

**PROJECT TITLE:-**

**COLLECTION OF TWITTER DATA USING TWITTER STREAMING API’S AND STORE, ANALYZE, AND VISUALIZE TWEETS**

**SUBJECT:-** Principles of Bigdata Management

**PHASE-1**

**COLLECTION OF TWITTER DATA USING TWITTER API’S AND RUN WORDCOUNT ON THAT DATA**

1. Kantha Rao Patchava (16280105)
2. Premchand Lingamgunta (16279756)
3. Kranthi kiran Reddy Vanga (16274077)

Instructor

**Dr. PRAVEEN RAO, Ph.D.**

**ABSTRACT**

In this project collection of Twitter data by using Twitter Streaming API’s in CSV format. The collected data pushed into the HDFS and by using Python code Hashtags and URLs are extracted. Then Wordcount program is run on Extracted Hashtags and URLs files in both Hadoop and Spark.

**PROBLEM STATEMENT**: Data analysis is done by collecting data manually. This is a frantic work. Social networking websites is helping in analyzing data easily. But the problem is it is unstructured data.

**EXISTING SYSTEM:** In existing system we can store only structured data and low amount of data. In addition to this, it had scalability program.

**PROPOSED SYSTEM:** In this HDFS can store high amount of data like petabytes. It can store unstructured data and process data across the clusters not only in single node.

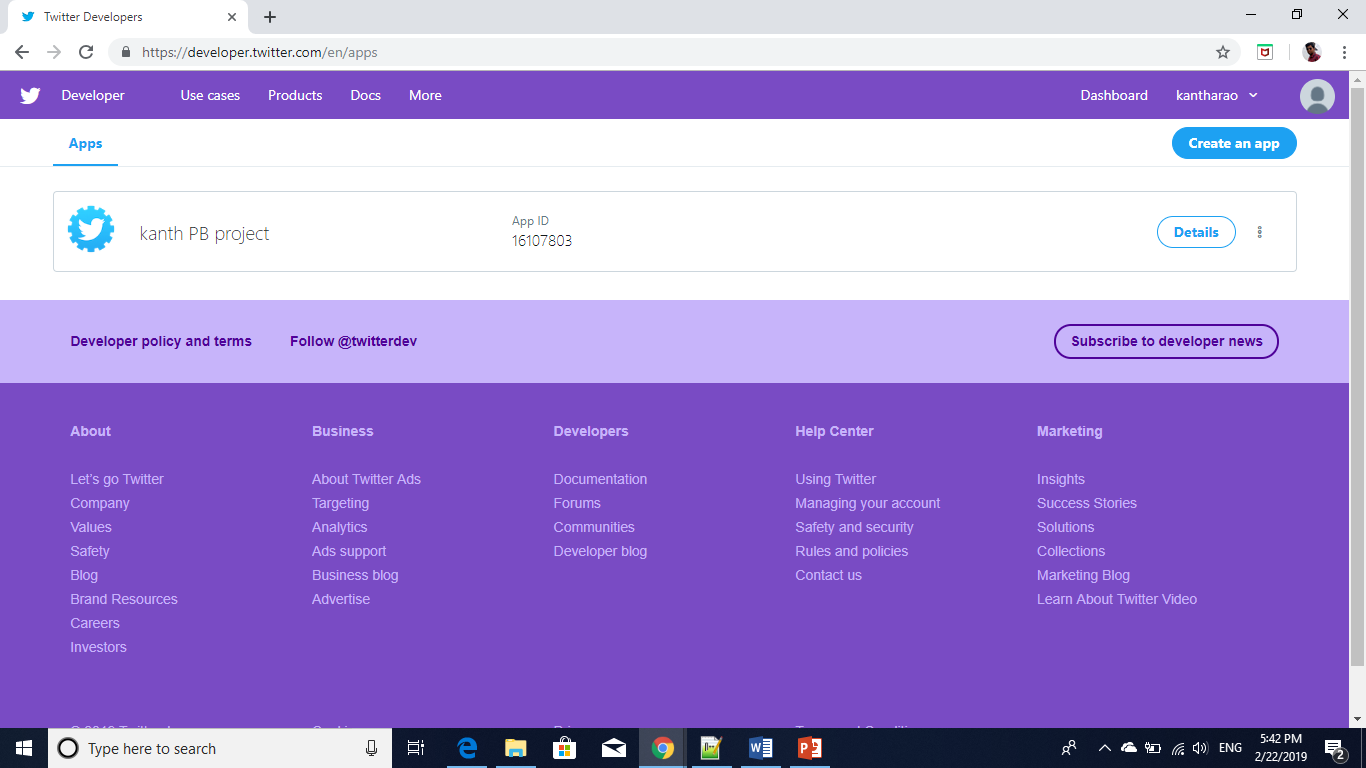
.

**TWITTER DEVELOPER SIGN UP FOR TWEETS**

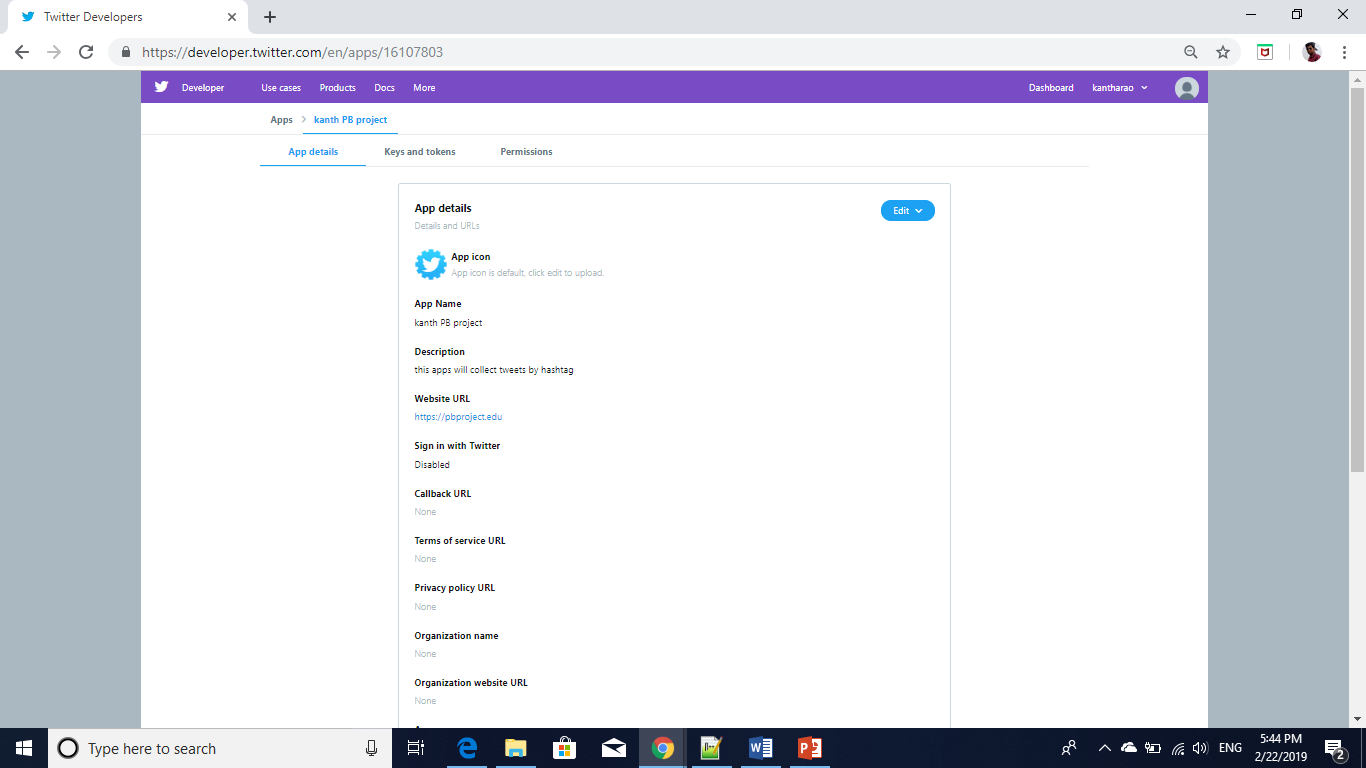
Before collection of tweets, it is important to sign in with twitter and join the documentation part with the following link :[https://dev.twitter.com/resources/signup.](https://dev.twitter.com/resources/signup.%20)

Create a Twitter Application with these we can create 4 keys – Consumer key, Consumer secret key, Access token and Access token secret which are later used for the collection of tweets

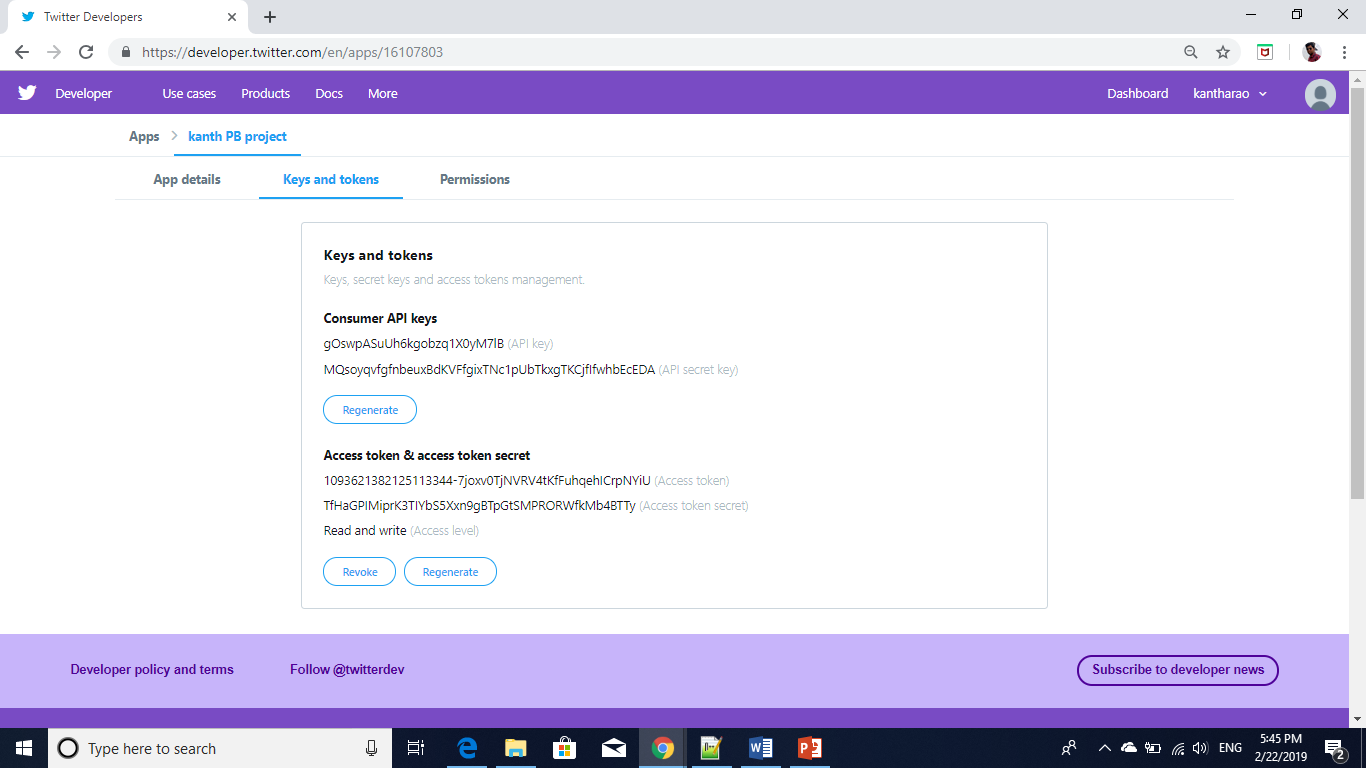
**Project creation**

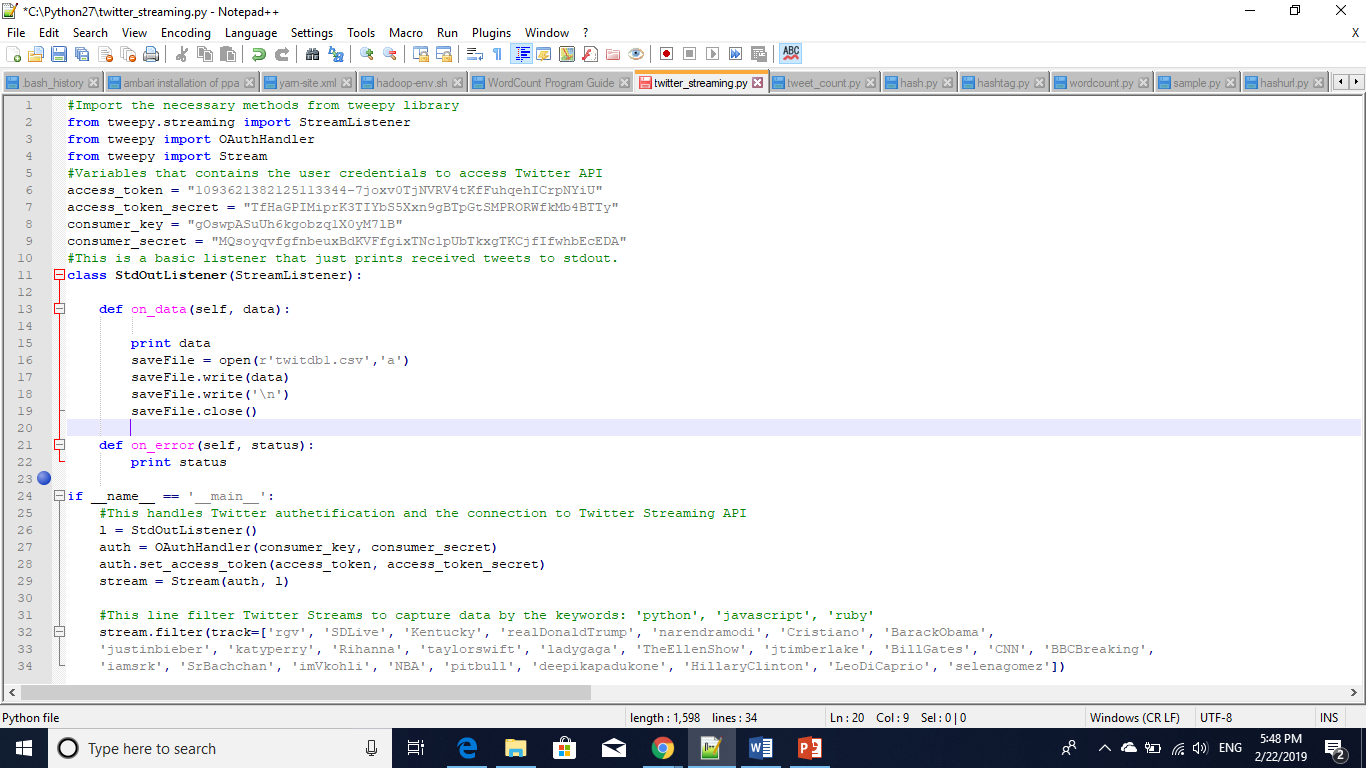


**App details**



**Keys and tokens**



**Code to collect tweets**

**Source Code: twitter\_straming.py**

#Import the necessary methods from tweepy library

from tweepy.streaming import StreamListener

from tweepy import OAuthHandler

from tweepy import Stream

#Variables that contains the user credentials to access Twitter API

access\_token = "1093621382125113344-7joxv0TjNVRV4tKfFuhqehICrpNYiU"

access\_token\_secret = "TfHaGPIMiprK3TIYbS5Xxn9gBTpGtSMPRORWfkMb4BTTy"

consumer\_key = "gOswpASuUh6kgobzq1X0yM7lB"

consumer\_secret = "MQsoyqvfgfnbeuxBdKVFfgixTNc1pUbTkxgTKCjfIfwhbEcEDA"

#This is a basic listener that just prints received tweets to stdout.

