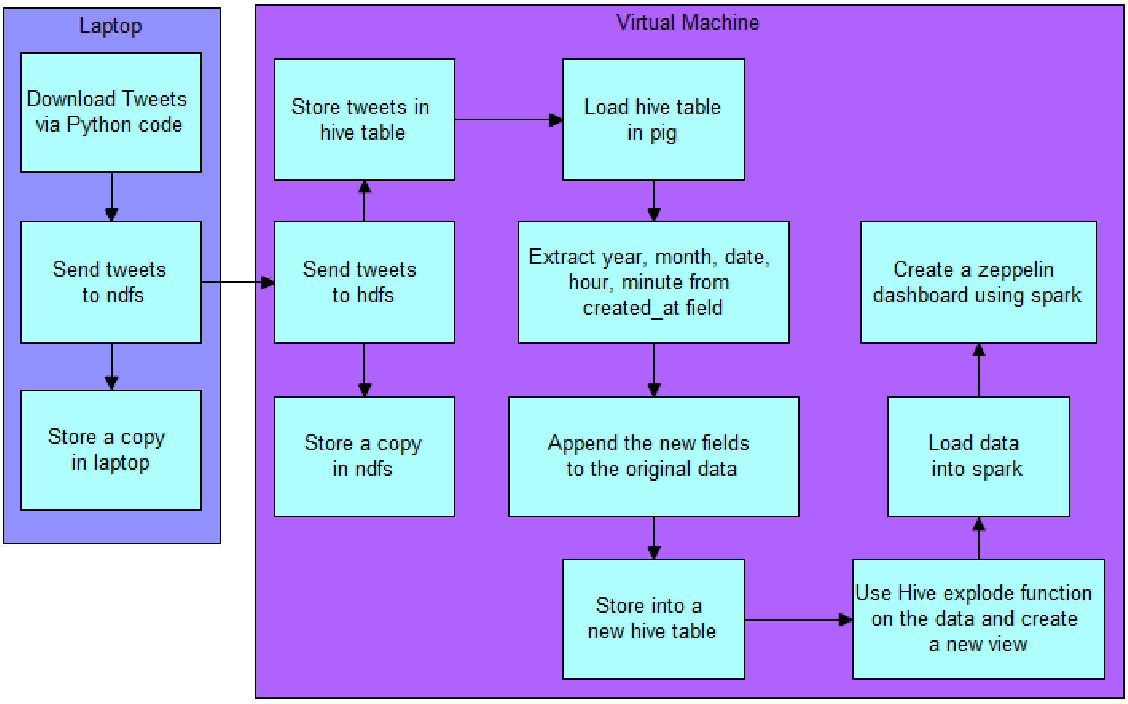
**Created By** Faran Shahid
**Powered By** Dice Analytics
Project Commands

# 1. Project Flow

The project in this course is to make an Extract, Transform, Load (ETL) pipeline which touches all the tools taught in the course so that you can have a full hands-on experience on all of them. Keep in mind that the ETL pipeline proposed for the project is a highly inefficient one and is only given so that you can have some practice.

This document contains all the commands for the project along with their usage to help you in your project.

## 1.1. ETL pipeline


Laptop
Download Tweets via Python code
Send tweets to ndfs
Store a copy in laptop
Virtual Machine
Store tweets in hive table
Load hive table in pig
Send tweets to hdfs
Extract year, month, date, hour, minute from created_at field
Create a zeppelin dashboard using spark
Store a copy in ndfs
Append the new fields to the original data
Load data into spark
Store into a new hive table
Use Hive explode function on the data and create a new view

A high level overview of the ETL pipeline is given in the above diagram. We will use a python program to fetch tweets from Twitter using the tweepy API every hour. A new file will be created by the python program which will be stored on the laptop and a copy will be sent to the ndfs of the sandbox. A copy of the file will then be sent to the HDFS and stored there as well.

Then we will load the file in the Hive table created using Hive's native JSON SERDE (**external SERDE can be used as well**) which will parse the tweets and extract the data required. Next, the hive data will be loaded into Pig where we will break the CREATED_AT column into year, month, date, hour and minute columns, append them to the text and id columns and store the result in a new hive table.

After this we will use the remove any new line character from the text column and use the Hive explode function on the text column. Furthermore, we will load the data into Spark using the zeppelin notebook and create a dashboard from the data.

