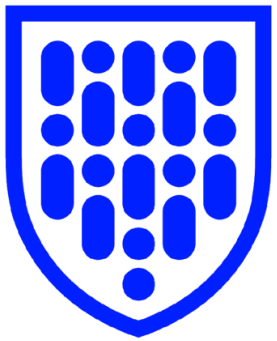


Institute of Data

2022

Data Science and AI

Module 3
Part 2:

APIs

Agenda: Module 3 Part 2

- What is an API?
- APIs for data services
- APIs for analytic services
- APIs for visualisation services
- APIs for cognitive services
- Creating an API

Institute of Data

What is an API?

- Definition, examples
- Interfaces
- Authentication protocols
- Documentation

What is an API?

- What does “API” stand for?
  - Application Programming Interface

- Examples?
  - automation in Microsoft Office
    - e.g. generating a Word document or an Outlook reminder from another application
  - high-level database drivers
    - e.g. PyMongo
  - programming libraries for mobile & wearable devices
  - programmable web services
  - other?

Use Cases for APIs

- integrate remote data access
  - repetitive analyses of an **evolving dataset**
  - **up-to-the-moment** forecasting

- **integrate** familiar functionality
  - location sharing using Google Maps
  - simplified app login via Facebook
  - in-app purchases
  - in-app YouTube viewing






Some Popular Web Service APIs

Name	Nature	URL
Twitter	Networking, marketing, trending	https://developer.twitter.com/en.html
Facebook	Networking, marketing	https://developers.facebook.com/tools/
Amazon S3	Cloud storage, Big Data analytics	https://aws.amazon.com/s3/
LinkedIn	Networking	https://developer.linkedin.com/
eBay	E-commerce	https://developer.ebay.com/
Google API Console	Data access & analytics, e-commerce, etc.	https://developers.google.com/apis-explorer/#p/
New York Times	News	http://developer.nytimes.com/

Interfaces for Web Service APIs

- SOAP
  - *Simple Object Access Protocol*
  - early, widespread web service protocol
  - exposes components of application logic as services
  - XML
- REST
  - *Representational State Transfer*
  - now > 70% of public APIs
  - accesses data
  - variety of data formats, coupled with JSON
  - generally faster and uses less bandwidth
  - easier to integrate with existing websites

Overview of RESTful API Description Languages:
https://en.wikipedia.org/wiki/Overview_of_RESTful_API_Description_Languages

roll your own:
https://www.restapitutorial.com/
https://aws.amazon.com/api-gateway

HTTP

- HyperText Transfer Protocol
- underlies RESTful APIs

- 4 major methods
  - GET fetches data from web server
  - PUT edits data on web server
  - POST adds new data
  - DELETE removes data

- HTTP Status Codes
  - 1xx informational
  - 2xx success
  - 3xx redirection
  - 4xx client error
  - 5xx server error

https://www.restapitutorial.com/httpstatuscodes.html

Elements of an API call

- ***endpoint***
  - URL of a server page that provides data or functionality via ***requests*** and ***responses***
- ***protocol***
  - the communication standard for passing requests to an endpoint
- ***authentication***
  - secure **identification** of user making request
  - if a developer creates an app for other users, the app needs to obtain **authorisation** from the owner of the API for both the developer's access *and* the user's access

Authentication Protocols

- HTTP Basic Access Authentication
  - username + password
  - transmitted in header of HTTP request
  - weakly encoded, no encryption
- OAuth 1.0
  - uses encrypted tokens
- OAuth 2.0
  - simpler, more robust than OAuth 1.0

OAuth 2.0

- token-based
  - e.g. ***client_id*** & ***client_secret***
  - allows a 3rd-party app to access a user's/developer's account **without knowing the account password**
  - allows an end-user to access an API via *your* app, using *their* token

- redirect URL
  - **registered** when app created
  - OAuth 2.0 service **returns user to this URL** after authorising (and issuing a user token)
  - protects access token from **interception**

https://www.oauth.com/oauth2-servers/background/

Developer Access

- some API's have **a developer mode** that may allow access without requesting a user token
- options for connect/request include:
  - use developer's *user_id* and *password*
  - use *app_id*, developer's *client_id*, developer's *secret*
- access granted **may** include
  - read developer's posts, comments, profile, etc.
  - post to developer's account
  - read other users' posts, comments, profiles, etc.

