

# Getting Started: How To Download Data From Google Analytics

Before we begin, I want to start with a few caveats:

1. **Google APIs often change over time.** The approach I'm showing you works with [Version 4 of the Google Analytics API](#). Google APIs often change over time, and I can't guarantee that the code I show will work months or years from now.
2. **R isn't one of the official Google languages listed in their documentation.** I use the excellent `googleAnalyticsR` package written by Mark Edmondson in my dashboard, but Google only officially documents Java, Python, and PHP interfaces.
3. **This may not be the most efficient or best solution.** I hope readers who find simpler, easier, or better methods of authorizing the use Google Analytics API will document their methods as well. I encourage you to share them with the RStudio community using the `googleAnalyticsR` tag at <https://community.rstudio.com/tag/googleanalyticsr>.

With all that said, the dashboards that use this API provide insights into our blog use that would require a great deal of manual work to reproduce using the GA web interface.

## Install These Packages If You Don't Have Them Already

While the finished dashboards use 16 different R packages, the essential ones I use are:

- `gargle`. This package helps us set up our Google Analytics (henceforth abbreviated as GA) authorization and credentials.
- `googleAnalyticsR`. This essential package allows us to download the raw visitor data using the Google Analytics API.
- `flexdashboard`. This package allows us to present the results in a simple Web interface using R Markdown.
- `reactable`. This package allows users of the dashboard to browse, search, reorder, and interact with the data presented.

All of these packages are available for download at CRAN using `install.packages()`.

You should also create an R project for your dashboard at this time. We will need a place to store our Google Analytics credentials, and having a project ready to store them will keep things organized.

## Google Analytics credentials: The secrets to success

I want to begin by talking about what I found to be one of the most challenging pieces of the entire project: Creating, authorizing, and applying Google Analytics credentials. It's not hugely difficult, but it does have a lot of steps you must get right before you can get any data.

Here's a high-level overview of what we'll need to get the visitor data for our Google Analytics dashboard. To use the Google Analytics API, we need to present two types of credentials that represent:

1. **The user represented by the client:** The API only provides service on behalf of an authorized *user*. Most people are pretty familiar with this type of authorization; it's the equivalent of logging into Google with an email address and password. The trick in this case is that the email address we'll use will be one Google creates for our particular client.

2. **The client requesting service:** The API requires that we authorize and provide credentials for each *client* making requests. In our case, our client will be an R program, which is considered a desktop client. We'll request a *service account* to represent this our R program and allow direct server-to-server interactions without human interaction.

For any of this to work, the author of the dashboard has to be an authorized user of Google Analytics. You can test this by going to the [Google Analytics Home page \(analytics.google.com\)](https://analytics.google.com). If you are an authorized user, you'll see the web dashboard. If you aren't, you'll get an error message and will have to ask for access from your Google Analytics Administrator. Keep the contact information for your Google Analytics administrator handy; we'll need that information again later.

We must perform six steps to download data using the GA API. We need to:

- **Step 1:** Request a service account from Google.
- **Step 2:** Download the service account key and securely store its JSON file.
- **Step 3:** Enable API access to your project.
- **Step 4:** Add your service account credentials to your project.
- **Step 5:** Create and download your project's OAUTH credentials from Google.
- **Step 6:** Submit both pieces of information to the Google Analytics API and make a test data request.

I'll walk you through each step individually in the following sections. For readers not interested in the gory details, you can skip ahead to [the conclusion of this piece](#) where I'll recap what we got out of this process and what the next steps are.