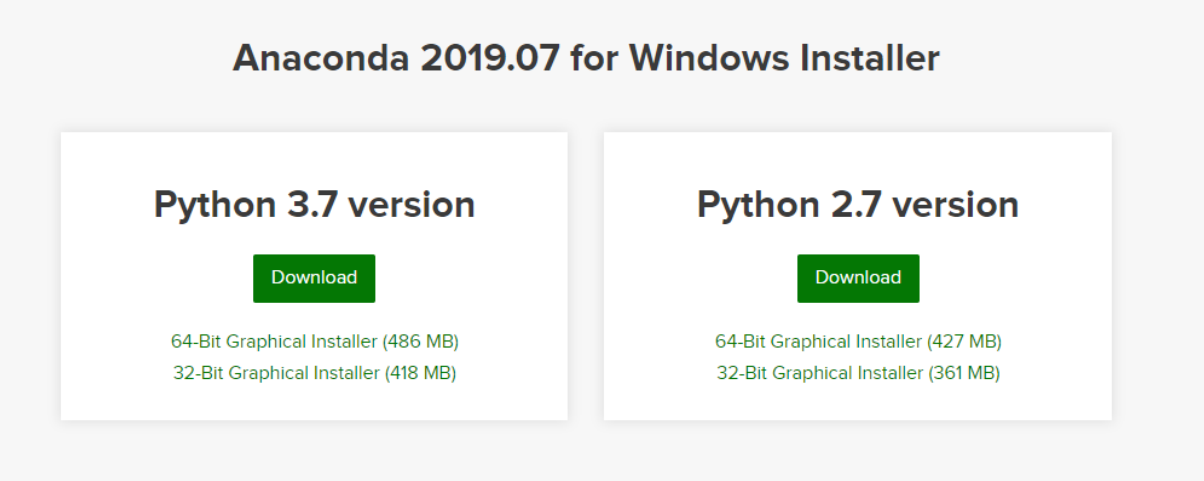
# 2. Pre-requisites

## 2.1. Installing Anaconda

Anaconda is a complete package for Python which provides a stable version of Python and a complete IDE and notebook for python coding. In addition, it also provides tools for debugging and testing making it a go-to application for python development. You will be provided with the Anaconda 5.0.1 executable for Windows 64-bit systems or you can download it directly from downloads section on Anaconda's website. **The python code provided in a separate .py file.**

Here's the download link for Anaconda's website.  Choose Python 3.7 installer according to your machine specs (32/64 bit).

https://www.anaconda.com/download/



Anaconda 2019.07 for Windows Installer

Python 3.7 version

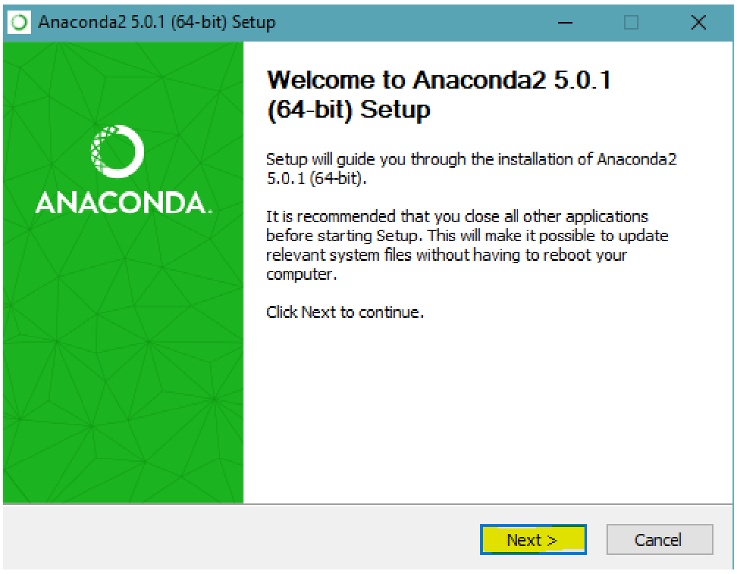
64-Bit Graphical Installer (486 MB)
32-Bit Graphical Installer (418 MB)

Python 2.7 version

64-Bit Graphical Installer (427 MB)
32-Bit Graphical Installer (361 MB)

Follow the installation wizard and install Anaconda. Make sure to follow the process described below or there can be problems with your installation:

- Open the installer and click Next



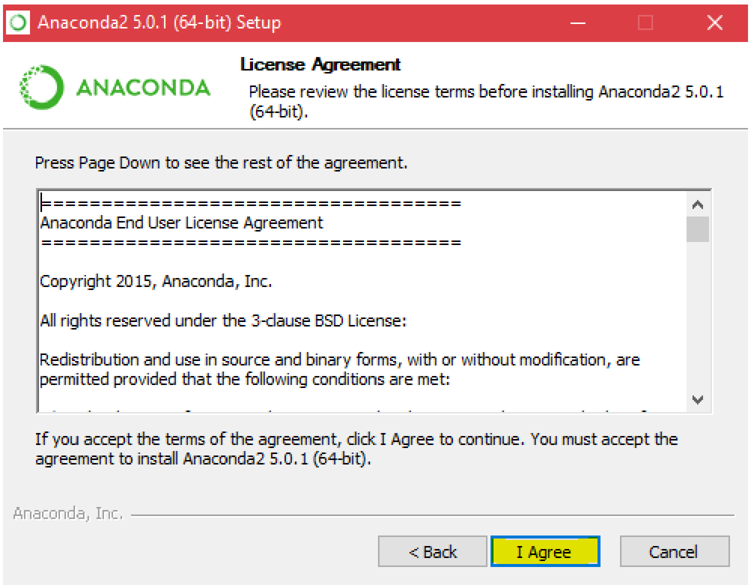
**Welcome to Anaconda2 5.0.1 (64-bit) Setup**

Setup will guide you through the installation of Anaconda2 5.0.1 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

