Chuqing He

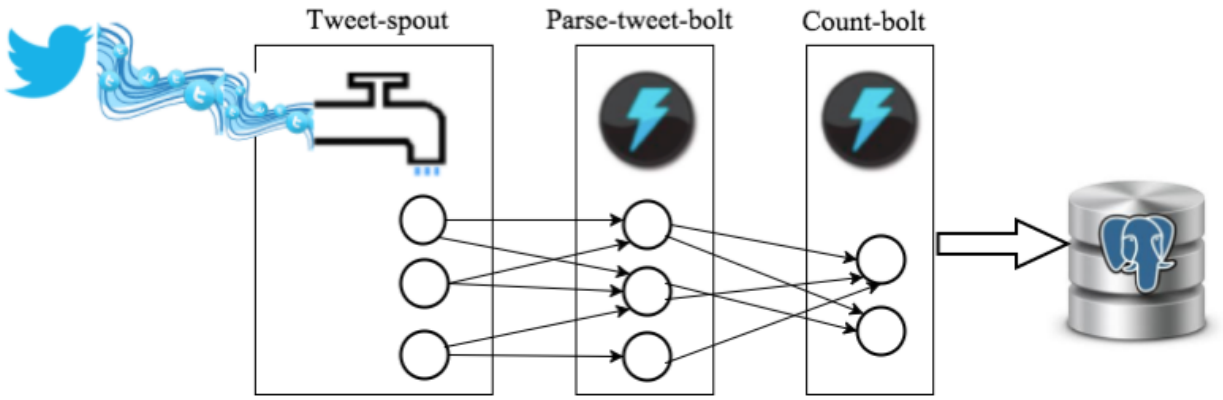
W205 Exercise 2 Architecture

This application captures and analyzes live twitter data to show people's interest at any given time. It uses a variety of tools to accomplish its purposes such as AWS EC2, AWS EBS, Apache Storm, Postgres, streamparse, Twitter API, Python, and Python libraries.

We will use an AWS EC2 instance to host our application and attach an EBS volumn to our instance for storage. For data persistence, we will use Postgres to persist and query the data. The Postgres table tweetwordcount has two fields, word and count. Word is of type Text and also acts as the primary key. Count is of Integer type and represents the number of times the Word has appeared so far. We must also setup a Twitter Application in order to get the necessary Authentication information.

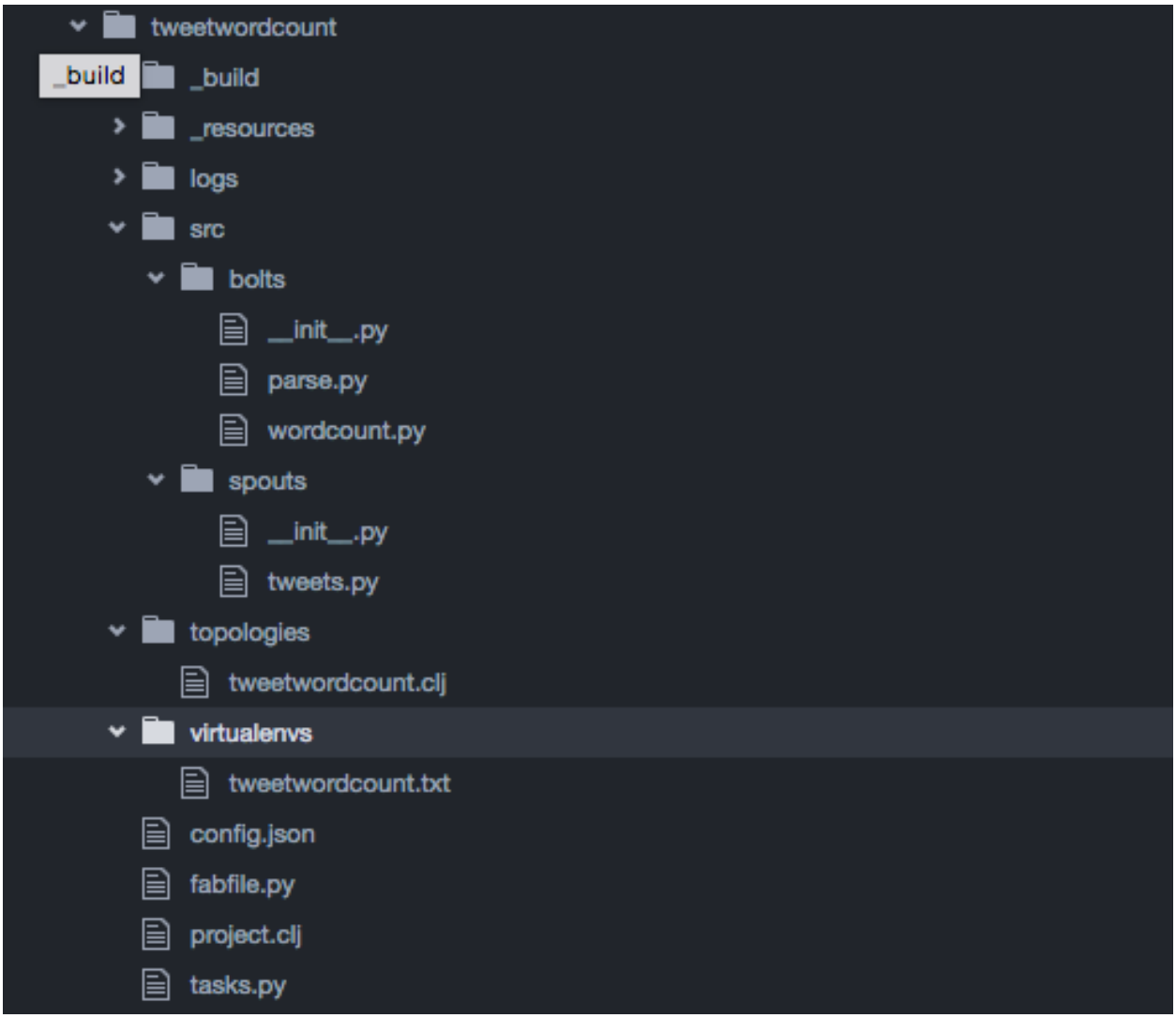
The main application will be a streamparse application written in python. We will need the psycopg2 library to connect our application to Postgres and Tweepy to talk to our Twitter Application. After those are correctly installed, we can start setting up the topology for our streamparse application. There is one spout and two bolts in this application. The Tweet spout takes in the necessary tweets and passes it to the ParseTweet bolt, which will remove the special characters and pass each word to the Count bolt. In the Count bolt, we will query the Postgres DB for the word; if the word already exists in the db, we will increment the count by 1,

otherwise we will insert a new entry for the word with a count of 1. The topology is illustrated below:



Finally, the finalresults.py and histogram.py provides the analytic capabilities. Finalresults.py will enable the user to query for any word in the Postgres DB and find out the corresponding count. If no word is passed in as parameter, Finalresults.py will display all the words and counts in the database. Histogram.py provides the capability to find words that have count within a range k1 and k2.

The folder structure for our TweetCount application is as follows:


tweetwordcount
_build
_build
_resources
logs
src
bolts
__init__.py
parse.py
wordcount.py
spouts
__init__.py
tweets.py
topologies
tweetwordcount.clj
virtualenvs
tweetwordcount.txt
config.json
fabfile.py
project.clj
tasks.py