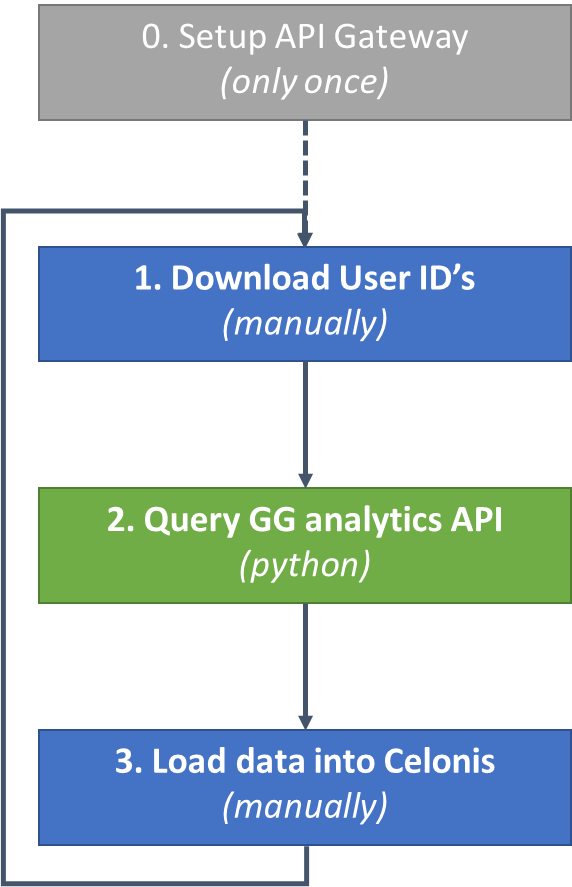
Building Event-log from GG Analytics

Author: Youp Suurmeijer

This document provides a brief overview on how to build an event-log for process mining/customer journey mining using the google analytics API. Getting the data from the API and processing it roughly follows the workflow depicted in picture 1. The whole workflow is based around a Python script that handles the querying and restructuring of the GG analytics data automatically.

# Step 0: Setup API Gateway

This step mainly involves making sure you have the right authorizations to query the GG analytics API. Similar to using the browser to access the GG analytics website the API also needs to know you are who you say you are and ensure you have access to the website’s data. The way this is done is for the GG analytics API is using the OAuth protocol. You’ll need the following things in order to be able to run the python script:

* Enable API
* Client secrets key value pair
* View ID

This document should contain all the information you need, but for more details and background info you can check the [GG analytics API reference guide](https://developers.google.com/analytics/devguides/reporting/core/v4/quickstart/installed-py).

**Enable API**

In order to use the API you’ll need a special kind of user account for GG analytics, called a project. This project will have its own credentials and will link to the source data. You can pretty much see it as a shared dropbox account, which your computer and other computers can use to interact with the same data.

To get started using Analytics Reporting API v4, you need to first [use the setup tool](https://console.developers.google.com/start/api?id=analyticsreporting.googleapis.com&credential=client_key), which guides you through creating a project in the Google API Console, enabling the API, and creating credentials. You will need to separately create credentials for your project using the following instructions:

Create credentials

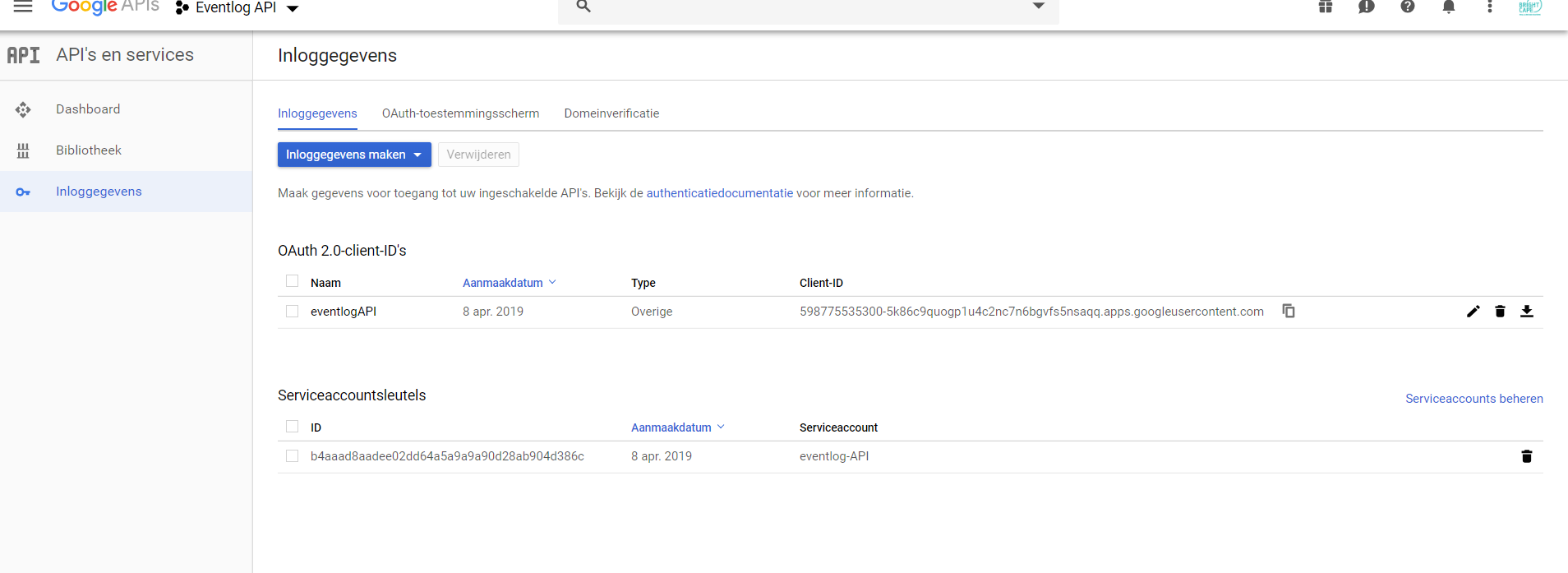
* Open the [Credentials page](https://console.developers.google.com/apis/credentials).
* Click **Create credentials** and select **OAuth client ID**
* For the **Application type** select **Other**.
* Name the client ID **quickstart** and click **Create**.

