**Entire Code base(Git):**

<https://github.com/jainsourabh2/GrosvenorLocation>

**Twitter:**

**Twitter Data Load:**

<https://github.com/jainsourabh2/GrosvenorLocation/tree/master/twitter>

**Functional Implementations:**

Twitter Load is a continuous streaming process to fetch data from Twitter and put it into Apache Kafka.

**Technical Implementations:**

1. NodeJS
2. Apache Kafka
3. Hortonworks Data Platform
4. MySQL
5. Flume

**Scripts:**

***twitteruser.js***

This script converts the slug configuration of twitter into the screen names and persist into the mysql database. As the list of members returned across the slugs are duplicate, a deduplication process is executed before persisting the data into MySQL database.

The slug configuration is available at <<Base Directory>>/GrosvenorLocation/config/config.js

The extracted twitter handles are stored into twitterlist table under the screen\_name attribute.

***twitteridderivation.js***

This script converts the twitter handle derived in earlier process into the corresponding twitter id.

The derived twitter id is updated in the MySQL table twitterlist against twittered field.

Note: As there are Rate Limits applicable for the twitter’s user/show api , the script needs to be executed multiple times for the derivation of all the twitter ids.

***twitterstream.js***

This script initiates the stream from twitter. The filters applied for the stream is as mentioned below. The above 2 processes were needed in order to prepare for the 1st filter criteria.

List of Restaurant Twitter Handles in London: Available in MySQL table twitterlist

1. London geography coordinates: -0.5103, 51.2868, 0.3340, 51.6923
2. Keywords: Grosvenor, Mayfair, Belgravia

The script checks the content of the tweet against a configured list of keywords to categorize it as a possibly adult content.

The list containing adult words is available at below location:

<<Base Directory>>/GrosvenorLocation/config/abusivewords

Also sentiment gets derived based out of the tweets text. Finally, the tweet gets pushed into kafka topic ***newtwittercategoryflume.*** The derived sentiment is a score from 0 to 10.

For tweets that is not a geo tagged, a randomized tweet coordinate gets generated based on the place coordinate that gets associated with the tweet.

The log file is available at below location:

/var/log/GrosvenorLocation/twitterstream.log

***Notes:***

1. Twitter ends the streaming connection after approximately 7 hours. Hence a script is written to check if the twitterstream process is currently running or not every 1 minute. If the process is found not to be running, it restarts the process.

The script is available at below location:

<<Base Directory>>/GrosvenorLocation/checkRestartTwitterStream.sh

The log files get generated for this script is available at below location:

/var/log/GrosvenorLocation/shellRestartTwitterStream.log

1. The frequency of the process is set to 1 minute and is configured within the crontab.

**Flume**

The flume twitter agent is setup to configure the kafka topic ***newtwittercategoryflume.*** The flume hdfs sink is configured to write data into the hdfs path:

hdfs://sandbox.hortonworks.com:8020/grosvenor/twitter/newtwittercategoryflume/current

The rollInterval property for the hdfs sink is set to be 60 seconds i.e. every 60 seconds, data will be flushed into the hdfs file. A new file gets created when the data gets flushed.

**Merging Data**

The final step in the capturing of twitter data is to be merge the data into a single file. The file that gets created by the flume process is moved into a day wise folder. The day wise folder gets created on a daily basis if it doesn’t exist. Once the folder gets created, the file from flume process gets merged into that single file every 1 minute.

The script is available at below location:

<<Base Directory>>/GrosvenorLocation/moveFilesCategorize.sh

**Hive:**

Hive external table twittercategorystream is created on the data stored in Hadoop and is partitioned on date as we there are separate folders for each date.

**Partitions:**

A midnight script creates daily partition for the next day. The script is schedule in crontab to run at 11:30 PM every night. <<Base Directory>>/GrosvenorLocation/Hive\_Partition.sh

**Flickr:**

**Flickr Data Load:**

<https://github.com/jainsourabh2/GrosvenorLocation/tree/master/flickr>

**Functional Implementations:**

Flickr Load is a batch process to fetch data from Flickr and put it into Apache Hadoop (HDFS).

