GENERAL ASSEMBLY

Intro to Data Engineering

Paul Singman
Data Engineer, Vroom

LEARNING OBJECTIVES

- Introduce some common DE technologies/concepts
- Give example of real world data pipeline
- Perform coding exercise using some best practices

What is Data Engineering?

- **Definition:** Moving data from Point A to Point B in a **Robust, Fault-Tolerant**, and **Reliable** manner.

Data Science Unicorn

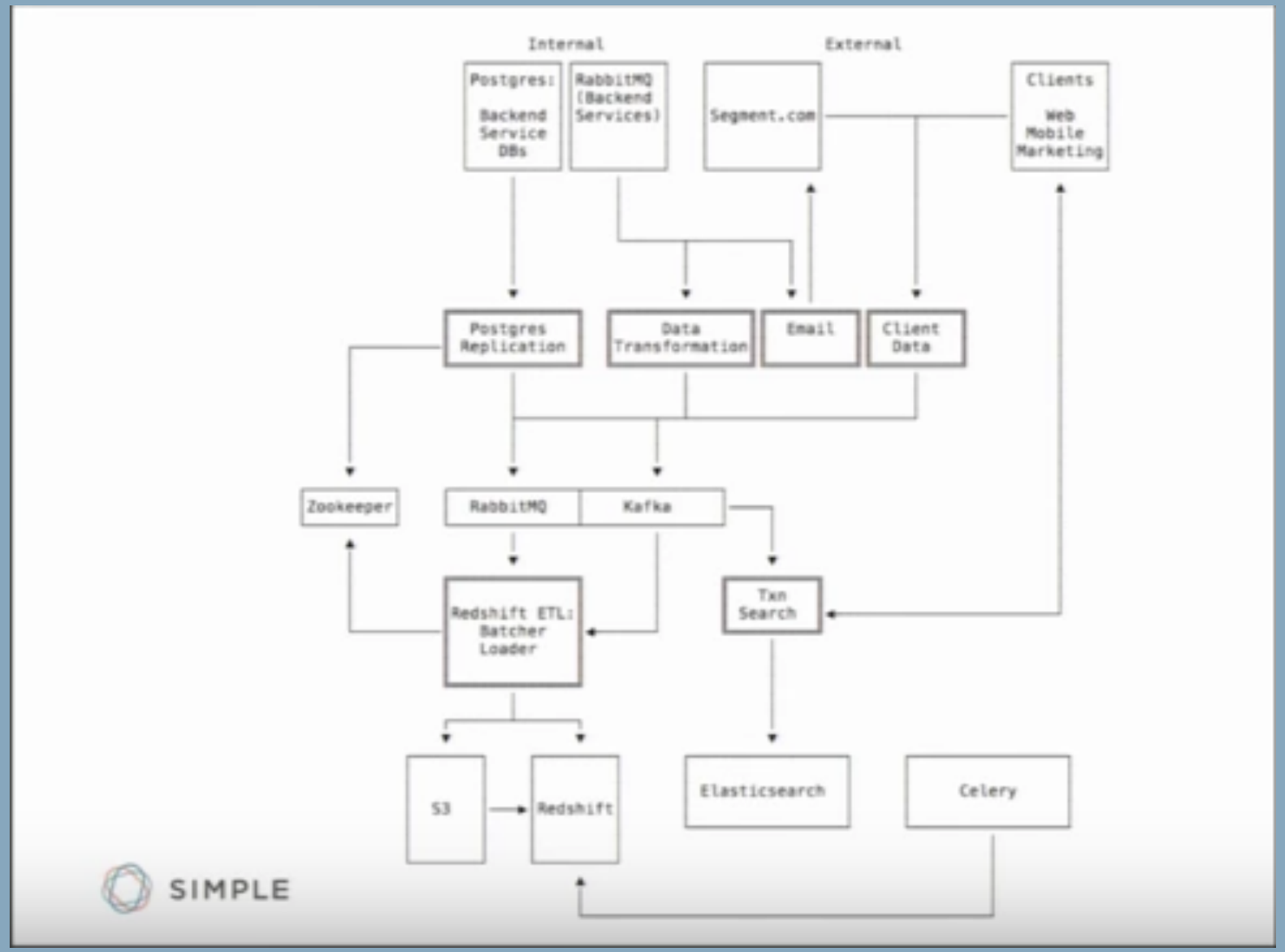
Robustness, Fault-Tolerance & Reliability

