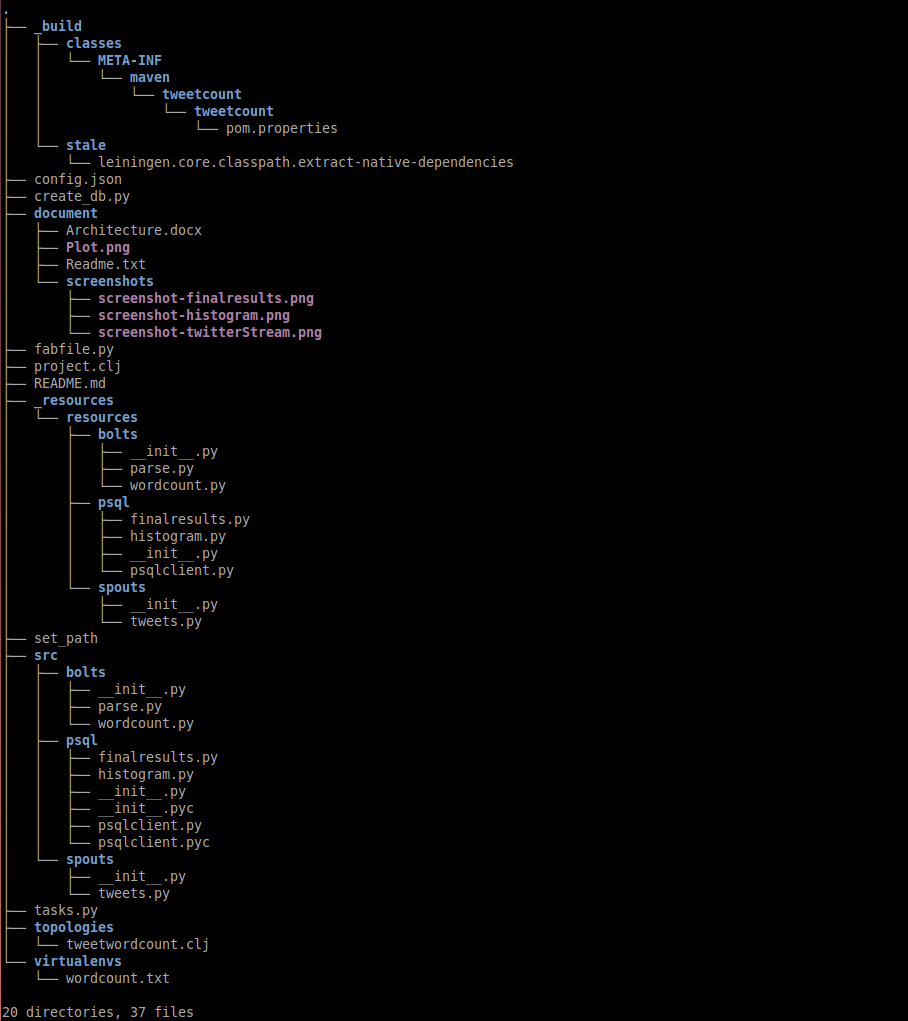
Project Architecture

# Project Hierarchy:



# Design Overview:

This is a pretty simple twitter word count app based on the Storm framework. The python script in **tweetwordcount/src/spouts** directory will serve as the data source for the app. The scripts in **tweetwordcount/src/bolts** directory will analyze the input stream, filter out duplicate tweets and invalid characters, and count the occurrence of each unique word. The scripts in **tweetwordcount/src/psql** directory will interact with the PostgreSQL database, and provide some basic query functionalities to the user. The **tweetwordcount/document** directory stores the required documentation and pictures.

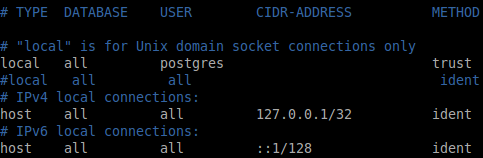
# System Requirements:

You should have the following packages pre-installed:

* psql (PostgreSQL) 8.4.20
* psycopg2 (2.6.2)
* tweepy (3.5.0)
* streamparse (2.1.4)

Besides, your PostgreSQL server should have the default superuser “postgres” and the default database “template1”. I emphasize this because I will use the db “template1” as the enter point to create the database “tcount”, and user “postgres” to interact with the PostgreSQL server through the entire project.

You might need to modify the default PostgreSQL client authentication configuration to grant the user “postgres” privilege to connect to the server locally. If the server refuses your connection, please modify your **/var/lib/pgsql/data/pg\_hba.conf** as following:



You can start the PostgreSQL server afterwards.



# File Descriptions:

This section will give a simple explanation to each new added code file or modified code file.

* **tweetwordcount/set\_path**:

This file will export the “$<current\_path>/src” path as a pythonpath to your environment. This pythonpath can enable python scripts within the “$<current\_path>/src” directory to recognize and import each other. **It is mandatory to source this file before running the app**.

* **tweetwordcount/create\_db.py**:

This script will create the database ‘”tcount” and the table “tweetwordcount” with the required schema. It is worth noticing that running this script will erase the existing database “tcount” if it exists, and create a fresh one. **It is mandatory to run this file before running the app, if your tcount database doesn’t already exist**.



* **tweetwordcount/src/psql/psqlclient.py**:

This script contains a class “PsqlClient”, which serves as a PostgreSQL connector, and provides several database manipulating and query APIs. The other parts of the app will talk to the PostgreSQL sever via this class. This design will improve the code reusability and separate the database access logic from the stream processing logic. You can see detailed explanation of each APIs inside the code file.

* **tweetwordcount/src/psql/finalresults.py**

This script will do what it should do as the exercise specification requires. This file will rely on **tweetwordcount/src/psql/psqlclient.py** to talk to the PostgreSQL database.

* **tweetwordcount/src/psql/histogram.py**

This script will do what it should do as the exercise specification requires. This file will rely on **tweetwordcount/src/psql/psqlclient.py** to talk to the PostgreSQL database.

* **tweetwordcount/src/spouts/tweets.py:**

Thispython script will stream the live English tweets from a twitter API. Compared with the original tweets.py, the only difference is that I add the twitter app credentials.

* **tweetwordcount/src/bolts/parse.py:**

This script parses the incoming tweet and divides it into separated words. On top of filtering out “hash”, “user mention”, “retweet”, and “url”, I convert all incoming tweets to lowercase only, because apparently “Trump” should be equivalent to “trump” when counting its occurrence. I also remove leading and trailing character which is neither digit nor letter.

* **tweetwordcount/src/bolts/wordcount.py:**

This script counts the occurrence of each incoming word. It will print the latest count to the console as well as store the count to the PostgreSQL database. It is worth noticing that, the count displayed on the console and the count stored in the database may be different even for the same word at the same time. It is because the count displayed on the console is only for the scope of the “current application execution”, while the count stored in the database also accounts for the “previous application execution”. This file will rely on **tweetwordcount/src/psql/psqlclient.py** to talk to the PostgreSQL database.