Published in Towards Data Science



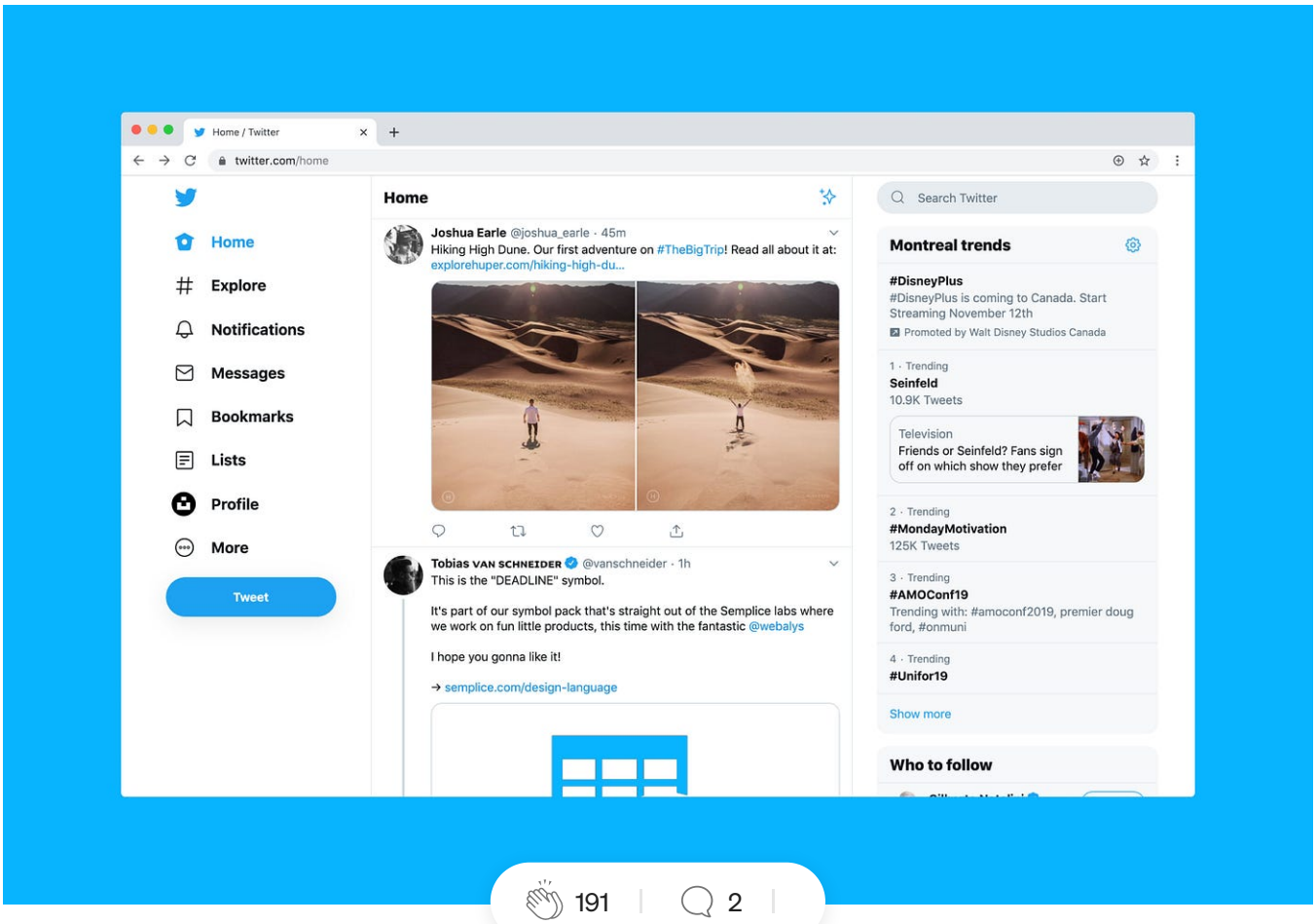
Ahmed Besbes Follow

May 17, 2022 · 8 min read · ✨ · Listen

Save

# How To Extract Data From The Twitter API Using Python

## An overview of the latest release of Tweepy




191 2

Photo by Luke Chesser on Unsplash

Accessing external data from social networks like Twitter is valuable in many situations.

Whether you're a big company or an individual, this helps conduct and enrich analyses on particular topics, monitor the competition, evaluate the social reputation of a given product, analyze customer reviews, follow reactions to recent events, etc.

Extracting data from Twitter and other social networks was a nightmare a few years ago: there were no APIs, no clear documentation, no libraries, and you had to manually build complex web crawlers to collect data from the underlying HTML code.

Needless to say that this was painful and clunky 😰.

Hopefully, things have hanged a lot since.

> In this post, I'll show you how you can extract useful and actionable data from Twitter. To do this, we'll be using a Python package called **Tweepy** that'll interact with the Twitter API to fetch different types of metadata and automate tasks.

Hope you're all set. Let's have a look 🔍

## Why do we need the Twitter API for?

Imagine the following: you're a web developer and want to extract the tweets of a given account in order to embed them into your app. You'd also like to fetch and update these tweets on a daily basis to get the recent ones.