1) Gracefully handle unexpected inputs / schema changes
2) Gracefully handle failing or unresponsive nodes
3) Provide logging and metrics such as compute time or No. of bytes processed

Engineering Best Practices

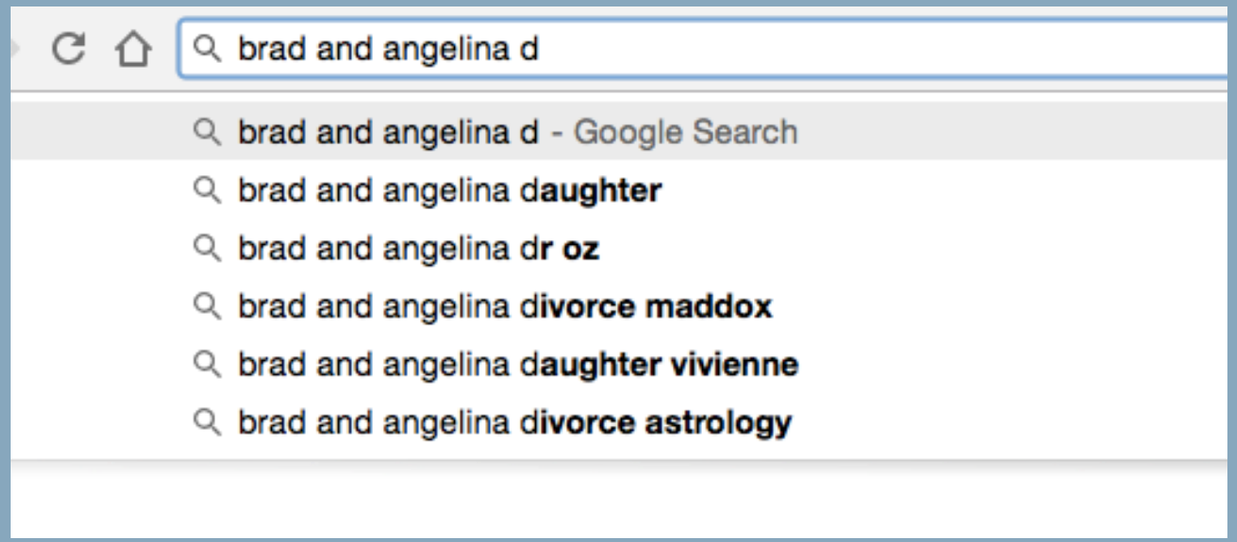
1) Operate in isolated development/production environments
2) Use version control for code (Git)
3) Write unit tests on code and automatically run tests when deploying new code (Continuous Integration)

Source: https://www.youtube.com/watch?v=9nX35zrN20E



Batch vs Streaming Pipelines

1) Historically companies had batch data pipelines (e.g. job runs once a day at 2:00 AM)
2) As technology improves, move towards streaming architectures (real-time processing)



Motivating Example: Google autocomplete suggestions

Hot DE Technologies

1) **Spark** (Berkeley) and **Flink**: Distributed Stream Computing

2) **Kafka** (LinkedIn) and **Kinesis** (AWS): Distributed Queue

3) **Airflow** (Airbnb) and **Luigi** (Spotify): Task Scheduling and Monitoring

4) Cloud Computing: Google Cloud Compute (AWS replacement)

5) Environment creation via containers: **Docker**

Lab Time!