Note: To create a Web Client ID or an Installed Application Client, you need to set a product name in the consent screen. If you have not done so already you will be prompted to [*Configure consent screen*](https://console.cloud.google.com/apis/credentials/consent).

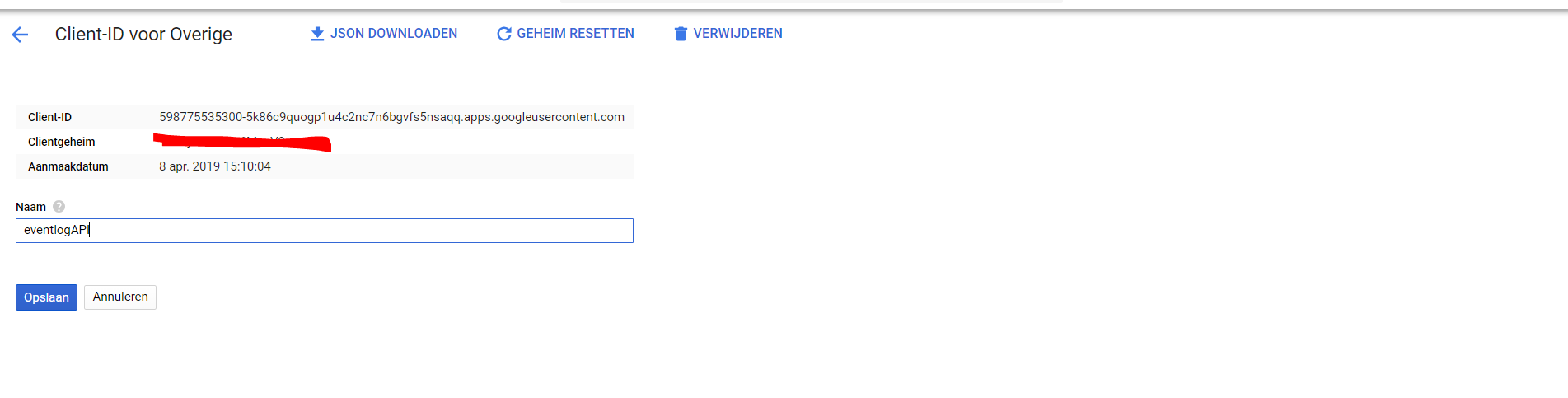
**Client secrets key value pair**

Once you have created a project and you have initiated your project credentials you need to download a JSON file containing the key value pair for the account. This JSON file is used by the Python script to log into the API.

->In your project screen go to login accounts



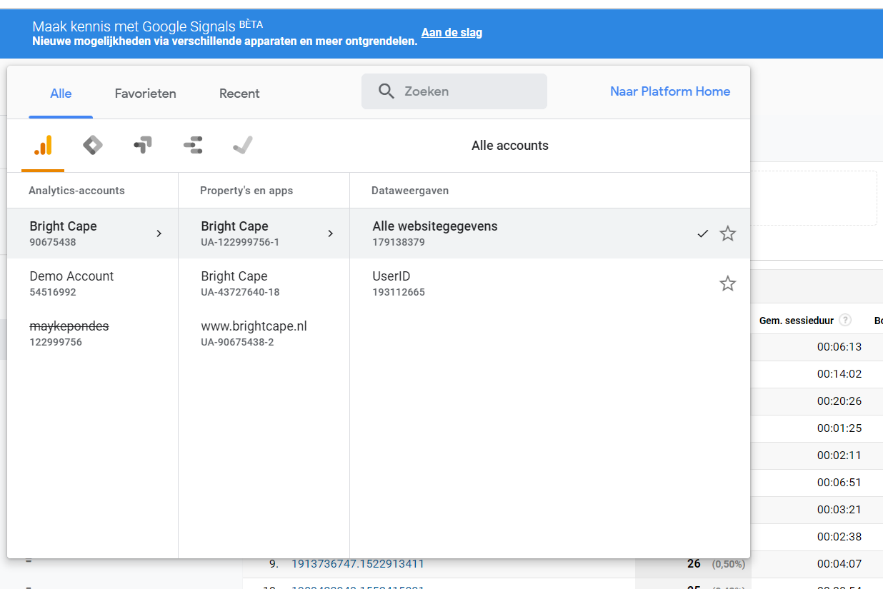
-> Select the OAuth 2.0 client ID, the following screen should appear:



-> Click download JSON file and save the file to your computer

**View ID**

Once you have set up a client to query the GG analytics API you also need to determine the right dataset on which to query. In GG analytics data is defined using the following setup:

**Account -> Properties -> Views**

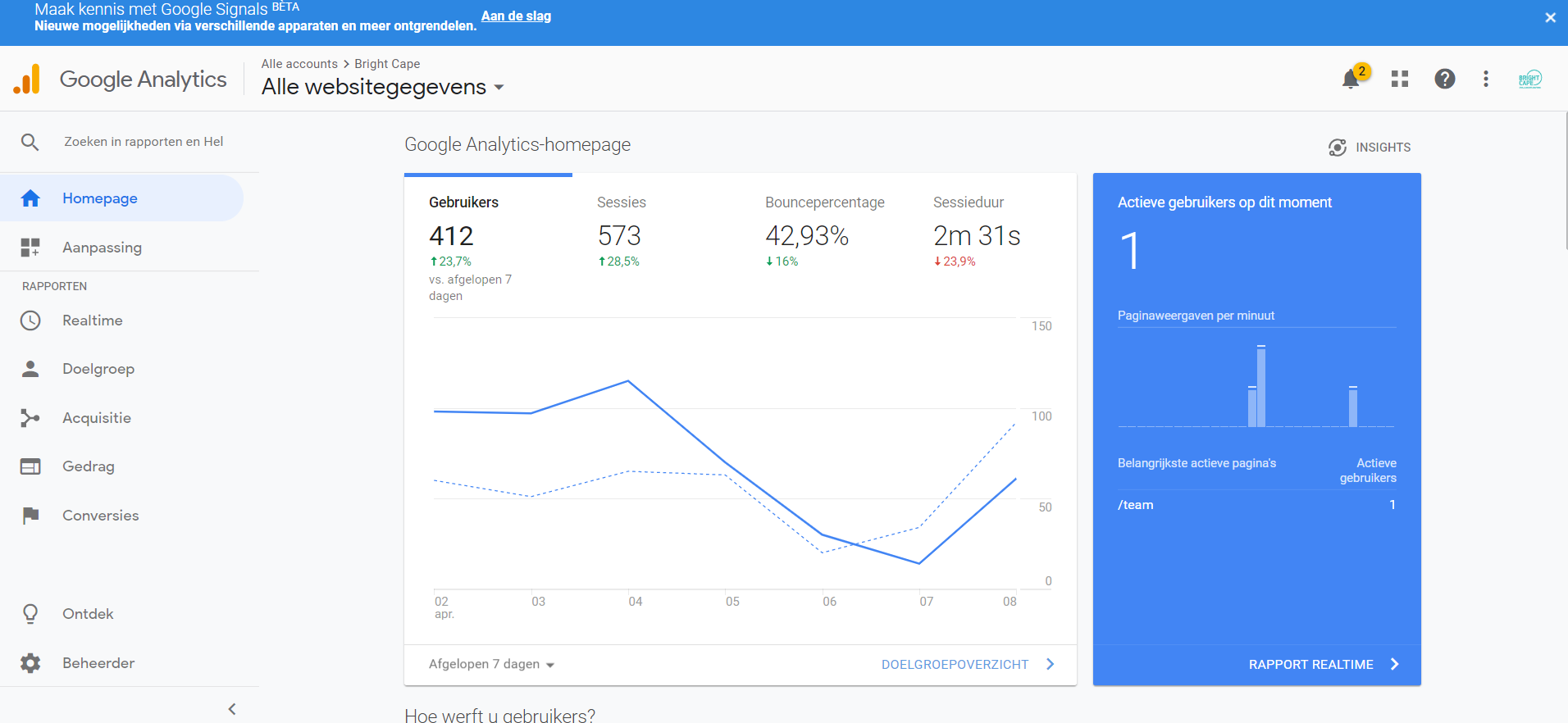
For example, the Bright Cape google account has several domains under its name, Bright Cape, Demo Account and MaykePondes. In the Bright Cape account, we have several properties, one being the production website (Bright Cape) others being test environments (I presume). Within these properties you can have different views (similar to database setup). The whole purpose of this structure is to be able to provide different sets of access rights for different accounts/users/websites. What we need to do is tell the API which view to query, in this case the “ga:119835346” view. You can use a handy tool from google to look up the different views in more complicated accounts quite easily (<https://ga-dev-tools.appspot.com/account-explorer/>).