class StdOutListener(StreamListener):

def on\_data(self, data):

print data

saveFile = open(r'twitdb1.csv','a')

saveFile.write(data)

saveFile.write('\n')

saveFile.close()

def on\_error(self, status):

print status

if \_\_name\_\_ == '\_\_main\_\_':

#This handles Twitter authetification and the connection to Twitter Streaming API

l = StdOutListener()

auth = OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_token\_secret)

stream = Stream(auth, l)

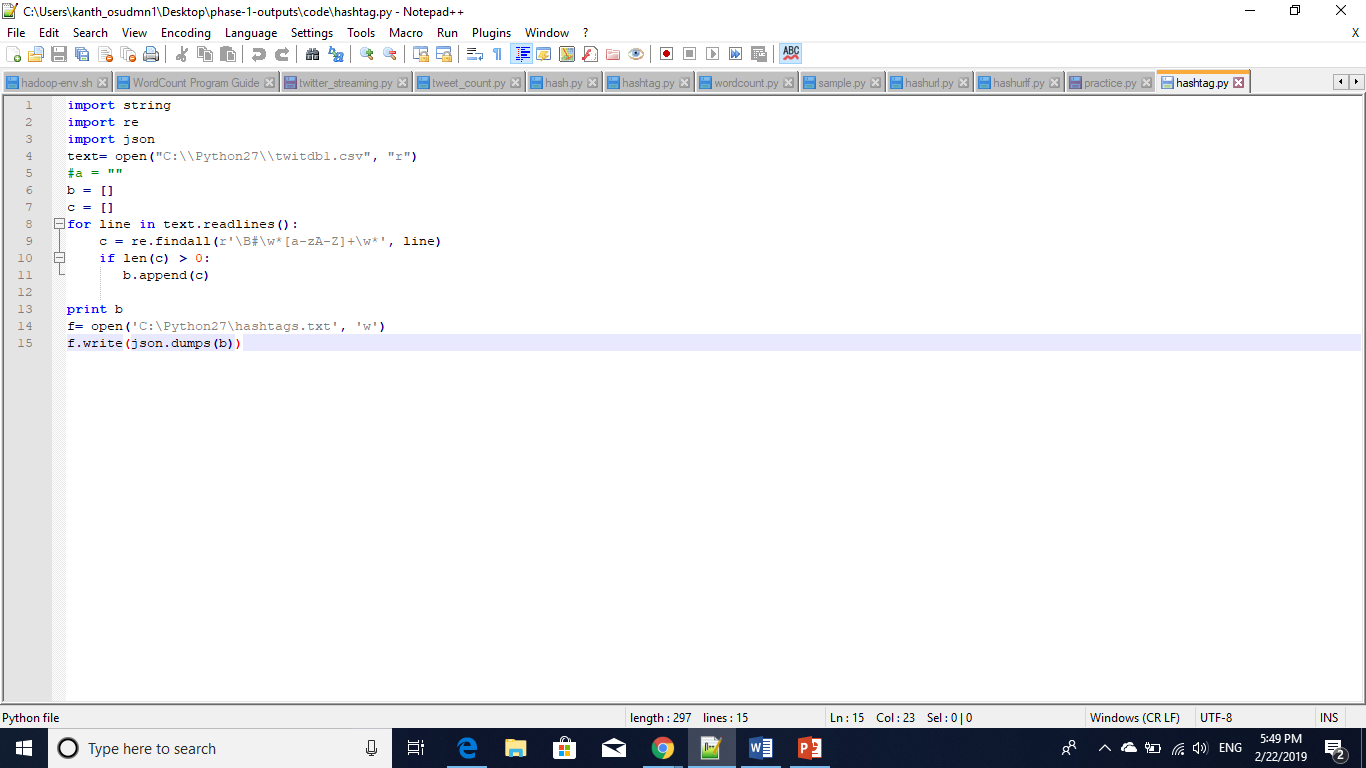
#This line filter Twitter Streams to capture data by the keywords:

stream.filter(track=['rgv', 'SDLive', 'Kentucky', 'realDonaldTrump', 'narendramodi', 'Cristiano', 'BarackObama',

'justinbieber', 'katyperry', 'Rihanna', 'taylorswift', 'ladygaga', 'TheEllenShow', 'jtimberlake', 'BillGates', 'CNN', 'BBCBreaking',

'iamsrk', 'SrBachchan', 'imVkohli', 'NBA', 'pitbull', 'deepikapadukone', 'HillaryClinton', 'LeoDiCaprio', 'selenagomez'])

**Code to extract hashtags**



**Source code: hashtag.py**

import string

import re

import json

text= open("C:\\Python27\\twitdb1.csv", "r")

#a = ""

b = []

c = []

for line in text.readlines():

c = re.findall(r'\B#\w\*[a-zA-Z]+\w\*', line)

if len(c) > 0:

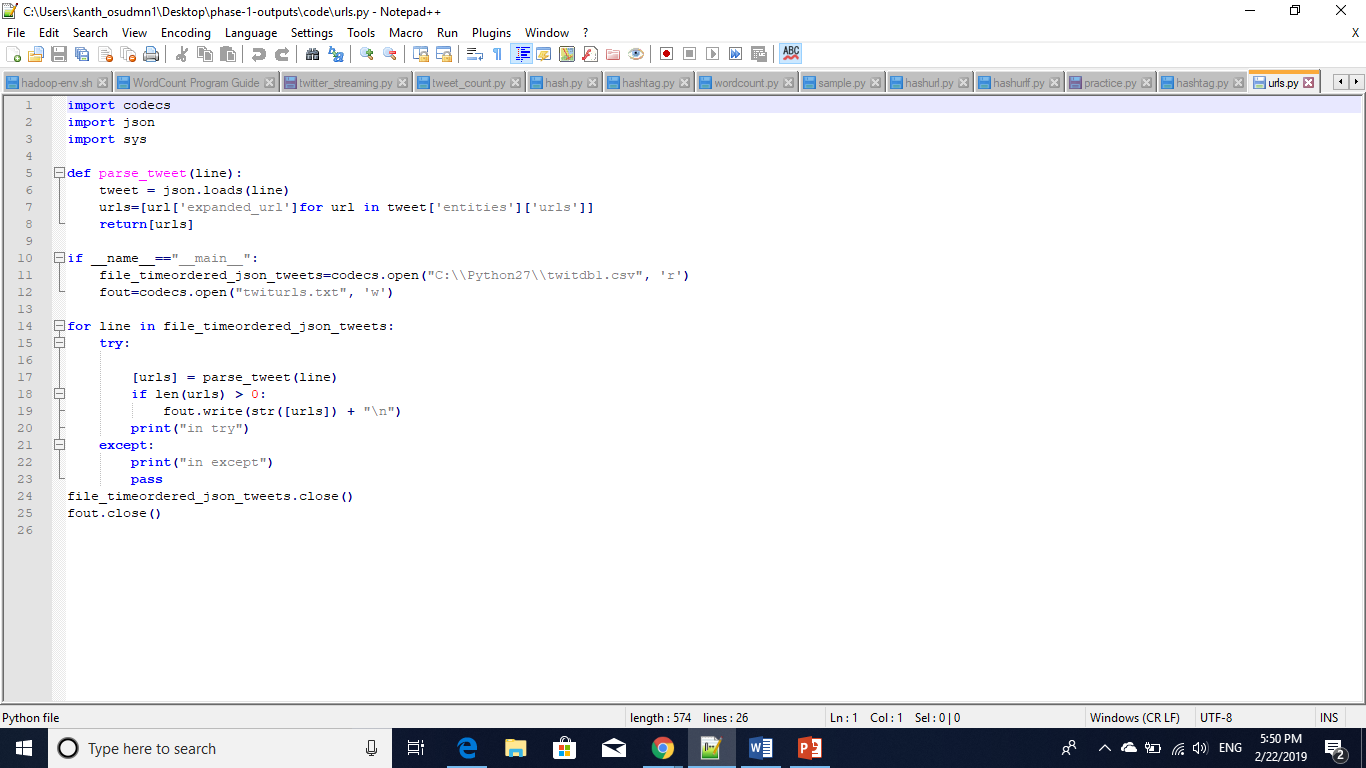
b.append(c)

print b

f= open('C:\Python27\hashtags.txt', 'w')

f.write(json.dumps(b))

**Code to extract urls: urls.py**



**Source code:** urls.py

import codecs

import json

import sys

def parse\_tweet(line):

tweet = json.loads(line)

urls=[url['expanded\_url']for url in tweet['entities']['urls']]

return[urls]

if \_\_name\_\_=="\_\_main\_\_":

file\_timeordered\_json\_tweets=codecs.open("C:\\Python27\\twitdb1.csv", 'r')

fout=codecs.open("twiturls.txt", 'w')

for line in file\_timeordered\_json\_tweets:

try:

[urls] = parse\_tweet(line)

if len(urls) > 0:

fout.write(str([urls]) + "\n")

print("in try")

except:

print("in except")

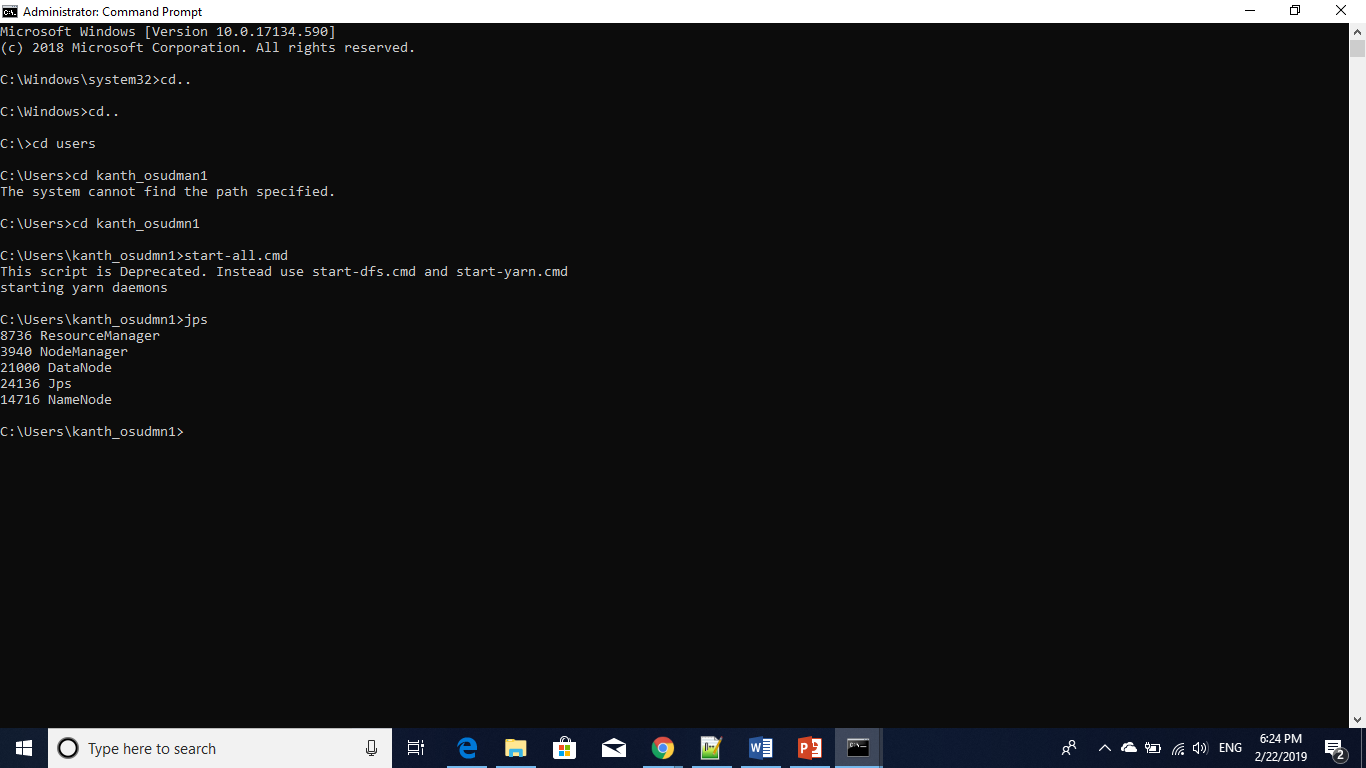
pass

file\_timeordered\_json\_tweets.close()

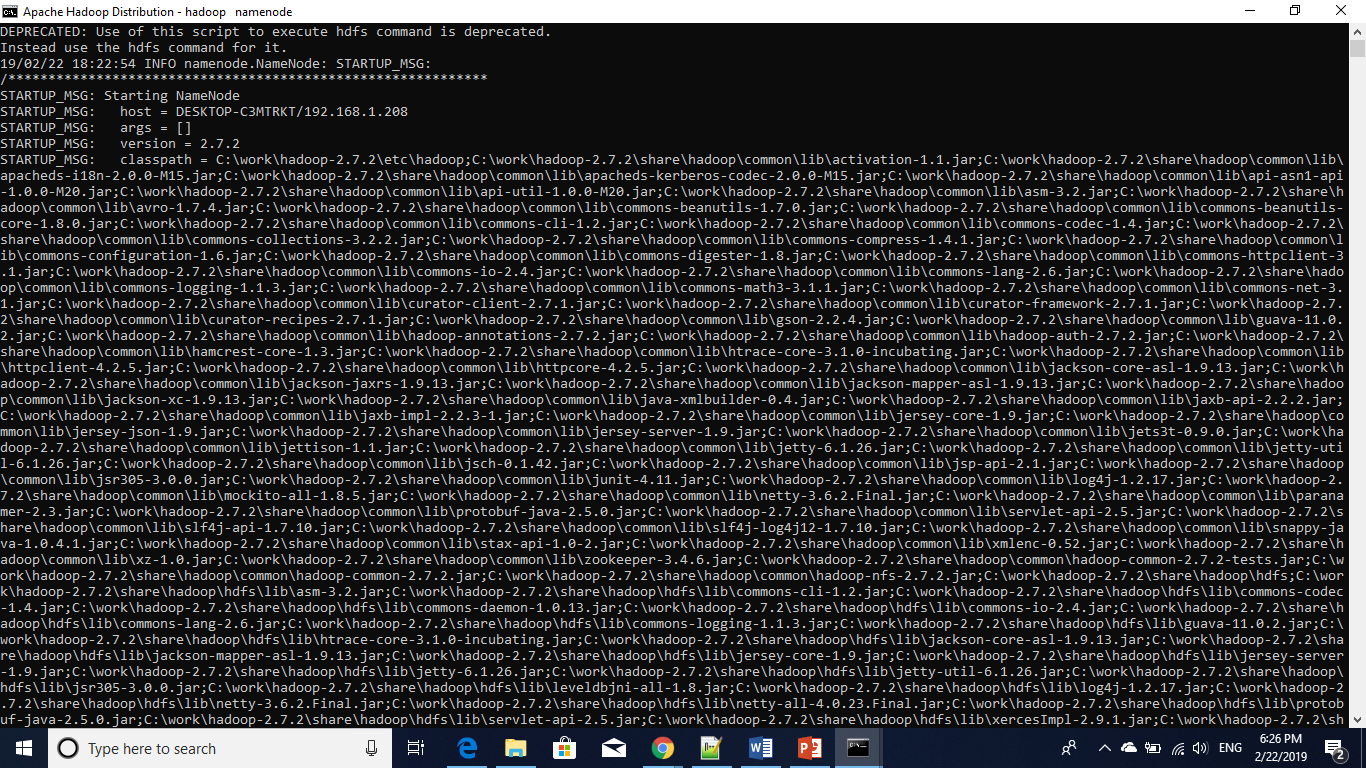
fout.close()