> *How would you do that?*

In order to consume Twitter data, you don't need to have access to Twitter's internal servers and databases. You don't need to understand how Twitter is coded either.

All you need is an intermediary between you and the Twitter backend, an intermediary that you can request (with some parameters) to get some data or interact with Twitter services.

This is what an API is in a nutshell: it's simply a junction between two independent applications that communicate with each other.

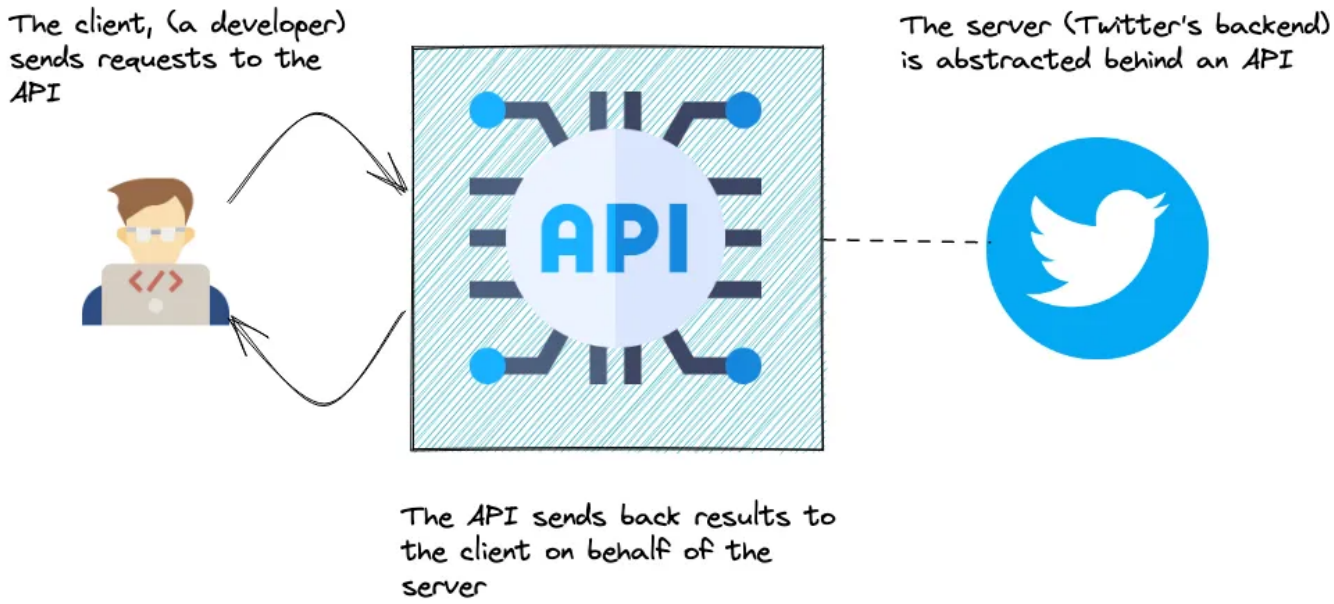



Image by the author

As an **abstraction,** the Twitter API **simplifies the integration with third parties** (developers, apps, other APIs, etc.).

Here's what you could do with the Twitter API:

- Programmatically search tweets based on hashtags, keywords, geolocation, etc.
- Build Twitter bots that automatically retweet a list of predefined accounts
- Stream tweets in real-time based on a series of filters
- Automatically follow a list of users
- etc.

The API lets you programmatically execute any action you'd manually perform from the interface.

Looks like fun: let's see now how Python interacts with the Twitter API.

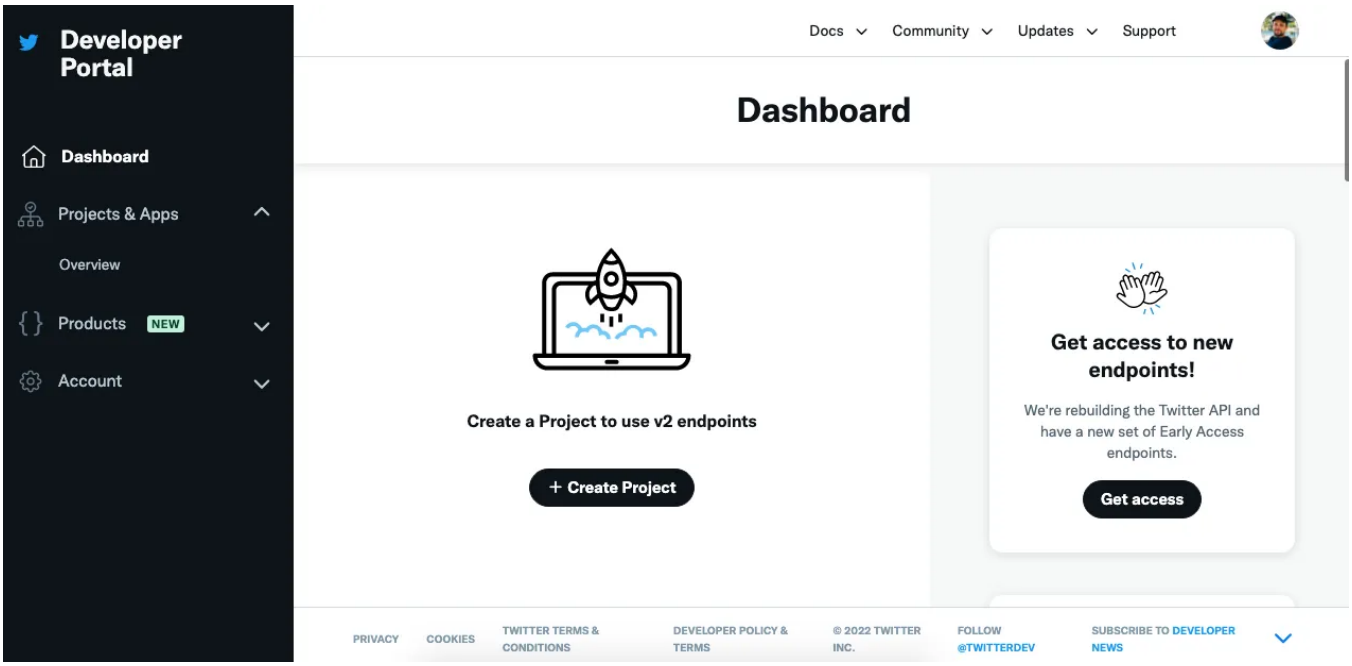
## Create a Twitter application and setup credentials