## Step 1: Request a service account from Google

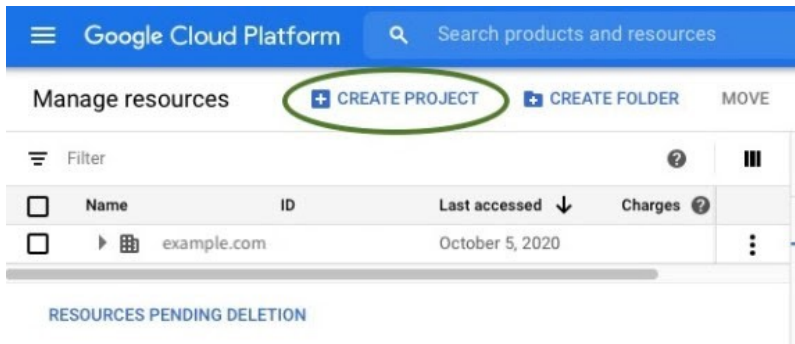
Google has written [a comprehensive document on how to do API authentication](#). Because we want to build a stand-alone dashboard, we're going to use the service account option, which Google describes this way:

Service accounts are useful for automated, offline, or scheduled access to Google Analytics data for your own account. For example, to build a live dashboard of your own Google Analytics data and share it with other users.

This sounds like exactly what we want, so let's use that option. It will take a few sub-steps, but they are fairly straightforward. Jenny Bryan has written a nice overview about how this process works as part of her [gargle package](#); the description of service accounts is at the bottom of the page.

To create your service account, you should:

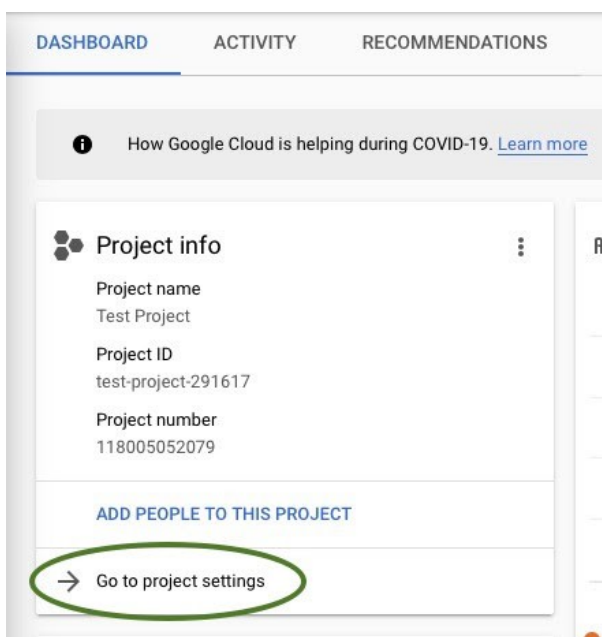
1. **Go to <https://console.cloud.google.com/cloud-resource-manager> and click on **Create Project**** (see figure below). While you could also use the [web-based Google setup tool](#) recommended by the Google document, I find that using the cloud resource manager page referenced above simplifies naming your project something other than "My Project".



2. **Give your project a name.** Here, we've named our project *Test Project*. Click *Create* once you've entered a name.

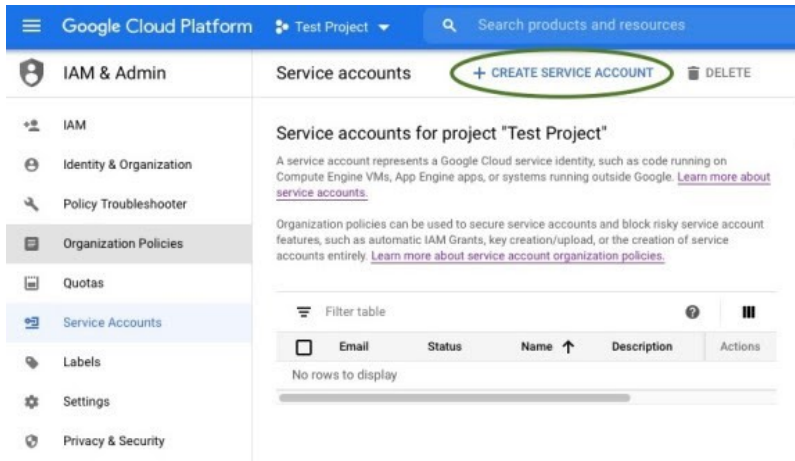
The screenshot shows the 'New Project' form in Google Cloud Platform. The 'Project name' field is filled with 'Test Project' and is circled in green. Below it, the 'Project ID' is 'test-project' and it's noted that it cannot be changed later. The 'Organization' field is filled with 'example.com'. The 'Location' field is also filled with 'example.com'. At the bottom, the 'CREATE' button is circled in green, and there is a 'CANCEL' button next to it.