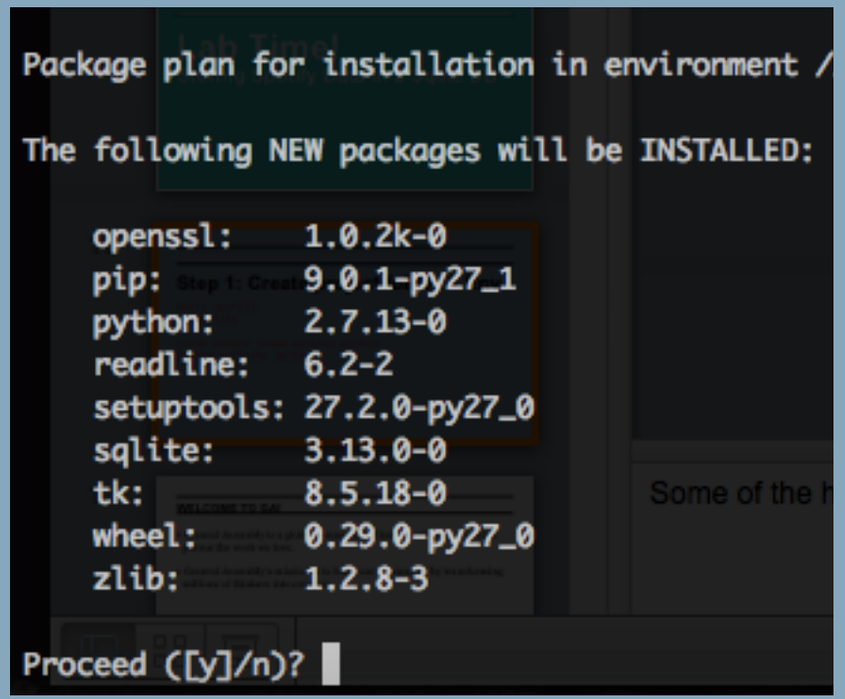
Storing Spotify Song Data in a Sqlite DB

Step 1: ???

Step 1: Create project folder & Env

```
Mkdir spotify
Cd spotify

Conda create –name spotenv python
Source activate spotenv
```



Step 2: Open Sublime and create python file get_data.py

```
touch get_data.py
vim get_data.py (to open file and edit in terminal)
```

For fun, type: `import pandas as pd`
Save and run: `python get_data.py`

```
(spotenv) Pauls-MacBook-Air:spotfiy paulsingman$ python get_data.py
Traceback (most recent call last):
  File "get_data.py", line 1, in <module>
    import pandas as pd
ImportError: No module named pandas
```

Step 3: Add to get_data.py

get_data.py | create_db.py

```python
import json
import requests
import sqlite3


def fetch_album_songs(album_id):

    pass


def store_in_database(songs):

    pass


if __name__ == '__main__':

    songs = fetch_album_songs()

    store_in_database(songs)
```

Step 4: Plan fetch_album_songs

1) Build api endpoint url with album_id
2) Use requests to get API response
3) Convert response into json object
4) Parse response for relevant fields
5) Return parsed data

https://open.spotify.com/special/thebeatles

https://developer.spotify.com/web-api/get-albums-tracks/

Step 6: code to fetch_album_songs

1) Build custom url with album_id
2) Use requests to get API response
3) Convert response into json object
4) Parse response for relevant fields
5) Return parsed data

```python
def fetch_album_songs(album_id):

    url = URL_BASE + 'albums/' + album_id + '/tracks'  # 1

    response = requests.get(url)  # 2

    json_response = json.loads(response.text)  # 3

    songs = []

    for item in json_response['items']:  # 4

        track_num = item['track_number']
        name = item['name']
        duration = item['duration_ms']

        songs.append((track_num, name, duration))

    return songs  # 5
```

{u'items': [{u'name': u"Sgt. Pepper's Lonely Hearts Club Band - Remastered", u'external_urls': {u'spotify': u'https://open.spotify.com/track/4fUKE8EULjQdHF

