The Twitter API and Authentication

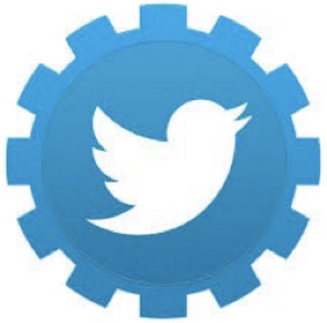
INTERMEDIATE IMPORTING DATA IN PYTHON

Hugo Bowne-Anderson
Data Scientist at DataCamp

datacamp

Herein, you'll learn

- How to stream data from the Twitter API
- How to filter incoming tweets for keywords
- About API Authentication and OAuth

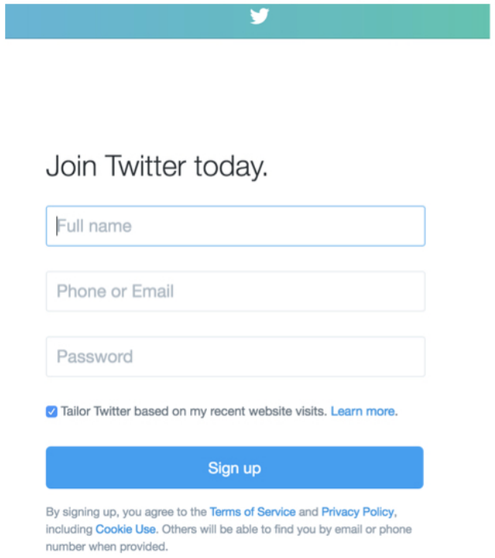


Herein, you'll learn

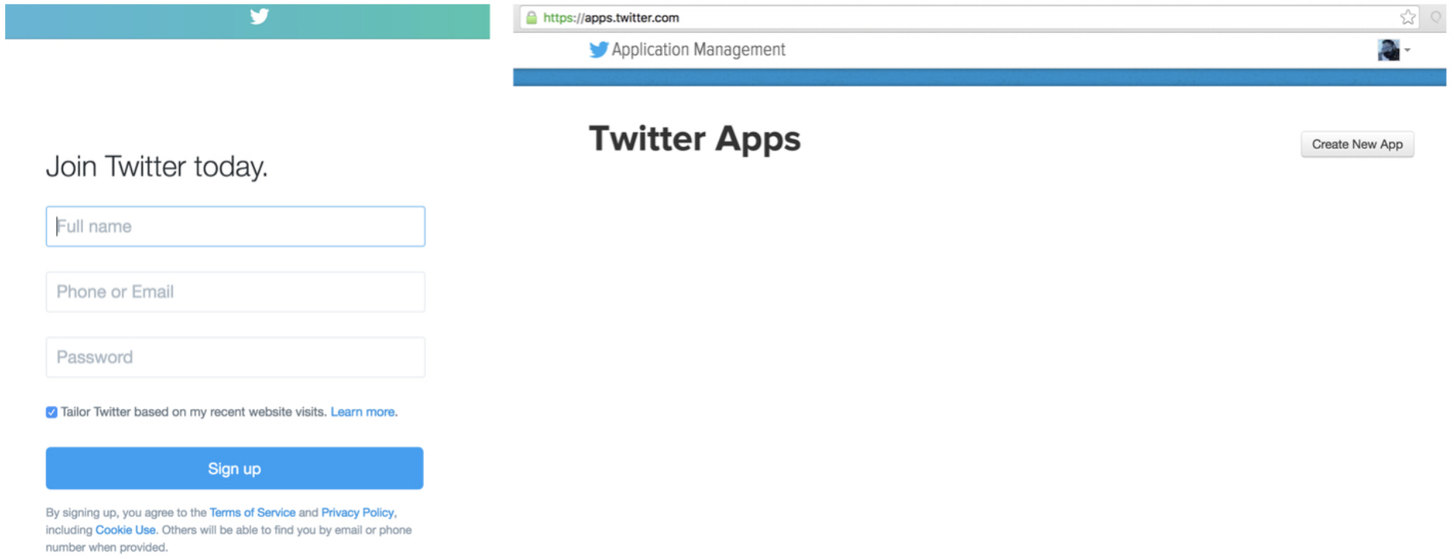
- How to stream data from the Twitter API
- How to filter incoming tweets for keywords
- About API Authentication and OAuth
- How to use the Tweepy Python package