3. **Click on *Go to project settings* on the Google Cloud Dashboard project card.** Usually the new project's card will be at the top left of the page and should have the project name and number. You'll now go to your project settings page to create a service account to access this project.



4. **Select *Create Service Account*.** You now have a project created, but you don't yet have

a Google user account that can be used with that project. We'll create that on the *Service Account Details* screen.



5. **Give your service account a name.** The name will automatically populate the Service Account ID field. **Record the full Service Account ID generated somewhere; we'll need to register that account with your Google Analytics administrator.** It's also a good idea to provide a long description of what you intend to do with this account. When you've finished with filling in these fields, click *Create*.

A screenshot of the 'Create service account' form in the Google Cloud Platform console. The form has two steps: '1 Service account details' (active) and '2 Grant this service account access to project (optional)'. Under 'Service account details', there are three input fields: 'Service account name' with the value 'GA-analysis' circled in green, 'Service account ID' with the value 'ga-analysis' circled in green, and 'Service account description' with the value 'This service account is for accessing Google Analytics through googleAnalyticsR' circled in green. At the bottom, there is a 'CREATE' button circled in green and a 'CANCEL' button.

This completes the creation of our service account.

## Step 2: Download the service account key

1. **Now that your service account exists, download your key from the three vertical dots menu.** Once your account is created, the dashboard will take you back to the *Service Accounts* page as shown below.

Service accounts + CREATE SERVICE ACCOUNT DELETE

### Service accounts for project "Test Project"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. [Learn more about service accounts.](#)

Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. [Learn more about service account organization policies.](#)

Filter table

<input checked="" type="checkbox"/>	Email	Status	Name ↑	Description	Key ID	Key creation date	Actions
<input checked="" type="checkbox"/>	<a href="mailto:ga-analysis@test-project-291617.iam.gserviceaccount.com">ga-analysis@test-project-291617.iam.gserviceaccount.com</a>	✓	GA-analysis	This service account is for accessing Google Analytics through googleAnalyticsR	No keys		⋮

2. **Create your private service account key.** Make your browser window wide enough to see the *Actions* menu with the three vertical dots. Click on those 3 vertical dots, and you'll see a pop-up menu. Click on *Create key*.

Service accounts + CREATE SERVICE ACCOUNT DELETE

### Service accounts for project "Test Project"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. [Learn more about service accounts.](#)

Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. [Learn more about service account organization policies.](#)

Filter table

<input checked="" type="checkbox"/>	Email	Status	Name ↑	Description	Key ID	Key creation date	Actions
<input checked="" type="checkbox"/>	<a href="mailto:ga-analysis@test-project-291617.iam.gserviceaccount.com">ga-analysis@test-project-291617.iam.gserviceaccount.com</a>	✓	GA-analysis	This service account is for accessing Google Analytics through googleAnalyticsR	No keys		⋮

Edit  
Disable  
**Create key**  
Delete

3. **Select JSON as your key format.** The `googleAnalyticsR` package requires the key to be in JSON format. Once you've selected that format, click *Create*.

through  
googleAnalyticsR

### Create private key for "GA-analysis"

Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.