*To be able to reproduce the following steps, you need a Twitter account.*

In order to use the Twitter API, you first have to register as a Twitter developer on the developers' website.

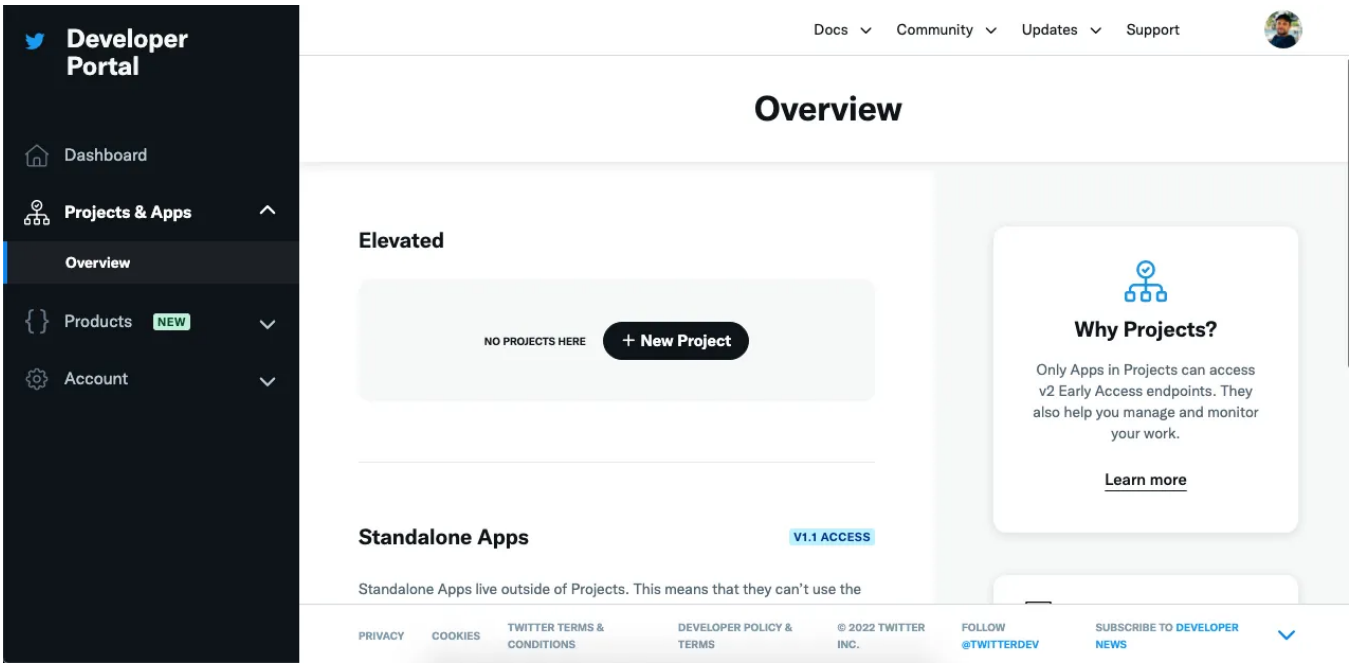
Once registered, you need to create a Twitter application that'll set up a bunch of credentials: these credentials will be later used by the Tweepy library in order to authenticate you.

First, go to the developer's dashboard.