Python Libraries: Utilities

requests

- HTTP library (“elegant and simple”)
- http://www.python-requests.org/en/latest/
- returns JSON-formatted byte strings

json

- JSON ↔ lists, dictionaries
- https://docs.python.org/2/library/json.html

untangle, xmltodict

- parses XML to Pythonic data structures

BeautifulSoup (bs4)

- parses HTML, XML to Pythonic data structures

Python Libraries: API Wrappers

- simplify usage of APIs by introducing a Python API into the loop
- use data types & structures familiar to Python developers

pyfacebook

linkedin

praw (Reddit)

bucketstore (Amazon S3)

python-forecastio (weather)

foursquare (location-based networking)

GooPyCharts (Google Charts)

indeed (indeed.com)

kiteconnect (stock trading)

pymaps (Google Maps)

pymed (PubMed)

pyspotify (Spotify)

newsapi

rottentomatoes (crowd-based movie reviews)

sportradar (sport APIs)

tesserocr (OCR)

bowshock (NASA)

geopy (geocoding)

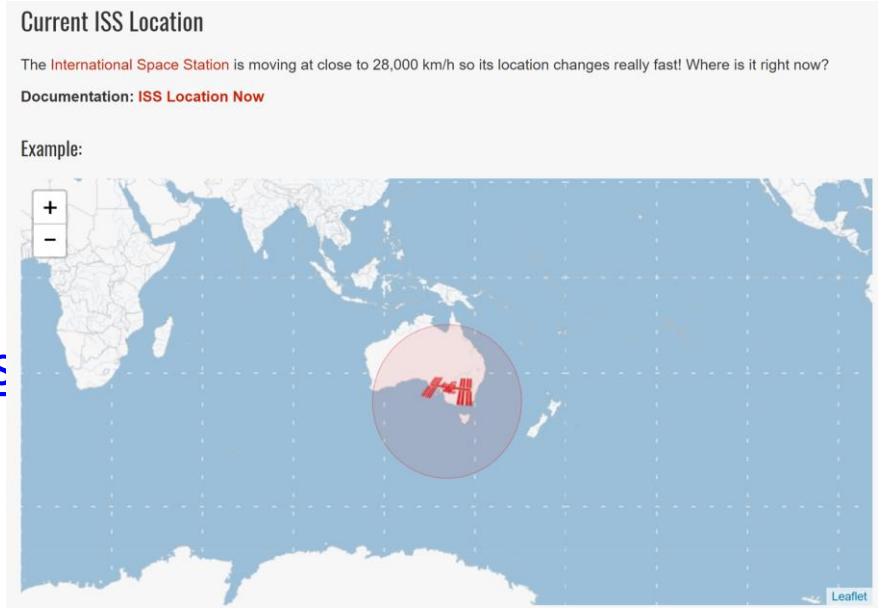
https://github.com/realpython/list-of-python-api-wrappers

Lab 3.2.1: Querying the ISS

- Purpose:
  - To become familiar with basic API requests and responses

- Resources:
  - API for the International Space Station:
    OpenNotify
    http://open-notify.org/Open-Notify-API/
  - HTTP response codes
    https://www.restapitutorial.com/httpstatus

- Materials:
  - 'Lab 3.2.1.ipynb'