# Step 1: Download Client ID’s

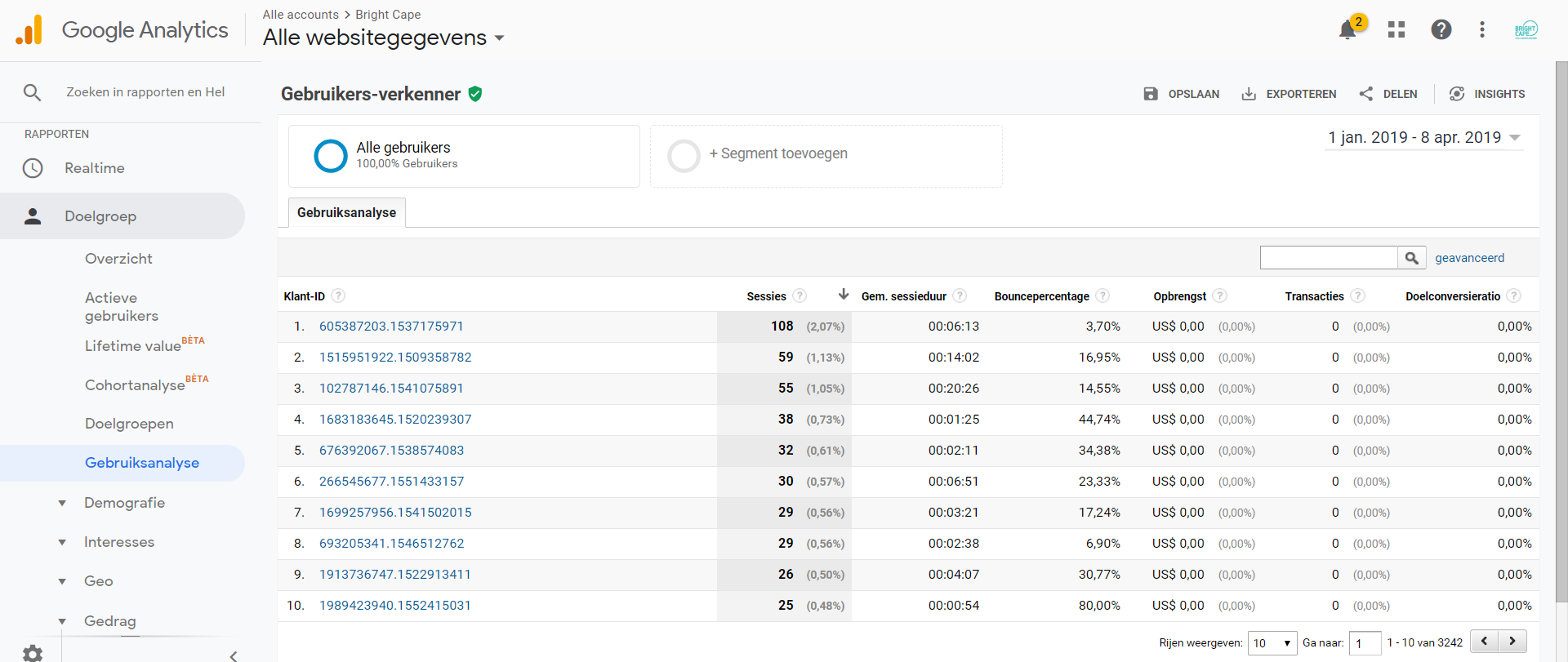
For some odd reason the GG Analytics API does not allow you to directly query the client ID’s using the API, so for now the only solution is to download the ID’s manually and subsequently use the API to retrieve all user data. The client ID is a randomly generated ID that GG analytics uses to distinguish between devices and browsers over different sessions, or basically a pseudo identifier for different users of the website (for more info check the [client ID reference guide](https://support.google.com/analytics/answer/6205850)).

To download all the client ID’s use the following steps:

-> Go to the correct GG Analytics view

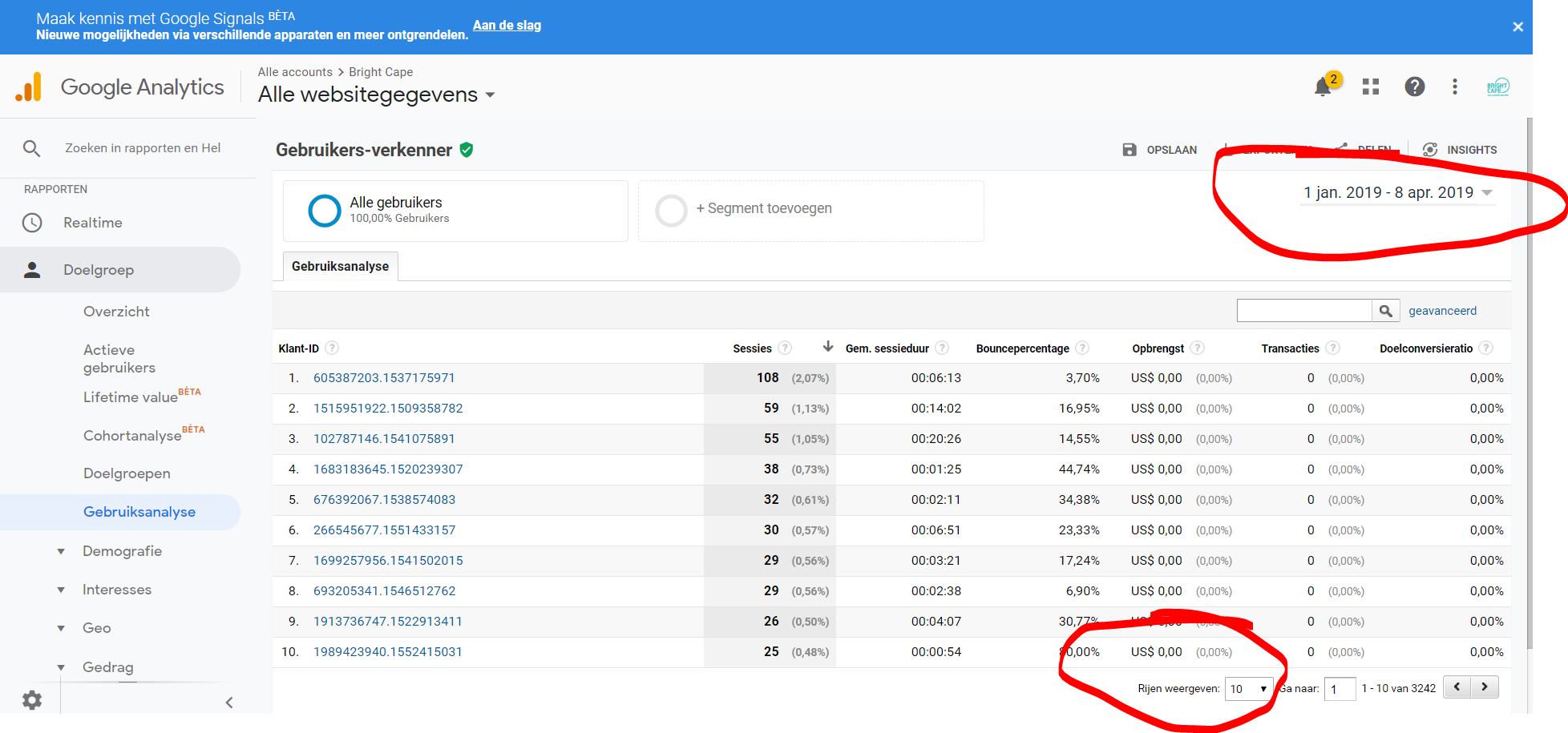


-> Under the ‘Target Audience’ tab select the ‘User Analysis’ report

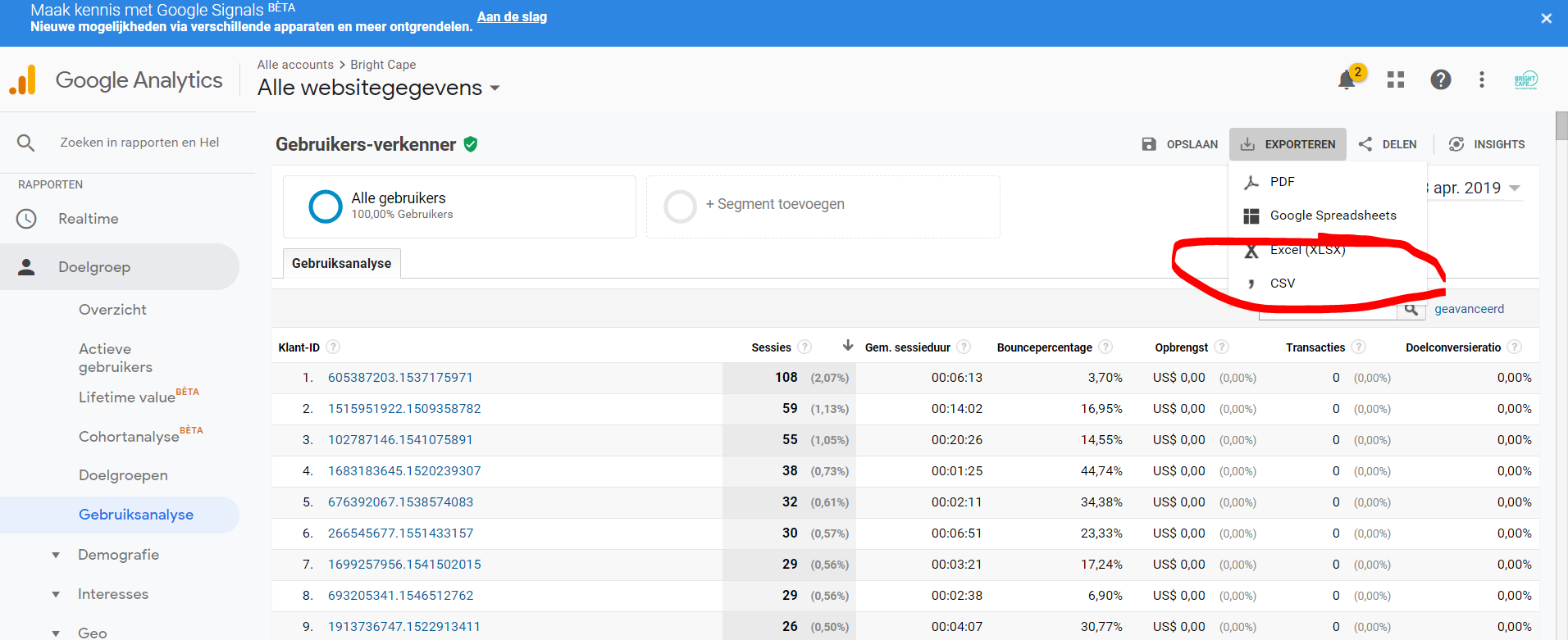


(If the user analysis report is not present please consult the notes at the end of this document)

-> Select a date range for which to download the data, and select rows to display at maximum (5000)



-> Download the results as a csv and save them to your computer



(Because we have to use the UI to download data, we can only download 5000 users at a time, which can be a problem for large website. I already have a solution in mind to automate this process, but it will require building some web scraping tooling)

# Step 2: Query GG Analytics API

This step uses a Python script to automatically query all the user data obtained in step 1. When running the python script for the first time some setup will be required. In the future this Python script can easily be converted to a standalone .exe and even be given a proper UI, for now we will suffice with just executing the script.