2. Herein, you'll learn
As a final deep dive, you're going to stream data from the Twitter API. You'll learn how to filter incoming tweets for keywords, you'll learn about the principles of API authentication and OAuth. You'll also learn the basics of the package

Access the Twitter API

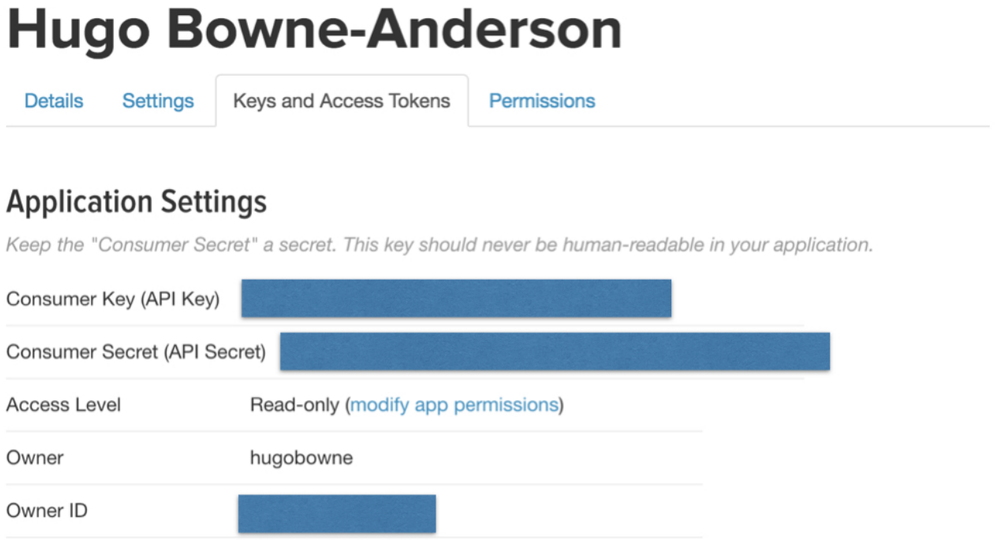


Access the Twitter API

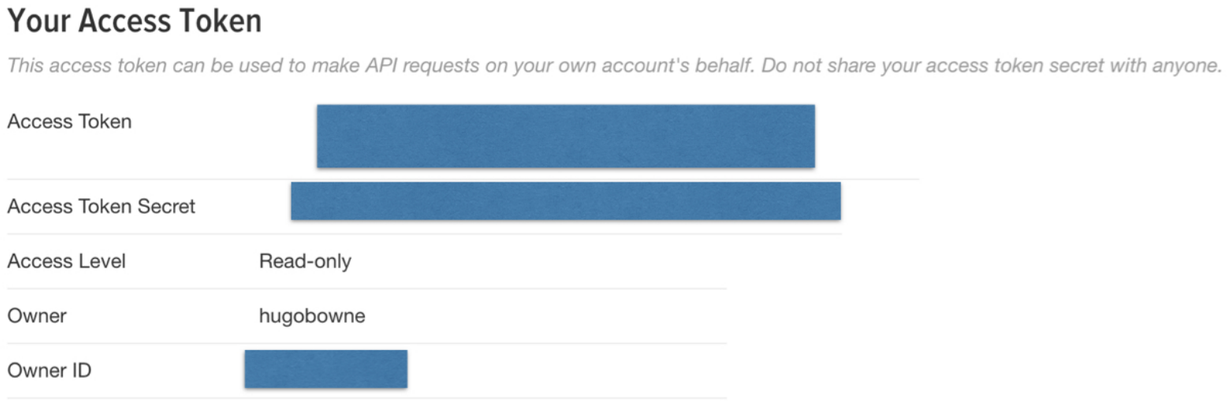


4. Access the Twitter API
One of the first major differences between the Twitter API and all the APIs you have seen so far is that you were able to access all the others anonymously and Twitter requires that you have an account. In order gain access to the Twitter API, one needs to create a twitter account if you don't already have one,