Institute of Data

Extracting Data from APIs

- Twitter API
- Google Public Data and BigQuery API

Institute of Data

Twitter API

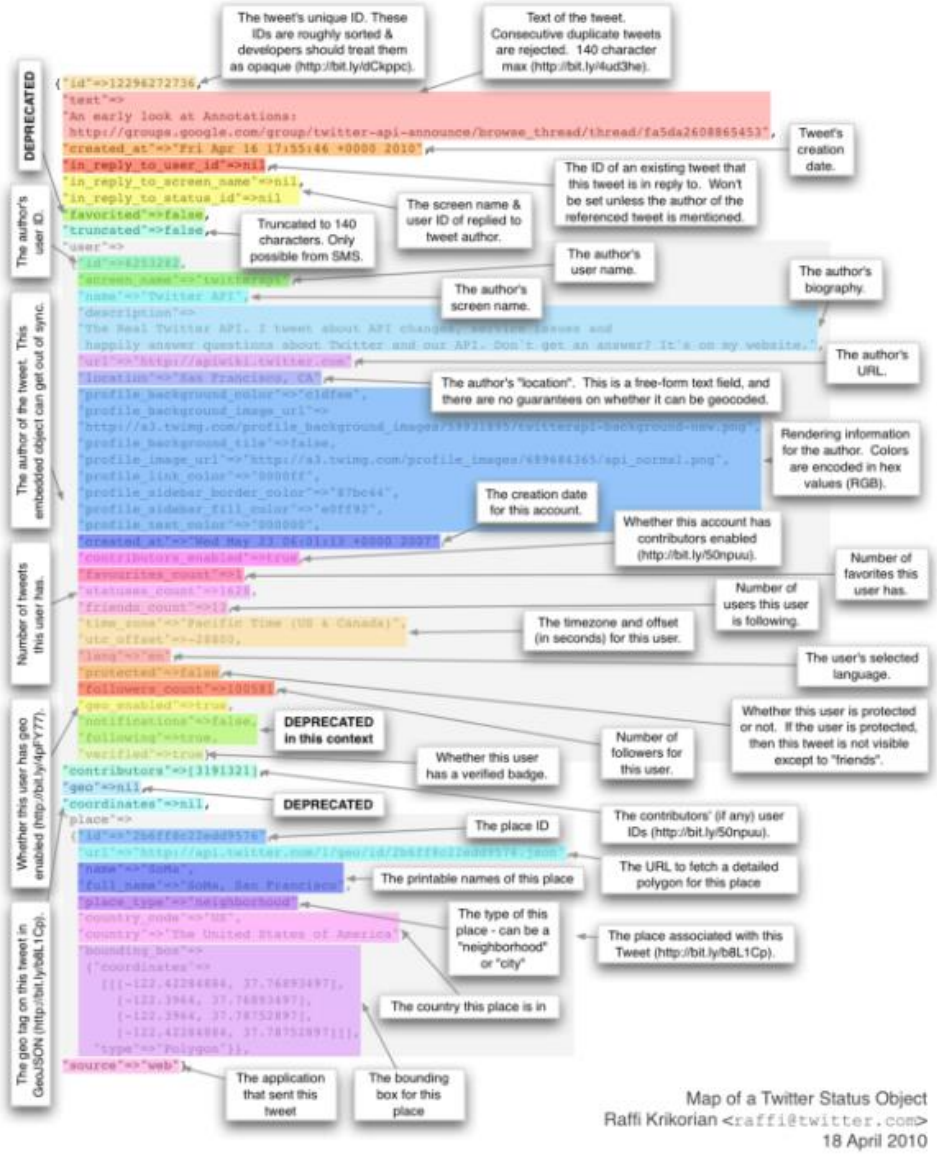
- Twitter API structure
- API Usage restrictions
- Developer / App approval

Twitter API

- Tweets are **status updates**
- main API object = **status**
  - root-level attributes:
    - *id*, *created_at*, *text*, ...
  - child objects:
    - *user*, *entities*, *extended_entities*, ...
    - *place* (if Tweet was geo-tagged)

https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object.html

http://socialmedia-class.org/twittertutorial.html



Twitter API

1. If you haven't got one, open a Twitter user account
2. Create a Twitter app (https://developer.twitter.com/en/apps)
3. Register the app for API access
4. Store your credentials
   - for accessing your account:
     - user name
     - password
   - for authenticating your app:
     - user agent (information describing your app)
     - client ID (a unique identifier for your app)
     - client secret (secure token for authorising your app to access the API)