A look At the API Response

```
(spotenv) Pauls-MacBook-Air:spotfiy paulsingman$ python get_data.py
{u'name': u"Sgt. Pepper's Lonely Hearts Club Band - Remastered", u'external_urls': {u'spotify': u'https://open.spotify.com/track/4fUKE8EULjQdHF4zb0M8FO'},
u'uri': u'spotify:track:4fUKE8EULjQdHF4zb0M8FO', u'explicit': False, u'preview_url': u'https://p.scdn.co/mp3-preview/7ae81e104c9b55dfd0c203678d29a264801711
c6?cid=null', u'track_number': 1, u'disc_number': 1, u'href': u'https://api.spotify.com/v1/tracks/4fUKE8EULjQdHF4zb0M8FO', u'artists': [{u'name': u'The Bea
tles', u'external_urls': {u'spotify': u'https://open.spotify.com/artist/3WrFJ7ztbogyGnTHbHJFl2'}, u'uri': u'spotify:artist:3WrFJ7ztbogyGnTHbHJFl2', u'href'
: u'https://api.spotify.com/v1/artists/3WrFJ7ztbogyGnTHbHJFl2', u'type': u'artist', u'id': u'3WrFJ7ztbogyGnTHbHJFl2'}], u'duration_ms': 122893, u'type': u'
track', u'id': u'4fUKE8EULjQdHF4zb0M8FO', u'available_markets': [u'CA', u'MX', u'US']}
```

```
{u'name': u'With A Little Help From My Friends - Remastered', u'external_urls': {u'spotify': u'https://open.spotify.com/track/2RnPATK99oG0ZygnD2GT06'}, u'u
ri': u'spotify:track:2RnPATK99oG0ZygnD2GT06', u'explicit': False, u'preview_url': u'https://p.scdn.co/mp3-preview/2574cc919f6f70013598de262fc8c3d39b55fbbc?
cid=null', u'track_number': 2, u'disc_number': 1, u'href': u'https://api.spotify.com/v1/tracks/2RnPATK99oG0ZygnD2GT06', u'artists': [{u'name': u'The Beatle
s', u'external_urls': {u'spotify': u'https://open.spotify.com/artist/3WrFJ7ztbogyGnTHbHJFl2'}, u'uri': u'spotify:artist:3WrFJ7ztbogyGnTHbHJFl2', u'href': u
'https://api.spotify.com/v1/artists/3WrFJ7ztbogyGnTHbHJFl2', u'type': u'artist', u'id': u'3WrFJ7ztbogyGnTHbHJFl2'}], u'duration_ms': 164106, u'type': u'tra
ck', u'id': u'2RnPATK99oG0ZygnD2GT06', u'available_markets': [u'CA', u'MX', u'US']}
```

Step 7: code to store_in_database

```
def store_in_database(songs):

    connection = sqlite3.connect(DATABASE_NAME)

    cursor = connection.cursor()

    cursor.executemany("""INSERT INTO sgt_pepp_songs VALUES(?,?,?)""", songs)

    connection.commit()

    connection.close()
```

Full Script

get_data.py | create_db.py

```python
import json
import requests
import sqlite3

URL_BASE = 'https://api.spotify.com/v1/'
DATABASE_NAME = 'spotify.db'

def fetch_album_songs(album_id):

    url = URL_BASE + 'albums/' + album_id + '/tracks'  # 1

    response = requests.get(url)  # 2

    json_response = json.loads(response.text)  # 3

    songs = []

    for item in json_response['items']:  # 4

        track_num = item['track_number']
        name = item['name']
        duration = item['duration_ms']

        songs.append((track_num, name, duration))

    return songs  # 5


def store_in_database(songs):

    connection = sqlite3.connect(DATABASE_NAME)

    cursor = connection.cursor()

    cursor.executemany("""INSERT INTO sgt_pepp_songs VALUES(?,?,?)""", songs)

    connection.commit()

    connection.close()


if __name__ == '__main__':

    songs = fetch_album_songs('6QaVfG1pHYl1z15ZxkvVDW')

    store_in_database(songs)
```

Step 8: Create Database in create_db.py

create_db.py get_data.py

```python
import sqlite3

with sqlite3.connect('spotify.db') as connection:

    c = connection.cursor()

    c.execute("""CREATE TABLE sgt_pepp_songs
                (track_num INT,
                name TEXT UNIQUE,
                duration INT)
                """)
```

Run: python create_db.py

Step 8: Create Database in create_db.py

create_db.py get_data.py

```python
import sqlite3

with sqlite3.connect('spotify.db') as connection:

    c = connection.cursor()

    c.execute("""CREATE TABLE sgt_pepp_songs
                (track_num INT,
                name TEXT UNIQUE,
                duration INT)
                """)
```

Run: python create_db.py

Step 9: Query Away!

Q: Can you write a query that returns the name and duration of the second longest song?

Step 10 (Optional): Adding Command Line Args

```
43  if __name__ == '__main__':
44
45      parser = argparse.ArgumentParser(description='Specify an album id')
46      parser.add_argument('-a', '--album_id', required=True, help='Spotify Album id')
47      args = parser.parse_args()
48
49      songs = fetch_album_songs(args.album_id)
```

Run: python create_db.py –a 6QaVfG1pHYI1z15ZxkvVDW

Where to go from here

1) Check out Spotipy on Github for a generalized python wrapper for Spotify API

2) Use credentials to access all API endpoints

3) Add documentation and logging to the code

4) Get more Spotify data into a database!

EXIT TICKET

DON’T FORGET TO FILL OUT YOUR EXIT TICKET