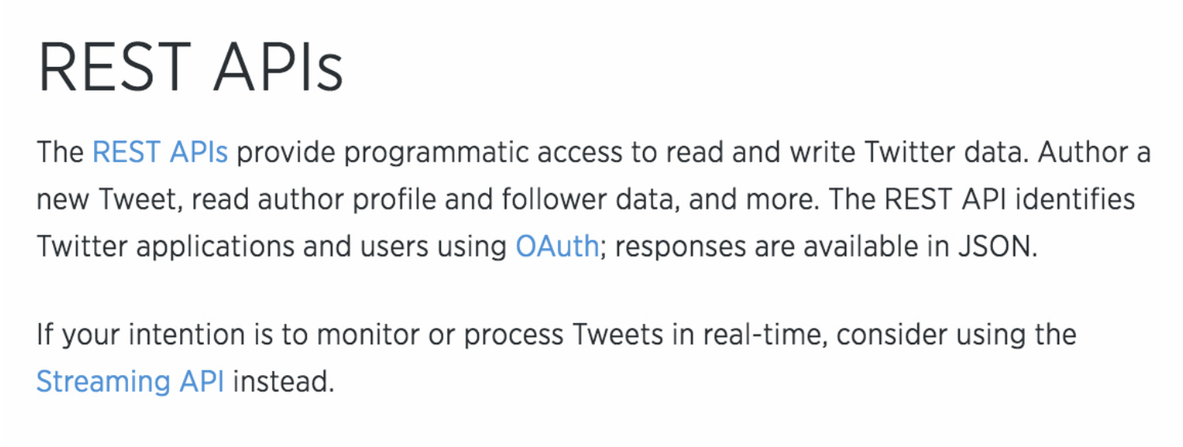
Access the Twitter API



Access the Twitter API

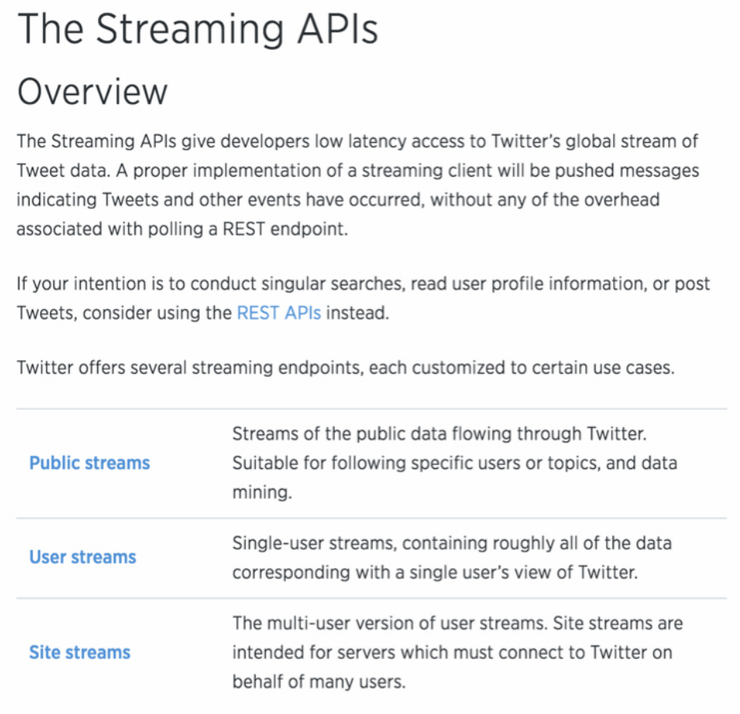


Twitter has a number of APIs



8. Twitter has a number of APIs
It is now important to mention that Twitter has a number of APIs. Firstly, they have a REST API; we won't go into the gory details of REST APIs here but I'll say two things - one: REST is short for Representational State Transfer; two: Twitter's REST API allows the user to "read and write Twitter data"; In order "monitor or process Tweets in real-time",