**Running the script the first time**

When running the Python script for the first time on your computer please ensure the following steps are taken:

1. Install python
2. Open the script and check package dependencies
3. Install folder
4. Set variables
5. Test run script

*1. Install Python*

I used Anaconda to buid/run the code <https://www.anaconda.com/distribution/>

*2. Open the script and check package dependencies*

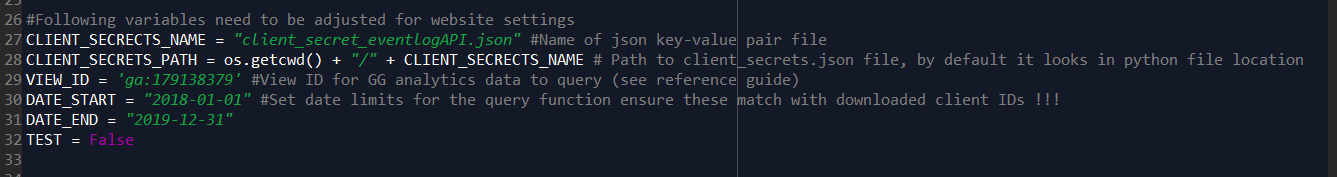
In the future this code should be properly packaged and documented, for now please refer to the code to see which packages are required to run the script

*3. Install folder*

Create a new folder and put the Python script in the folder, together with the JSON key value pair obtained in step 0 (or if you wish to keep the key value pair in a different directory adjust the python script accordingly).

*4. Set variables*

At the top of the Python script the environmental variables for the script are set:



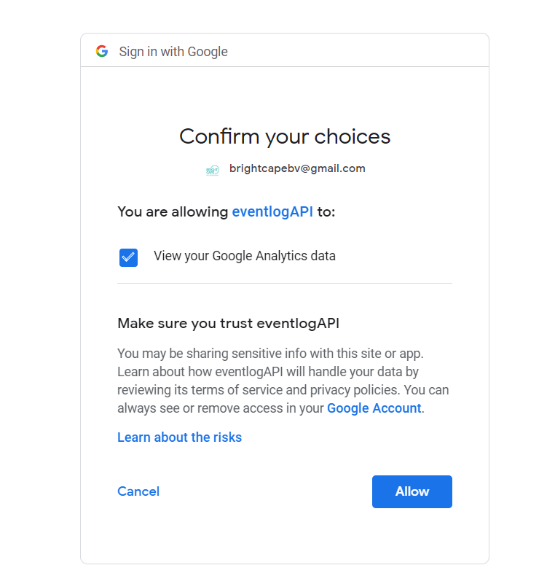
These include reference to the location of the key-value pair, the view ID, and the date ranges for the GG analytics query. Please adjust these accordingly when installing the script, for instructions on how to obtain the required info please refer to ‘Step 0: Setup API Gateway’

*5. Test run the script*

It is always a good idea to do a test run before trying to pull large amounts of data via the API. To perform a test run set the variable TEST = True at the top of the Python script, this will limit the amount of API calls for the user ID’s to 3.