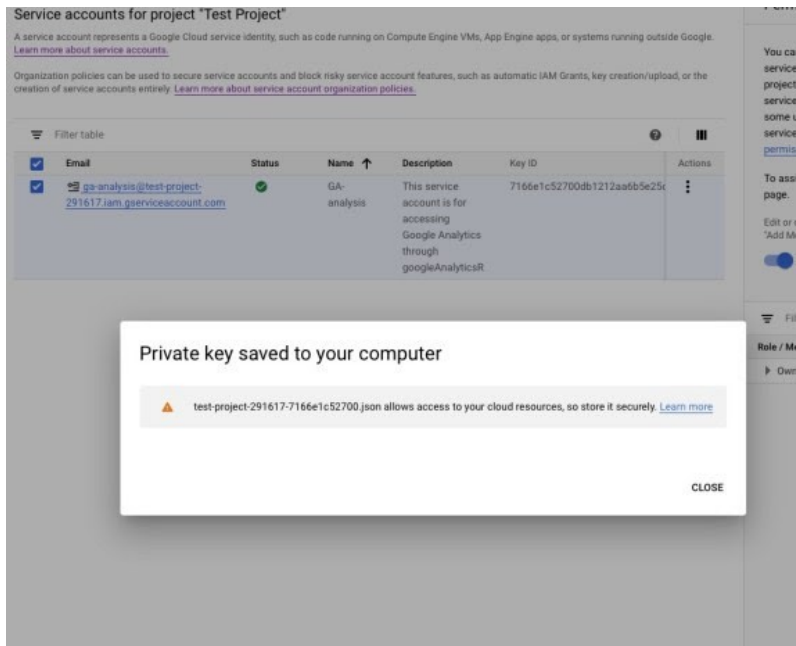
Key type

☒ JSON  
Recommended

☐ P12  
For backward compatibility with code using the P12 format

CANCEL **CREATE**

4. **Store the downloaded key in a folder within your R project.** I typically create a folder in my dashboard project named `.secrets` where I keep such keys.

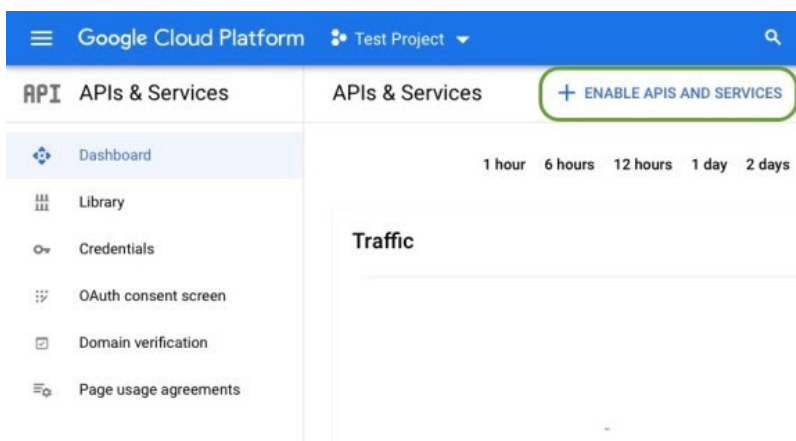


At this point, you have the service key credentials you need to make requests. However, we still have a couple more steps to do before we can use the API.

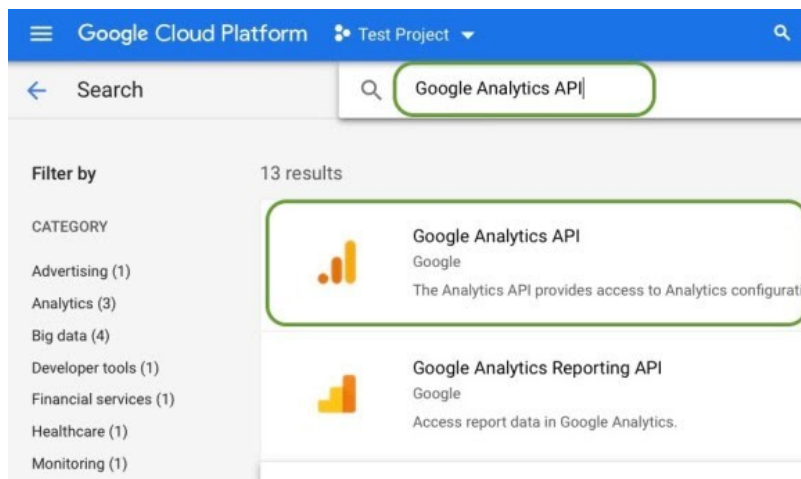
### Step 3: Enable API access to your project