Twitter has a number of APIs



The Streaming APIs

Overview

The Streaming APIs give developers low latency access to Twitter's global stream of Tweet data. A proper implementation of a streaming client will be pushed messages indicating Tweets and other events have occurred, without any of the overhead associated with polling a REST endpoint.

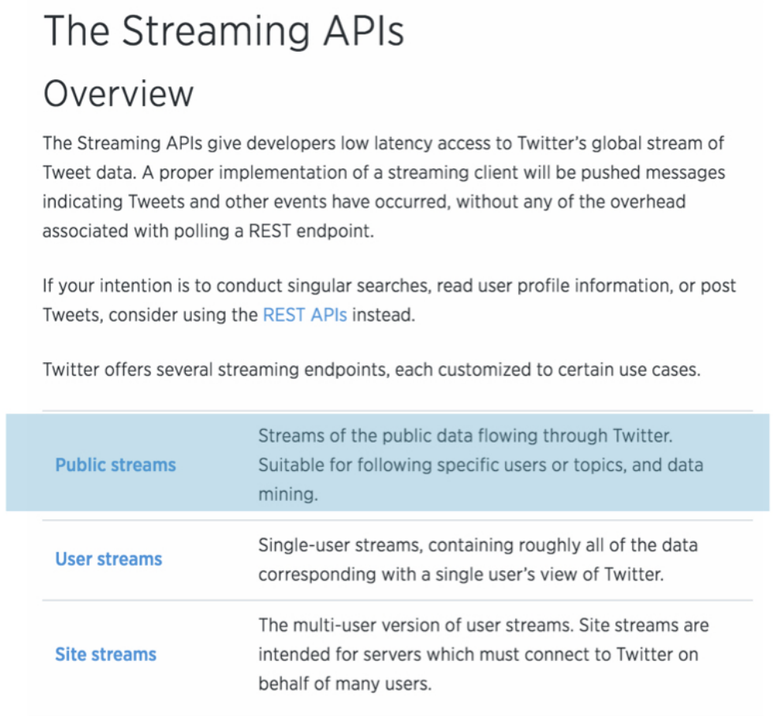
If your intention is to conduct singular searches, read user profile information, or post Tweets, consider using the REST APIs instead.

Twitter offers several streaming endpoints, each customized to certain use cases.

Public streams	Streams of the public data flowing through Twitter. Suitable for following specific users or topics, and data mining.
User streams	Single-user streams, containing roughly all of the data corresponding with a single user's view of Twitter.
Site streams	The multi-user version of user streams. Site streams are intended for servers which must connect to Twitter on behalf of many users.

9. Twitter has a number of APIs
that is, to stream Twitter data, however, we'll want to use Twitter's Streaming API. In particular,

Twitter has a number of APIs



The Streaming APIs

Overview

The Streaming APIs give developers low latency access to Twitter's global stream of Tweet data. A proper implementation of a streaming client will be pushed messages indicating Tweets and other events have occurred, without any of the overhead associated with polling a REST endpoint.

If your intention is to conduct singular searches, read user profile information, or post Tweets, consider using the REST APIs instead.

Twitter offers several streaming endpoints, each customized to certain use cases.

