

# Gathering Data with APIs

---

R

PYTHON

DR. AMI GATES

# What is an API?

---

**API:** Application Programming Interface  
**Goal:** To connect to a Server as a Client to gain data or information.

**Common Method:** POST/GET URL query with the correct key=value pairs.

## Examples:

AirNow.gov

[http://www.airnowapi.org/aq/forecast/latLong/?format=text/csv&latitude=39.0509&longitude=-121.4453&date=2020-08-04&distance=25&API\\_KEY=D9AA91E7-070D-4221-867C-EFF5E0D8C2C7](http://www.airnowapi.org/aq/forecast/latLong/?format=text/csv&latitude=39.0509&longitude=-121.4453&date=2020-08-04&distance=25&API_KEY=D9AA91E7-070D-4221-867C-EFF5E0D8C2C7)

<https://newsapi.org/>

<https://newsapi.org/v2/top-headlines?country=us&apiKey=8f4134f7d0de43b8b49f91e22100f22b>

# Which Sites/Companies Offer APIs?

---

The best method to answer this question is to **search**.

Some Examples:

<https://docs.airnowapi.org/about>

<https://newsapi.org>

<https://developer.twitter.com/en/apply-for-access>

[https://www.mediawiki.org/wiki/API:Main\\_page](https://www.mediawiki.org/wiki/API:Main_page)

<https://pypi.org/project/yahoo-finance/>

<https://finance.yahoo.com/quotes/API,Documentation/view/v1/>

<https://www.api-football.com/documentation>

<https://wonder.cdc.gov/wonder/help/WONDER-API.html>

<https://api.data.gov/docs/fbi/>

# Using an API – The Steps

---

1) Find the API you want to use.

2) READ the documentation 😊

3) Register and get the necessary keys, passcodes, etc.

**EACH API is a little different** – it is best to read the documentation and look at examples on the site

4) Be sure you know the **endpoint URL** (also called base URL).

5) Read about the attributes – the **KEY = VALUE pair options**.

6) Build a practice URL and test it.

7) Code it in R and Python

# APIs and Building URLs in Python

---

```
## https://newsapi.org/
```

```
import requests
```

```
import json
```

```
BaseURL="https://newsapi.org/v1/articles"
```

```
URLPost = {'apiKey': '8f4134f7d0de43b8b49f91e22100f22b', ## USE YOUR KEY – NOT MINE!
```

```
    'source': 'bbc-news',
```

```
    'pageSize': 85,
```

```
    'sortBy': 'top',
```

```
    'totalRequests': 75}
```

```
response1=requests.get(BaseURL, URLPost)
```

```
jsontxt = response1.json()
```

What does this build?  
Try it!

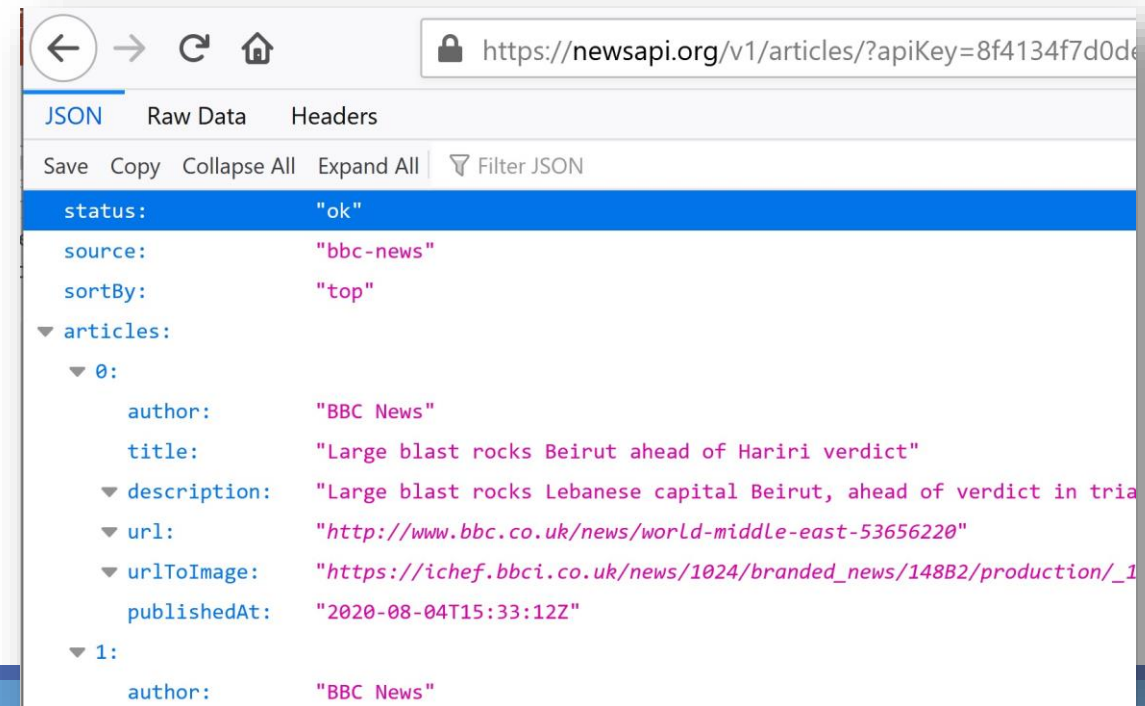
# The URL Built by Python

---

<https://newsapi.org/v1/articles/?apiKey=8f4134f7d0de43b8b49f91e22100f22b&source=bbc-news&pageSize=85&sortBy=top&totalRequests=75>