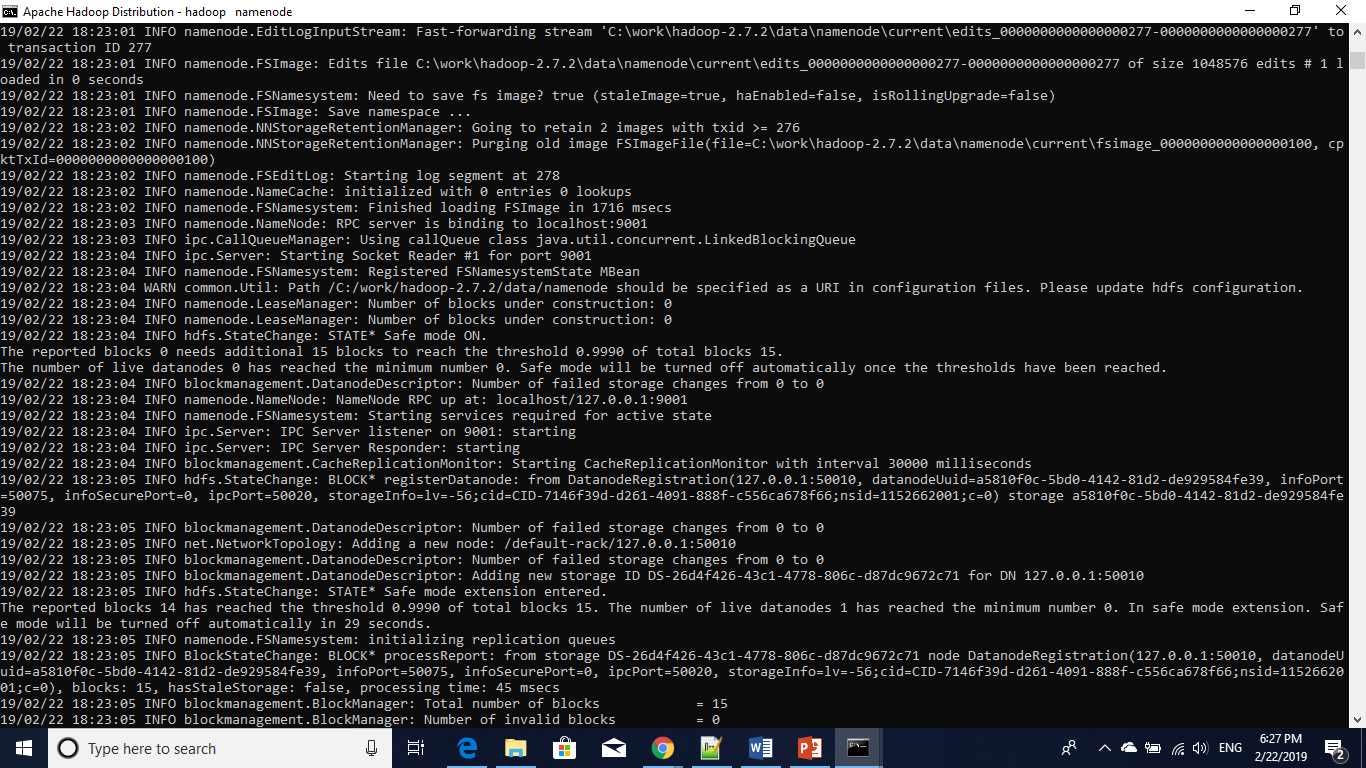
**output screens:**

**to start Hadoop**

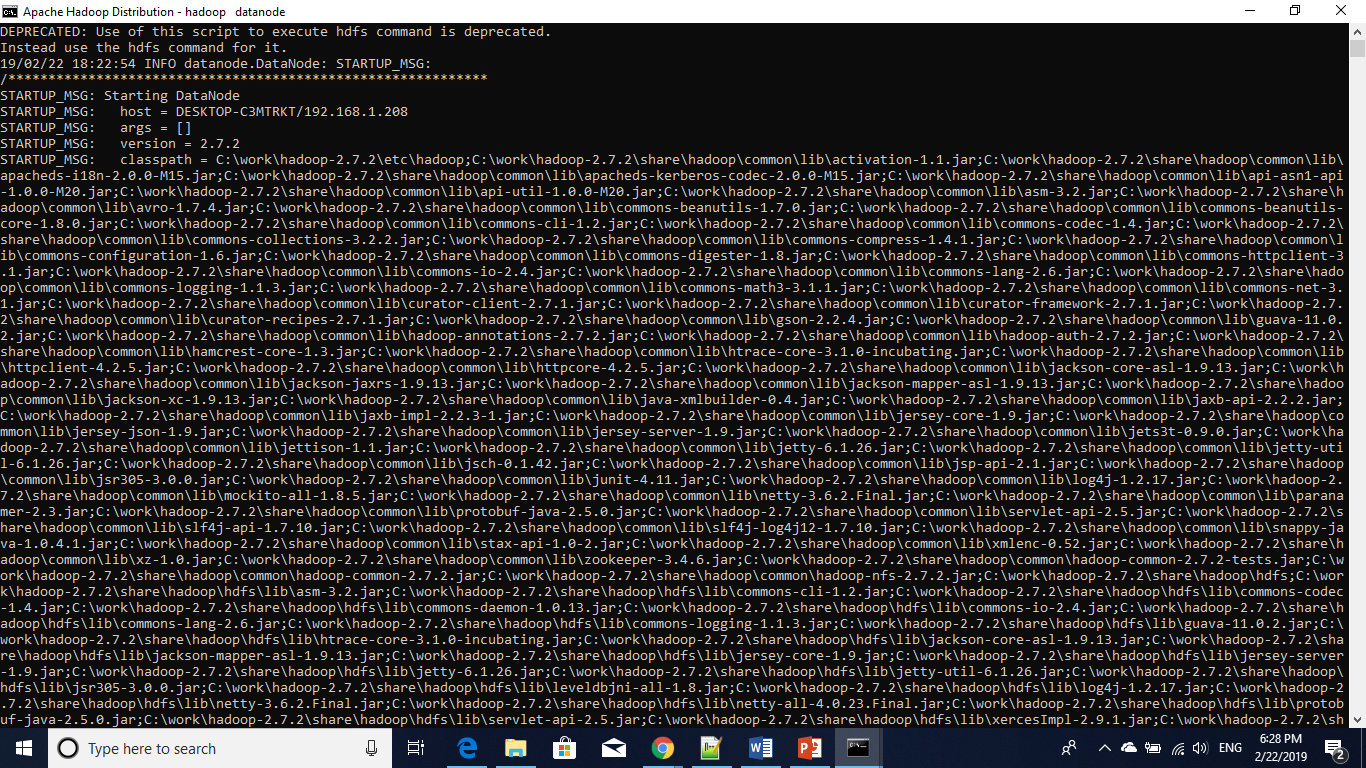


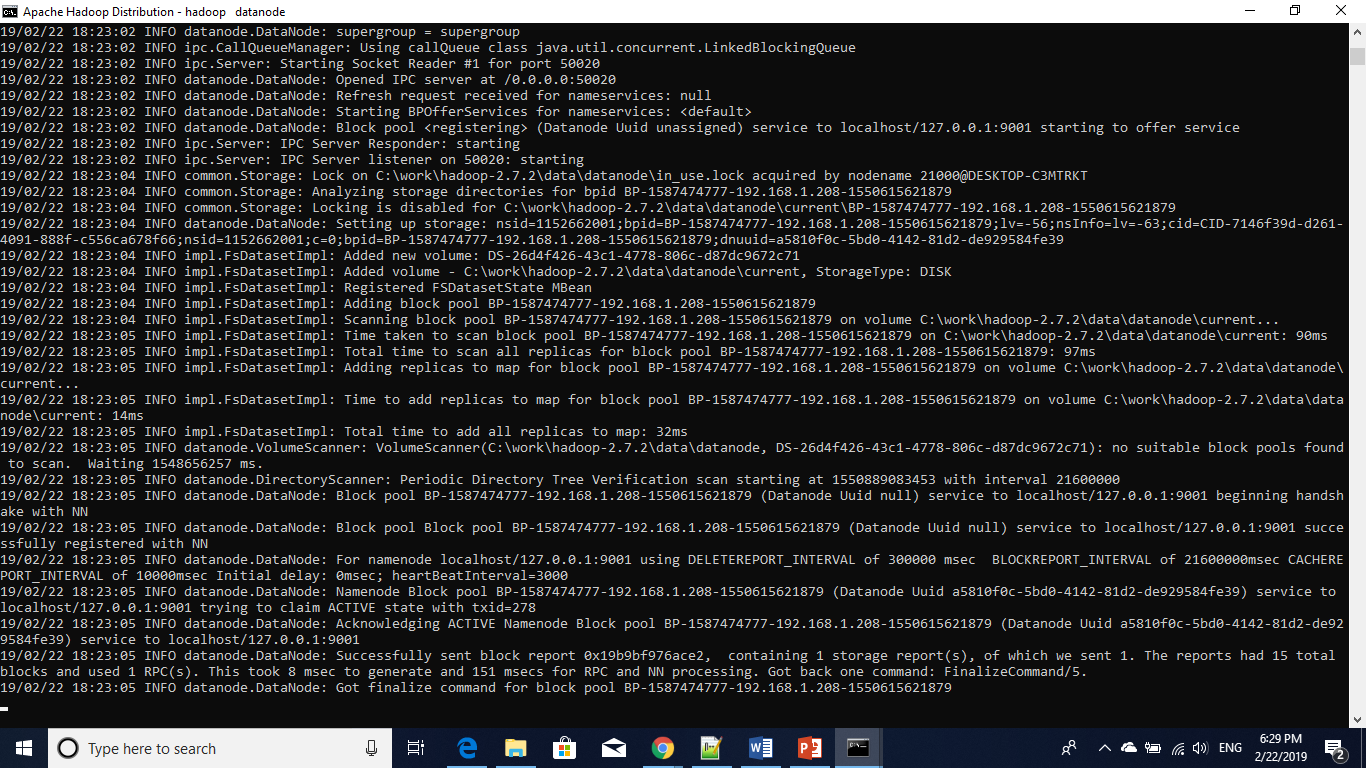
**Namenode starting windows:**



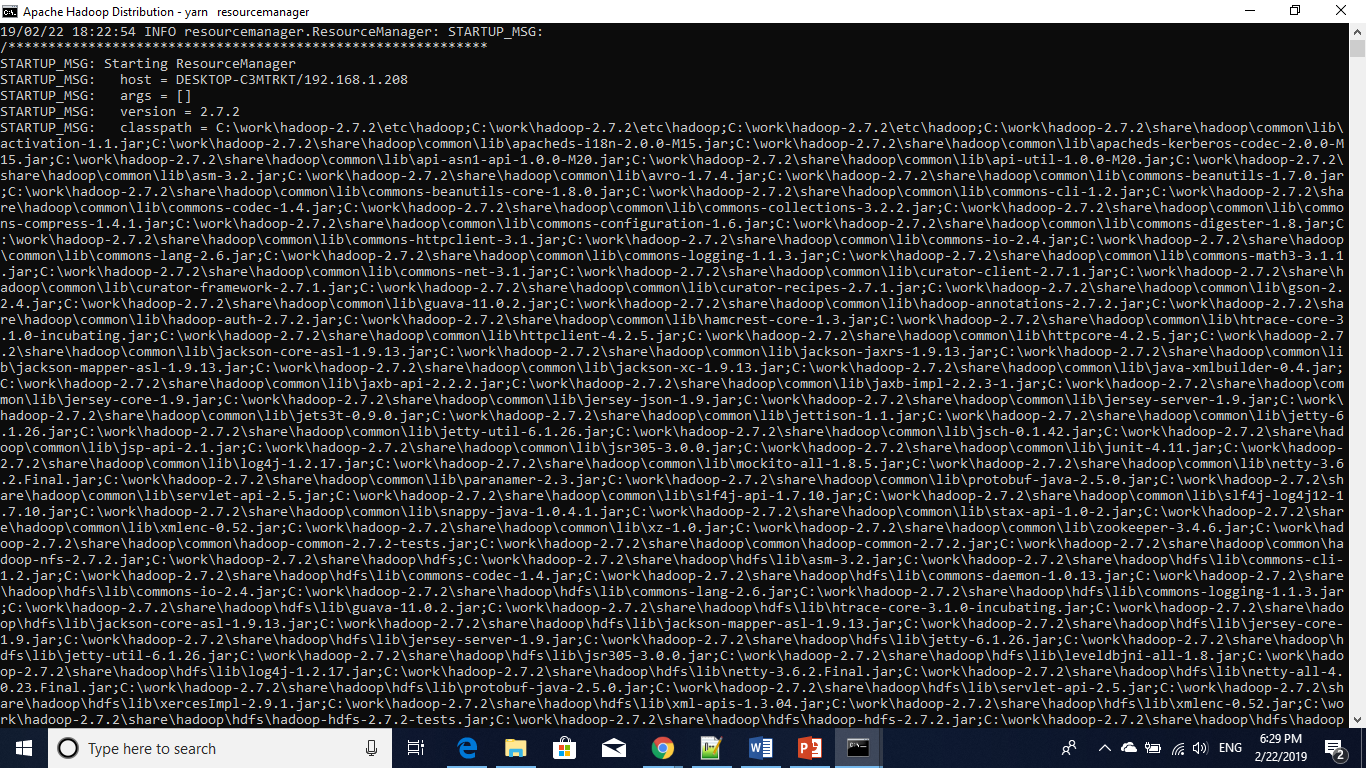


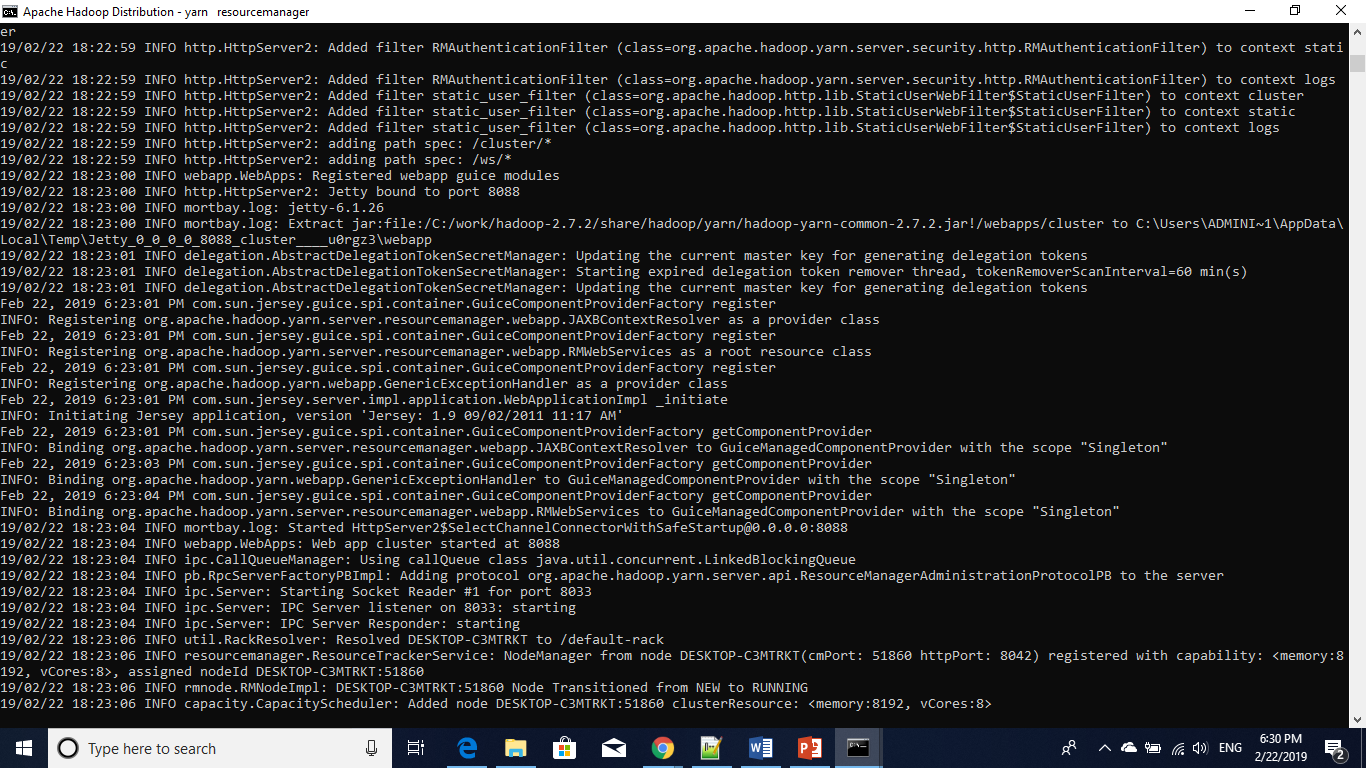