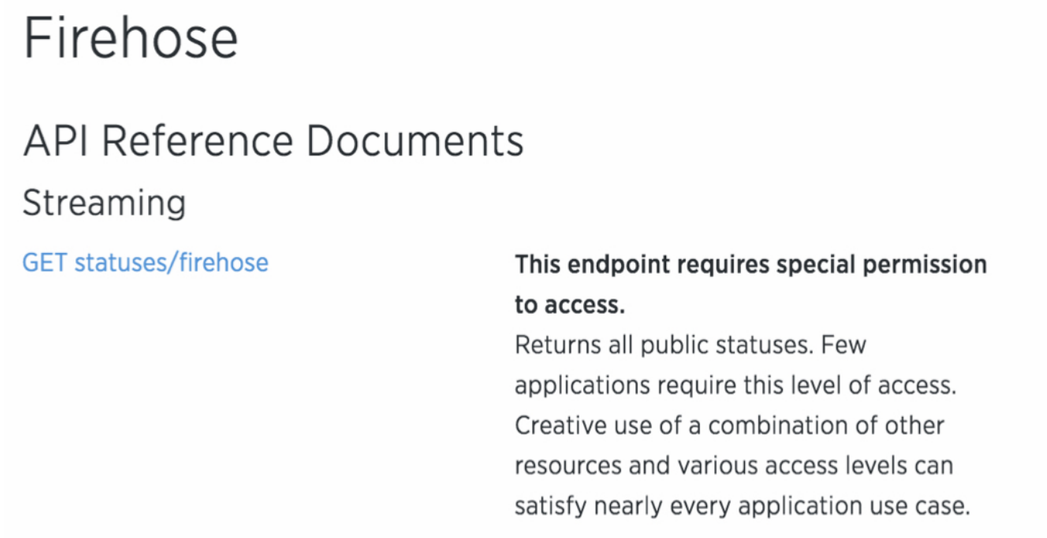
Public streams	Streams of the public data flowing through Twitter. Suitable for following specific users or topics, and data mining.
User streams	Single-user streams, containing roughly all of the data corresponding with a single user's view of Twitter.
Site streams	The multi-user version of user streams. Site streams are intended for servers which must connect to Twitter on behalf of many users.

10. Twitter has a number of APIs
we'll use the public stream, which Twitter's API documentation states "Streams of the public data flowing through Twitter.". The Public Stream itself contains a number of options. As we want to read and process tweets,

Twitter has a number of APIs

11. Twitter has a number of APIs
we'll want to use the GET statuses/sample API, which "Returns a small random sample of all public statuses."