**Technical Implementations:**

1. NodeJS
2. Hortonworks Data Platform
3. Alasql

**Scripts:**

***flickrsample.js***

This script loads data from the different flickr api services and the data is stored into in-memory javascript database alasql. Once the data is loaded, all the tables are joined based on the photoid and the output is pushed into a file. The generated output is then pushed into a hdfs folder. A daily file gets created and the same is stored into Hadoop into a daily folder.

**Hive:**

Hive external table facebookdata is created on the data stored in Hadoop and is partitioned on date as we there are separate folders for each date.

**Partitions:**

A midnight script creates daily partition for the next day. The script is schedule in crontab to run at 11:30 PM every night. <<Base Directory>>/GrosvenorLocation/Hive\_Partition.sh

**Troubleshoot Flickr:**

* If the flickr data is not getting generated, check the crontab script if the process is scheduled.
* If the flickr data is not getting generated and the process is also scheduled, check the /var/log/GrosvenorLocation/flickrservice.log file for any errors.
* If the errors are related to access issues or api key expiry , update the new key in the configuration file <<Base Directory>>/GrosvenorLocation/config/config.js

**Facebook:**

**Facebook Data Load:**

<https://github.com/jainsourabh2/GrosvenorLocation/tree/master/facebook>

**Functional Implementations:**

Facebook Load is a batch process to fetch data from Facebook and put it into Apache Kafka.

**Technical Implementations:**

1. NodeJS
2. Apache Kafka
3. Hortonworks Data Platform
4. MySQL
5. Flume

**Scripts:**

***facebook.js***

The script derives the facebook handles and stores into mysql table “facebooklist”. This script derives local facebook business handles around the various tube stations for a particular type i.e. Restaurant, Café, Theatre etc. This type can be configured in the configuration file and script can be executed.

***facebookgraphapi.js***

This script reads the facebook handles from the facebooklist table and calls the facebook api to fetch all the posts and associated data. The fetched data is put into Apache Kafka against topic “**facebooktopic**”. The script also derives sentiments for the posts and tags if the message contains any abusive keywords. The list of configured abusive list is available at <<Base Directory>>/GrosvenorLocation/config/abusivewords

**Flume**

The flume facebook agent is setup to configure the kafka topic **facebooktopic*.*** The flume hdfs sink is configured to write data into the hdfs path:

hdfs://sandbox.hortonworks.com:8020/grosvenor/facebook/facebooktopic/

**Troubleshoot Facebook:**

* If the facebook data is not getting generated, check the crontab script if the facebook process is scheduled.
* If the facebook data is not getting generated and the process is also scheduled, check the /var/log/GrosvenorLocation/fbingestion/\*.log file for any errors.
* If the errors are related to access issues or api key expiry , update the new key in the configuration file <<Base Directory>>/GrosvenorLocation/config/config.js

**Troubleshooting:**

**Apache Drill:**

* If the Apache Drill queries are running slow or getting stuck.
  + Log into the server using root credentials.
  + Identify the drill process on the server using the below command

ps –eaf | grep –i drill

* + Note the processed from the above output and check its resident memory usage using the below command

top –p <<processid>>

* + If the memory used is too high, restart the drill process.
* Clearing the drill logs:
  + Visit the below path and remove files which are 2 days older than the current date

<<Apache Drill Home>>/log

* + Visit the below path and remove files which are 2 days older than the current date

<<Apache Drill Home>>/log/profiles

**Yarn Memory Usage:**

* If the Yarn Memory usage is too high, wait for some time as MapReduce generated application containers might be under execution.
* If the Yarn Memory usage is too high and no jobs are expected to run, visit the below Resource Manager UI (http://<<hostname>>:8088) and see if any applications are stuck. Kill them if it is not expected to be executing. Identify the cause of it by going through the application logs in event it is expected to run.

**Memory Usage:**

* Login to the server using root credentials.
* Shut down the services which are not required for the application from HortonWorks dashboard
* If the memory usage is too high, execute the below command from command prompt:

free && sync && echo 3 > /proc/sys/vm/drop\_caches && free

**HDFS Disk Usage:**

* Observe and clean the log files on daily basis from below folders and folders within which are 2 days older than the current date.
  + /var/log/
* If the hdfs disk usage if above 90% even after clearing all the data, you need to raise an alert to add more space to the cluster.