- Click on I Agree



**License Agreement**

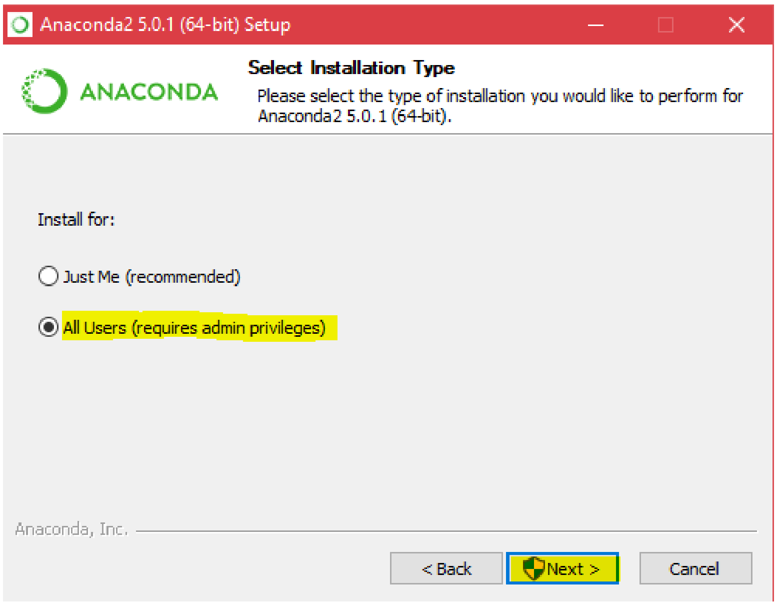
Please review the license terms before installing Anaconda2 5.0.1 (64-bit).

Press Page Down to see the rest of the agreement.

```
==================================
Anaconda End User License Agreement
==================================

Copyright 2015, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:

Redistribution and use in source and binary forms, with or without modification, are
permitted provided that the following conditions are met:
```

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda2 5.0.1 (64-bit).

- Select All Users installation type



Anaconda2 5.0.1 (64-bit) Setup

**Select Installation Type**

Please select the type of installation you would like to perform for Anaconda2 5.0.1 (64-bit).

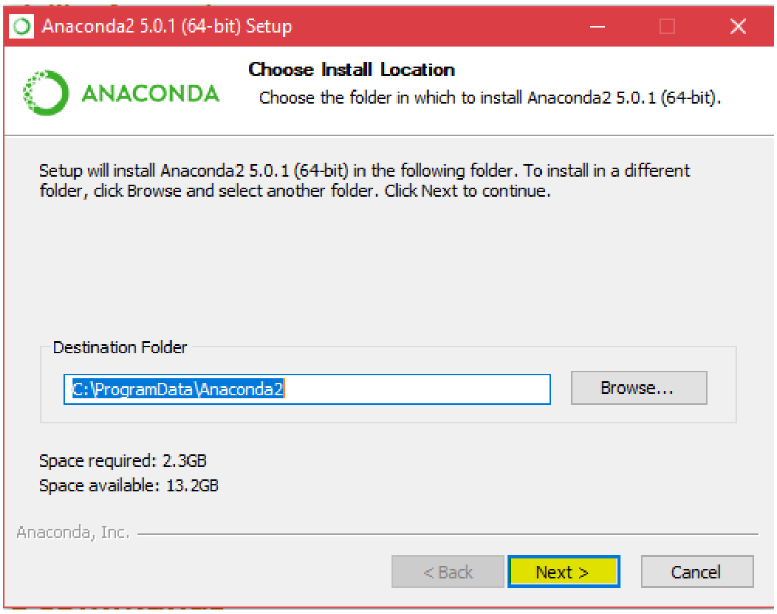
Install for:

○ Just Me (recommended)

● All Users (requires admin privileges)

Anaconda, Inc.

- Select the folder you want to install Anaconda in and click next**. Take note of the installation path entered in the Destination Folder as we will need it later on**. Its best to copy the path in a notepad file for later use.



Anaconda2 5.0.1 (64-bit) Setup

**Choose Install Location**

Choose the folder in which to install Anaconda2 5.0.1 (64-bit).

Setup will install Anaconda2 5.0.1 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

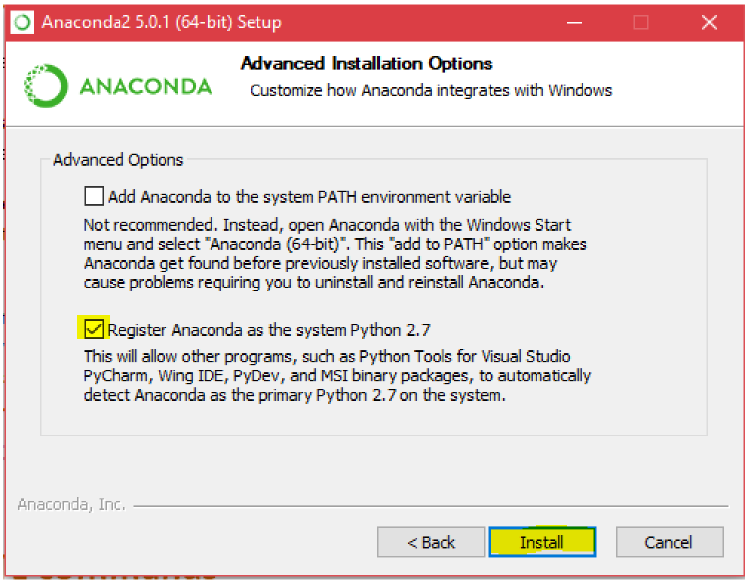
Destination Folder

C:\ProgramData\Anaconda2

Space required: 2.3GB
Space available: 13.2GB

Anaconda, Inc.