The fact you have a valid service key is not enough to start making requests. You still need to enable the API from the Google Dashboard. To do this you:

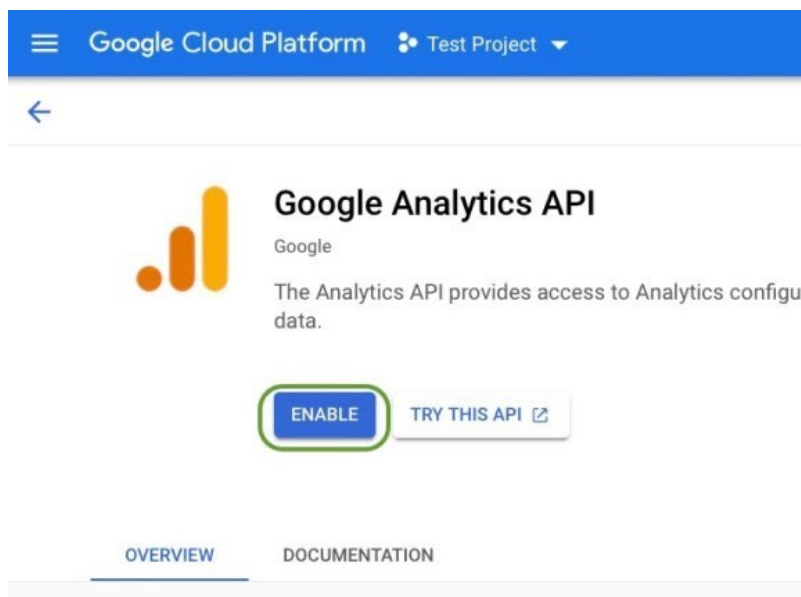
1. Go to <https://console.cloud.google.com/apis> as shown in the screenshot below and then click on **Enable APIs and Services**.



2. Search for and click on the **Google Analytics API**.



3. Click on **Enable** to make the API for your project active.



Sadly, the fact you have a valid service key is not enough to start making requests yet. We still need to authorize the user account with GA.

#### Step 4: Add your service account user credentials to your project

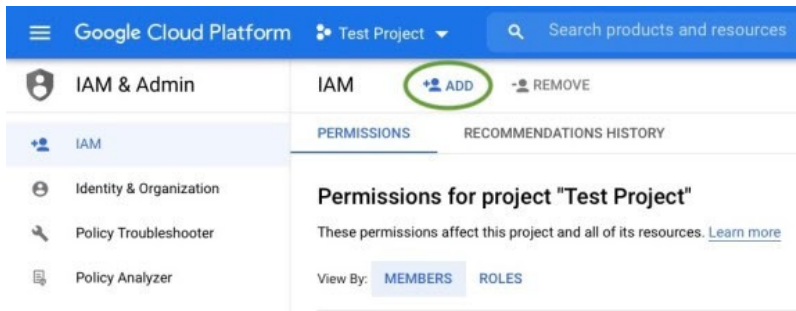
You now need to add the email associated with that key to the list of authenticated project users. To do this, we're going to return to the Cloud Resource Manager pane at <https://console.cloud.google.com/cloud-resource-manager>.

Please note that for many Google Analytics configurations, **only GA administrators may add new members to a project**. If that is the case for you, you'll will not see the screens shown below. Instead, you must contact your GA administrator and ask them to add your service account email to the project with Viewer rights.

