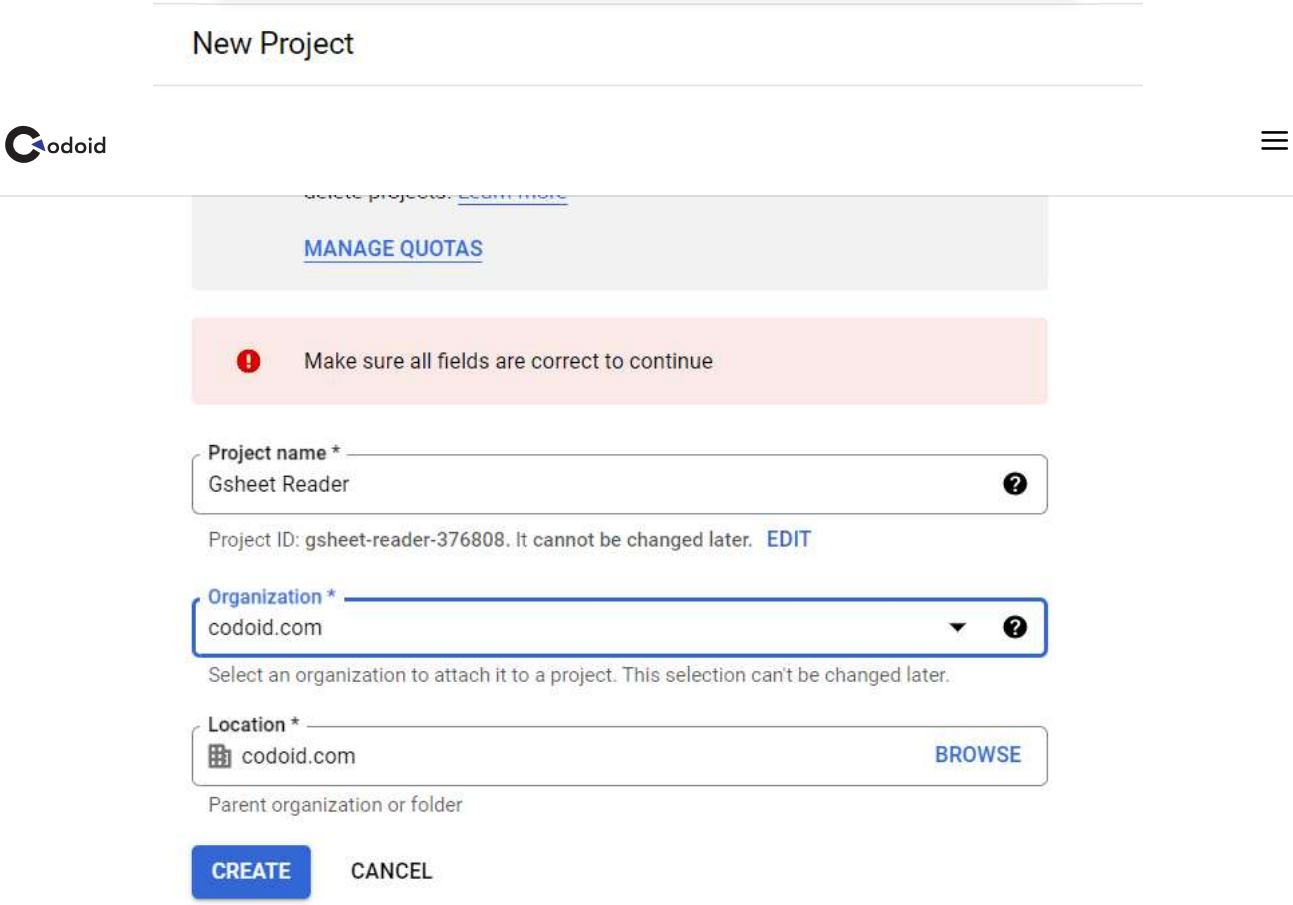
Once you have installed the library, you'll need to set up authentication. And to do that, you'll have to create a "service account" in Google API Console and grant access to the Google Sheet you want to automate. Let's see what steps have to be followed to do it.

Create a New Project

Navigate to Google Developers Console and click on 'Create Project'.

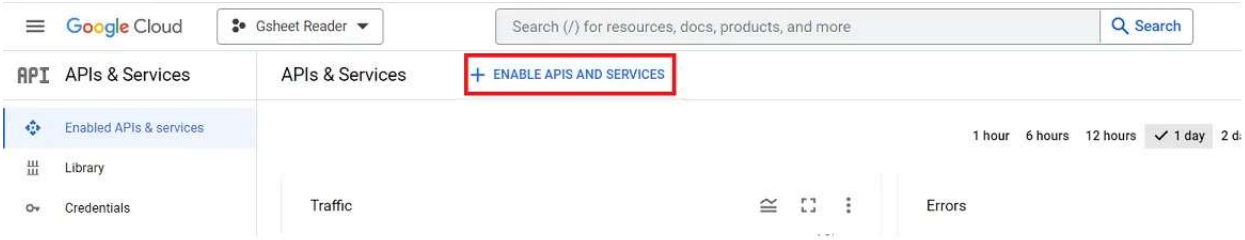


For explanation purposes, we have created a project called 'Gsheet Reader' for our gspread Python Tutorial. You can and clicking the 'Create' button.

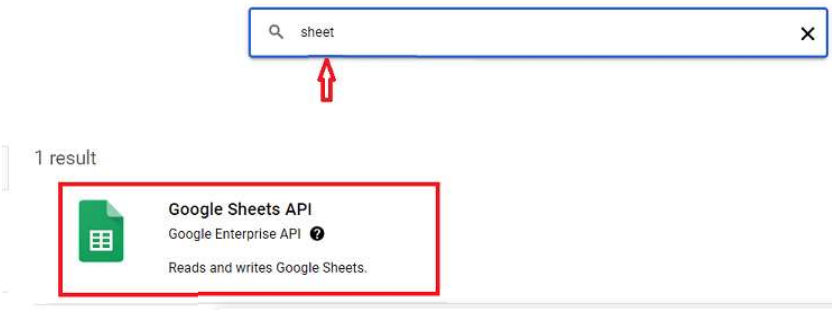


Enable API & Services

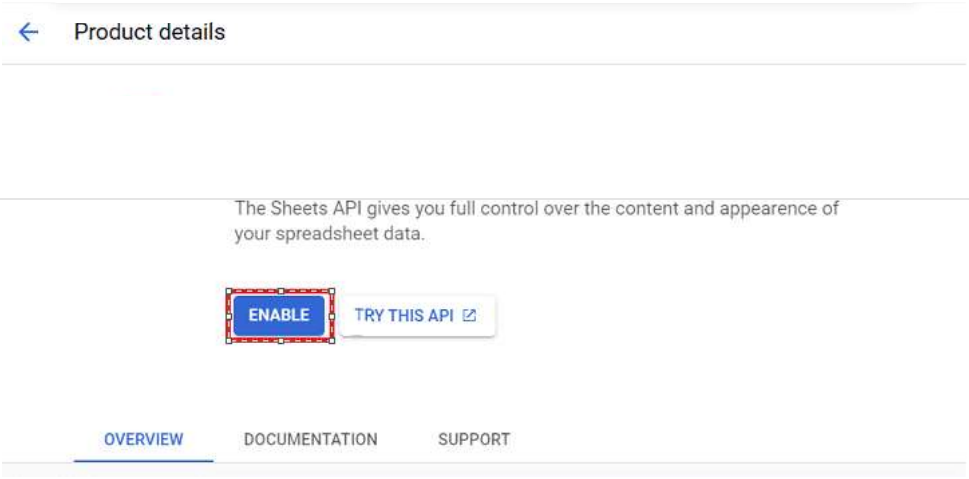
As the project has been created, you will see a new option to 'Enable APIs and Services' as shown below. You'll have to click on it to add the Google Sheet API.



There will be a list of APIs and you can enter an appropriate keyword such as 'sheet' in the search bar and select the 'Google Sheets API'.



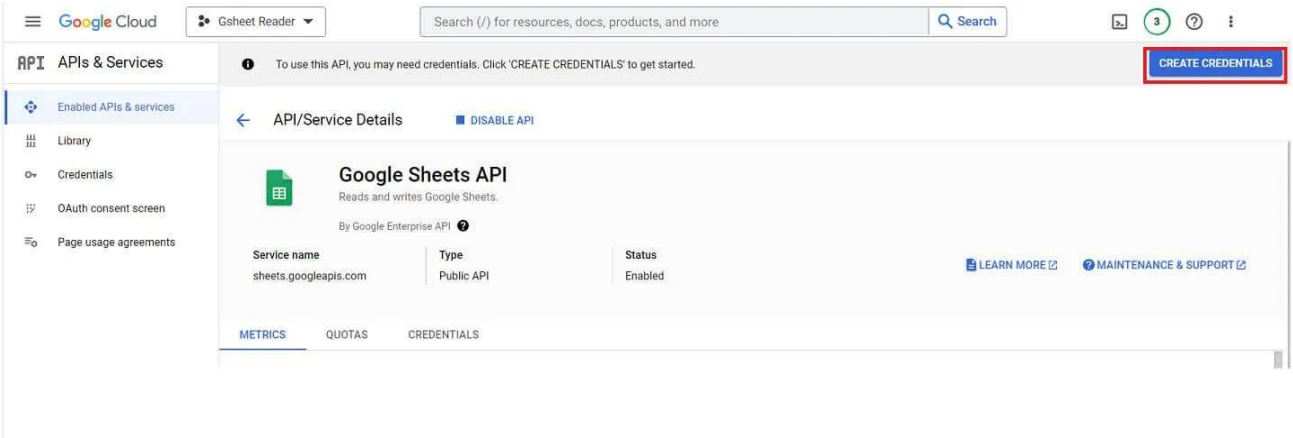
You can then click



Create Credentials:

Google Sheets API will now appear in your Enabled APIs & Services tab. You will see 3 sections namely Metrics, Quotas, and Credentials under it. So the next step in our gspread Python Tutorial would be to see how to create the required credentials to access the Google Sheet API.

So click the 'Create Credentials' button and follow the below steps.



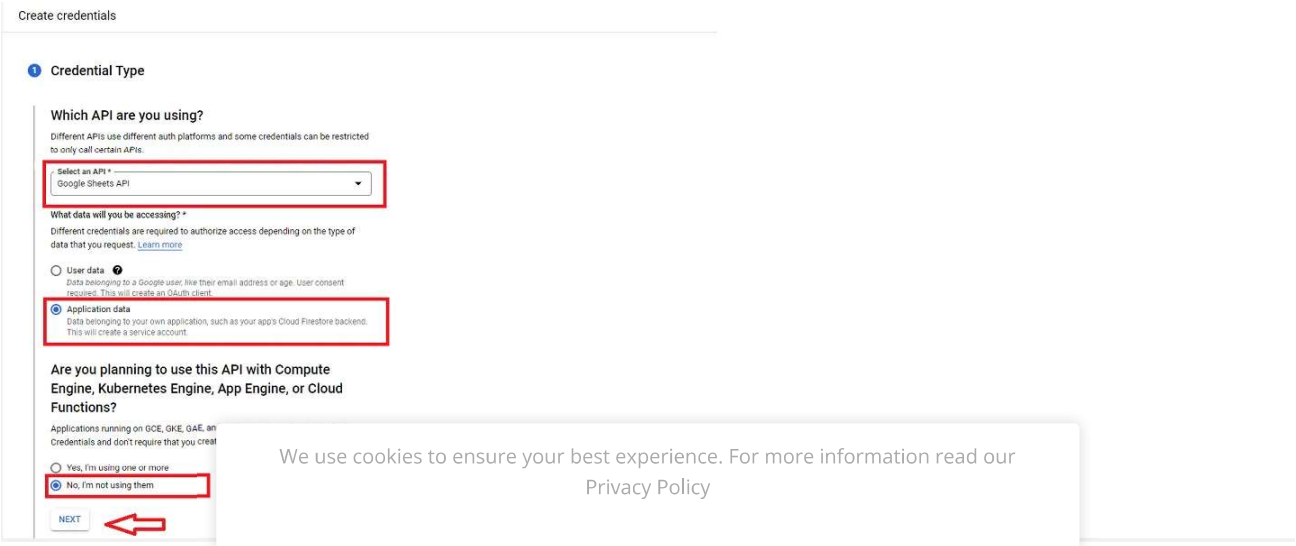
Credential Type

Select 'Google Sheets API' in the 'Select an API' drop-down

Select the 'Application Data' radio button, and

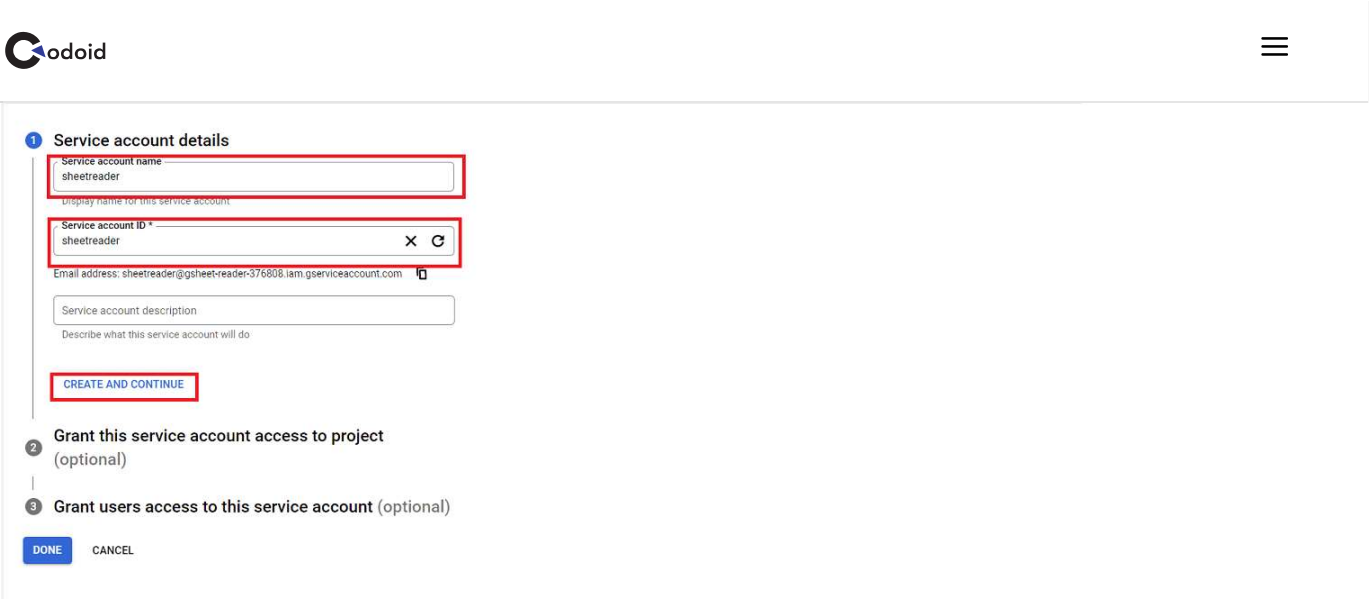
Choose the 'No, I'm not using them' option in the last question.

Click 'NEXT'.



Service Account Details

After which, you'll have to enter the Service account name & Service account ID in the respective fields