- In the Advanced options step for the make sure the Add Anaconda to my PATH environment variable is unchecked as below. This is an option but as the warning below it states, it can cause problems. Check the other option and click install



Anaconda2 5.0.1 (64-bit) Setup

ANACONDA

**Advanced Installation Options**

Customize how Anaconda integrates with Windows

Advanced Options

□ Add Anaconda to the system PATH environment variable

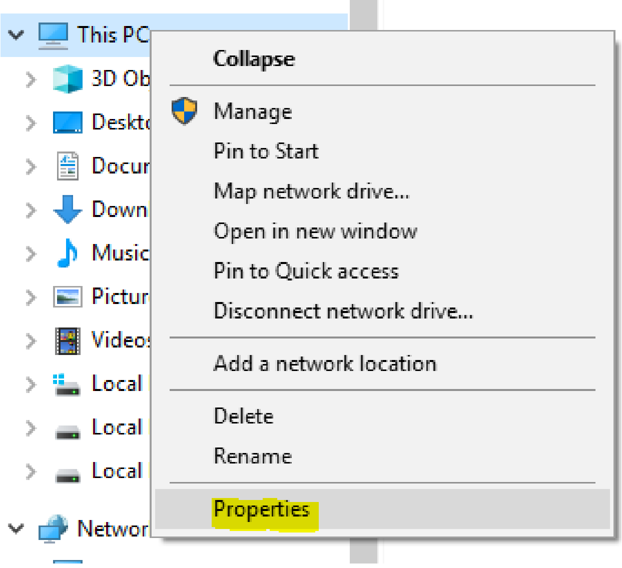
Not recommended. Instead, open Anaconda with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

☑ Register Anaconda as the system Python 2.7

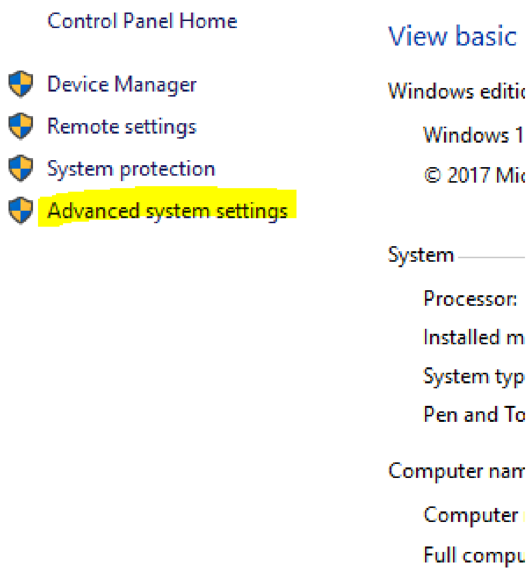
This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 2.7 on the system.

Anaconda, Inc.

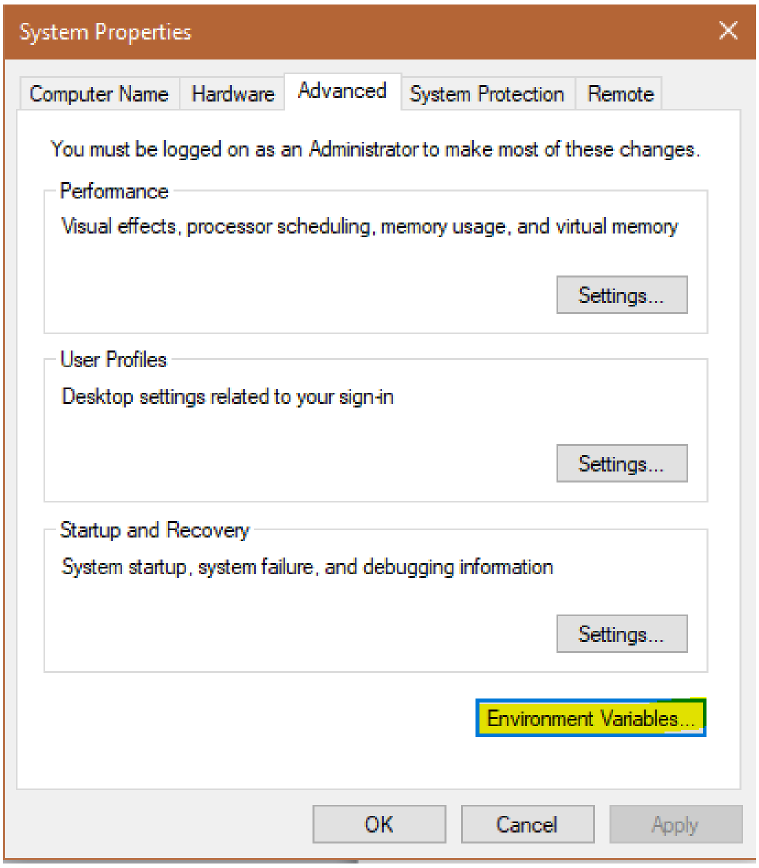
- The installation will be done and just click finish. Now we need to add the path for Anaconda ourselves as we unchecked the option in the previous point. Open file explorer and on the left side right click the computer name (usually named This PC but can be different). This will open the system's properties.



- Click on Advanced System Settings on the left hand side of the system properties window. A new window will open.



- Click on Environment Variables button. A new window will open.



