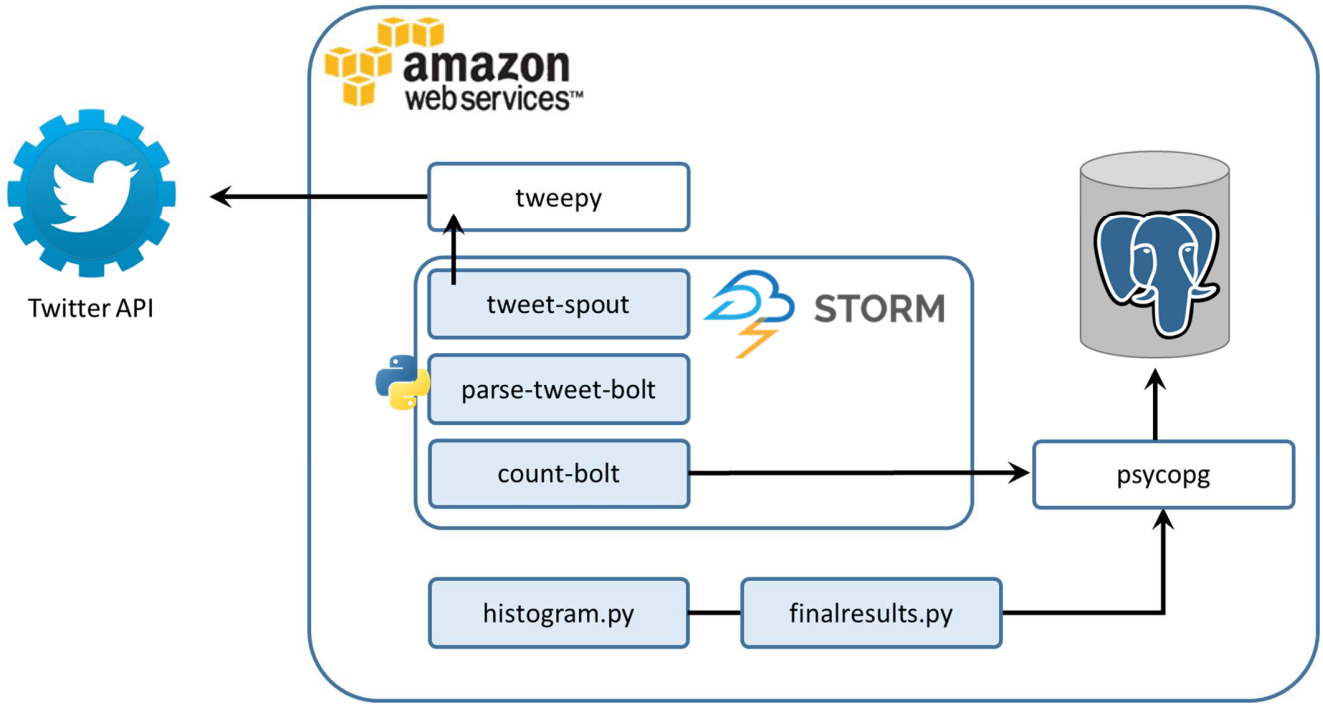
Exercise 2 W205

Overview of the Application

The purpose of the software is to access to an API provided by Twitter, select the tweets written in English with a simple algorithm (described later), parse all the words and populate a Postgresql table where to store the frequencies of the words retrieved from these tweets.
The components that retrieve the tweets, parse them and populate the table are managed by Apache Storm. The following picture sketches the architecture of the system.



All the components are written in Python (light blue rectangles in the picture).

Dependencies and Requirements

In order to work correctly, the software requires:

1. Version 2.7 of Python;
2. tweepy library, necessary to connect to the Twitter API through python.
   To install: `pip install streamparse`
3. streamparse libray, necessary to run the python code inside Apache storm.
   To install: `pip install tweepy`
4. pycopg library, necessary to connect to a postgresql db through python.
   To install: `pip install psycopg2`

Moreover, it's necessary a Twitter account and to create an application on it in order to obtain the grants to access the Twitter API (see https://apps.twitter.com for more details). The software is using my credentials.

Of course are also required Apache storm and Postgresql installed on the server.

A table Tweetwordcount where to store the words complete the requirements for the software. [CREATE TABLE Tweetwordcount(word TEXT PRIMARY KEY NOT NULL,count INT NOT NULL)]

File Structure

Following is the structure of the files for the project:

```
/root/lab
|   finalresults.py
|   hello-stream-twitter.py
|   histogram.py
|   psycopg-sample.py
|   Twittercredentials.py
|
└──ex2
    |   config.json
    |   fabfile.py
    |   project.clj
    |   README.md
    |   tasks.py
    |
    ├──src
    |   ├──bolts
    |   |       parse.py
    |   |       wordcount.py
    |   |       __init__.py
    |   |
    |   └──spouts
    |           tweets.py
    |           __init__.py
    |
    ├──topologies
    |       EX2Tweetwordcount.clj
    |
    └──virtualenvs
            EX2Tweetwordcount.txt
```

The core of the software is located inside the directory ex2.
In particular, the most important files are:

- **EX2Tweetwordcount.clj**: topology for the Apache storm application, with the definition of the spouts and bolts;
- **tweets.py**: spout that connects to the Twitter API through tweepy using Twitter credentials obtained from the website. The algorithm used to retrieve English tweets is very simple: it looks for tweets that contain the following words: "a", "the", "i", "you", "u".
- **parse.py**: bolt that parses the tweets dividing each word;
- **wordcount.py**: bolt that counts the words parsed by the previous bolt and update a Postgresql table using the library pycopg.
- **finalresults.py**: script that accepts a word as an argument and returns the total number of that word retrieved from Twitter. If used without argument, it returns a list of tuples (word, count) ordered alphabetically by words;
- **histogram.py**: script that accepts as argument “K1, K2” (2 numbers separated by a comma) and returns a list of tuples (word, count) ordered by count descending.

There are two files useful for testing the infrastructure:

- **hello-stream-twitter.py**: that tests the access to the Twitter API (the file **Twittercredentials.py** must be updated with the credentials obtained from Twitter);
- **psycopg-sample.py**: that tests the access to the Postgresql database (and it creates the database and the table if they are not present).

Run the software

To run the software that collects tweets from Twitter, we must change the working directory to the /root/lab/ex2/ and run the following command: `sparse run`.

Some small modifications I did to solve some problems:

- Increased the timeout for the connection to Twitter (in the spout file) to resolve some exceptions (3 seconds);
- Removed the dependencies to Redis (presents in the original template of the project)
- Renamed the name of the project from EX2Tweetwordcount to ex2 to solve a non-well defined exception that blocked the correct execution of the software.