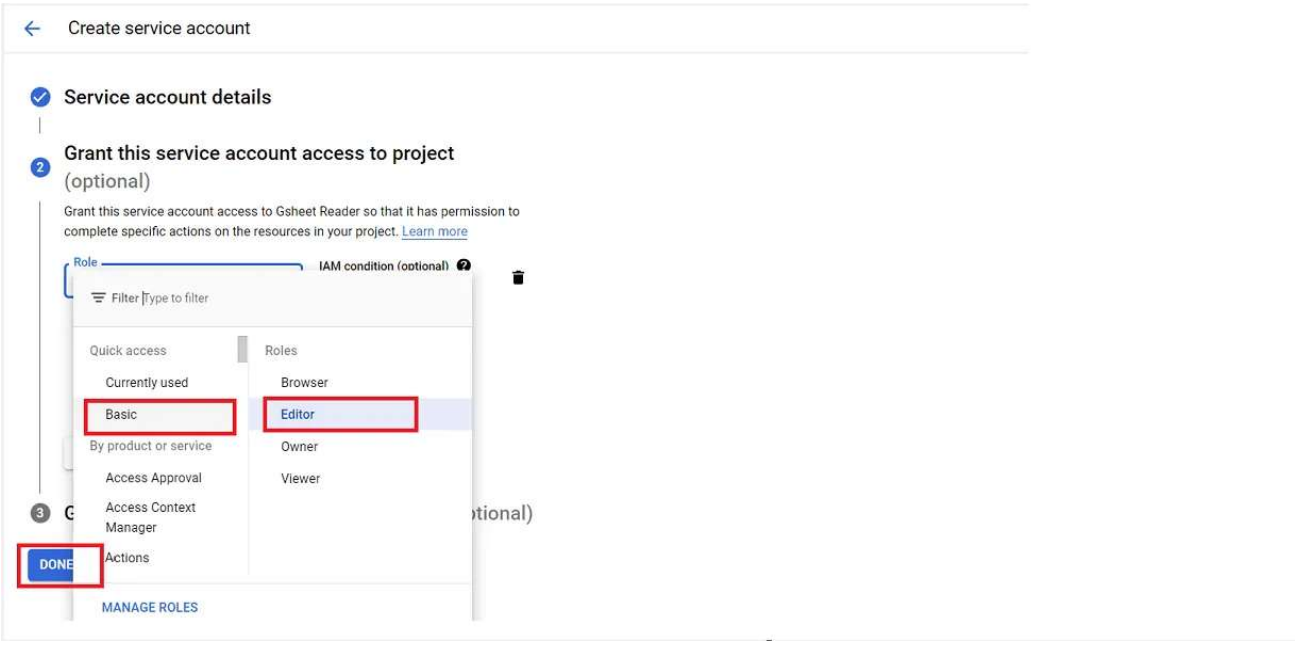


Grant Access

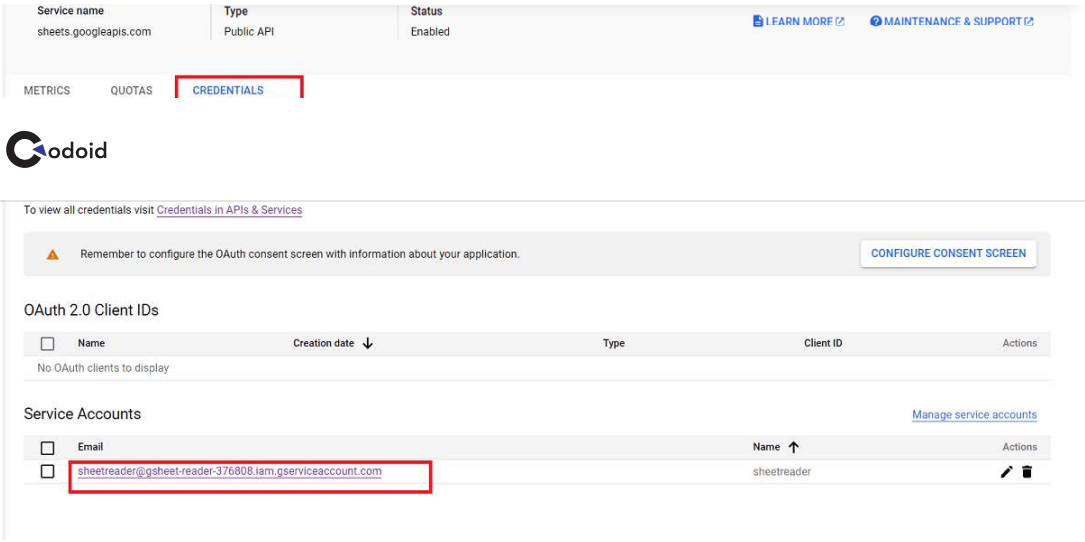
Now, you'll have to specify the level of access you are going to provide for the service account.

Click on 'Select a Role' and select 'Editor' under the 'Basic' section. Based on your needs, you can choose whichever role will be the aptest.

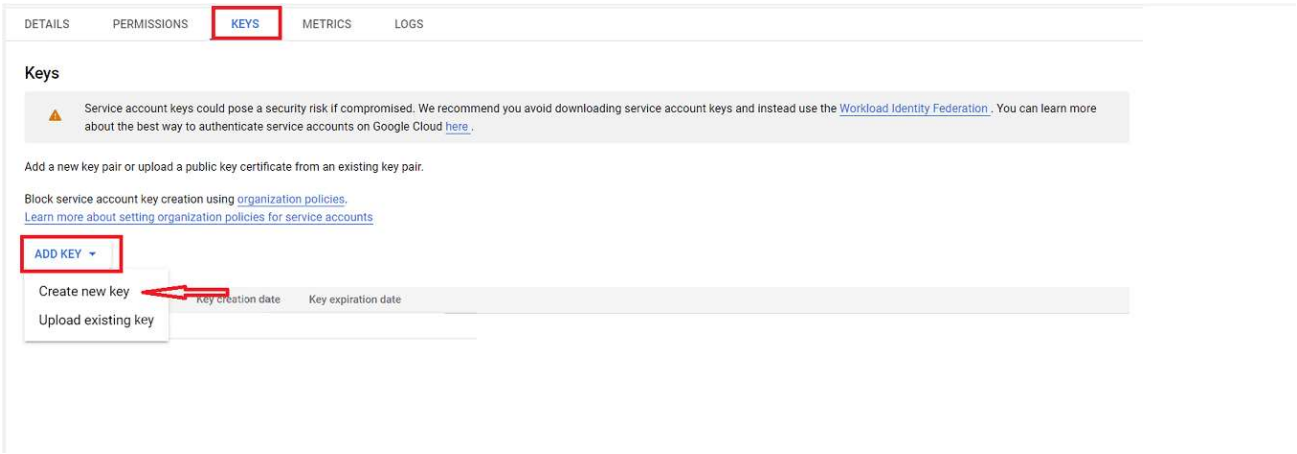
Press 'CONTINUE' and leave the other optional fields and click DONE.



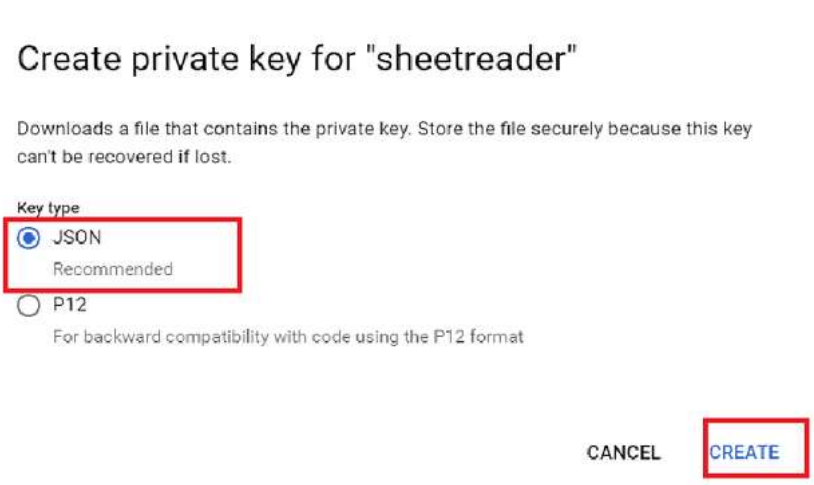
You will now be able to see the newly created Service account under the Credentials section.