**Test it!** How? Paste it into a Browser.

**You should get this →**



The screenshot shows a web browser displaying the JSON response from the NewsAPI endpoint. The browser's address bar shows the URL: `https://newsapi.org/v1/articles/?apiKey=8f4134f7d0de43b8b49f91e22100f22b&source=bbc-news&pageSize=85&sortBy=top&totalRequests=75`. The JSON response is displayed in a structured format with the following fields:

```
status: "ok"
source: "bbc-news"
sortBy: "top"
articles:
  0:
    author: "BBC News"
    title: "Large blast rocks Beirut ahead of Hariri verdict"
    description: "Large blast rocks Lebanese capital Beirut, ahead of verdict in trial"
    url: "http://www.bbc.co.uk/news/world-middle-east-53656220"
    urlToImage: "https://ichef.bbci.co.uk/news/1024/branded_news/148B2/production/_14882220.jpg"
    publishedAt: "2020-08-04T15:33:12Z"
  1:
    author: "BBC News"
```

# There are Many Ways To Do the Same Thing

### WAY 2

```
import requests
import json
url = ('https://newsapi.org/v2/everything?'
      'q=Sports&'
      'from=2020-08-04&'
      'sortBy=relevance&'
      'source=bbc-news&'
      'pageSize=100&'
      'apiKey=8f4134f7d0de43b8b49f91e22100f22b')

response2 = requests.get(url)
jsontxt2 = response2.json()
```

This code builds and posts THIS and returns the following

<https://newsapi.org/v2/everything?q=Sports&from=2020-08-04&sortBy=relevance&source=bbc-news&pageSize=100&apiKey=8f4134f7d0de43b8b49f91e22100f22b>

```
status:      "ok"
totalResults: 1455
articles:
  0:
    source:
      id:      null
      name:    "Gizmodo.com"
      author:  "Sam Rutherford"
    title:    "Olympus' OM-D E-M10 Mark IV Sports a New Sensor and a Flip-Down Display"
    description: "Even though Olympus is planning to sell off its camera business later than the refreshed OM-D E-M10 Mark IV.Read more..."
    url:      "https://gizmodo.com/olympus-om-d-e-m10-mark-iv-sports-a-new-sensor-and-c"
    urlToImage: "https://i.kinja-img.com/gawker-media/image/upload/c_fill,f_auto,fl_progressive,w_1000,h_1000,q_80/olympus-om-d-e-m10-mark-iv-sports-a-new-sensor-and-c"
    publishedAt: "2020-08-04T13:30:00Z"
    content:    "Even though Olympus is planning to sell off its camera business later than the refreshed OM-D E-M10 Mark IV.\r\nStarting at $70... [+2524 chars]"
  1:
```

# APIs and Building URLs in R

---

```
library("httr")
library("jsonlite")
base <- "http://www.airnowapi.org/aq/forecast/zipCode/"
FM<-"text/csv"
zipCode="20002"
date="2019-11-16"
## DO NOT USE MY KEY!!
API_KEY="D9AA91E7-070D-4221-867C-EFF5E0D8C2C7"
distance="25"
```

```
(call1 <- paste(base,"?",
  "format", "=", FM, "&",
  "zipCode", "=", zipCode, "&",
  "date", "=", date, "&",
  "API_KEY", "=", API_KEY, "&",
  "distance", "=", distance,
  sep=""))
```

```
(AirNowAPI_Call<-httr::GET(call1))
```

```
(MYDF<-httr::content(AirNowAPI_Call))
```



# What Was Built - Posted – and Retrieved?

DO NOT USE MY API KEYS

This is the POST/GET URL that R Builds – try it in a Browser! What happens?