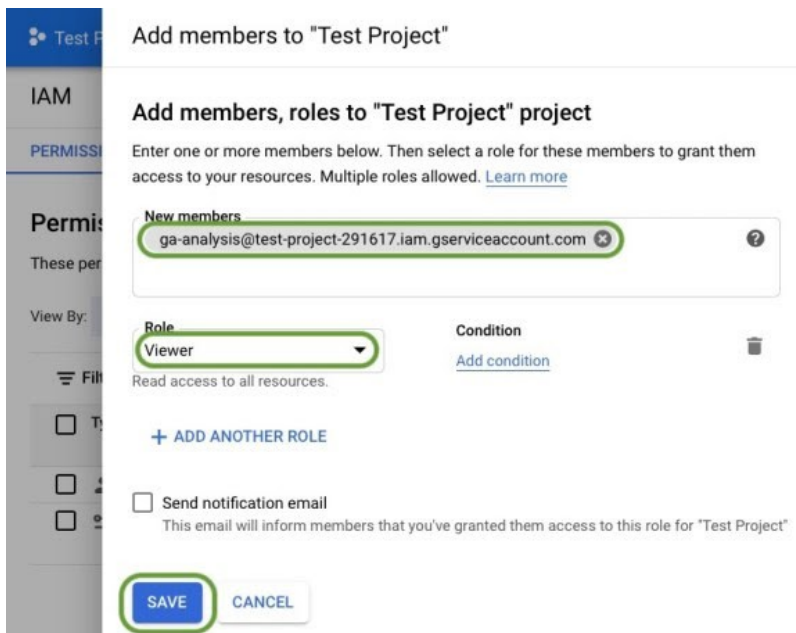
If you do have the appropriate permissions, however, perform the following 3 tasks:

1. Click on the **IAM** selection on the left-hand-side menu and select **ADD** from the top submenu as shown below:

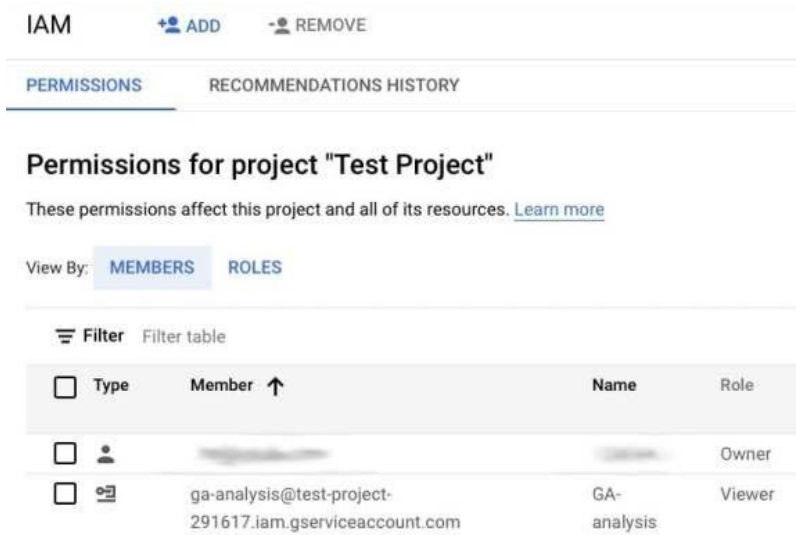




2. **Add your service account user to the project.** Enter the email address for your service account, select *Viewer* as the role, and click *Save* as shown below.



3. **Verify that your service account email has now been added** by observing it in the list of members for this project.



## Step 5: Create and download your project's OAUTH credentials from Google

While you may be questioning why you ever started this seemingly endless project at this point, fear not; we're almost done. All that remains to do is to create and download the OAUTH credentials for your service key.