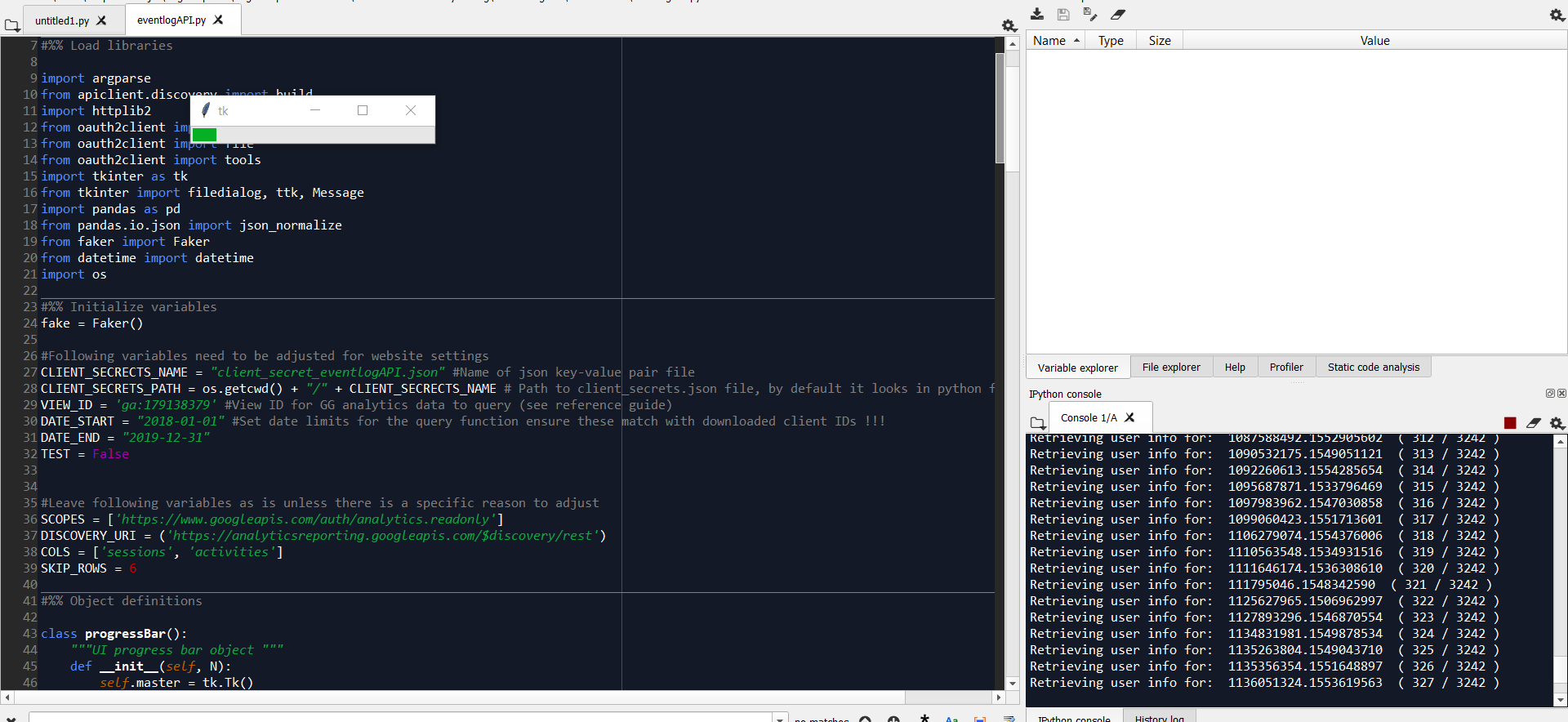
**Running the Python script**

In order to run the API script use F5 or go through the command line to execute the script. The script will prompt you to select the .csv file with the client ID’s that you downloaded in Step 1. Select the file and select OK.

When running the script for the first time you will be guided through the OAuth login via your webbrowser. Use the website’s google account credentials to log in.

If this is not the first-time logging in a file named “analyticsreporting.dat” should appear in the folder. The next time running the script GG Analytics will no longer prompt you for authentication.

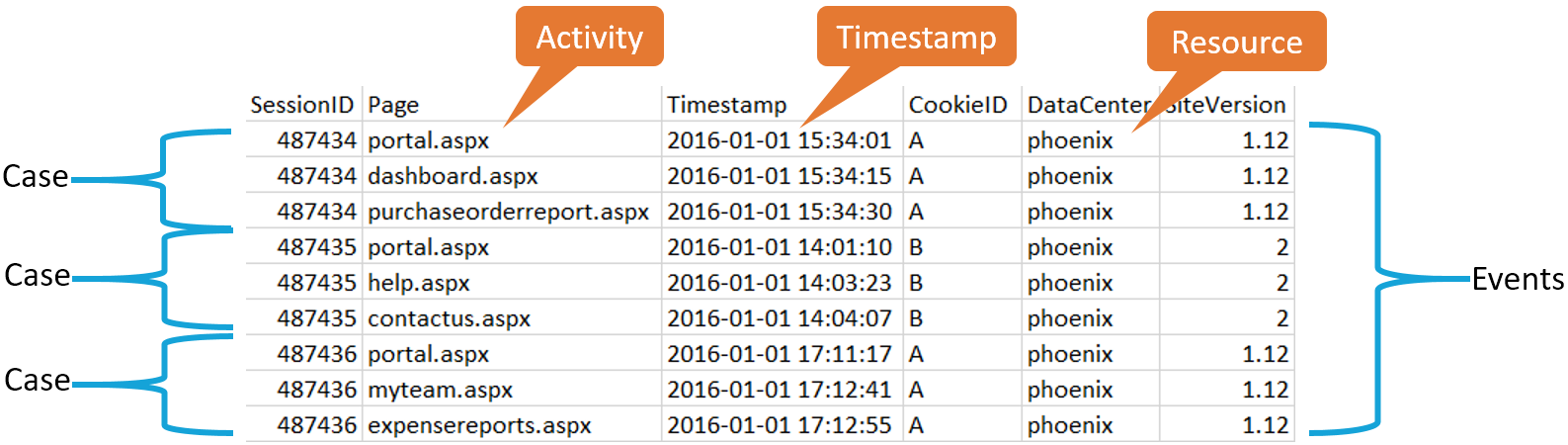
After authentication is complete the script will start querying the API. When using the command line to run the script a progress bar should appear. When using an IDE like Spyder a progress bar will appear as well as some extra info on progress in the console.