Twitter App Restrictions

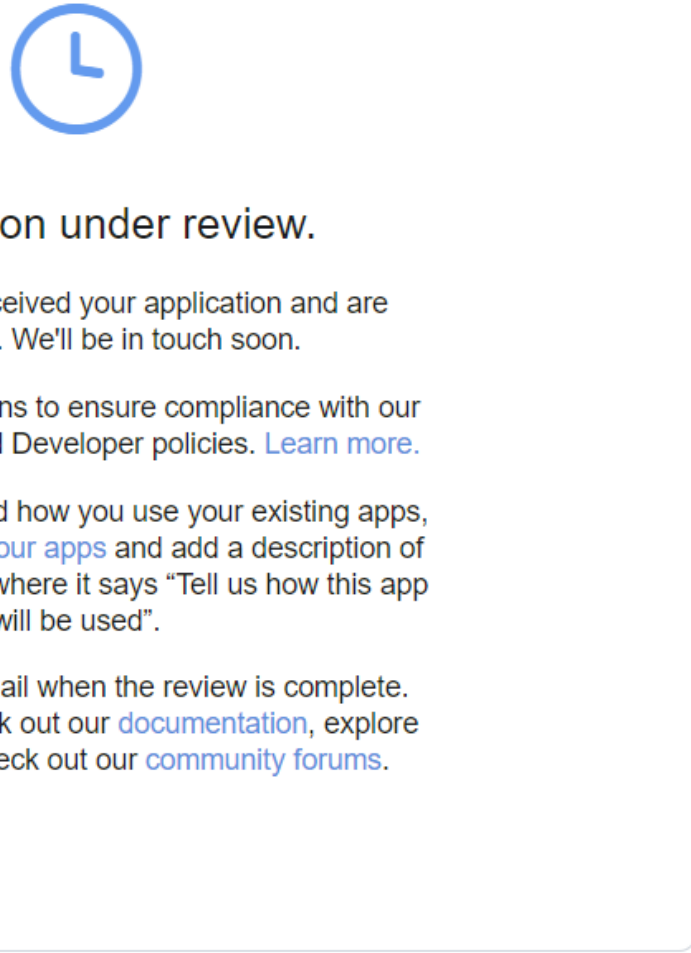
- Terms & Conditions:

C. **Geographic Data.** Your license to use Twitter Content in this Agreement does not allow you to (and you will not allow others to) aggregate, cache, or store location data and other geographic information contained in the Twitter Content, except in conjunction with the Twitter Content to which it is attached. Your license only allows you to use such location data and geographic information to identify the location tagged by the Twitter Content. Any use of location data or geographic information on a standalone basis or beyond the license granted herein is a breach of this Agreement.

Twitter App Approval

New developer requirements to protect our platform

https://blog.twitter.com/developer/en_us/topics/tools/2018/new-developer-requirements-to-protect-our-platform.html



Lab 3.2.2: Mining Social Media with Twitter

- Purpose:
  - To develop skills in using a more complex API
- Resources:
  - Python library for Twitter API: ***Tweepy***
    - http://docs.tweepy.org
- Materials:
  - 'Lab 3.2.2.ipynb'

Institute of Data

Google Cloud Platform

- public data sets / BigQuery
- APIs based on data science products

Google Cloud Platform

Google Cloud SDK	• https://cloud.google.com/sdk/gcloud/ • https://cloud.google.com/sdk/docs/initializing
Google Cloud Platform	• https://github.com/GoogleCloudPlatform/python-docs-samples • https://googlecloudplatform.github.io/google-cloud-python/ • https://googlecloudplatform.github.io/google-cloud-python/latest/
Google API Client Libraries	• https://developers.google.com/api-client-library/
Google BigQuery	• https://cloud.google.com/bigquery/public-data/ • https://cloud.google.com/bigquery/docs/quickstarts/quickstart-web-ui • https://cloud.google.com/bigquery/docs/reference/libraries • https://cloud.google.com/bigquery/create-simple-app-api • https://github.com/GoogleCloudPlatform/google-cloud-python/tree/master/bigquery