[http://www.airnowapi.org/aq/forecast/zipCode/?format=text/csv&zipCode=20002&date=2019-11-16&API\\_KEY=D9AA91E7-070D-4221-867C-EFF5E0D8C2C7&distance=25](http://www.airnowapi.org/aq/forecast/zipCode/?format=text/csv&zipCode=20002&date=2019-11-16&API_KEY=D9AA91E7-070D-4221-867C-EFF5E0D8C2C7&distance=25)

This is code to save the results of the query to a .csv file:

```
AirName = "AirFileExample.csv"
## Start the file
AirFile <- file(AirName)
## Write to the file
write.csv(MYDF, AirFile, row.names = FALSE)
```

This is .csv file with the results:

DateIssue	DateForecast	Reporting	State	Latitude	Longitude	Parameter	AQI	CategoryN	CategoryN	Action
11/15/2019	11/16/2019	Metropoli	DC	38.919	-77.013	PM2.5	33	1	Good	FA
11/15/2019	11/17/2019	Metropoli	DC	38.919	-77.013	PM2.5	29	1	Good	FA
11/15/2019	11/18/2019	Metropoli	DC	38.919	-77.013	PM2.5	38	1	Good	FA
11/15/2019	11/19/2019	Metropoli	DC	38.919	-77.013	PM2.5	42	1	Good	FA
11/15/2019	11/20/2019	Metropoli	DC	38.919	-77.013	PM2.5	46	1	Good	FA

# Links to APIs Code: Python and R

---

Python

[https://drive.google.com/drive/folders/146La-4Nq\\_DRUdhY7bqJ0B5vIQkVd4iDI?usp=sharing](https://drive.google.com/drive/folders/146La-4Nq_DRUdhY7bqJ0B5vIQkVd4iDI?usp=sharing)

R

<https://drive.google.com/drive/folders/17K0kgeNxiK8TWP-aMrepvgPxe2-hlcee?usp=sharing>

# Using the Twitter API

---

- 1) You will need a Twitter Dev Account. This can take time so get one now...
- 2) Use your .EDU email to apply for the Twitter Dev Account.
- 3) Learn to log into the Twitter Dev area and to get your codes and keys
- 4) Twitter does NOT make it easy. You will need to click, search, Google, explore, learn, and try.



https://developer.twitter.com/en



Developer

Use cases

Products

Docs

More

Labs

Developers

Tap

**Documentation**

---

**API reference index**

---

**Tutorials**

---

**Changelog**



Developer

Use cases

Products

Docs

More

Labs

Dashboard

DrGates309

Apps > **GatesTwitterMining**

App details

**Keys and tokens**

Permissions

## Keys and tokens

Keys, secret keys and access tokens management.

### Consumer API keys

API key:

API secret key:

Get started

Subscriptions

**Apps**

Apps

Dev environments

Billing

# What is a **REST API**?

---

RE: <https://dev.twitter.com/rest/public>

“The [REST APIs](#) provide programmatic access to read and write Twitter data. “

You can: Create a new Tweet, read a user profile, follower data, and more.

The REST API identifies Twitter applications and users using [OAuth](#); responses that are in JSON format.

If your intention is to monitor or process Tweets in real-time, consider using the [Streaming API](#) instead.”

# What is “REST”

---

## REST: Representational State Transfer.

- ❑ An **architecture style** that is a **stateless, client-server based, and cacheable communications protocol that generally works with HTTP.**
- ❑ A **RESTful API** refers to an application program interface (API) that uses **HTTP** and **GET, PUT, POST, and DELETE**; most commonly GET and POST.

Tutorial: <http://www.restapitutorial.com/lessons/whatisrest.html>

## REST:

- Six Constraints: uniform interface, stateless, client-server, cacheable, layered, code on demand.
- Resourced-based (like user or address or thing) (rather than action based in SOAP-RPC such as get data).

# What is OAuth: Send secure authorized requests

---

<https://dev.twitter.com/oauth/overview/single-user>

## OAuth: Application-only authentication

**Twitter** offers the ability for you to retrieve a single access token (complete with `oauth_token_secret`) from application detail pages found on [dev.twitter.com](https://dev.twitter.com).

This is ideal for applications with single-user use cases. You shouldn't ever share the combination of your OAuth **consumer key, secret, access token, and access token secret**.

By using a single access token, you don't need to implement the entire OAuth token acquisition dance. Instead, you can pick up from the point where you are working with an access token to make signed requests for Twitter resources.



# Twitter and OAuth Code in Python:

---

```
import tweepy
```

```
#conda install -c conda-forge tweepy
```

```
from tweepy import OAuthHandler
```

```
auth = OAuthHandler(consumer_key, consumer_secret)
```

```
auth.set_access_token(access_token, access_secret)
```

```
api = tweepy.API(auth)
```

# Other Methods of Gathering Data

---

1) Experiments

2) Surveys (online, phone, in-person)

3) Downloading from Data Sites

- Kaggle
- CDC
- .gov sites
- Etc.

4) Web Scraping – this method is generally not permitted and it is always better to use an API.

# OAuth

## Send secure authorized requests to the Twitter API

---

Twitter uses OAuth to provide authorized access to its API.



## Features

- **Secure** - Users are not required to share their passwords with 3rd party applications, increasing account security.
- **Standard** - A wealth of client libraries and example code are compatible with Twitter's OAuth implementation.

## Twitter API Authentication Model

There are two forms of authentication, both leveraging OAuth 1.0A.