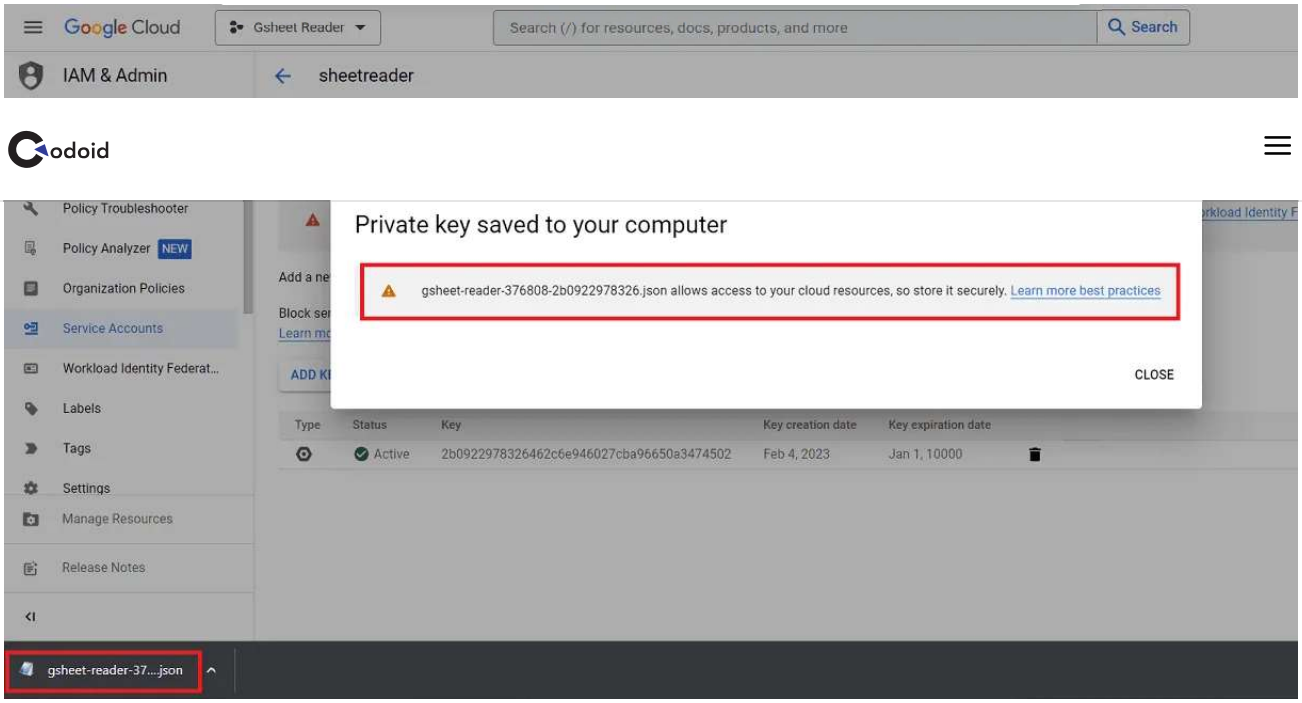
Under the Keys Section, you can click on 'ADD KEY' and choose 'Create new key' to create a credential secret key.



There are two key types, and as suggested by Google, it is better to choose the JSON key type. After making the selection, click the 'CREATE' button.



Once you click the button, the key will be downloaded to your computer. Please make sure to not share this file with anyone else and keep it safe.

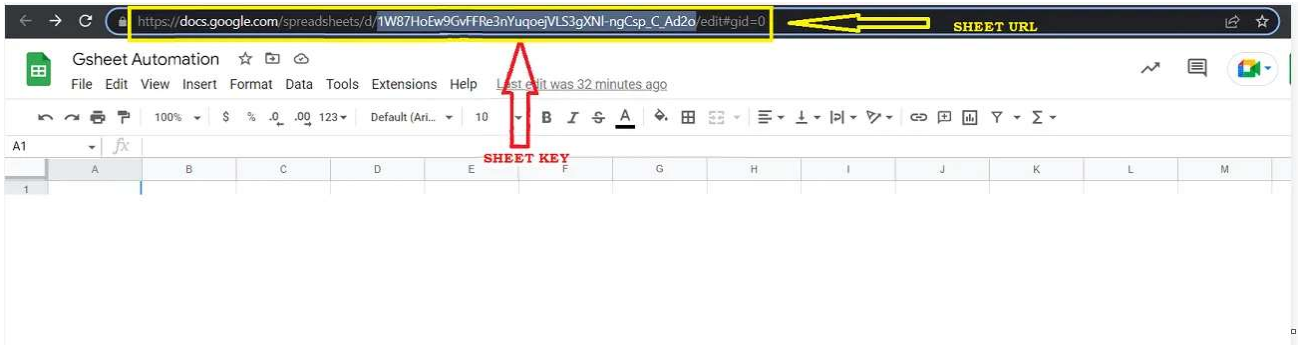


Tip: It will be helpful if you rename the file to credentials.json as it will be easy for you to remember the file name.

Create a Google Sheet

So the last step in our setup process is to create a Google Sheet that we can use to store our data. Once we are clear with that, we can go ahead and see the different Python code commands you can use to perform Google Sheet Automation.

If you already have a Google sheet, you can just copy the sheet url or sheet id as shown in the image as this copied string will be needed to access Google Sheets.



gspread Python Tutorial

Everything you'll need to achieve Google Sheet Automation is now ready. So you can start automating the listed actions by typing the Python code commands we have mentioned in a code editor.

- Opening a Spreadsheet
- Get a Worksheet's Name
- Read Data from a Sheet
- Insert Data into the Sheet
- Update Multiple Ranges

Opening a S

First up in our gspread Python Tutorial, we'll be seeing how to open a spreadsheet using Python code commands. Remember the URL we had copied after creating a new Google Sheet? We will now use it to

Code:

```
import gspread


Sheet_credential = gspread.service_account("credential.json")


# Open Spreadsheet by URL
# spreadsheet = Sheet_credential.open_by_url('paste your sheet url')


# Open Spreadsheet by key
spreadsheet = Sheet_credential.open_by_key('paste your sheet key')
print(spreadsheet.title)
```

Output:

You can see that the name of the Google Sheet we created (Gsheet Automation) has been fetched here.

Get a Worksheet's Name

A Spreadsheet can have more than one worksheet and if you wish to print the name and ID of a worksheet, you can use the below Python code commands.

Code:

```
# to print worksheet name using sheet id
worksheet = spreadsheet.get_worksheet(0)


# to print worksheet name using sheet name
worksheet = spreadsheet.worksheet('Sample Sheet')
print(worksheet)
```

Output:

Read Data from a Sheet

Next up in our gs use to read data from a Google Sh ific worksheet and printing it.

Code:

```
all_values = worksheet.get_all_records()
for value in all_values:
    print(value)
```

Output:

Insert Data into the Sheet

Reading the data alone isn't enough. You'll have to insert data into a sheet as well. So, let's look at the Python Code command you'll have to use to achieve that. In our example, we will be updating multiple values after a defined row and read & print the updated values.

Code:

```
# Update multiple values after A6 row
worksheet.update('A7', [["106", "Robert","J","Pass" ], ["107", "Robert","G","Fail"]])

# read data after update
update_data = worksheet.get_all_values()
for valu in update_data:
    print(valu)
```

Output:

Update Multiple Ranges

Finally in our gspread Python tutorial, we are going to see the commands you can use to update multiple ranges of data at the same time.

Code:


```
# Update multipl
```

```
worksheet.batch_update([{
    'range': 'E1:F2',
    'range': 'G1:H2',
    'values': [['A', 'AB'], ['C', 'CD']],
}])
# To Read data after the update
update_data = worksheet.get_all_values()
for valu in update_data:
    print(valu)
```


Output:

Related Blogs

- Desktop App Automation Testing using Python
- Selenium WebDriver with Python Cheat Sheet
- Java Excel Libraries

CONCLUSION

So in our gspread Python tutorial, we have laid the foundations you'll have to know to achieve Google Sheets automation using Python. By following these simple steps, you can easily automate tasks such as data entry, data analysis, and report generation. We hope this will help you significantly improve your workflow and save you time. Being a leading big data analytics testing company, we have used Python libraries and found gspread to be the best of the lot. Hope you find it useful too.

Codoid

Automation Testing

A Conclusive Allure Report Tutorial with Sample Report

Automation Testing

Allure report vs Extent report. Which is Better?

Automation Testing

Cypress Limitations you should be aware of

Automation

Best Cypress Must Know

Search..

SUBSCRIBE

Sign up for our newsletter to never miss out on our latest blogs.

username@mail.com

Talk to our Experts

Full Name*

Business Email*

Phone Number

Message*

SUBMIT