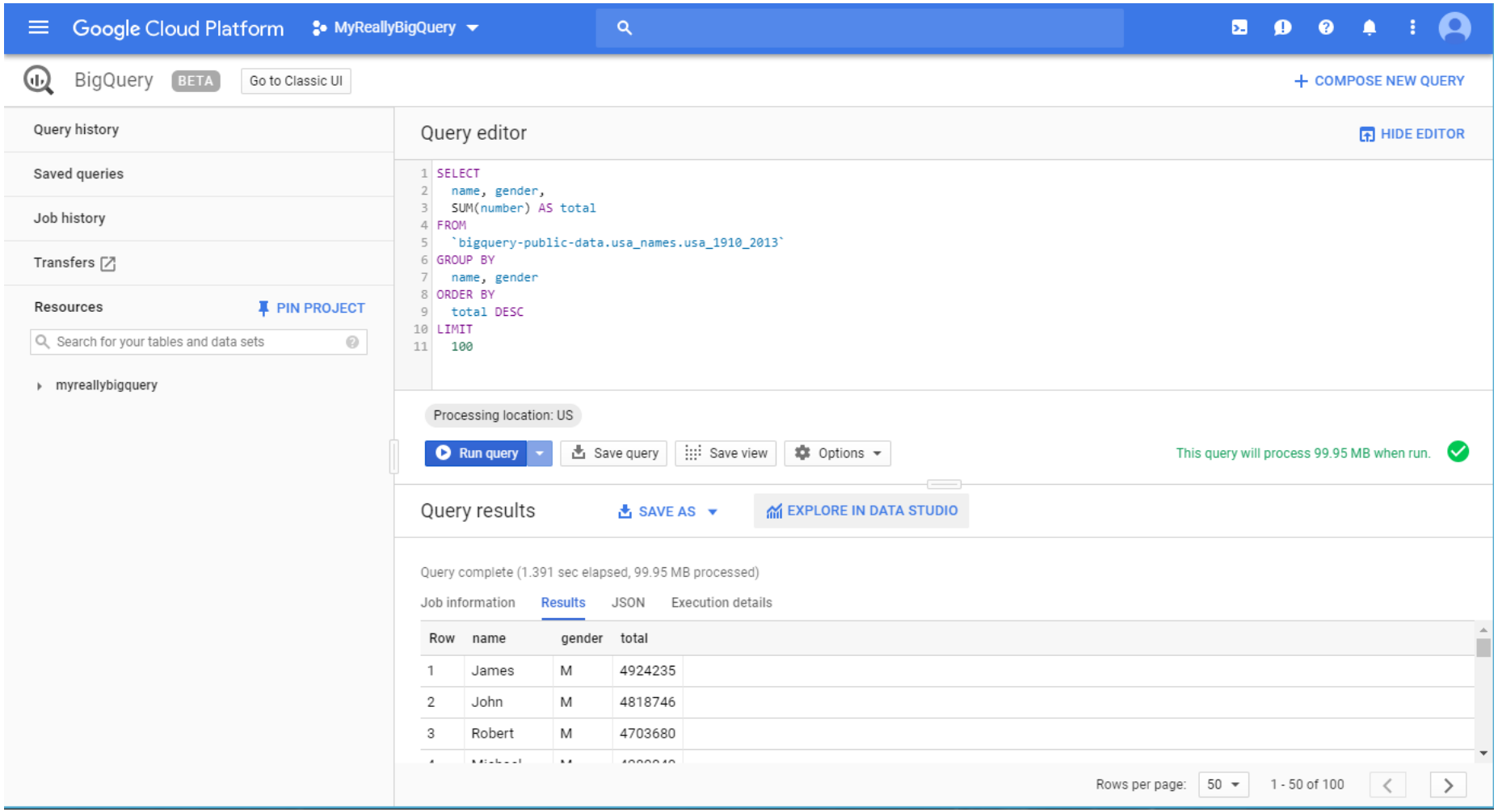
Google Public Data sets

- accessible via Google BigQuery
- free for 1st TB / month
- subject areas:
  - genomics
  - medicine & epidemiology
  - geo imagery (Earth science, weather, etc.)
  - transport & service utilisation
  - annotated images
  - etc.
- https://cloud.google.com/public-datasets/

Google BigQuery

Quickstart to BigQuery Web UI:

https://cloud.google.com/bigquery/docs/quickstarts/quickstart-web-ui



BigQuery API: Authentication

Service accounts

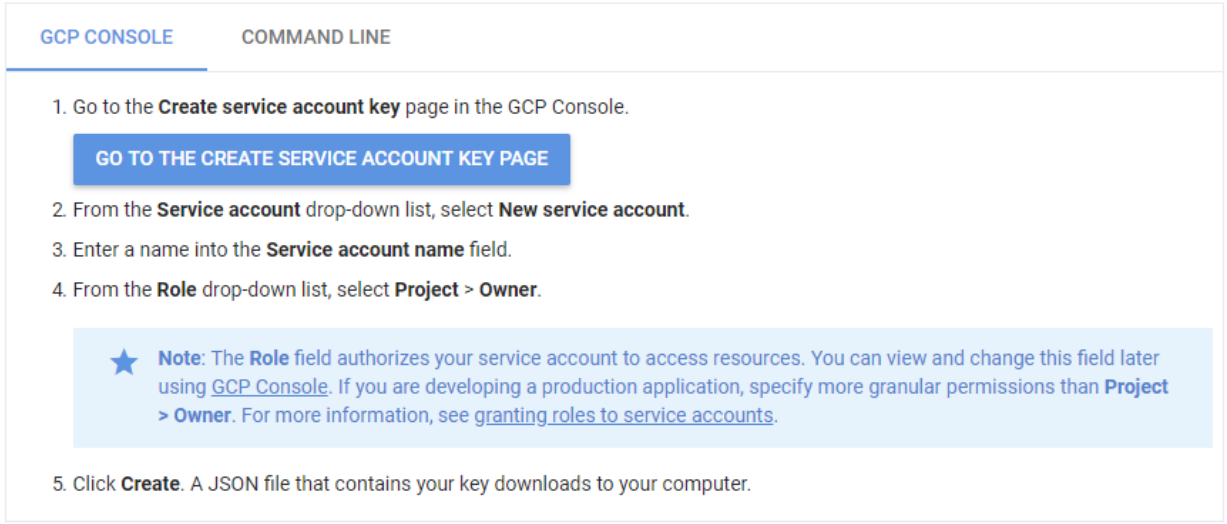
- for client apps that you will run
  - e.g. dev/test, batch processing pipelines
- authentication via your service credentials

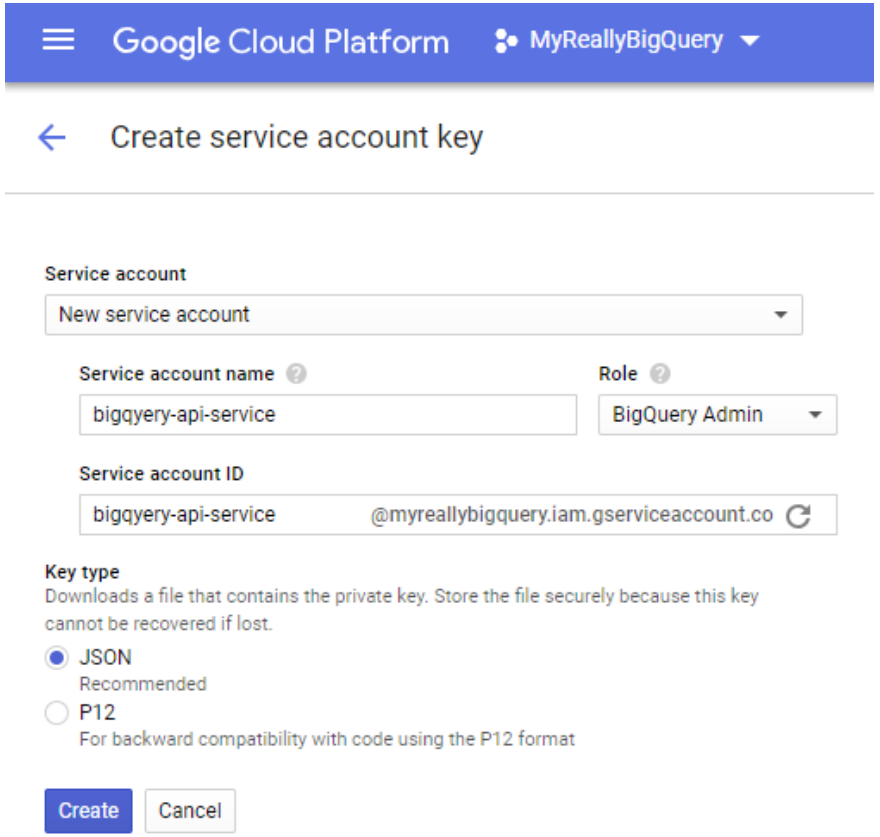
User accounts

- for apps you create for other end-users
  - e.g. data products
- authentication via end-users credentials
  - app can only access BigQuery tables that the end-user is authorised to access
  - end-user gets billed for queries

https://cloud.google.com/bigquery/docs/authentication/

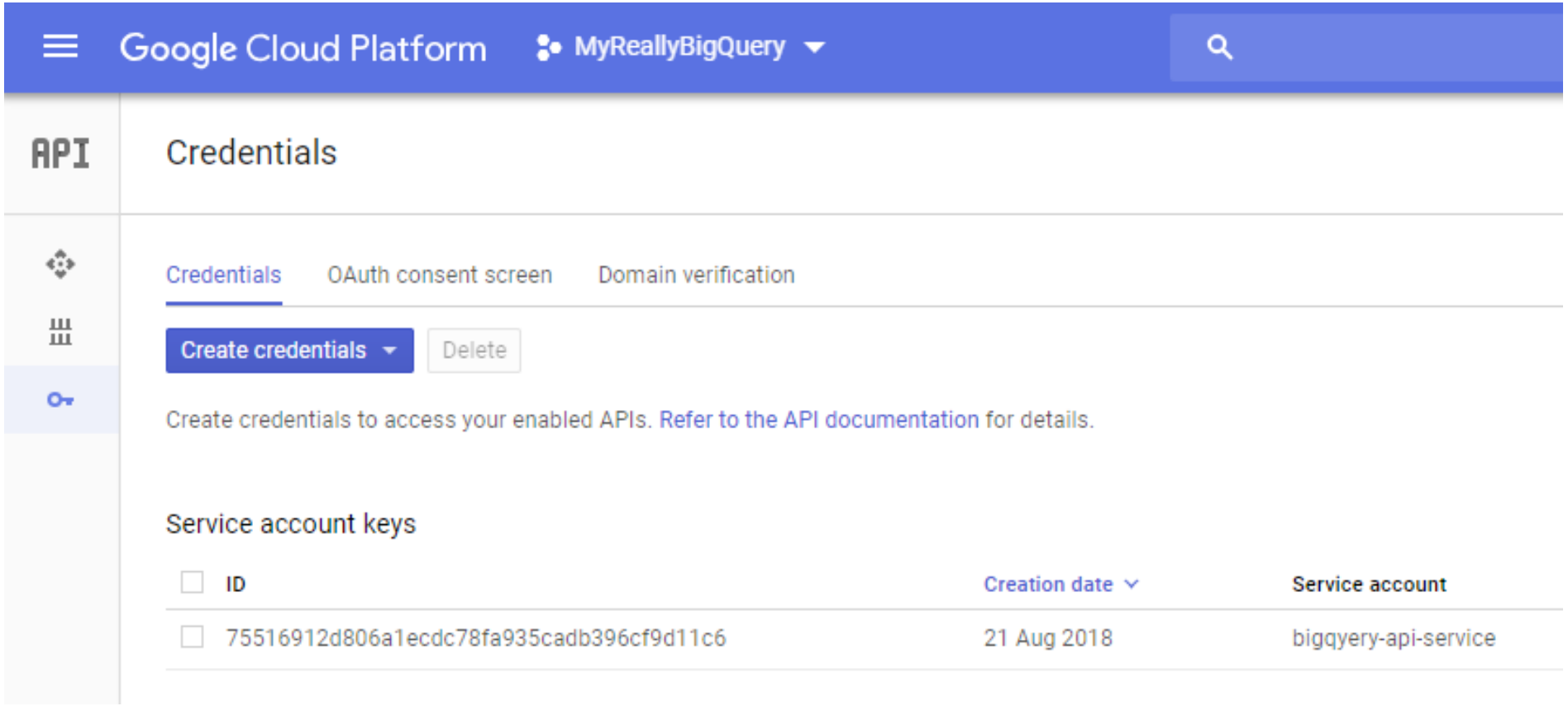
BigQuery API: Authentication – cont'd





https://cloud.google.com/docs/authentication/production

BigQuery API: Authentication – cont'd



Using the Google Authentication Key

Option 1: Set GOOGLE_APPLICATION_CREDENTIALS environment variable

- Linux / MacOS

```
$ export GOOGLE_APPLICATION_CREDENTIALS="[PATH]"
```

- Windows

```
$ set GOOGLE_APPLICATION_CREDENTIALS="[PATH]"
```

Option 2: Pass the path to the service account key in code

```
from google.cloud import storage
storage_client = storage.Client.from_service_account_json('[PATH]')
```

where '[PATH]' is the full file path of the json key file

Google BigQuery API: Top-Level Object

client object:

- connection
  - authenticated connection to the BigQuery service
  - determines credentials
    - implicitly from the environment,
    - or directly via *from_service_account_json* and *from_service_account_p12*
- project
  - top-level container
  - tied to billing
  - can provide default access control across all its datasets
  - access control list (ACL)
    - grants reader / writer / owner permission to one or more entities
    - must be managed using the Google Developer Console (not API)

BigQuery API Object Hierarchy

bigquery

- .projects
- .datasets
  - .get, .delete, .insert, .list, .update, ...
- .tabledata
- .tables
- .jobs
  - .get, .cancel, .insert, .list, .query, ...
- . . .

https://developers.google.com/apis-explorer/#p/bigquery/v2/

Lab 3.2.3: Big Data Analytics with BigQuery

- Purpose:
  - (1) To learn how to the Google BigQuery Web UI for discovering public data sets and performing basic analytics.
  - (2) To become proficient with the Google BigQuery API for wrangling Google's public datasets.

- Materials:
  - 'Lab 3.2.3.ipynb'

Lab 3.2.3 – cont'd

- Python packages :
  - pyarrow (pip)
  - google-cloud-bigquery (conda-forge)
  - google-cloud-storage (conda-forge)

- Resources:
  - Google BigQuery Public Datasets https://cloud.google.com/bigquery/public-data/
  - BigQuery UI https://cloud.google.com/bigquery/docs/quickstarts/quickstart-web-ui
  - Python client for BigQuery API https://github.com/GoogleCloudPlatform/google-cloud-python/tree/master/bigquery

Discussion

- Extracting data using APIs
  - applications?

Lab/ HOMEWORK

1. Create a mini-project based on any skills from the course so far:
   - select an interesting public data set or form a question you are interested answer and identify data needed to answer the question
   - use Jupyter Notebook to access, analyse and visualise the data
2. Prepare a 5-minute presentation
   - use Jupyter Notebook
   - organise as:
     - question
     - dataset & analysis
     - conclusion
3. plan to present to the class

Presentations

- each team
  - 5 minute presentation

Analytics-Based APIs

- **Google**
  - Google Analytics
    - https://developers.google.com/analytics/
  - Google Cloud Vision
    - https://cloud.google.com/vision/
  - Google Cloud AI
    - https://cloud.google.com/products/ai/
- **IBM Watson**
  - Developer Cloud
    - https://www.ibm.com/watson/developercloud/
    - https://github.com/watson-developer-cloud/python-sdk
  - Mashups
    - https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?subtype=SP&infotype=PM&htmlfid=LBS03048USEN&attachment=LBS03048USEN.PDF

Analytics-Based APIs – cont'd

- AWS
  - Boto3
    - low-level ("client") and high-level ("resource") APIs for all AWS products
    - https://aws.amazon.com/sdk-for-python/
  - API Explorer
    - https://developers.google.com/apis-explorer/#search/analytics/analytics/v3/

- Azure
  - Code samples, Cognitive Services API, etc.
    - https://docs.microsoft.com/en-us/python/azure/?view=azure-python
  - Python API Browser
    - https://docs.microsoft.com/en-au/python/api/?view=azure-python

Machine Vision APIs

- use cases:
  - autonomous vehicles
  - industrial control & QA
  - face recognition
  - number plate recognition
  - biometric identity verification
  - print & handwriting transcription
  - image annotation
    - detecting and labelling objects or themes in an image

Creating APIs

- Why would a data scientist/engineer want to create their own API?
  - for building an interface to your data product
  - for enforcing control over how your application's data and services can be used
  - for isolating the IP that your data product is based on

- References:
  - https://www.fullstackpython.com/application-programming-interfaces.html

Discussion

More Open Data APIs

- List of Open APIs (Wikipedia)
  https://en.wikipedia.org/wiki/List_of_open_APIs
- List of Open Data APIs (Programmable Web)
  https://www.programmableweb.com/category/open-data/api
- todmotto Public APIs
  https://github.com/toddmotto/public-apis

HOMEWORK

1. Investigate a data or analytic API for one of the following:
    - AWS
    - Microsoft Azure
    - IBM Cloud
2. Create a Jupyter notebook that demonstrates some basic operations (e.g. transporting, querying, or visualising data).

NOTES:

- The offerings of these platforms are myriad and complex. It may not be obvious which API you need to use at first, so try to start with published code examples.
- APIs (and the libraries that wrap them) change. Online examples may not work as documented.

Questions?

End of Presentation!