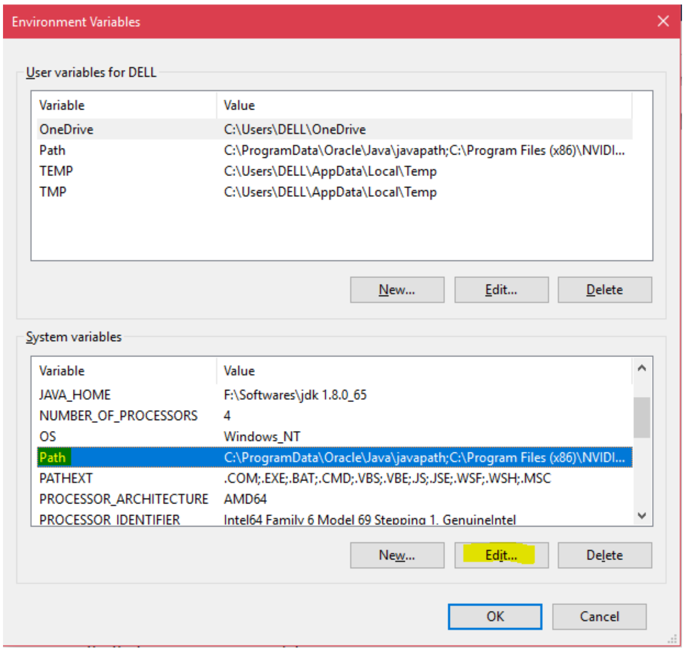
You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your sign-in

System startup, system failure, and debugging information

- Find the Path variable in the System variables tab, select it and click Edit. A new window will open.



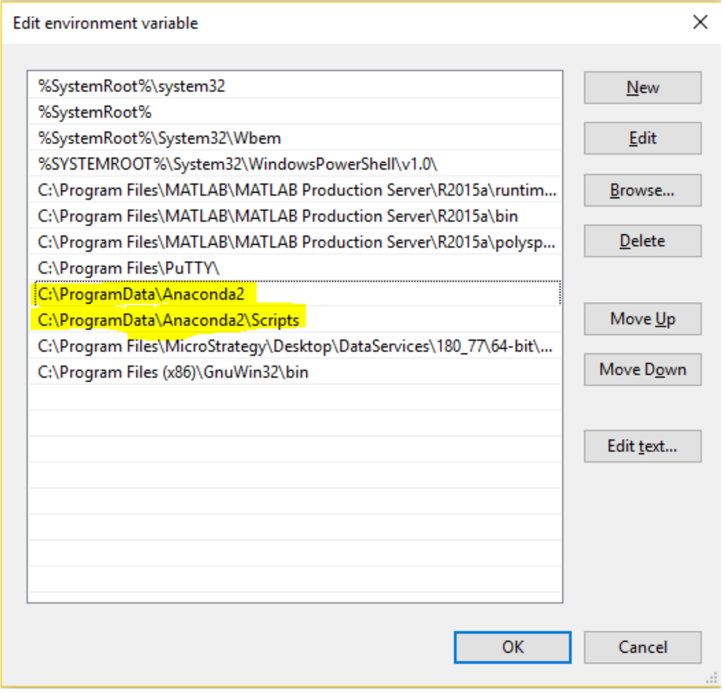
User variables for DELL

| Variable | Value |
|---|---|
| OneDrive | C:\Users\DELL\OneDrive |
| Path | C:\ProgramData\Oracle\Java\javapath;C:\Program Files (x86)\NVIDI... |
| TEMP | C:\Users\DELL\AppData\Local\Temp |
| TMP | C:\Users\DELL\AppData\Local\Temp |

System variables

| Variable | Value |
|---|---|
| JAVA_HOME | F:\Softwares\jdk 1.8.0_65 |
| NUMBER_OF_PROCESSORS | 4 |
| OS | Windows_NT |
| Path | C:\ProgramData\Oracle\Java\javapath;C:\Program Files (x86)\NVIDI... |
| PATHEXT | .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC |
| PROCESSOR_ARCHITECTURE | AMD64 |
| PROCESSOR_IDENTIFIER | Intel64 Family 6 Model 69 Stepping 1, GenuineIntel |

- Click New button and paste the path for Anaconda we saved earlier which is usually (**C:\ProgramData\Anaconda2**). Make sure to add **(C:\ProgramData\Anaconda2\scripts)** into variables as well (as shown below).
- **Make sure that you DON'T delete previously set environment variables.**
- Then click OK. Click OK again on the Environment Variables window and OK once again on the System Properties window. Now the path is set for Anaconda.