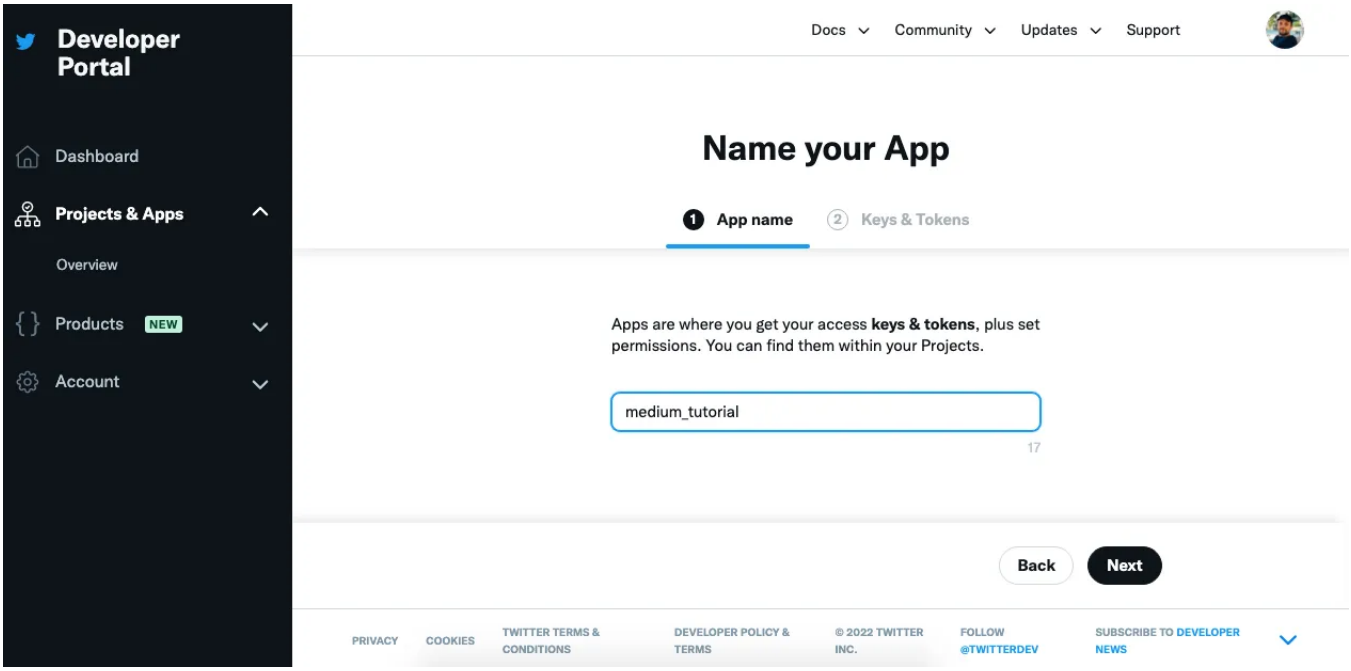

Screenshot by the author

Hit **Overview** from the left sidebar and click on the **Create App** button.



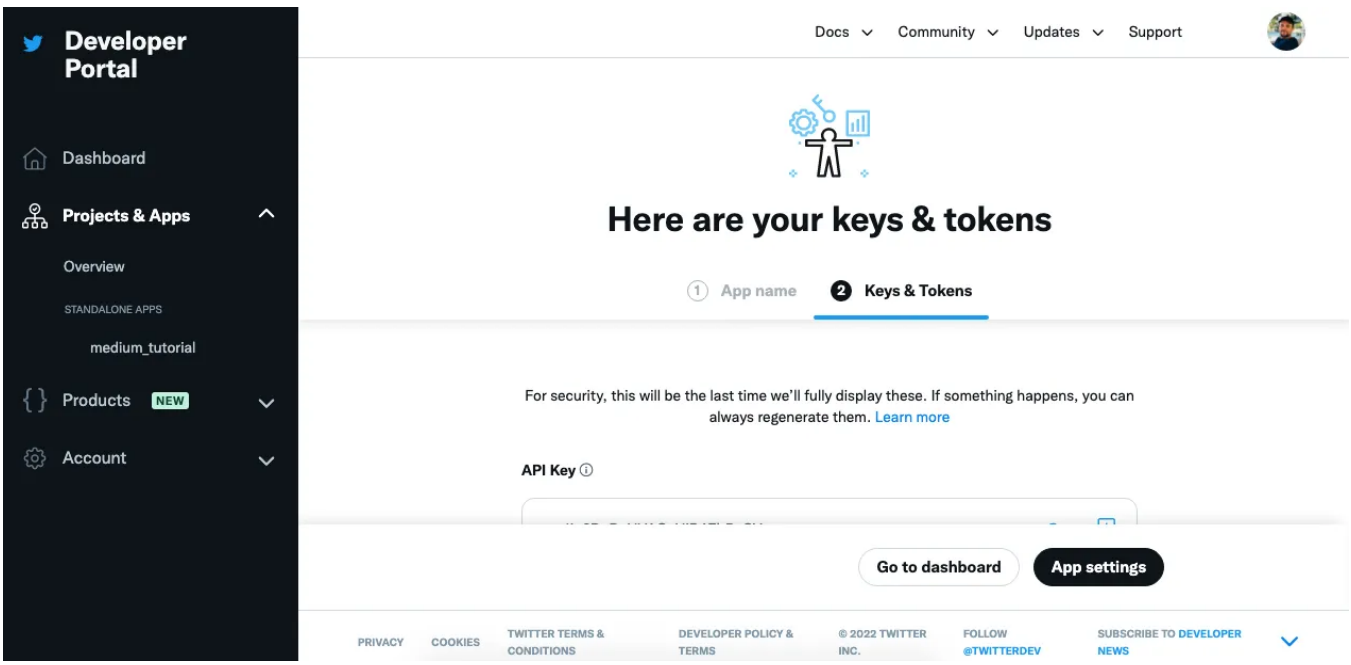

Screenshot by the author

Give your app a name.




Screenshot by the author

This will generate the following credentials. They're personal: don't share them with anybody.

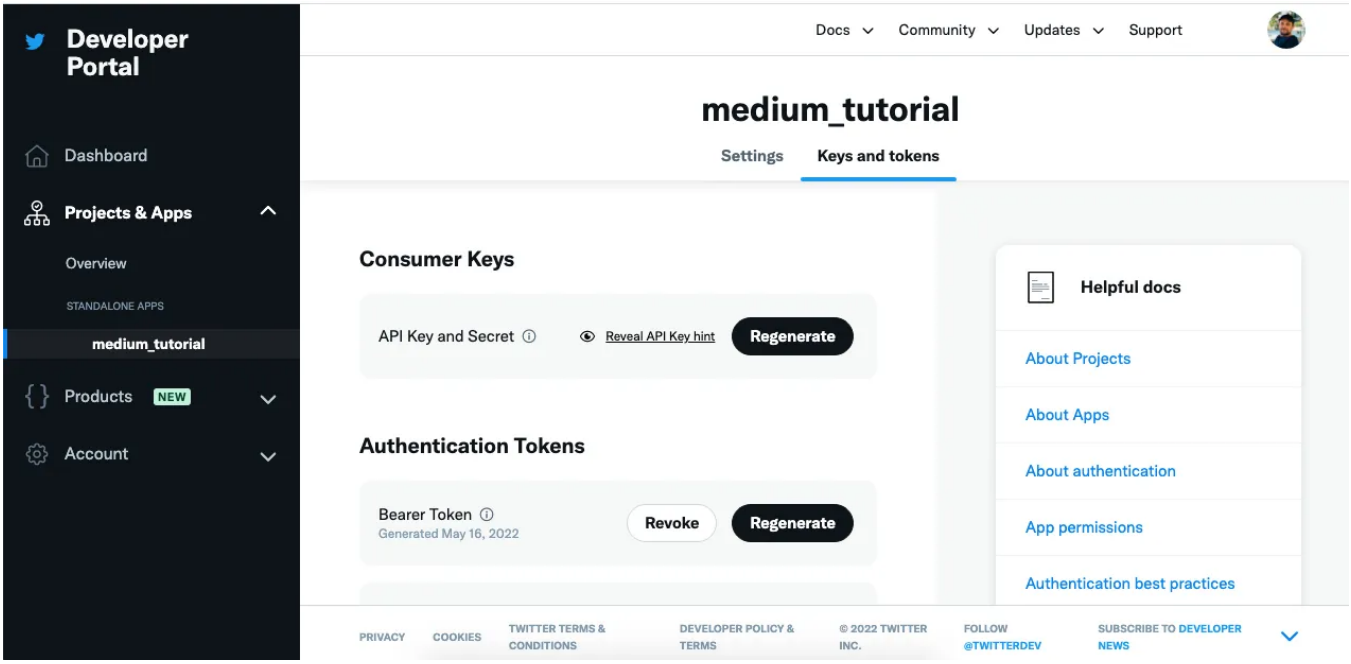



Screenshot by the author

If you didn't copy-paste the credentials the first time they were presented to you, you need to regenerate them. To do this, access your app from the left sidebar and hit the **Keys and tokens** sidebar.

Click on each **Regenerate** button and save the following credentials somewhere:

- **API_KEY**
- **API_KEY_SECRET**
- **ACCESS_TOKEN**
- **ACCESS_TOKEN_SECRET**
- **BEARER_TOKEN**




Screenshot by the author

You're now ready to use API.

## Connect to the Twitter API

To be able to interact with the Twitter API using Python, we'll use a library called Tweepy.

To be able to hide the credentials from the source code and load them as environment variables, we'll use python-dotenv.

```
pip install tweepy
pip install python-dotenv
```

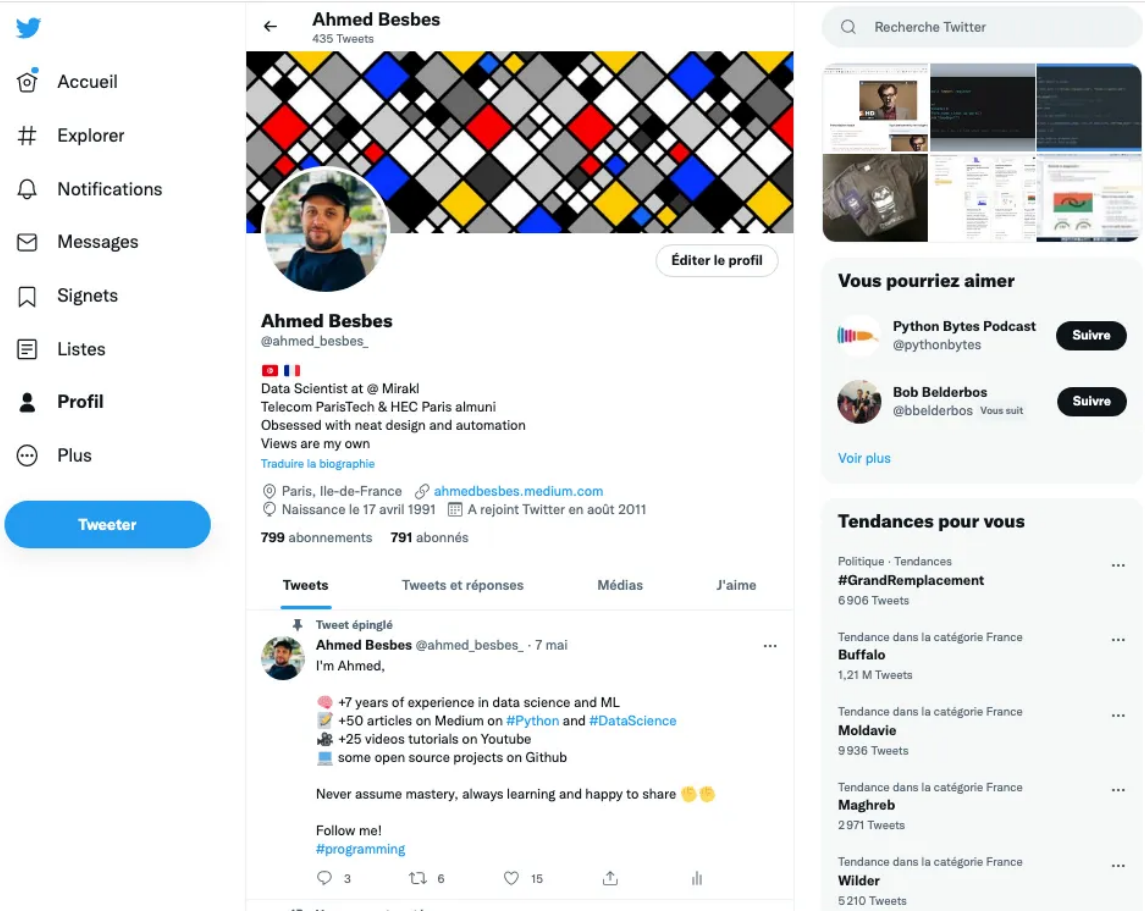
First, create a `.env` file to hold your credentials.


Screenshot by the author

Open a jupyter notebook or a python interactive shell in the same directory containing the `.env` file.

We can authenticate to the Twitter API using the following script.

Let’s check that the Twitter user who initiated this connexion is me.




Screenshot by the author

To do this, we call the `verifiy_credentials` method that returns a User object containing my profile data.

```
In [4]: me = api.verify_credentials()
In [4]: me = api.verify_credentials()

In [5]: me
Out[5]: User(_api=<tweepy.api.API object at 0x10a260a60>, _json={'id': 360328126, 'id_str': '360328126', 'name': 'Ahmed
Besbes', 'screen_name': 'ahmed_besbes_', 'location': 'Paris, Ile-de-France', 'description': '🇹🇳 🇫🇷\nData Scientist at @
irakl\nTelecom ParisTech & HEC Paris almuni \nObsessed with neat design and automation\nViews are my own', 'url': 'http
://t.co/eBYBr1vDUj', 'entities': {'url': {'urls': [{'url': 'https://t.co/eBYBr1vDUj', 'expanded_url': 'https://ahmedbes
es.medium.com/', 'display_url': 'ahmedbesbes.medium.com', 'indices': [0, 23]}]}, 'description': {'urls': []}}, 'protect
d': False, 'followers_count': 791, 'friends_count': 799, 'listed_count': 33, 'created_at': 'Tue Aug 23 01:34:19 +0000 2
11', 'favourites_count': 475, 'utc_offset': None, 'time_zone': None, 'geo_enabled': False, 'verified': False, 'statuses
count': 435, 'lang': None, 'status': {'created_at': 'Fri May 13 12:27:50 +0000 2022', 'id': 1525090442404061185, 'id_st
': '1525090442404061185', 'text': 'RT @TDataScience: How To Ignore Jupyter Notebook From Github Language Stats? by @ahm
d_besbes_  https://t.co/84IwFfw8vD', 'truncated': False, 'entities': {'hashtags': [], 'symbols': [], 'user_mentions': [
'screen_name': 'TDataScience', 'name': 'Towards Data Science', 'id': 788898706586275840, 'id_str': '788898706586275840'
 'indices': [3, 16]}, {'screen_name': 'ahmed_besbes_', 'name': 'Ahmed Besbes', 'id': 360328126, 'id_str': '360328126',
indices': [80, 94]}], 'urls': [{'url': 'https://t.co/84IwFfw8vD', 'expanded_url': 'https://buff.ly/3PcEjKU', 'display_u
l': 'buff.ly/3PcEjKU', 'indices': [96, 119]}]}, 'source': '<a href="http://twitter.com/download/iphone" rel="nofollow">
witter for iPhone</a>', 'in_reply_to_status_id': None, 'in_reply_to_status_id_str': None, 'in_reply_to_user_id': None,
in_reply_to_user_id_str': None, 'in_reply_to_screen_name': None, 'geo': None, 'coordinates': None, 'place': None, 'cont
ibutors': None, 'retweeted_status': {'created_at': 'Fri May 13 12:19:01 +0000 2022', 'id': 1525088221859827712, 'id_str
: '1525088221859827712', 'text': 'How To Ignore Jupyter Notebook From Github Language Stats? by @ahmed_besbes_  https:/
t.co/84IwFfw8vD', 'truncated': False, 'entities': {'hashtags': [], 'symbols': [], 'user_mentions': [{'screen_name': 'ah
```

Screenshot by the author

This object is a blob that contains a lot of metadata. We can access some of it as object attributes and print them on the screen.

Screenshot by the author

This user is definitely me. You can clearly see some of this data displayed on my profile.

## Search and extract tweets

The first application of the Twitter API is the ability to programmatically search tweets based on multiple types of filters.

> As an example, let's search some tweets that mention Ukraine 🇺🇦.

After setting up the authentication, we'll use the `search_tweets` method, loop over the extracted tweets and print their text.

Here's what happens after running the code snippet.

```
In [5]: tweets = api.search_tweets("ukraine", tweet_mode="extended")

In [6]: for tweet in tweets:
   ...:     try:
   ...:         print(tweet.retweeted_status.full_text)
   ...:         print("=====")
   ...:     except AttributeError:
   ...:         print(tweet.full_text)
   ...:         print("=====")
   ...:
@narrative_hole Remember some 6-7 years ago when Ukraine bombed one of their own cities in the Donbass  using a warplan
e ?  A video clip was all over the Internet of a woman laying in the street alive and talking, with one leg blown off.
 That's when we saw the true nature of that government.
=====
Russia's unprovoked invasion of Ukraine has involved some of the worst barbarism in Europe since Nazi Germany. Yet Euro
pe's streets are almost bereft of massive anti-Kremlin demos. Students are almost entirely silent. UK's NUS has nothing
 to say – except about Israel.
=====
Ukraine has defeated Russia in the Battle for Kharkiv and Ukrainian troops have now advanced to the Russian border. Sla
va Ukraini! https://t.co/SktMt74YXK
=====
Two Heroes died. The whole family was gone. Their son was left an orphan. And I couldn't do anything... I, the entire c
ommand, all of us are fighting around the clock to save our men. You must understand that whole families are erased her
e."

#Ukraine #Russia #WarCrimes
=====
Is Rand Paul still holding up US aid to Ukraine? Ukrainians are dying every hour while this buffoon grandstands, alread
y with a long history of defending Putin and courting Russian investment.
```

Screenshot by the author

👉 We didn't specify any filters in this example, but you can customize the search by setting up some parameters such as the number of tweets you want ( `count` ), the language ( `lang` ), the location ( `geocode` ), the stop date ( `until` ), etc.
You can learn about the full parameters here.

👉 You can also have a more advanced search query that directly embeds some of the filters in it.
For example, if you want to extract English Tweets that are not retweets and that talk about covid-19, you'd write the following query: `covid-19 lang:en -is:retweet`

Screenshot by the author

To learn more about Twitter search operators, please refer to this link.

## Search a larger number of tweets using pagination

In the previous example, we only extracted 15 tweets or so: that's because Twitter has a pagination mechanism and we only got the results of the first page.

In order to loop over multiple result pages/tweets, we need to use the Cursor class that wraps the `search_tweets` method.

Each instance of the Cursor class supports two methods, depending on what you want to iterate over:

- `items` : Maximum number of items to iterate over
- `pages` : Maximum number of pages to iterate over

Let's extract 250 tweets using the `items` method.

When using the `items` method, I suspect that the `count` parameter has no effect.

Screenshot by the author

Similarly, we can use the `pages` method to get the same number of tweets: we tell the Cursor object to iterate over 5 pages and extract 50 tweets for each one.

⚠️ if you aggressively hit the search API, you'll probably encounter a 429 error code that indicates that you reached a rate limit on a specific endpoint.

👉 For example, the rate limit on the `/search/tweets/` endpoint is 180 every 15 minutes. This means that, given this limitation, you can get at most 180 * `count` tweets every 15 minutes ( `count` is the parameter you specify in the `search_tweets` method)

You can learn more about Twitter rate limits here.

## Search tweets of a specific account

Let's say you want to analyze Elon Musk's tweets: that's pretty simple.

All you need to do is prepend your query with the `from` keyword.
If you want collect tweets about Tesla, here's what you'd write:

```
tesla from:elonmusk lang:en -is:retweet
```

Screenshot by the author

## Stream real-time tweets

One interesting functionality of Twitter API is the ability to stream a sample of real-time tweets. This is particularly helpful to follow live events and collect data about them.

In order to be able to use the streaming functionality of the API, you first need to grant **write access to your Twitter application** and **move it to a project.**

Then, you'll need to regenerate the credentials.

To be able to fetch streaming tweets using Tweepy, you'll have to define a custom class that subclasses the `StreamingClient` class.

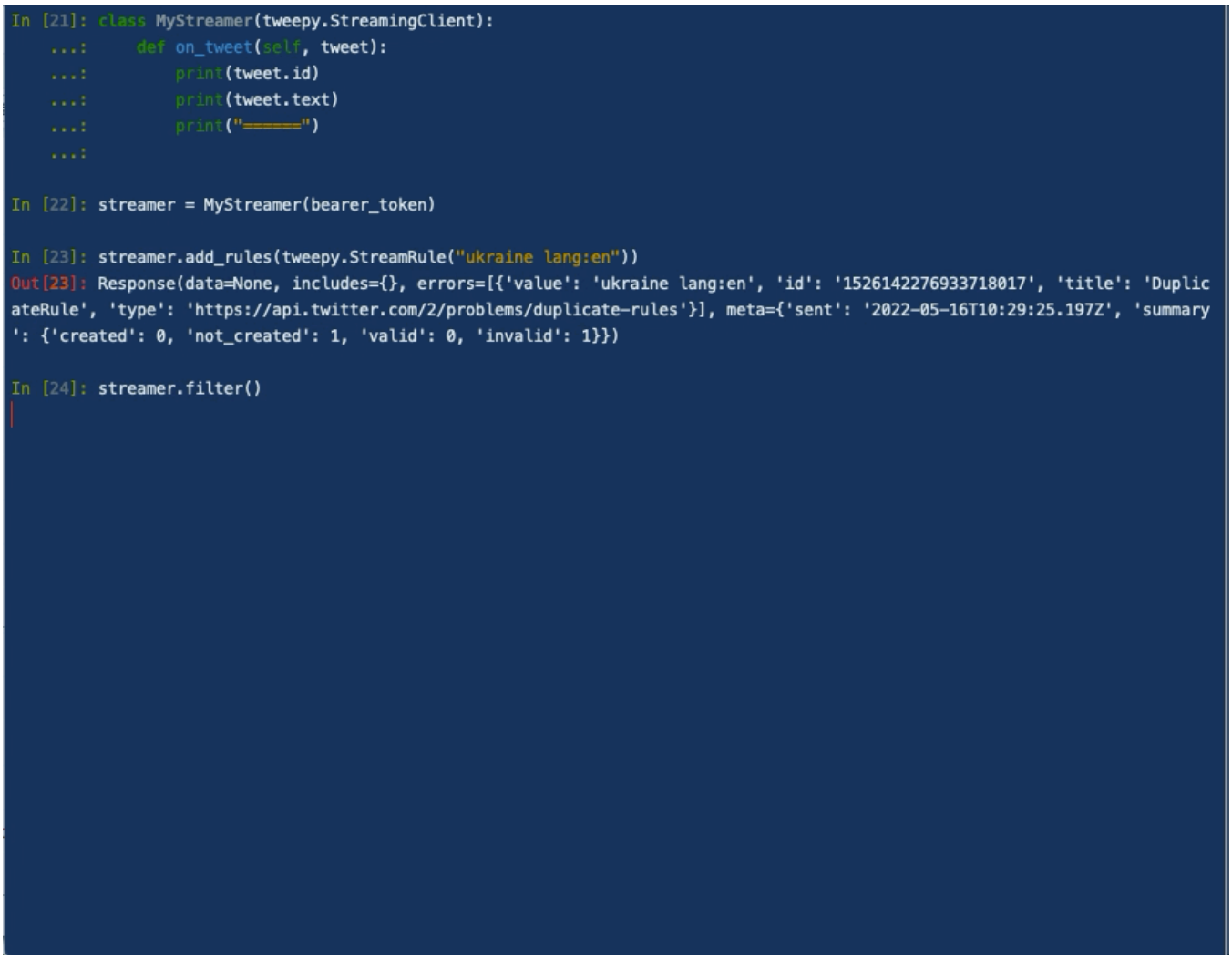
This class needs to override a method called `on_tweet` that gets executed every time a tweet is received: to make things simple we'll tell this method to print the tweet's id and text.

To instantiate a custom streamer out of this class, we need to pass the bearer token to the constructor.

Then, we'll need to attach some rules that define what we want to search (note that tweepy allows you to have multiple rules in a list but we'll only set one in this example).

Finally, calling the `filter` method opens up the stream and runs the `on_tweet` method on every incoming tweet.

The following GIF illustrates what happens once the stream is open.




GIF by the author

To learn more about streaming with Tweepy, have a look at this documentation.

## References

- https://developer.twitter.com/
- https://medium.com/towards-data-science/how-to-access-data-from-the-twitter-api-using-tweepy-python-e2d9e4d54978
- https://docs.tweepy.org/en/latest/getting_started.html

## Conclusion

This post was a small introduction to tweepy and the Twitter API. It gave us the opportunity to extract tweets and metadata in various ways from offline to streaming modes.

If you're a data scientist, adding tweepy to your skills will definitely give you a competitive edge and help you enrich your analyses with external data.

But we've just scratched the surface here: there's obviously more to learn. If you want to dig deeper I encourage you to have a look at the API documentation and as always experiment, experiment, experiment. (3 times is not enough said).

That'll be all for me today. Until next time for more programming tips and tutorials. 👋


Photo by charlesdeluvio on Unsplash

Python Twitter Programming Data Science API