Finally the script will

# Step 3: Load the data in Celonis

The obtained data can be imported into Celonis as a .csv file. Note that an eventlog is built up from the following components: Case ID, Activity, Timestamp, Resource. When defining the eventlog in Celonis the Session ID should be used as the case, and the User ID as the resource-identifier respectively.



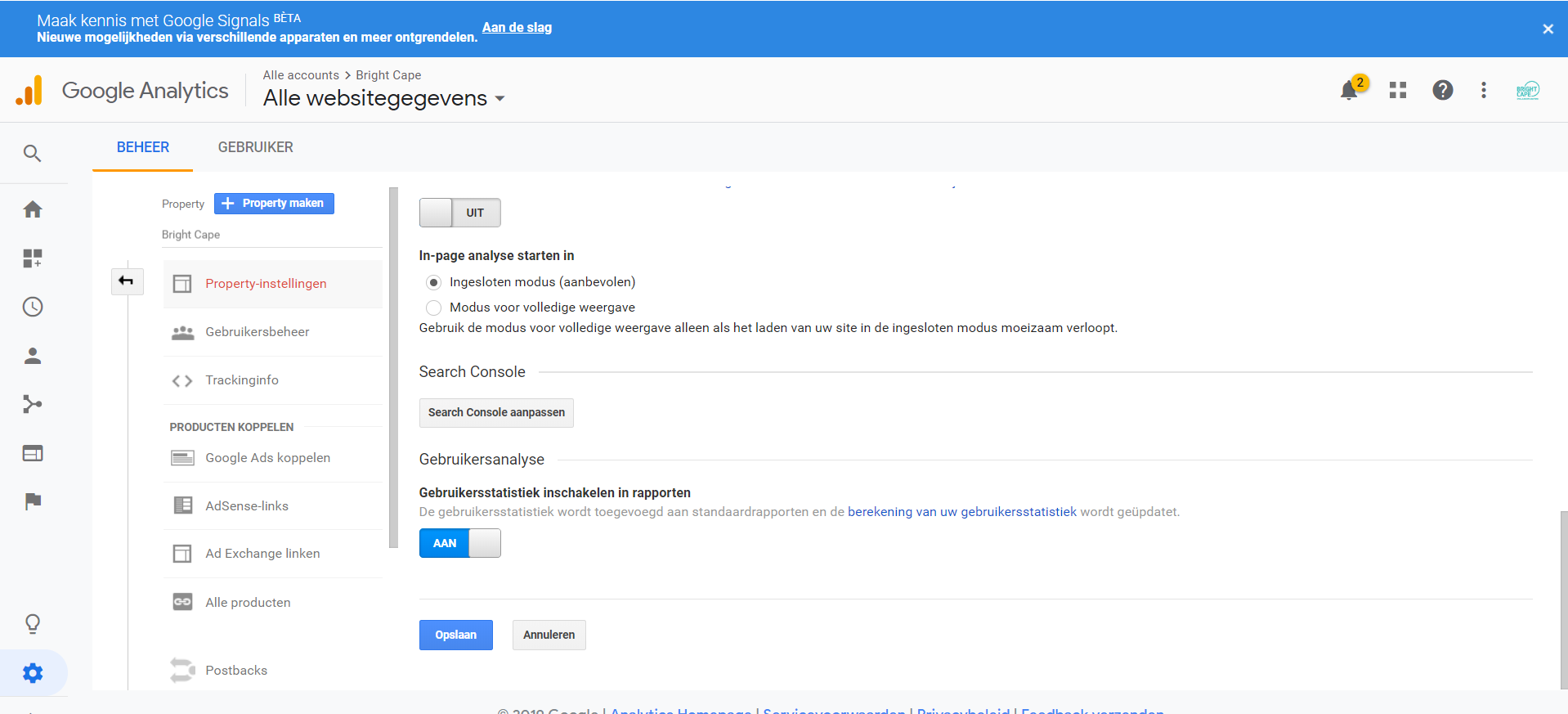
On an extra note, the output data contains 2 kinds of user ID’s, the default GG Analytics ID (client ID) and a person name. This name is fictional and consists of a randomly generated name for each unique client ID, this has no further significance other than that it might be easier to use in Celonis for demonstration purposes.

# Final Notes

**No User Analysis Report in GG Analytics**

If in step 1 you’re not able to find the user analysis report under the target audience tab, please activitate User ID tracking <https://www.youtube.com/watch?v=oXPODdoZvZ0>. I went through only step 1 and was not sure whether this unlocked the Client ID report or whether it was already there. If you have any additional questions please contact me.

Also you can adjust the settings for the user statistics reporting in the property settings of your analytics domain.



**Query limitations GG analytics API**

There are limits on the amount of API requests you can make per day and also per timespan. For the full overview please refer to: <https://developers.google.com/analytics/devguides/reporting/core/v4/limits-quotas#analytics_reporting_api_v4>

In the default API settings you can only make a 100 calls per 100 seconds to the analytics API using your credentials. You can adjust this setting to a maximum of 1000 per 100 seconds in the [Google API console](https://console.developers.google.com/apis/api/analytics-json.googleapis.com/quotas). Also the amount of daily API calls is limited to 50,000 so if you need more you’ll have to dive into the google documentation and probably start paying for the API.