Twitter has a number of APIs



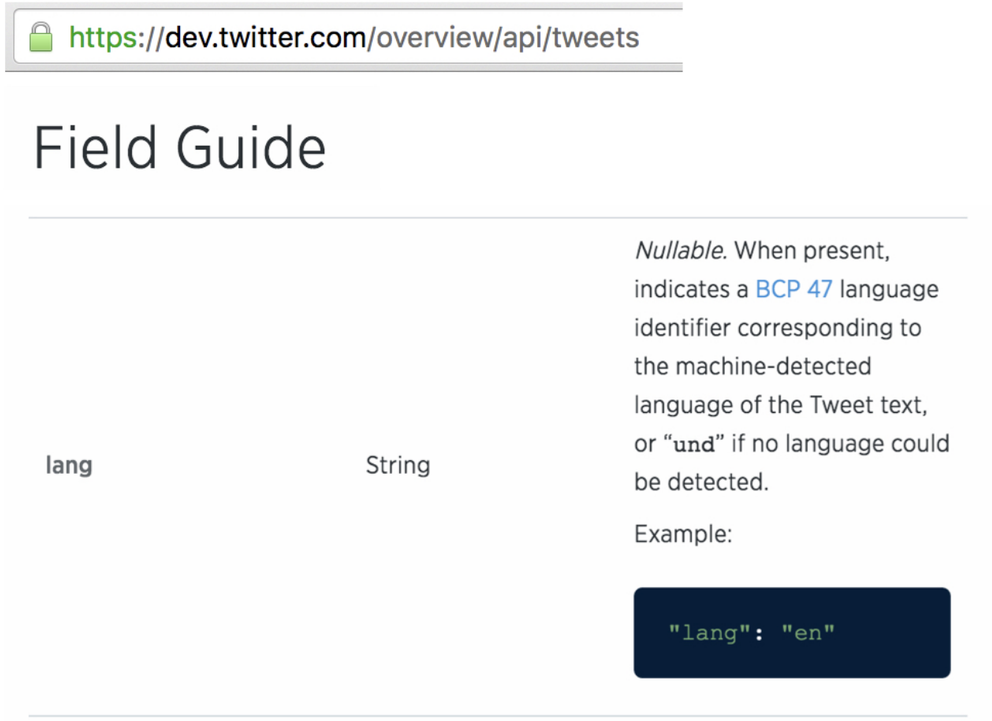
12. Twitter has a number of APIs

If you wanted to access absolutely "All public statuses", you would need to use Twitter's Firehose API, which is not publicly available and would most likely cost you a pretty penny.

Tweets are returned as JSONs

13. Tweets are returned as JSONs
One last point to note before we begin streaming tweets: tweets are returned to us as JSONs and they contain numerous possible fields. Check out the Twitter tweet field guide here. You can get tweet text, user, language, time of tweet,

Tweets are returned as JSONs



14. Tweets are returned as JSONs
among many other fields. Lets see how to access and stream data from the Twitter API!. For first-time Python tweet-streamers, I usually recommend the package tweepy,

Using Tweepy: Authentication handler

`tw_auth.py`

```
import tweepy, json
access_token = "..."
access_token_secret = "..."
consumer_key = "..."
consumer_secret = "..."
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
```

15. Using Tweepy: Authentication handler
which has a nice balance of usability and capability. Let's now use it to stream some tweets! First off, it has an OAuth handler which takes care of all of that nasty stuff for you: all you need to do is to pass the API Key and Secret to the handler and then to pass to access credentials using the set_access_token method.

Tweepy: define stream listener class

st_class.py

```
class MyStreamListener(tweepy.StreamListener):
    def __init__(self, api=None):
        super(MyStreamListener, self).__init__()
        self.num_tweets = 0
        self.file = open("tweets.txt", "w")
    def on_status(self, status):
        tweet = status._json
        self.file.write(json.dumps(tweet) + '\\n')
        tweet_list.append(status)
        self.num_tweets += 1
        if self.num_tweets < 100:
            return True
        else:
            return False
        self.file.close()
```

16. Tweepy: define stream listener class
After this, you'll need to define you Twitter stream listener Class. I wouldn't necessarily expect you to be able to do this yourself so I'm going to do that for you both here and in the interactive exercises that follow. Here I define a Tweet listener that creates a file called 'tweets dot txt', collects streaming tweets and writes them to the file 'tweets dot txt'; once 100 tweets have been streamed, the listener closes the file and stops listening.

Using Tweepy: stream tweets!!

`tweets.py`

```
# Create Streaming object and authenticate
l = MyStreamListener()
stream = tweepy.Stream(auth, l)
# This line filters Twitter Streams to capture data by keywords:
stream.filter(track=['apples', 'oranges'])
```

17. Using Tweepy: stream tweets!!
Now that we have written our Twitter Stream Listener Class, all you need to do is to create an instance of it and authenticate it. You can then stream tweets that containing keywords of choice by applying the filter method to the object stream! In the following exercises, you'll practice writing Python code to stream tweets and then you'll do some basic analysis of these tweets to see how often particular keywords are mentioned.

Let's practice!

INTERMEDIATE IMPORTING DATA IN PYTHON

Final Thoughts

INTERMEDIATE IMPORTING DATA IN PYTHON

Hugo Bowne-Anderson
Data Scientist at DataCamp

datacamp

What you've learned:

- Importing text files and flat files
- Importing files in other formats
- Writing SQL queries
- Getting data from relational databases
- Pulling data from the web
- Pulling data from APIs

Let's practice!

INTERMEDIATE IMPORTING DATA IN PYTHON