**Hadoop data node start windows**



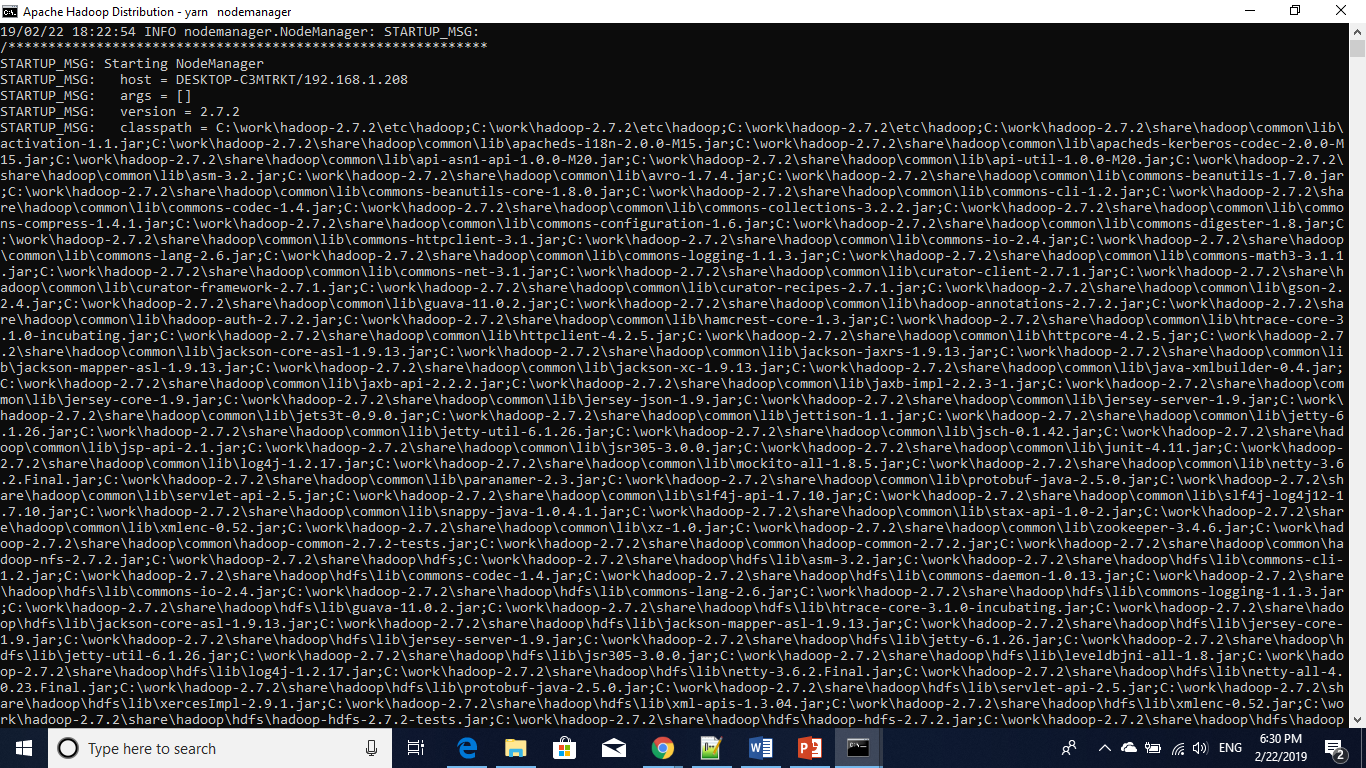


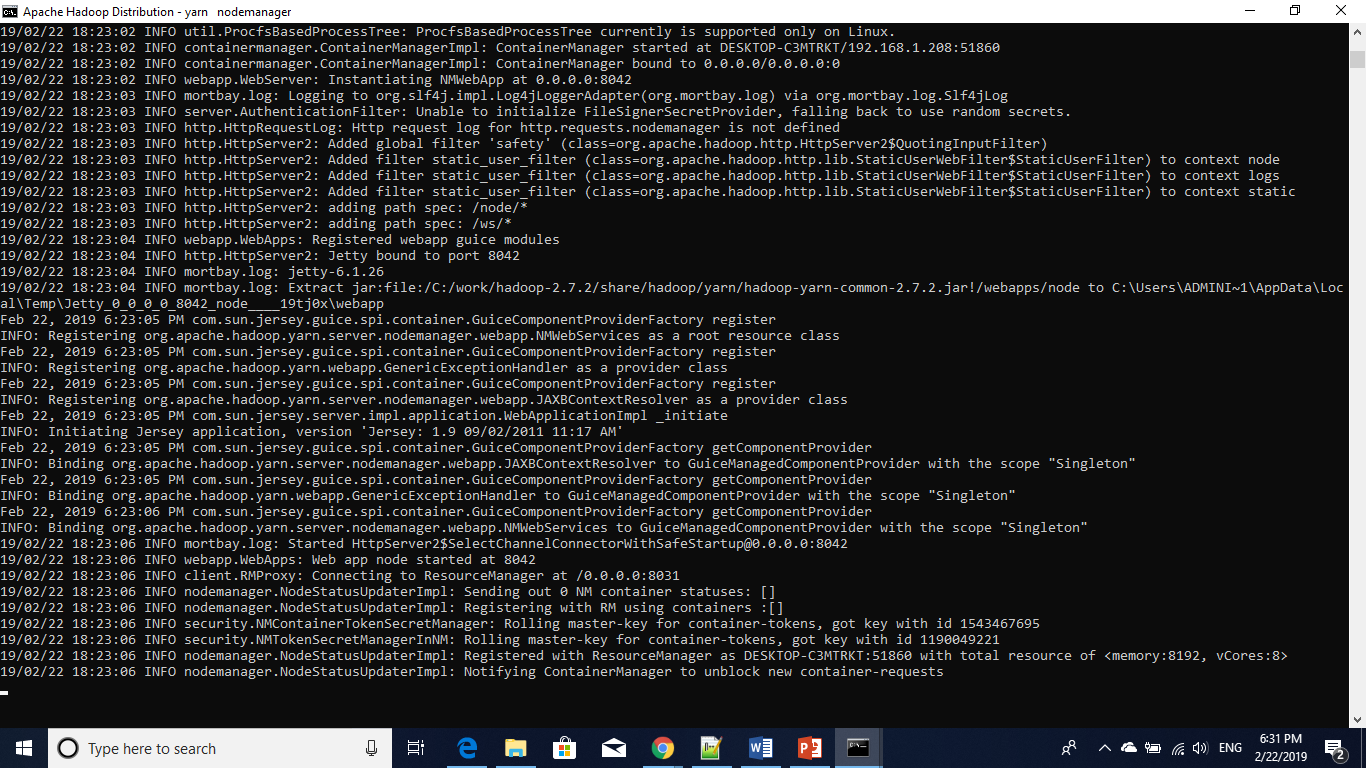
Resource manager starting window:



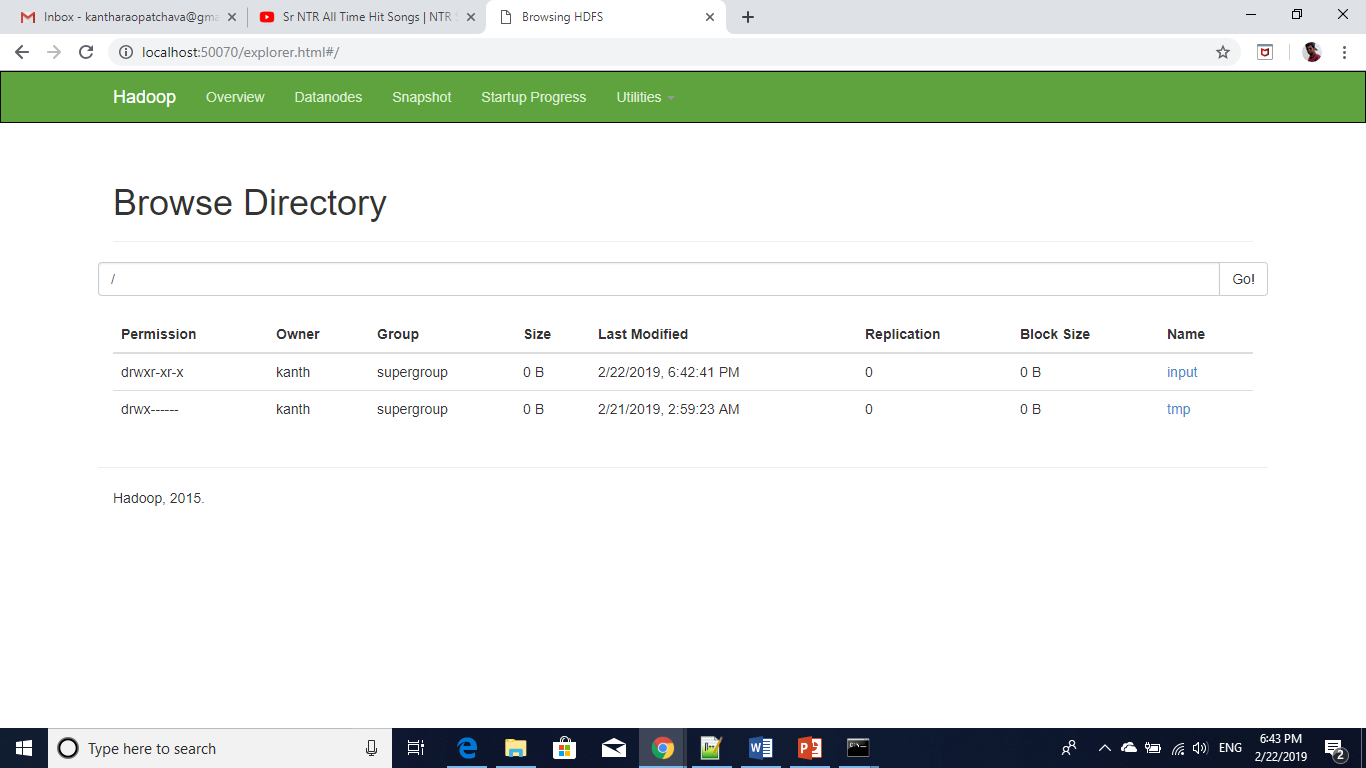


Nodemanager starting windows:

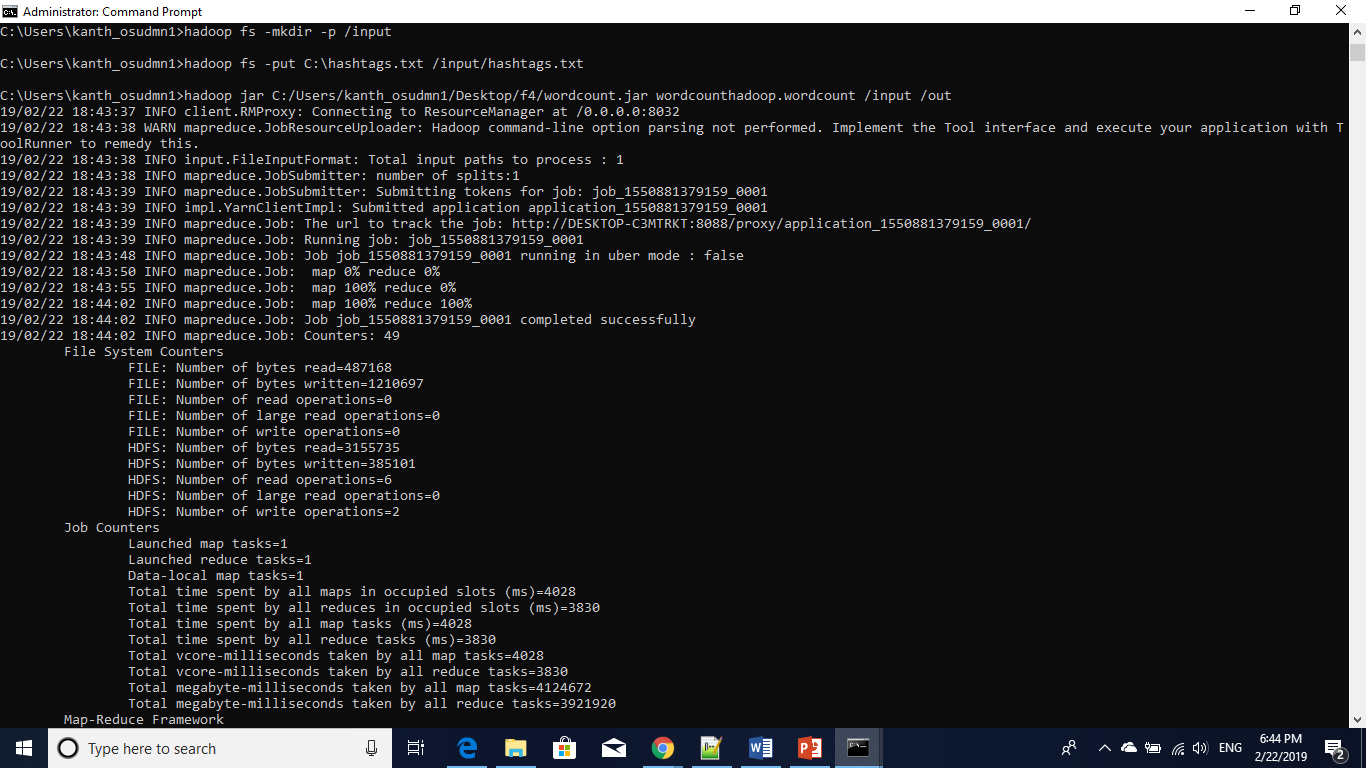




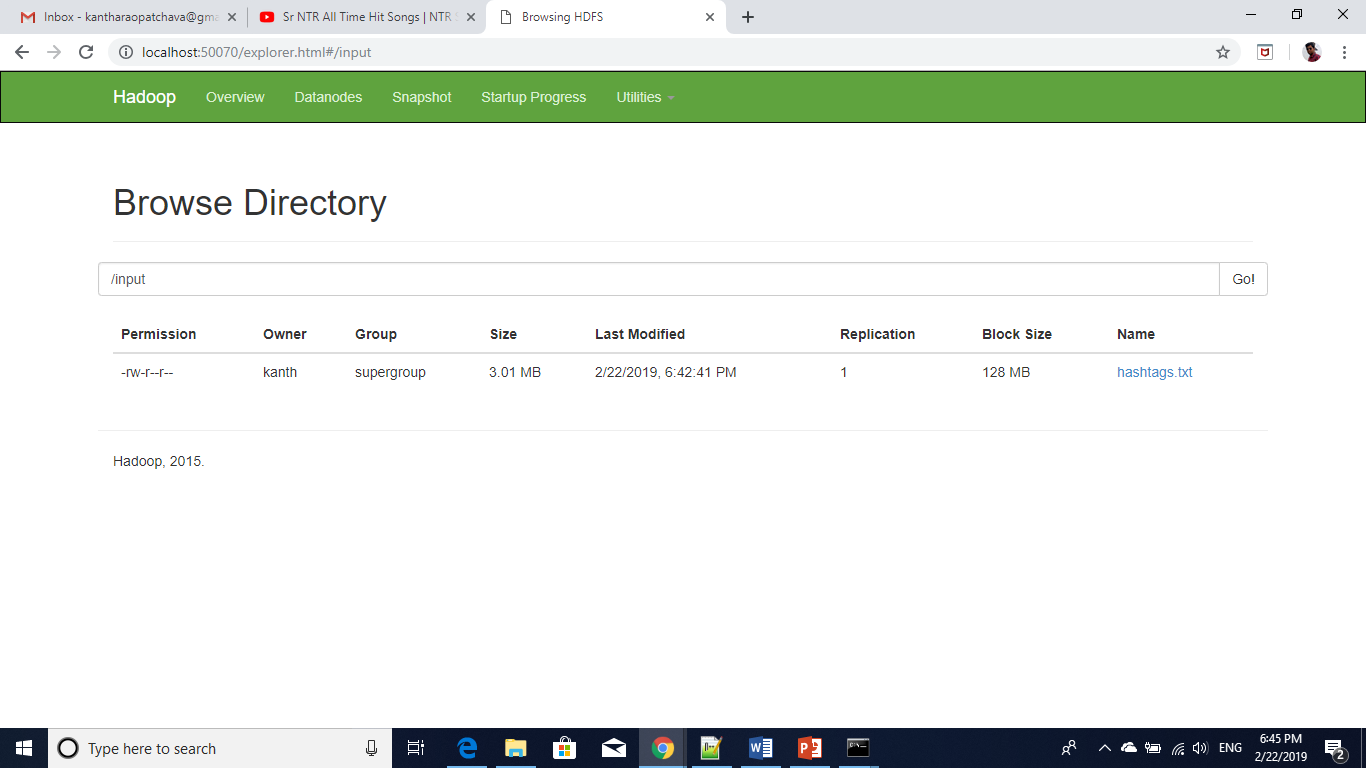
After input directory creation:



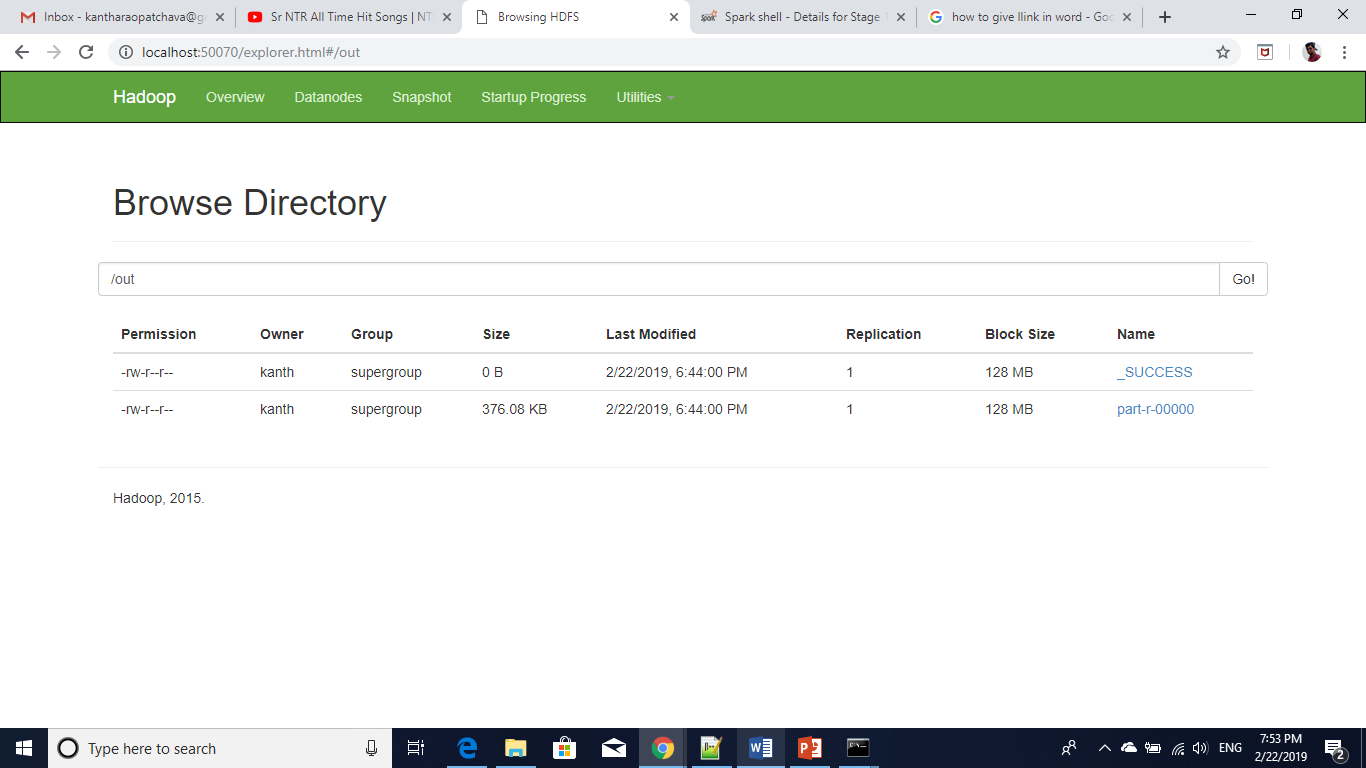
Mapreduce output on hashtahg:



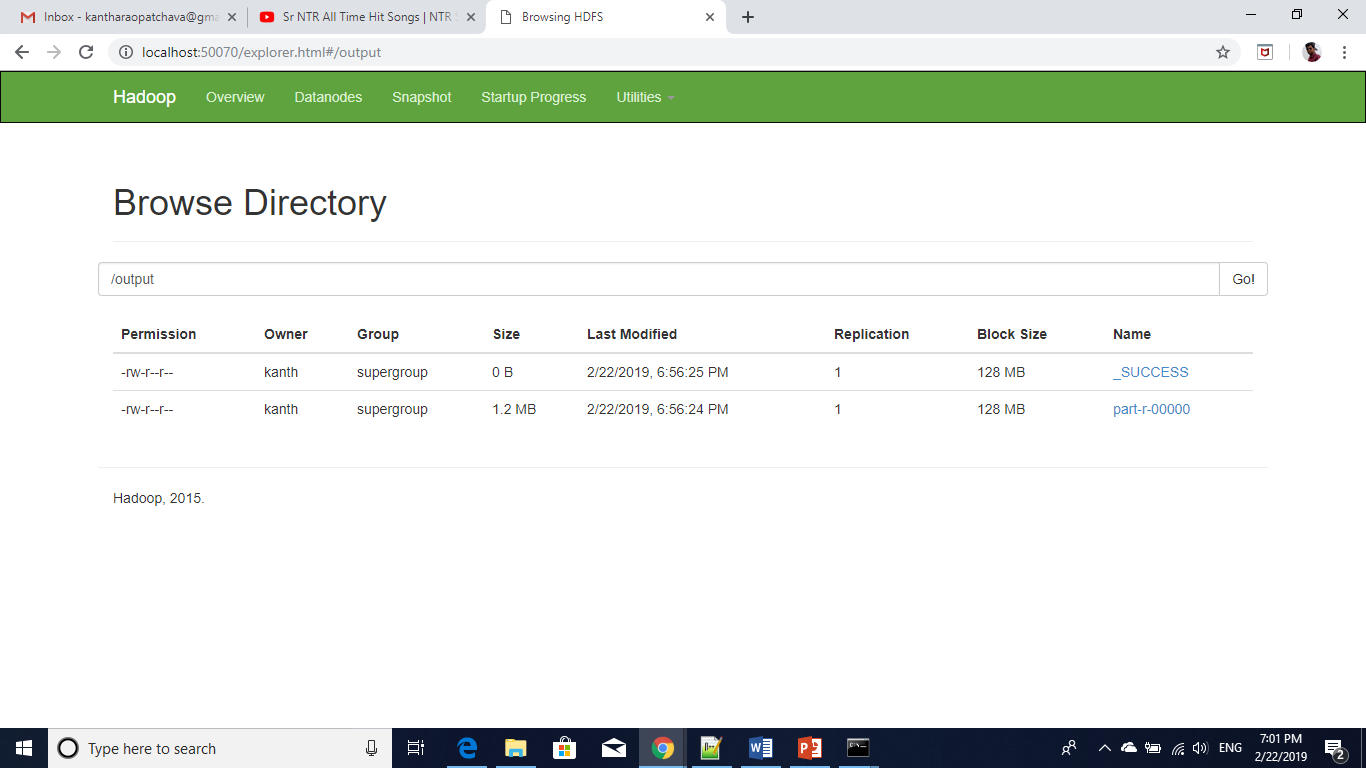
Input send to Hadoop directory



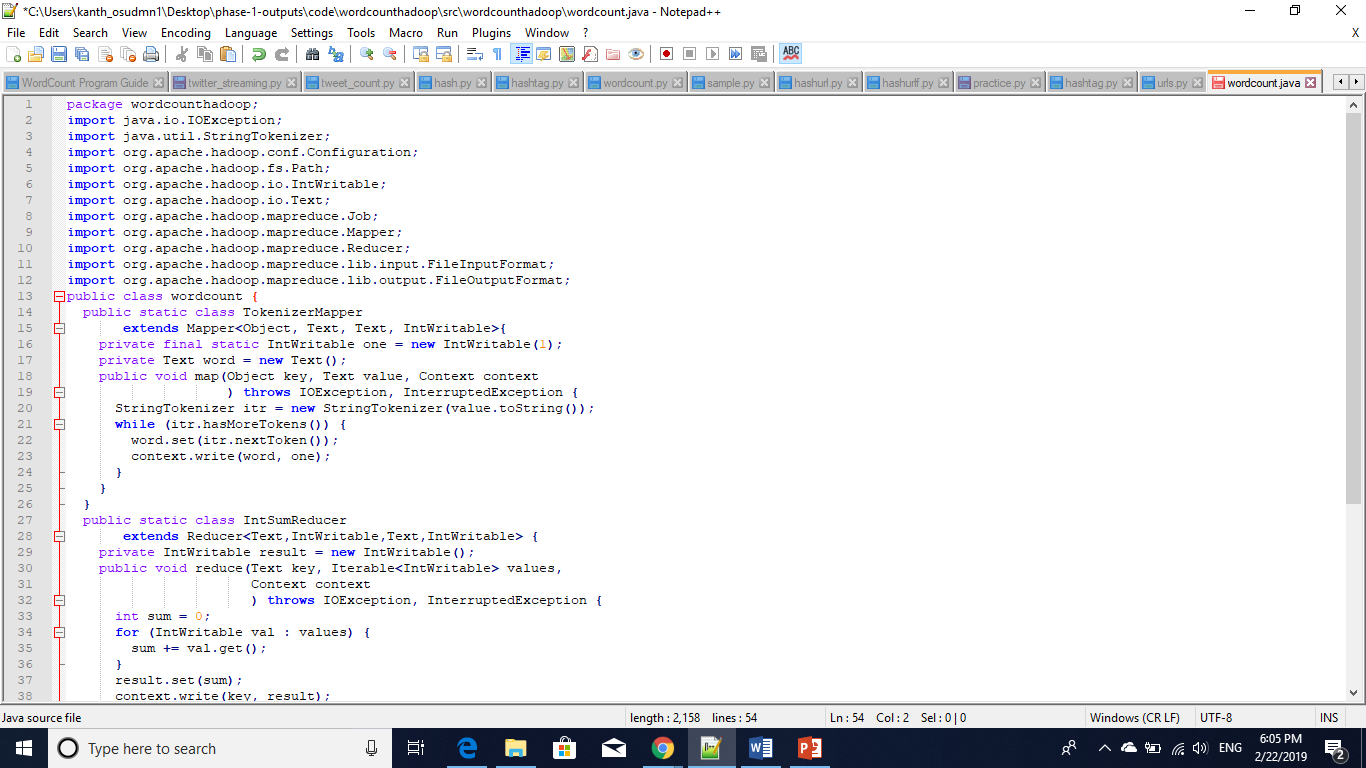
Hashtags output file in browser:



Hashtags output file in browser:



wordcount code:



**Source code:** wordcount.java

package wordcounthadoop;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class wordcount {

public static class TokenizerMapper

extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

public void map(Object key, Text value, Context context

) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.nextToken());

context.write(word, one);

}

}

}

public static class IntSumReducer

extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,

Context context

) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(wordcount.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

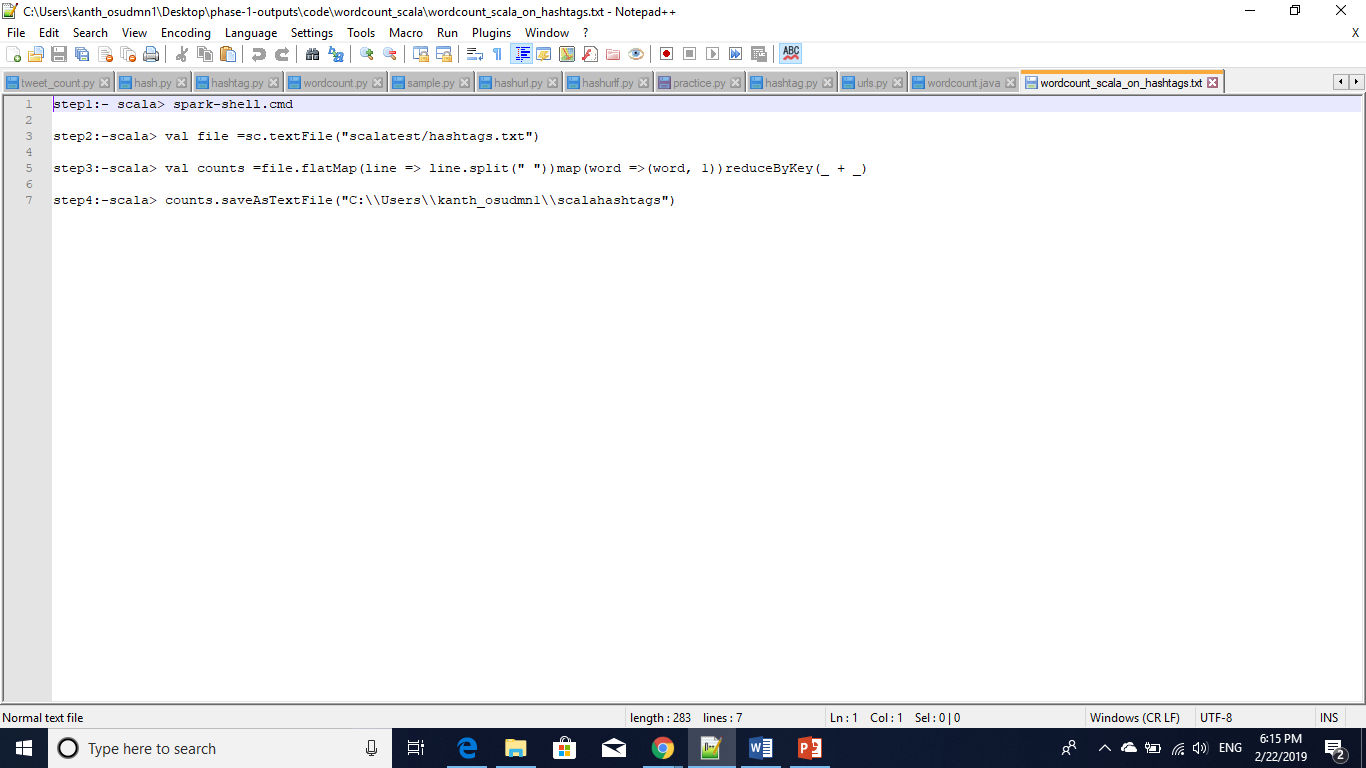
FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

Wordcount scala on hashtags: wordcount\_scala\_on\_hashtags.txt



Source Code:

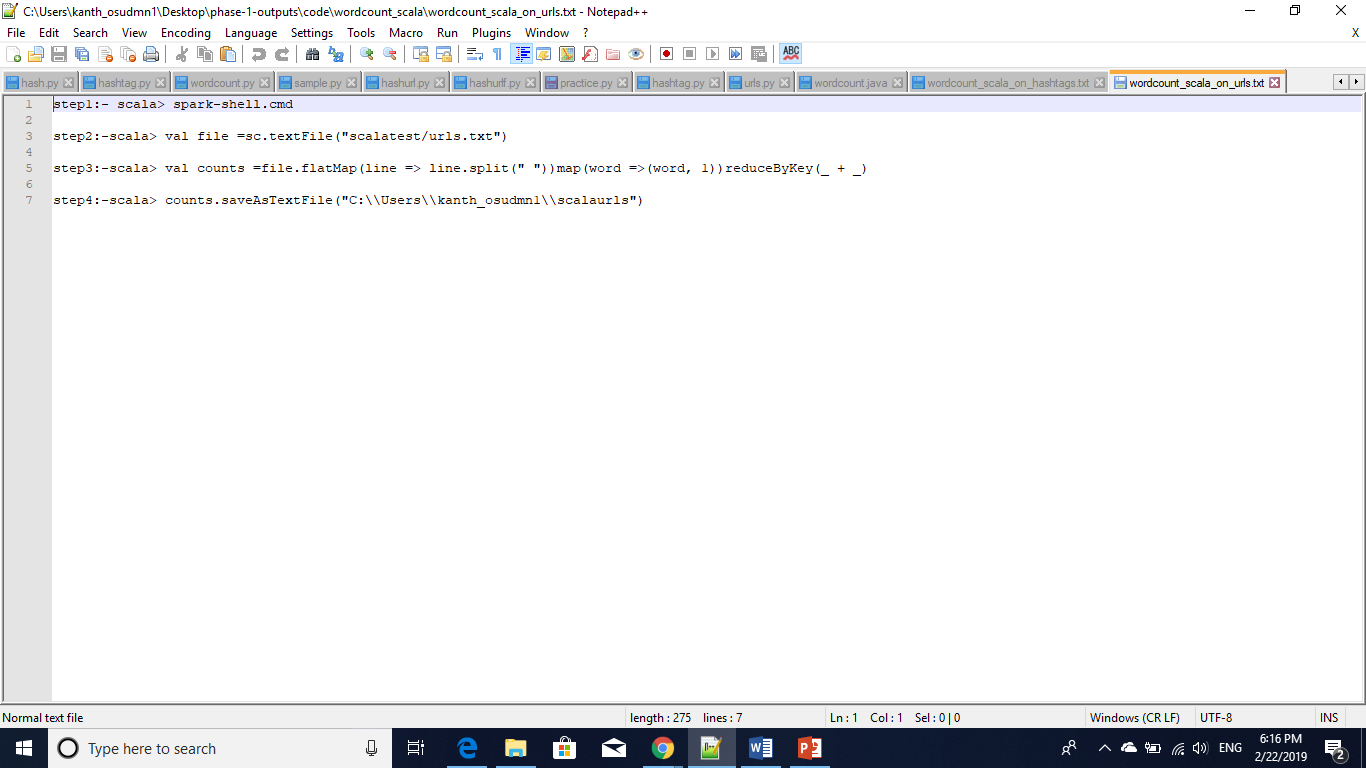
step1:- scala> spark-shell.cmd

step2:-scala> val file =sc.textFile("scalatest/hashtags.txt")

step3:-scala> val counts =file.flatMap(line => line.split(" "))map(word =>(word, 1))reduceByKey(\_ + \_)

step4:-scala> counts.saveAsTextFile("C:\\Users\\kanth\_osudmn1\\scalahashtags")

**wordcount scala on urls**: wordcount\_scala\_on\_urls.txt



Source code:

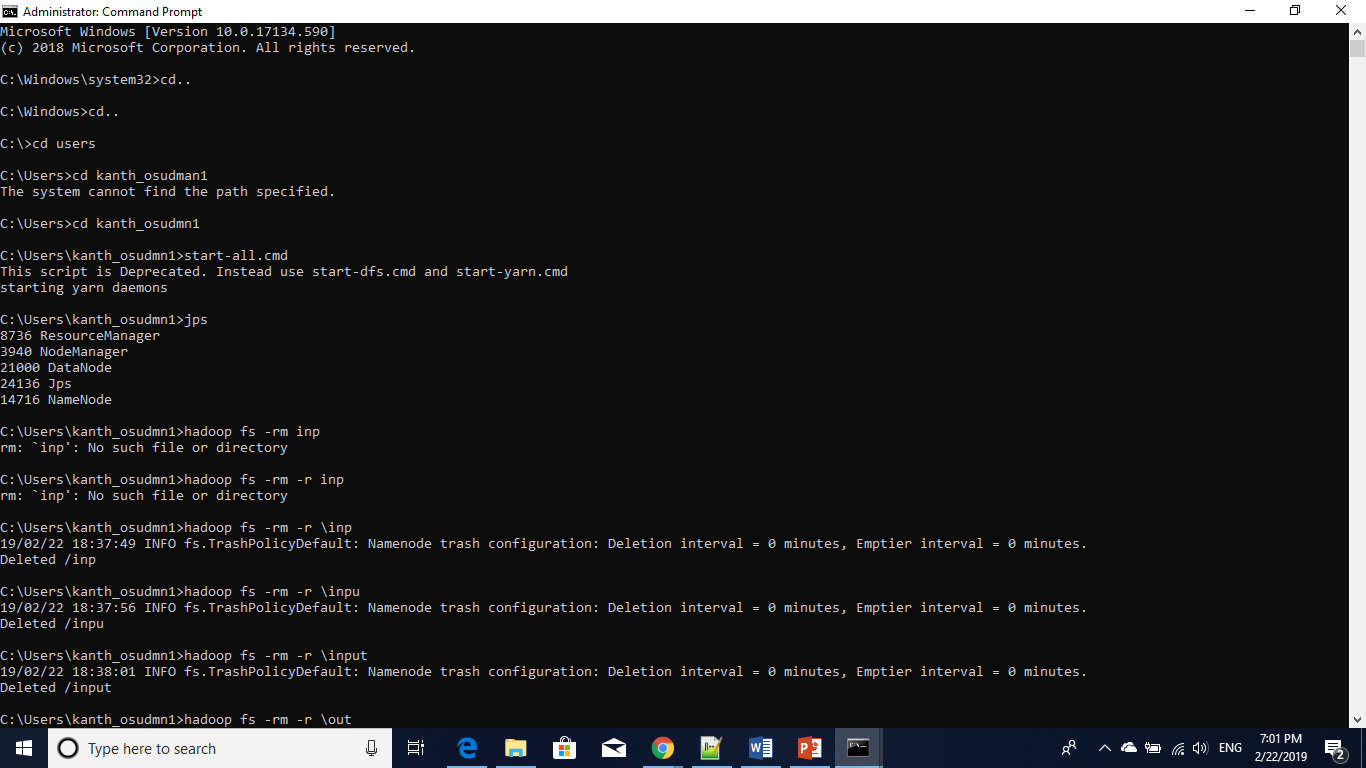
step1:- scala> spark-shell.cmd

step2:-scala> val file =sc.textFile("scalatest/urls.txt")

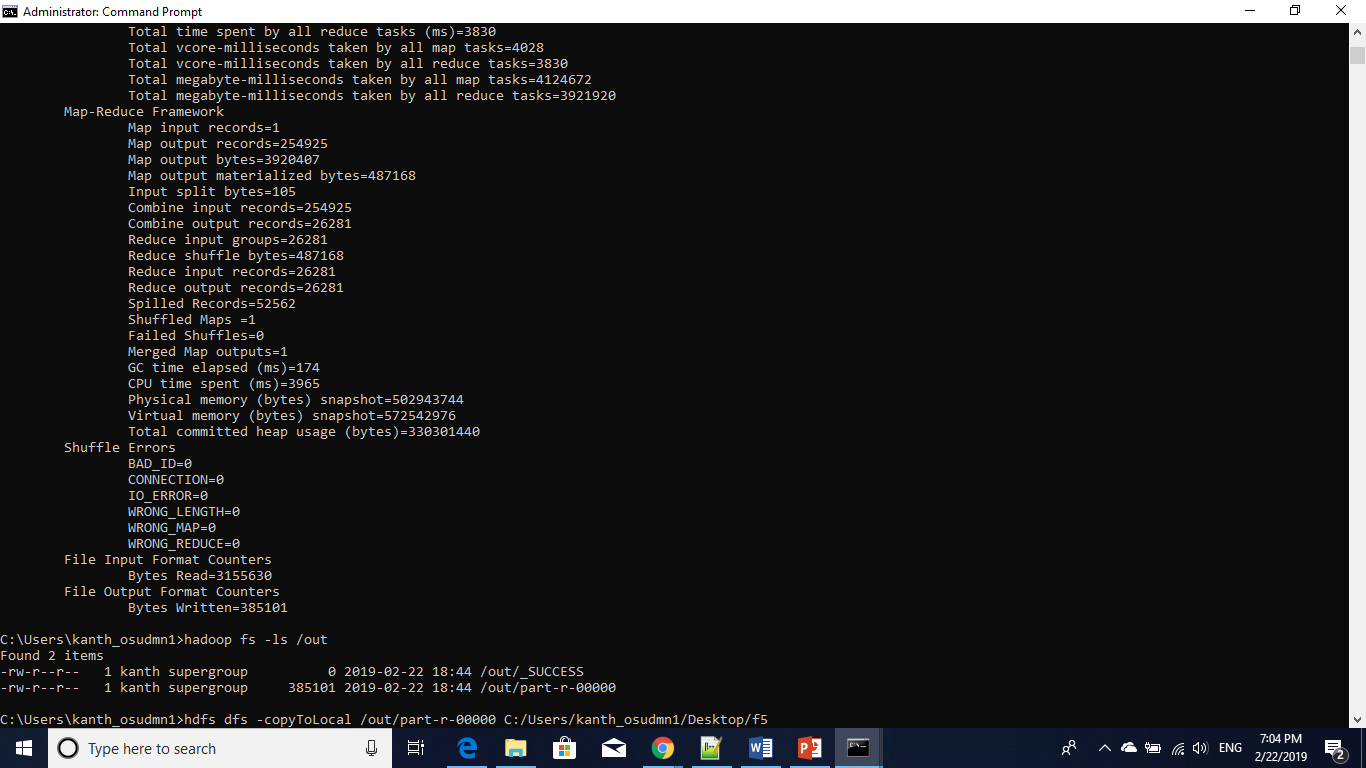
step3:-scala> val counts =file.flatMap(line => line.split(" "))map(word =>(word, 1))reduceByKey(\_ + \_)

step4:-scala> counts.saveAsTextFile("C:\\Users\\kanth\_osudmn1\\scalaurls")

