Directory Structure:

```
TwitterStormApp
├── extweetwordcount
│   ├── _build
│   │   ├── classes
│   │   │   └── META-INF
│   │   │       └── maven
│   │   │           └── extweetwordcount
│   │   │               └── extweetwordcount
│   │   │                   └── pom.properties
│   │   └── stale
│   │       └── leiningen.core.classpath.extract-native-dependencies
│   ├── config.json
│   ├── fabfile.py
│   ├── logs
│   ├── project.clj
│   ├── README.md
│   ├── _resources
│   │   └── resources
│   │       ├── bolts
│   │       │   ├── __init__.py
│   │       │   ├── parse.py
│   │       │   └── wordcount.py
│   │       └── spouts
│   │           ├── __init__.py
│   │           └── tweets.py
│   ├── src
│   │   ├── bolts
│   │   │   ├── __init__.py
│   │   │   ├── parse.py
│   │   │   └── wordcount.py
│   │   └── spouts
│   │       ├── __init__.py
│   │       └── tweets.py
│   ├── tasks.py
│   ├── topologies
│   │   └── tweetwordcount.clj
│   └── virtualenvs
│       └── wordcount.txt
├── finalresults.py
├── hello-stream-twitter.py
├── histogram.py
├── makePostgres.py
├── psycopg-sample.py
├── screenshots
│   └── placeholder.txt
├── Twittercredentials.py
└── Twittercredentials.pyc
```

Application Idea:

On a high level, what this application does is consume tweets from the twitter stream API, then consume and analyze them in real time using Apache Storm. The spout code initializes a tweet stream, applies some basic filters, and then emits each tweet in the stream. The parse tweet bolt consumes these tweets, applies some logic, and them emits valid words over to the count bolt, which performs queries on our postgres tweetwordcount table to increment the count on each word it sees. All of the spout and bolt operations occur asynchronously, so there is a constant mix of tweets, words, and counts being sent from one entity to the next.

Description of the Architecture:

This is already discussed and diagrammed in the lab document, but basically the twitter stream is consumed by the spout, which emits to the parsing bolt, which then emits words to the count bolt, which increments word counts in the database table. The tweetwordcount.clj topology file defines which spout/bolt a bolt receives from, along with the data being emitted and the stream grouping. In our application the parsetweet bolt uses a shuffle grouping which sends tuples to bolts in random, round robin sequencing.

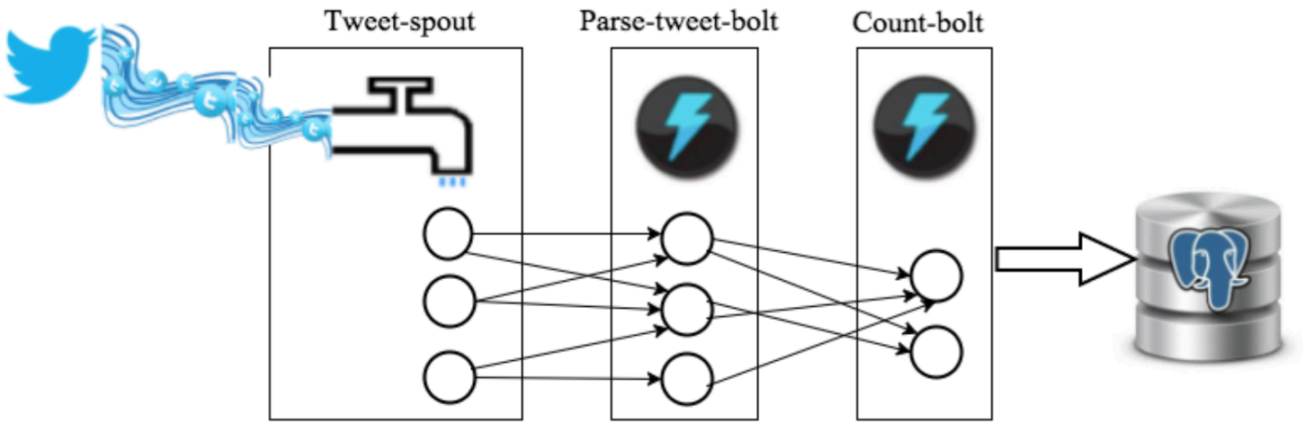

Figure 1: Application Topology

File Dependencies

Using this application requires python 2.7, postgres, and streamparse. The spout also requires one's Twitter API credentials in order to consume the tweet stream.