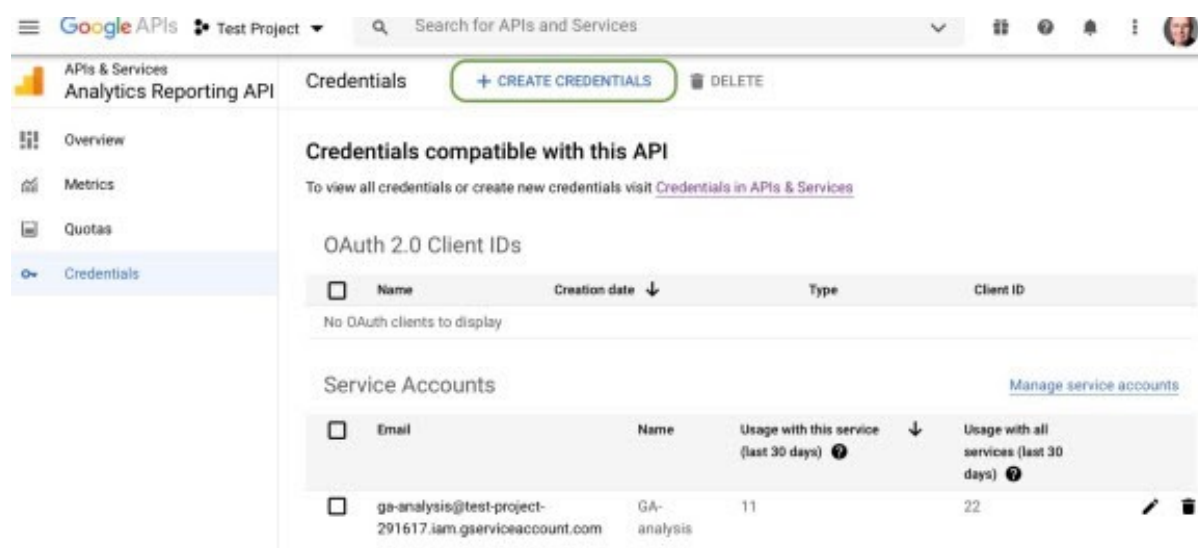
Now if you're anything like me, you're probably thinking "Wait a minute, I created a service key to bypass all this OAUTH complexity. Why do I need an OAUTH project file now?" I'm glad you asked; it's because Google:

- **Gathers API statistics on a per-project basis.** Google needs to know what project to aggregate your Google Analytics API calls under for reporting and accounting purposes.
- **Needs to defend against excessive API calls.** Because you are accessing the API from a computer program, Google has to defend its API against infinite loops and automated attacks. Should Google detect excessive API calls associated with your project, it can throttle its responses to you without affecting other users.

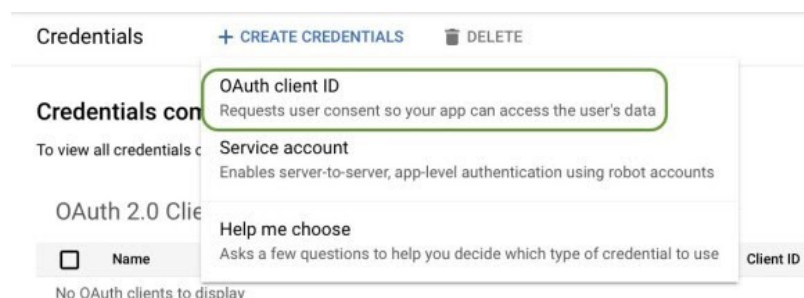
You don't actually need a project client ID for debugging purposes because the `GoogleAnalyticsR` package has a default project associated with it. However, this project ID is shared among all programs using the package, and you may find your API calls denied because too many users are actively using the package. You can avoid this issue entirely by setting your own project client ID as shown below.

In my opinion, acquiring an OAuth 2.0 client ID for a service account is poorly documented on the Google API dashboard, in the Google documentation, and in the `GoogleAnalyticsR` package. I found this process difficult to reproduce for our test project even though I'd been through it for my own dashboards. With that said, it's fairly straightforward if you start in the proper place as shown below:


1. Go to the site <https://console.developers.google.com/apis/api/analyticsreporting.googleapis.com/>. Please note that this is not the Google Cloud API dashboard we went to in Step 3; this is the Google Analytics Report API dashboard. You probably will have no OAuth 2.0 client IDs shown. Click on + *CREATE CREDENTIALS* at the top of the page.



2. Select *OAuth client ID* as the credential you wish to create.



3. Select **Desktop app** as the application type and enter your a name for your client. I chose the name “Test Google Analytics script.”

 Create OAuth client ID

---

A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, each will need its own client ID. See [Setting up OAuth 2.0](#) for more information.

Application type \*

Desktop app

[Learn more](#) about OAuth client types

Name \*

Test Google Analytics script

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

CREATE

CANCEL

4. Click **OK** to acknowledge the ID being created, which will return you to the Google Analytics dashboard.

## OAuth client created

The client ID and secret can always be accessed from Credentials in APIs & Services

**i** OAuth access is restricted to users within your organization unless the [OAuth consent screen](#) is published and verified.

Your Client ID  
118005052079-komp4qgpcj6-4d2jree7f1cjomseuf8t1\_apps..y

Your Client Secret  
Abu33N7nGmTmC10FQadot\_A

OK

5. **Click the down arrow button next to your new Client ID to download the client ID JSON file.** I typically put this file into my `.secrets` folder where I also keep my service account private key.

## Credentials compatible with this API

To view all credentials or create new credentials visit [Credentials in APIs & Services](#)

### OAuth 2.0 Client IDs

<input type="checkbox"/>	Name	Creation date ↓	Type	Client ID				
<input type="checkbox"/>	Test Google Analytics script	Nov 20, 2020	Desktop	118805052079-koeq...				

## Step 6: Submit both pieces of information to the Google Analytics API and make a test data request.

While this multi-step process which may have seemed like something out of *Lord of the Rings*, you now should have all the credentials and permission to make API requests to Google Analytics. So let's write code to fetch one day's Google Analytics data for the [rstudio.com](#) site.

```
library(googleAnalyticsR)
library(dplyr)
library(ggplot2)
library(lubridate)
library(reactable)
library(stringr)

## First, authenticate with our client OAUTH credentials from step 5 of
the blog post.
googleAuthR::gar_set_client(json = "secrets/oauth-account-key.json")

## Now, provide the service account email and private key
ga_auth(email = "ga-analysis@test-project-291617.iam.gserviceaccount.com",
        json_file = "secrets/service-account-key.json")

## At this point, we should be properly authenticated and ready to go.
We can test this
## by getting a list of all the accounts that this test project has
access to. Typically,
## this will be only one if you've created your own service key. If it
isn't your only
## account, select the appropriate viewId from your list of accounts.

my_accounts <- ga_account_list()
my_id <- my_accounts$viewId      ## Modify this if you have more than
one account

## Let's look at all the visitors to our site. This segment is one of
several provided
## by Google Analytics by default.

all_users <- segment_ga4("AllTraffic", segment_id = "gaid::-1")

## Let's look at just one day.
```

```
ga_start_date <- today()
ga_end_date <- today()

## Make the request to GA
data_fetch <- google_analytics(my_id,
                               segments = all_users,
                               date_range = c(ga_start_date,
ga_end_date),
                               metrics = c("pageviews"),
                               dimensions = c("landingPagePath"),
                               anti_sample = TRUE)

## Let's just create a table of the most viewed posts

most_viewed_posts <- data_fetch %>%
  mutate(Path = str_trunc(landingPagePath, width=40)) %>%
  count(Path, wt=pageviews, sort=TRUE)
head(most_viewed_posts, n=5)
```

Assuming you have the appropriate permissions, client ID, and service key, you should get a result that looks similar to this one I pulled...