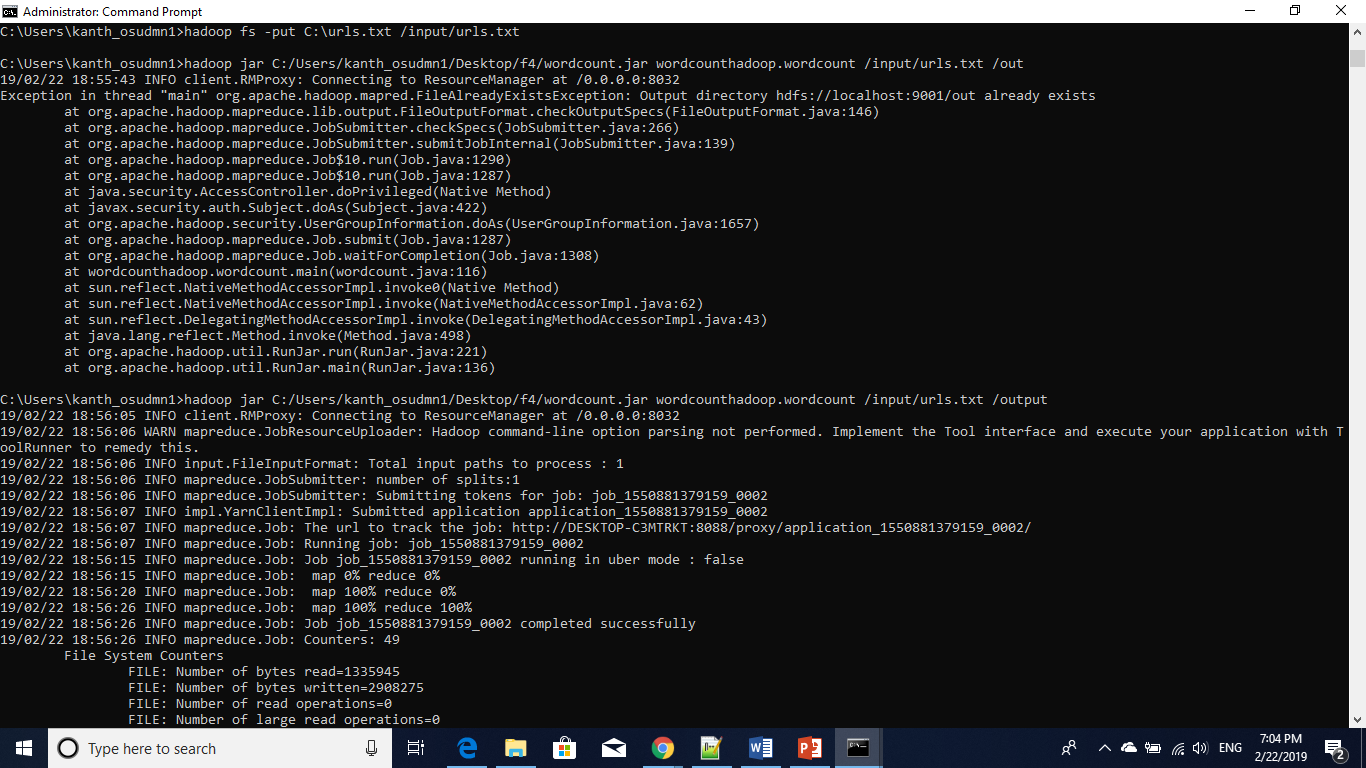
Hadoop installed



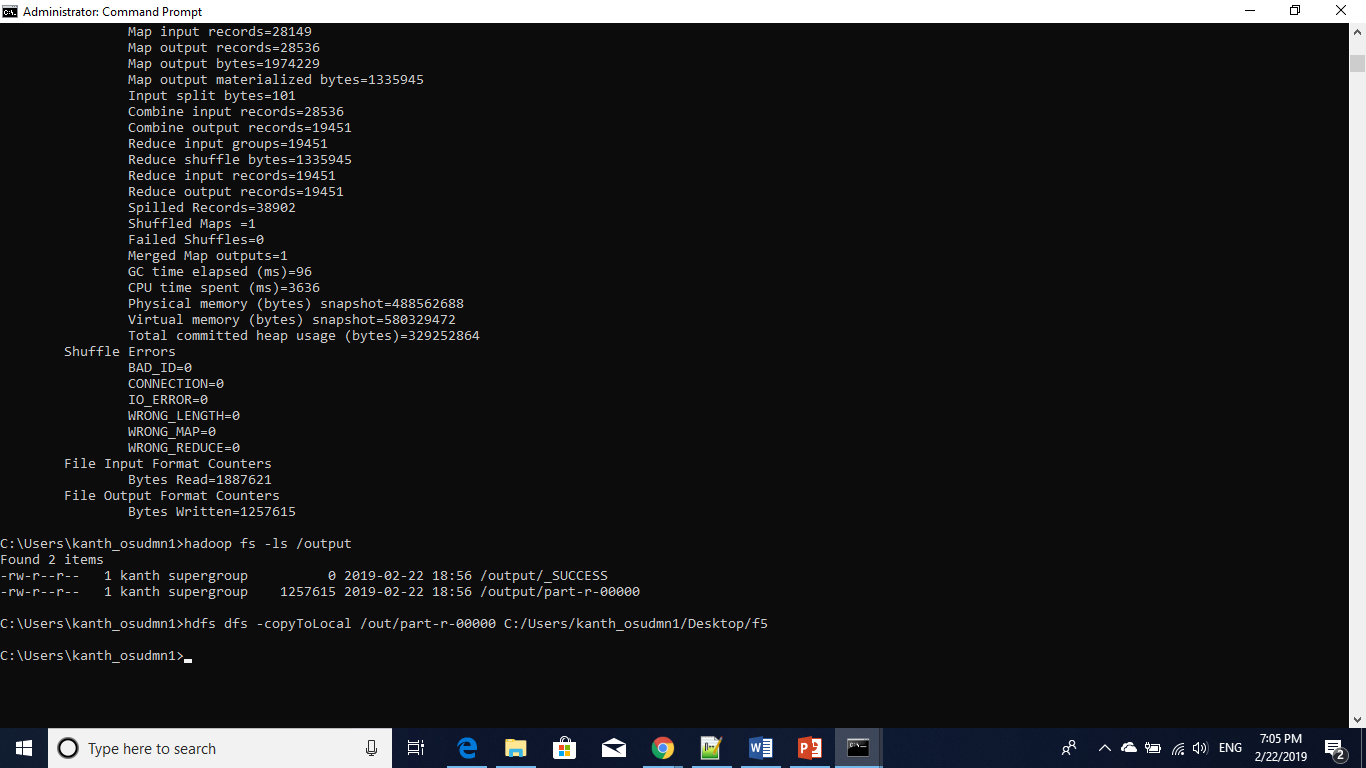
Wordcount hashtag output files



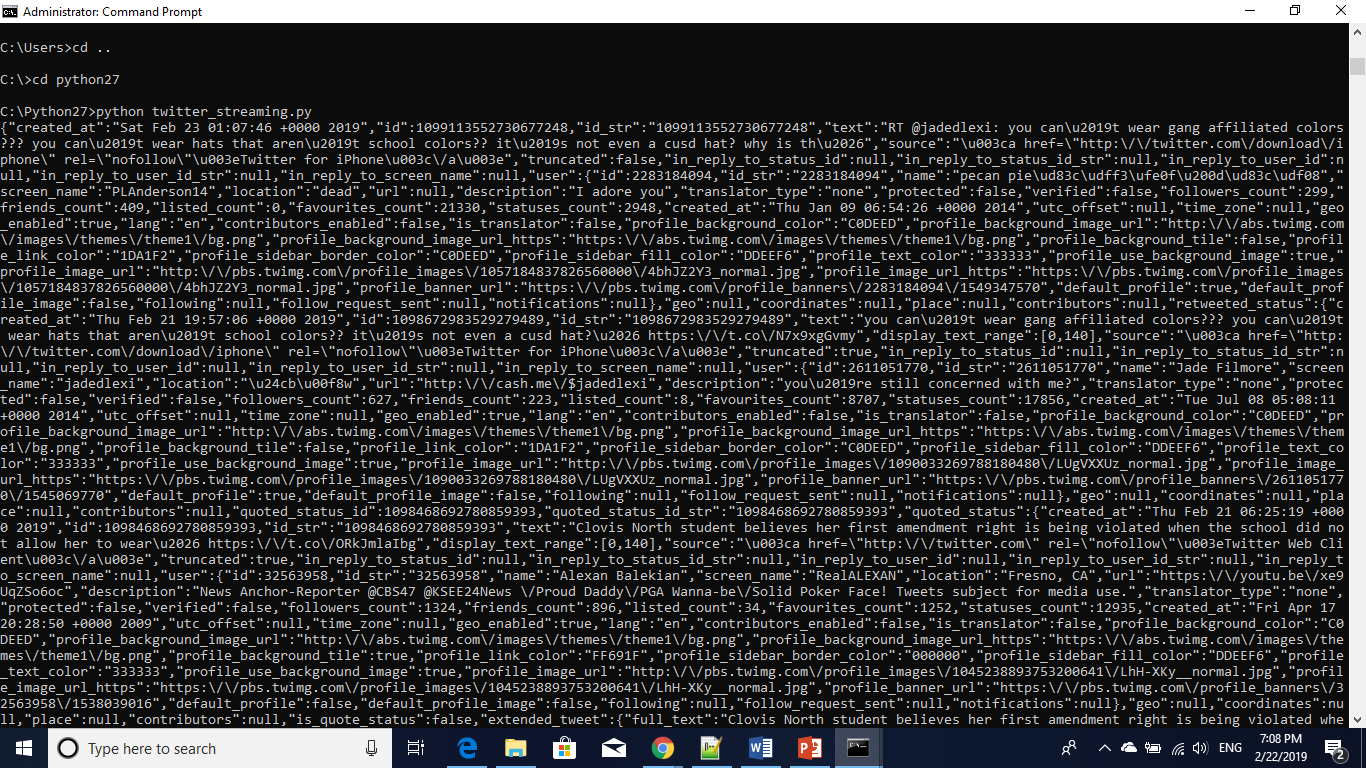
Wordcount on urls:



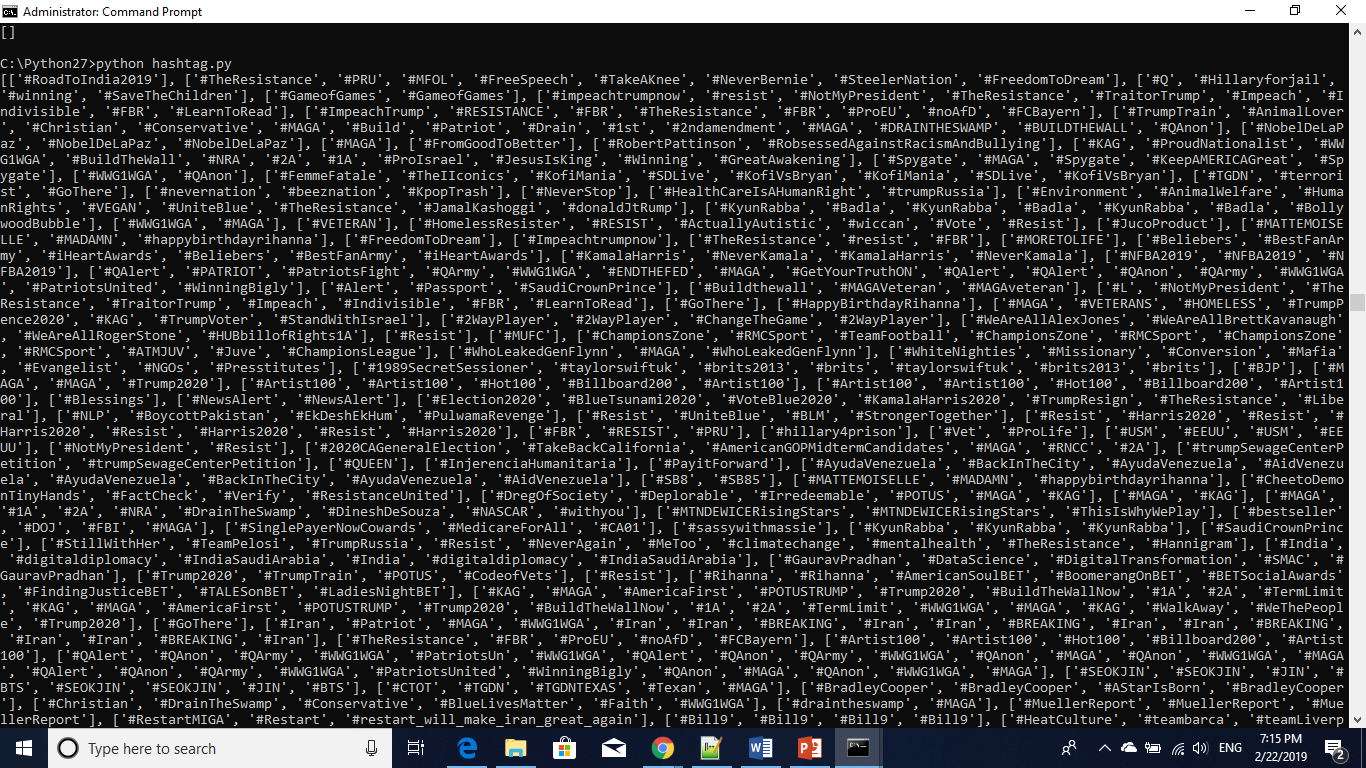
Wordcount output files on urls



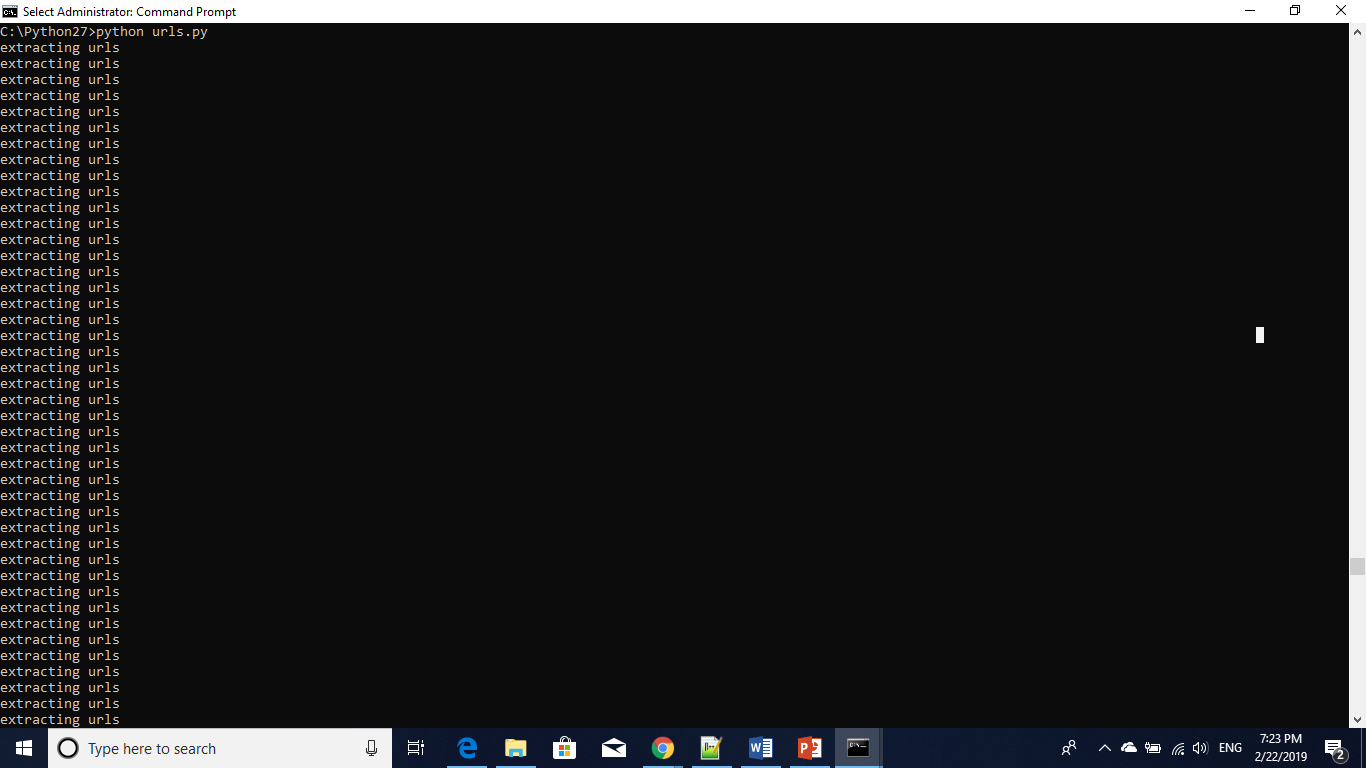
Twitter data streaming:



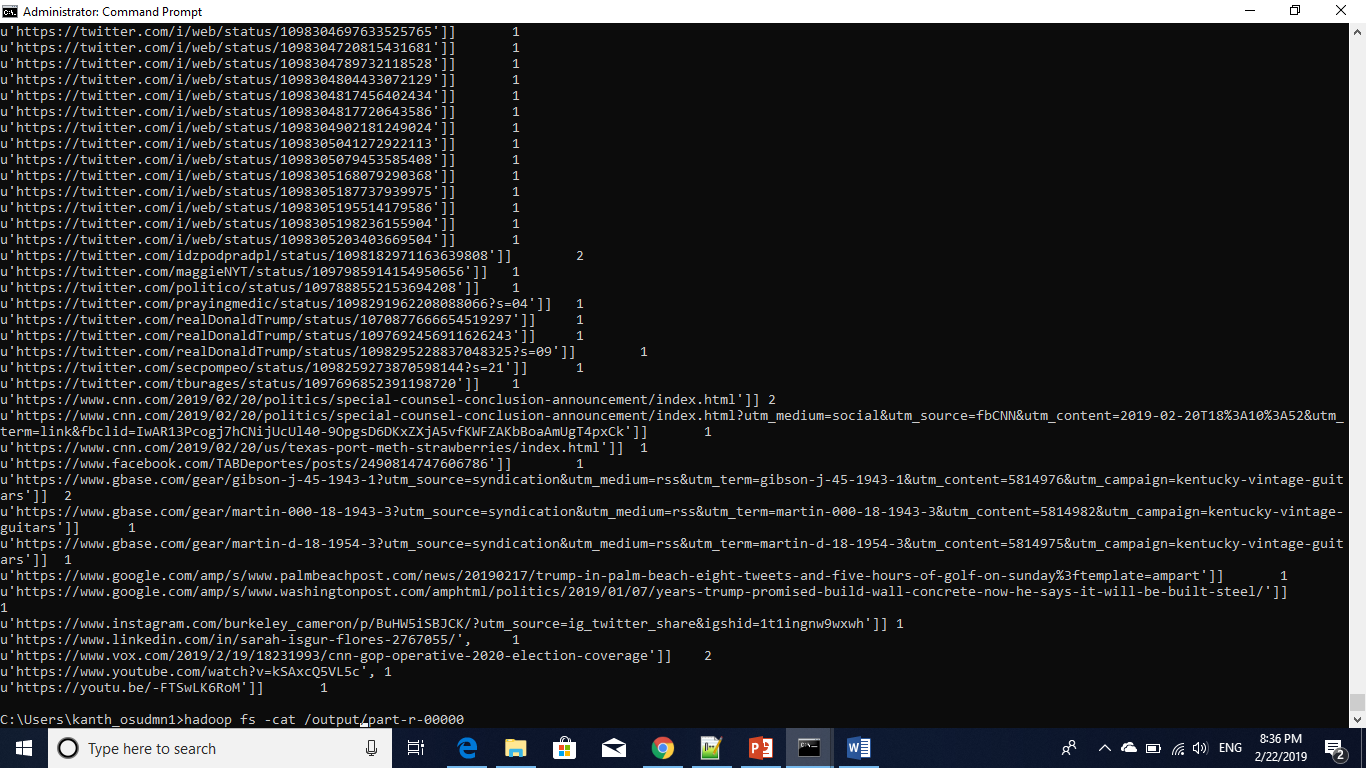
Hashtag extraction output

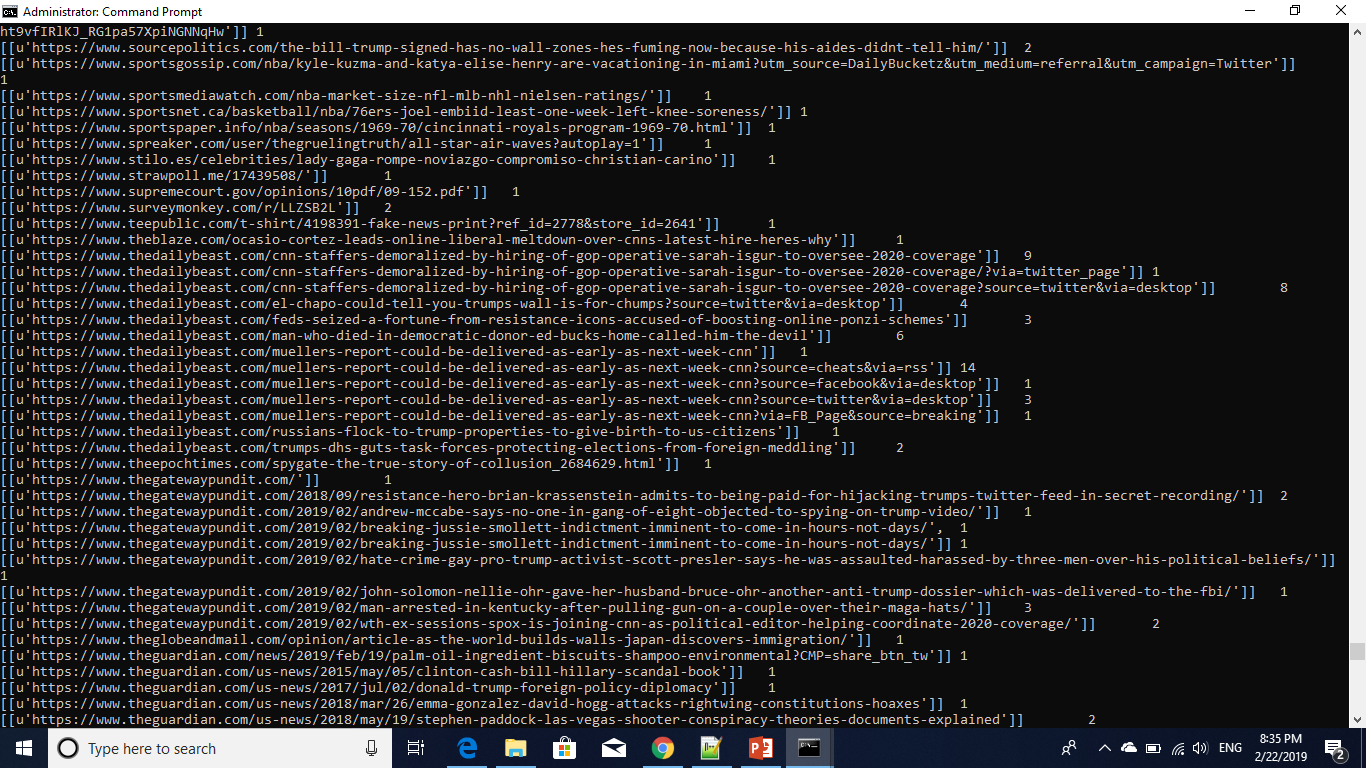


Urls extraction output:

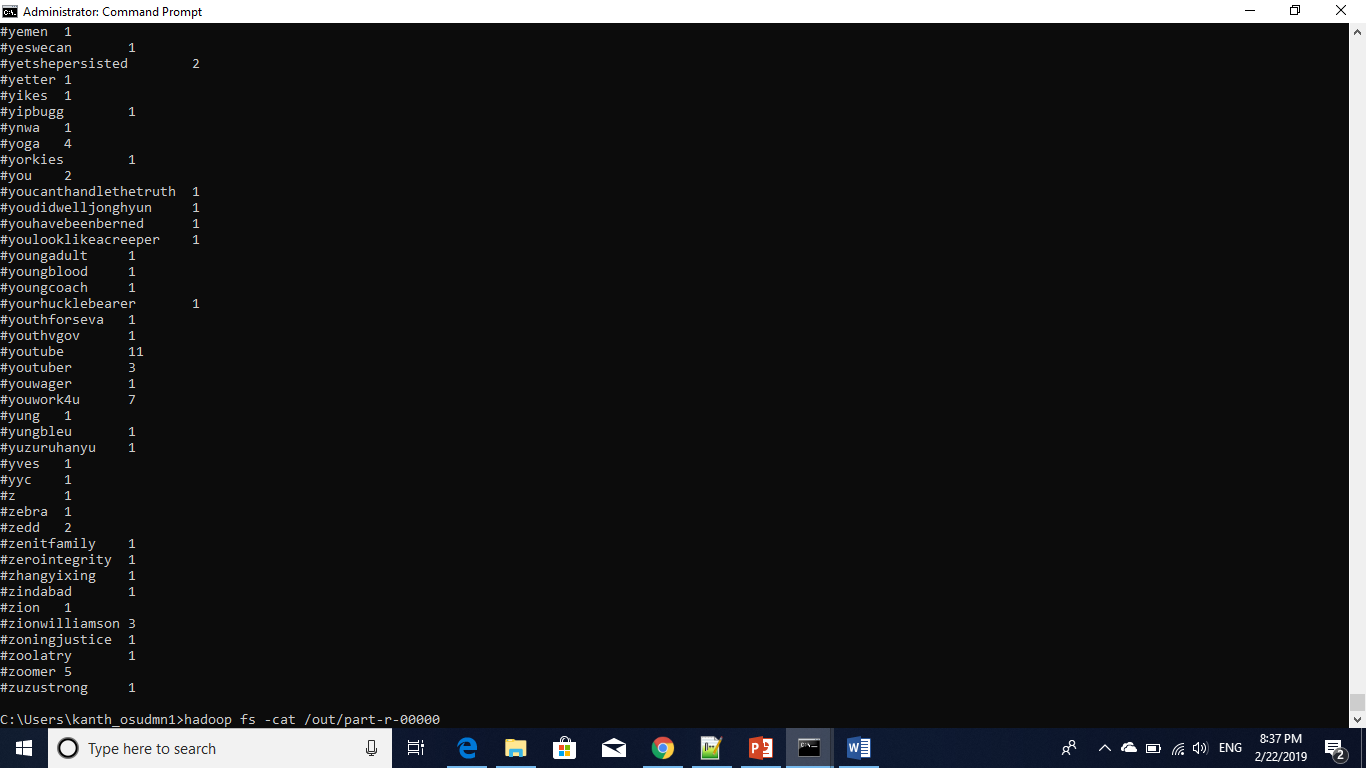


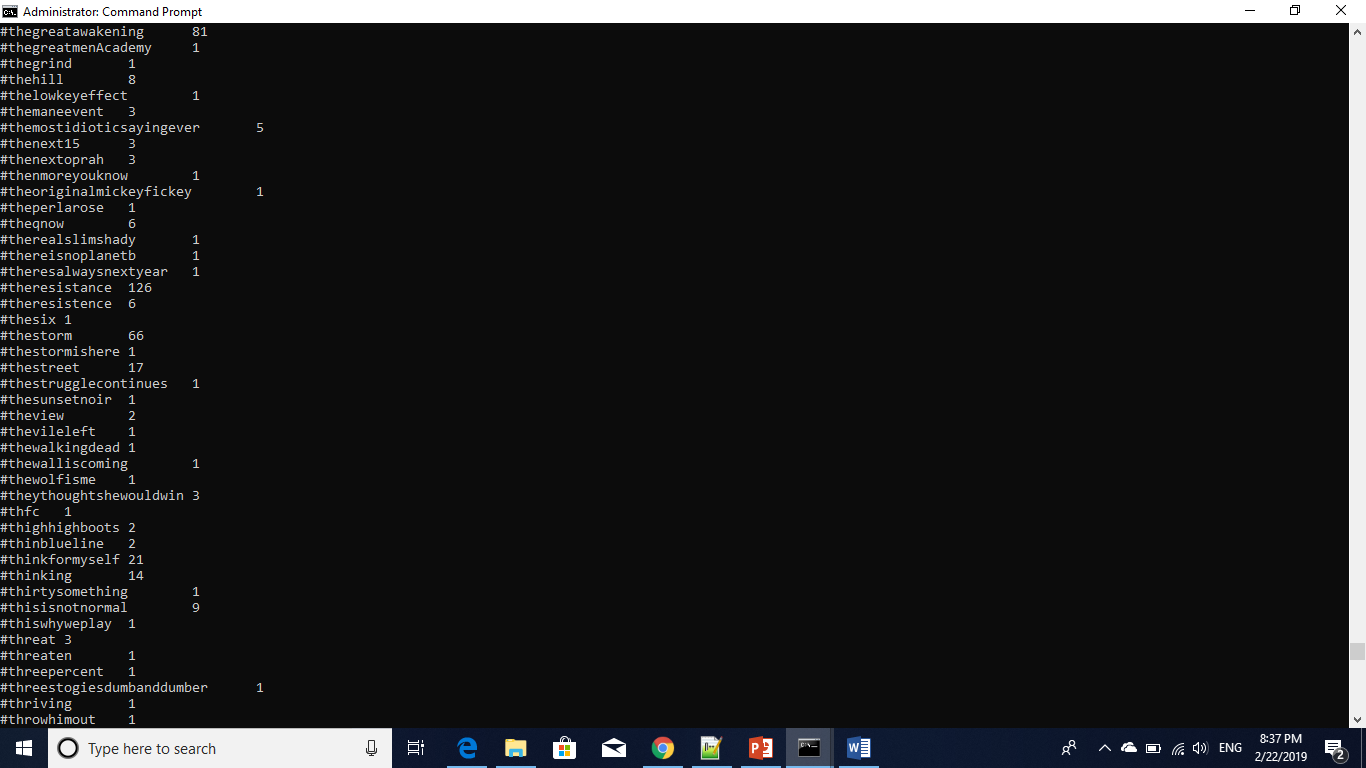
Word Count on Extracted URLs



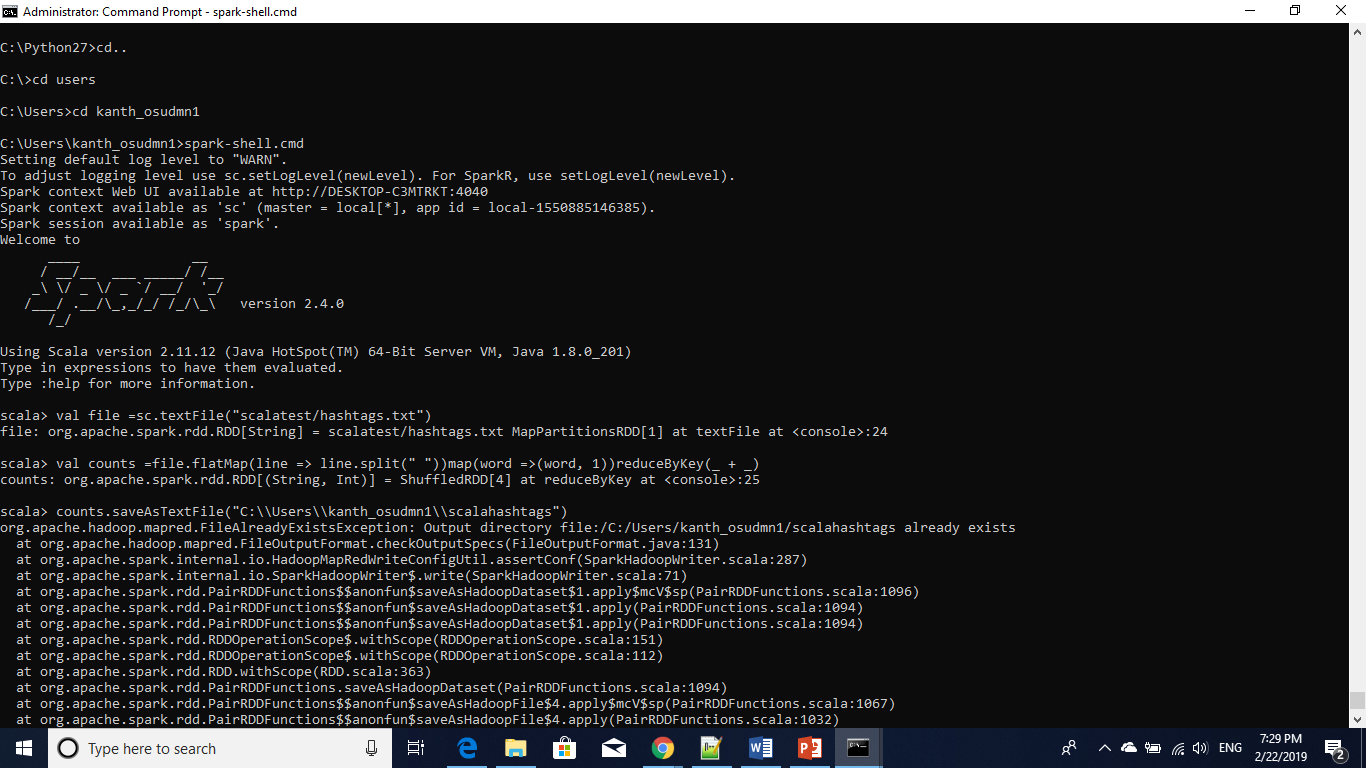


WordCount On Extracted Hashtags

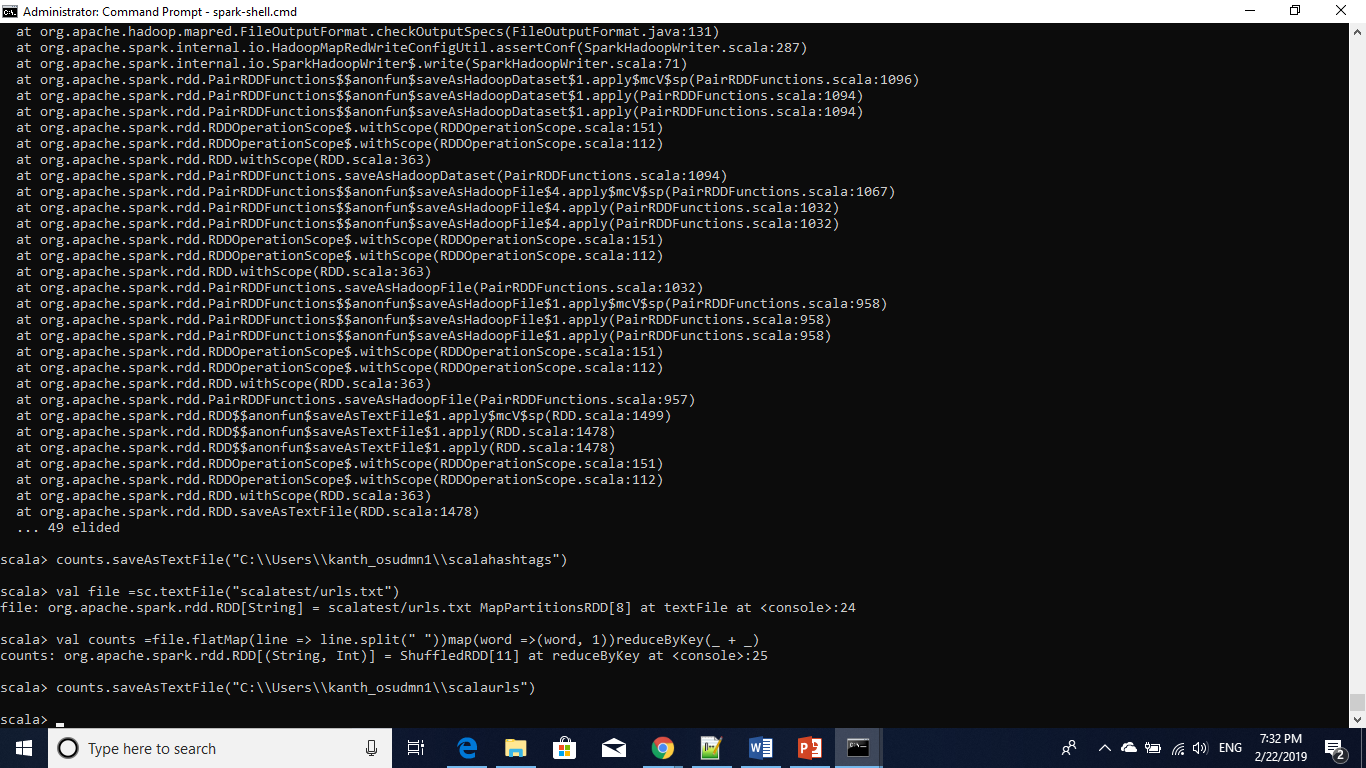




Spark wordcount on hashtags



Spark wordcount on urls



Spark output files in browser:

