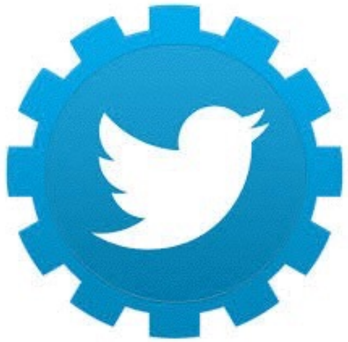
IMPORTING DATA IN PYTHON II

The Twitter API and Authentication

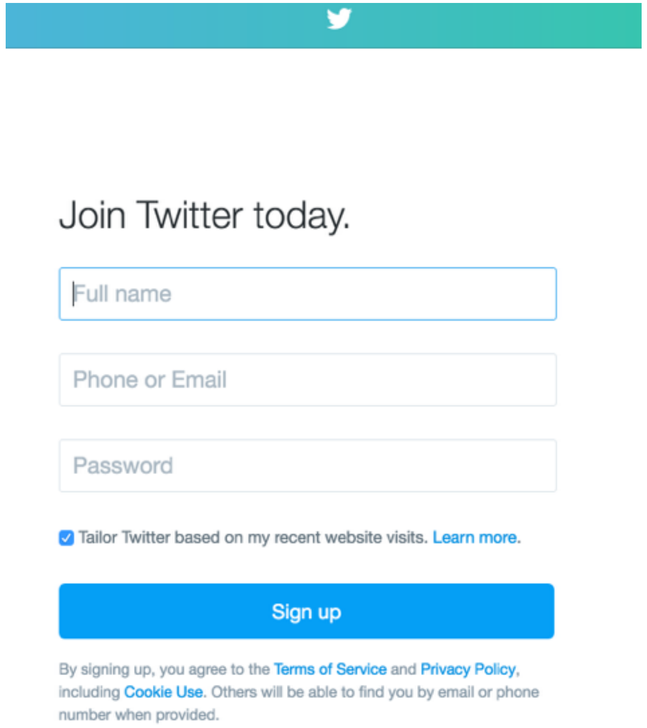
Herein, you'll learn

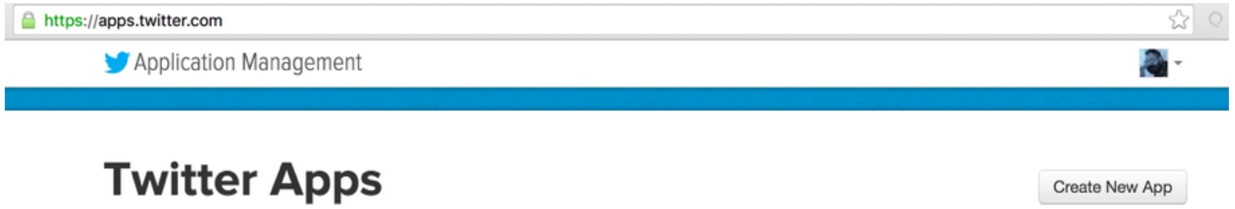
- How to stream data from the Twitter API
- How to filter incoming tweets for keywords
- About API Authentication and OAuth
- How to use the Tweepy Python package



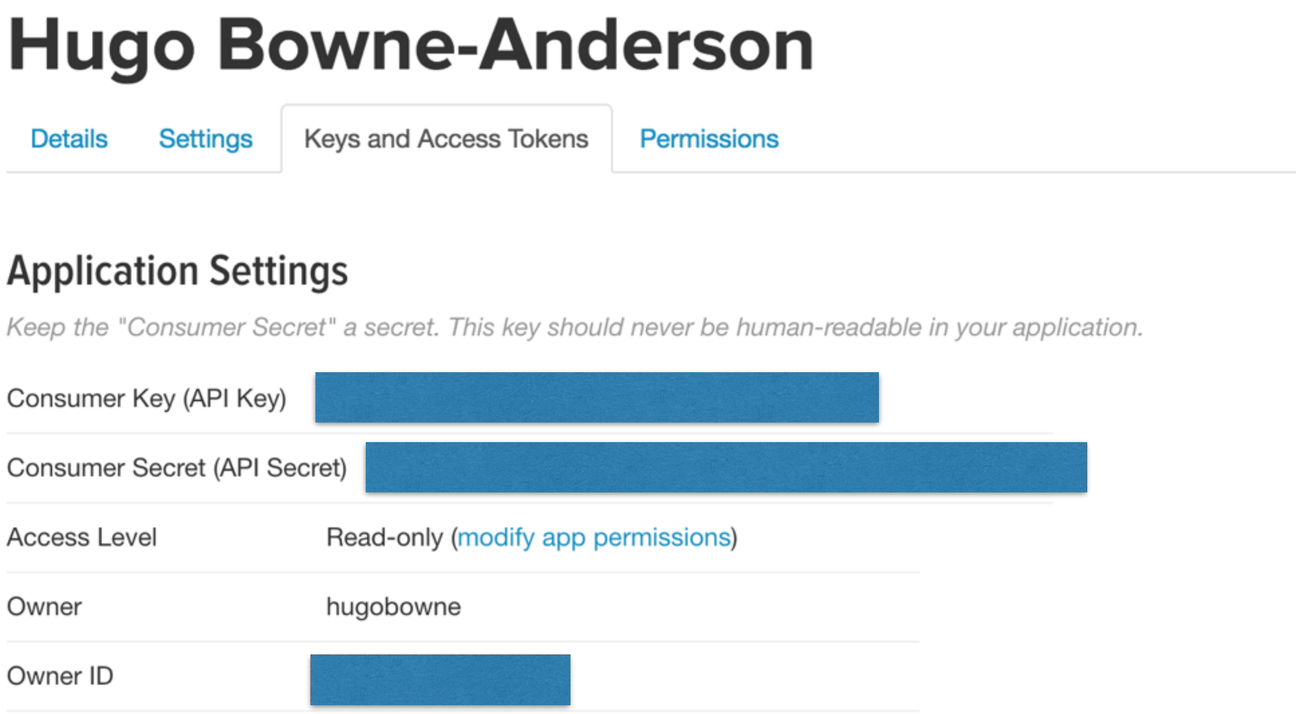
tweepy

Access the Twitter API





Access the Twitter API



Access the Twitter API

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	[redacted]
Access Token Secret	[redacted]
Access Level	Read-only
Owner	hugobowne
Owner ID	[redacted]

Twitter has a number of APIs

REST APIs

The REST APIs provide programmatic access to read and write Twitter data. Author a new Tweet, read author profile and follower data, and more. The REST API identifies Twitter applications and users using OAuth; responses are available in JSON.

If your intention is to monitor or process Tweets in real-time, consider using the Streaming API instead.

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.

Twitter has a number of APIs

GET statuses/sample

Returns a small random sample of all public statuses. The Tweets returned by the default access level are the same, so if two different clients connect to this endpoint, they will see the same Tweets.

Resource URL

`https://stream.twitter.com/1.1/statuses/sample.json`

Twitter has a number of APIs

Firehose

API Reference Documents

Streaming

GET statuses/firehose

This endpoint requires special permission to access.
Returns all public statuses. Few applications require this level of access. Creative use of a combination of other resources and various access levels can satisfy nearly every application use case.

Tweets are returned as JSONs

https://dev.twitter.com/overview/api/tweets

Field Guide

text	String	The actual UTF-8 text of the status update. See twitter-text for details on what is currently considered valid characters. Example: `"text":"Tweet Button, Follow Button, and Web Intents javascript now support SSL http:\/\/t.co\/9fbA0oYy ^TS"`

Tweets are returned as JSONs

https://dev.twitter.com/overview/api/tweets

Field Guide

lang	String	*Nullable.* When present, indicates a BCP 47 language identifier corresponding to the machine-detected language of the Tweet text, or "**und**" if no language could be detected. Example: `"lang": "en"`

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)
```

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()
```

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'])
```

IMPORTING DATA IN PYTHON II

Let’s practice!

IMPORTING DATA IN PYTHON II

Final Thoughts

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

IMPORTING DATA IN PYTHON II

Congratulations!