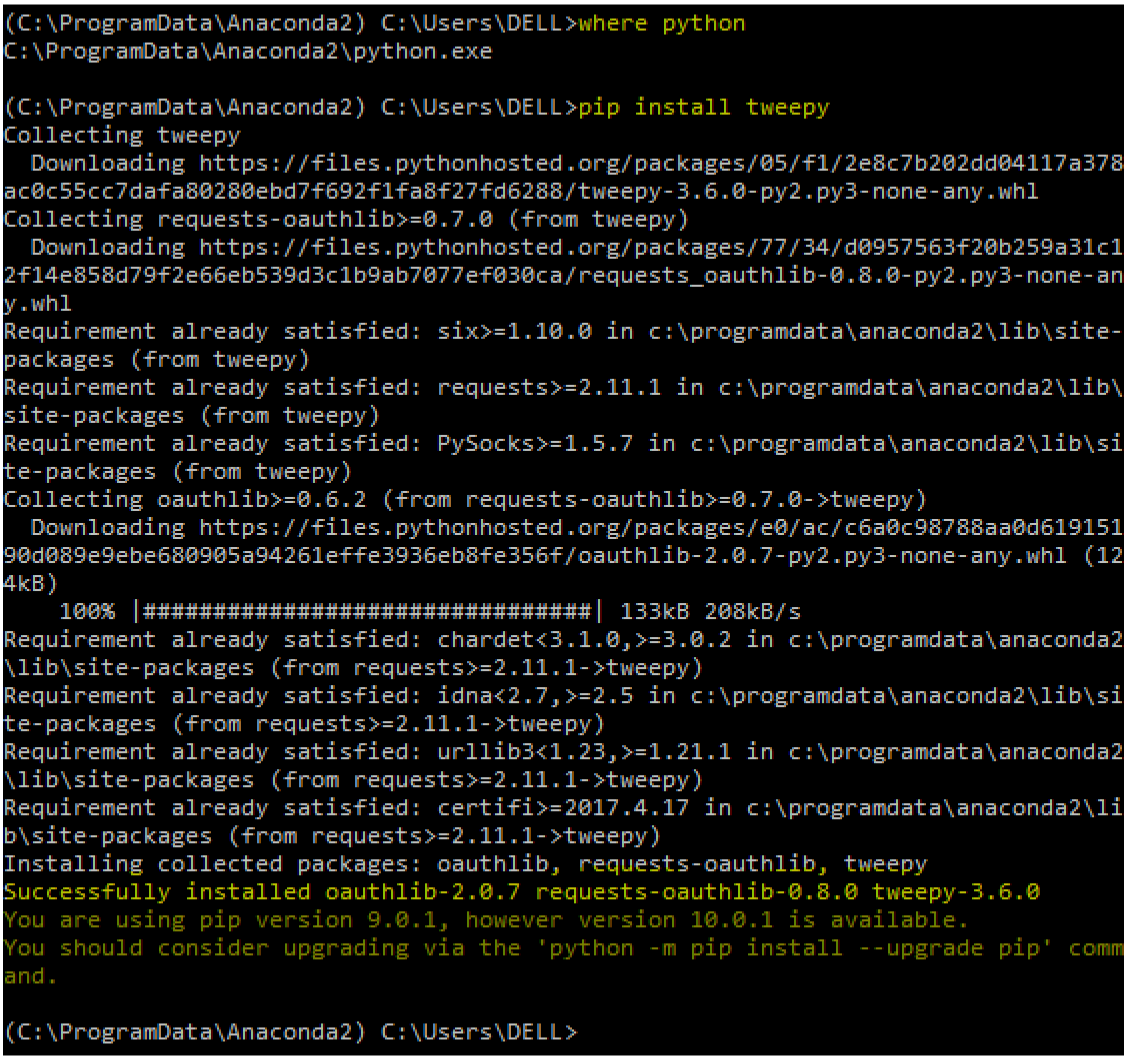


%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
C:\Program Files\MATLAB\MATLAB Production Server\R2015a\runtim...
C:\Program Files\MATLAB\MATLAB Production Server\R2015a\bin
C:\Program Files\MATLAB\MATLAB Production Server\R2015a\polysp...
C:\Program Files\PuTTY\
C:\ProgramData\Anaconda2
C:\ProgramData\Anaconda2\Scripts
C:\Program Files\MicroStrategy\Desktop\DataServices\180_77\64-bit\...
C:\Program Files (x86)\GnuWin32\bin

## 2.2. Installing tweepy

Next we need to install the tweepy library for Python. Tweepy provides a twitter API for Python and we can use it to fetch public tweets from Twitter.

- Open the Anaconda Prompt. Enter the command
  `where python`
  to make sure python is installed correctly. The command should return the path of the python installation. Next enter the command
  **`pip install tweepy`**
  to install tweepy. Pip will automatically find and install tweepy.



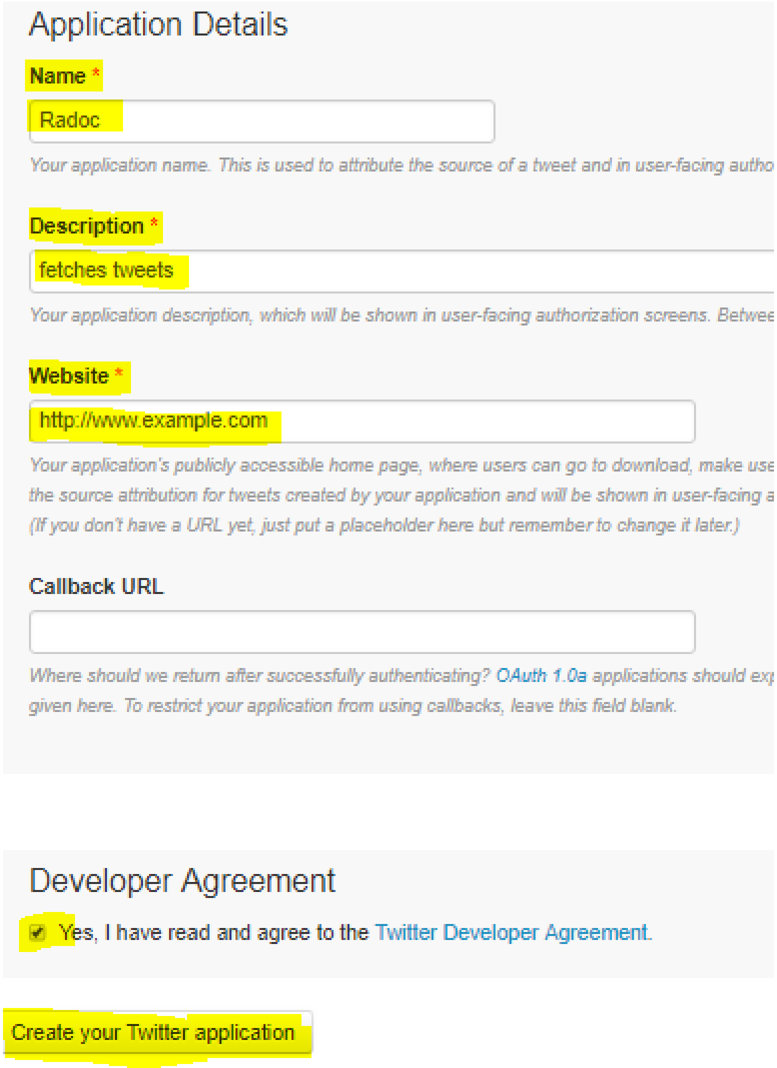
```
(C:\ProgramData\Anaconda2) C:\Users\DELL>where python
C:\ProgramData\Anaconda2\python.exe

(C:\ProgramData\Anaconda2) C:\Users\DELL>pip install tweepy
Collecting tweepy
  Downloading https://files.pythonhosted.org/packages/05/f1/2e8c7b202dd04117a378
ac0c55cc7dafa80280ebd7f692f1fa8f27fd6288/tweepy-3.6.0-py2.py3-none-any.whl
Collecting requests-oauthlib>=0.7.0 (from tweepy)
  Downloading https://files.pythonhosted.org/packages/77/34/d0957563f20b259a31c1
2f14e858d79f2e66eb539d3c1b9ab7077ef030ca/requests_oauthlib-0.8.0-py2.py3-none-an
y.whl
Requirement already satisfied: six>=1.10.0 in c:\programdata\anaconda2\lib\site-
packages (from tweepy)
Requirement already satisfied: requests>=2.11.1 in c:\programdata\anaconda2\lib\
site-packages (from tweepy)
Requirement already satisfied: PySocks>=1.5.7 in c:\programdata\anaconda2\lib\si
te-packages (from tweepy)
Collecting oauthlib>=0.6.2 (from requests-oauthlib>=0.7.0->tweepy)
  Downloading https://files.pythonhosted.org/packages/e0/ac/c6a0c98788aa0d619151
90d089e9ebe680905a94261effe3936eb8fe356f/oauthlib-2.0.7-py2.py3-none-any.whl (12
4kB)
    100% |################################| 133kB 208kB/s
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\programdata\anaconda2
\lib\site-packages (from requests>=2.11.1->tweepy)
Requirement already satisfied: idna<2.7,>=2.5 in c:\programdata\anaconda2\lib\si
te-packages (from requests>=2.11.1->tweepy)
Requirement already satisfied: urllib3<1.23,>=1.21.1 in c:\programdata\anaconda2
\lib\site-packages (from requests>=2.11.1->tweepy)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda2\li
b\site-packages (from requests>=2.11.1->tweepy)
Installing collected packages: oauthlib, requests-oauthlib, tweepy
Successfully installed oauthlib-2.0.7 requests-oauthlib-0.8.0 tweepy-3.6.0
You are using pip version 9.0.1, however version 10.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' comm
and.

(C:\ProgramData\Anaconda2) C:\Users\DELL>
```

## 2.3. Creating a twitter app

**If your account request rejected then you can skip this section and jump to section 3, as the API keys are also being provided within the python code.**

Next we need to create a twitter app on the twitter website so that we can get the authorization credentials to access the site using tweepy.

- Go to https://apps.twitter.com and click on Create New App on the right-hand side. Enter the info required on the next page. Name has to be unique and the Website URL must be a valid URL but not necessarily an active site. It can be any website but the URL has to be valid. Agree to the Developer Agreement and click Create your Twitter application.



Application Details

Name *

Radoc

Your application name. This is used to attribute the source of a tweet and in user-facing autho

Description *

fetches tweets

Your application description, which will be shown in user-facing authorization screens. Betwee

Website *

http://www.example.com

Your application's publicly accessible home page, where users can go to download, make use the source attribution for tweets created by your application and will be shown in user-facing a (If you don't have a URL yet, just put a placeholder here but remember to change it later.)

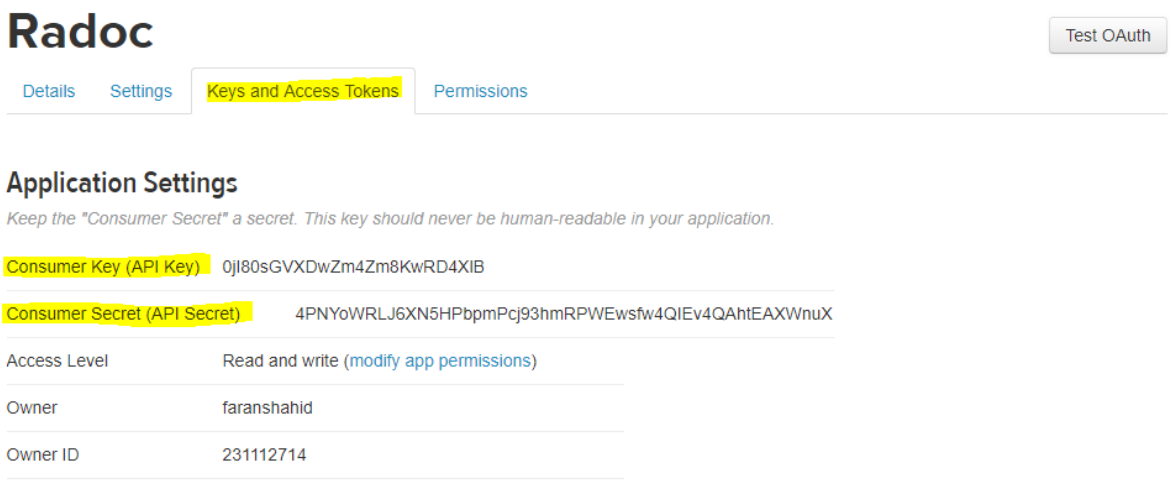
Callback URL

Where should we return after successfully authenticating? OAuth 1.0a applications should exp given here. To restrict your application from using callbacks, leave this field blank.

Developer Agreement

☑ Yes, I have read and agree to the Twitter Developer Agreement.

- Your app home will appear after the app has been created successfully. Click on the Keys and Access Tokens tab to view the keys required. On the top there will be the Consumer key and the Consumer Secret. Copy these two keys for later use in the python file we will create later.

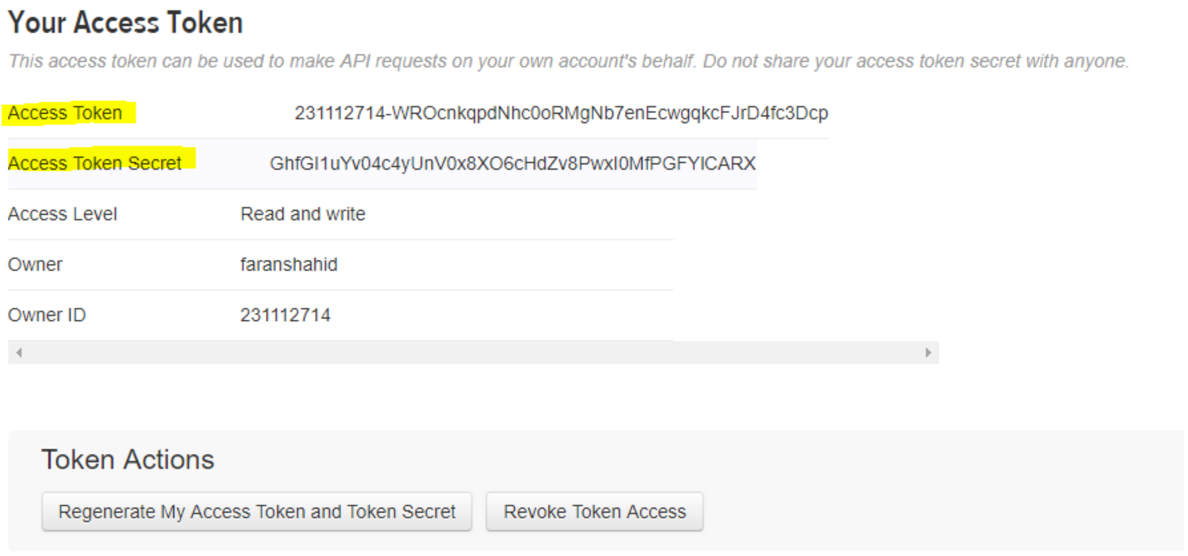


Application Settings

*Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.*

| | |
|---|---|
| Consumer Key (API Key) | 0jI80sGVXDwZm4Zm8KwRD4XIB |
| Consumer Secret (API Secret) | 4PNYoWRLJ6XN5HPbpmPcj93hmRPWEwsfw4QIEv4QAhtEAXWnuX |
| Access Level | Read and write (modify app permissions) |
| Owner | faranshahid |
| Owner ID | 231112714 |

- Scroll down and click on Generate Access Token to create your Access Token keys. Copy these two keys for later use as well.



Your Access Token

*This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.*

| | |
|---|---|
| Access Token | 231112714-WROcnkqpdNhc0oRMgNb7enEcwgqkcFJrD4fc3Dcp |
| Access Token Secret | GhfGI1uYv04c4yUnV0x8XO6cHdZv8PwxI0MfPGFYICARX |
| Access Level | Read and write |
| Owner | faranshahid |
| Owner ID | 231112714 |

You can visit https://apps.twitter.com anytime to get these keys again. You can also generate new keys using the specified buttons.

# 3. ETL commands

## 3.1. Generating tweets

**Note: Python code is provided as a separate .py file so please don't copy from here.**

The data we will be feeding into the ETL pipeline will be tweets pertaining to a particular hashtag. These tweets are fetched through a python program which uses the tweepy library to fetch tweets. The complete code along with comments is given below.

The code below is for Python 2.7. To run this code on Python 3 only minor changes are required. Python is quite sensitive to tabs and space indentations in code so there will be errors when copying the code below. It is recommended to write the code yourself or use the python file if it is provided. The lines in light grey are comments to help you understand what the code does.

```
# coding: utf-8
# In[26]:
# importing the required libraries.
import tweepy
import json
import time
import datetime
# Authentication details. Enter your own twitter app keys here.
consumer_key = ""
consumer_secret = ""
access_key = ""
access_secret = ""
# Enter the hashtag you want to search for here.
accountvar = "Football"
# getting the current date and time.
t = datetime.datetime.now()
# sorting the acquired date and time into the format we want as Windows
# doesn't allow us to include : in file names.
a = t.strftime('%Y-%m-%d-%H-%M')
# specifying the output file name.
outputfilejson = accountvar+"_"+str(a)+".json"
# This is the listener, resposible for receiving data.
class StdOutListener(tweepy.StreamListener):
    # this class will be called before any of the others once the
connection with the Twitter API is made.
    def __init__(self, time_limit=60):
        # setting current time as start time.
        self.start_time = time.time()
        # setting time limiter to pause stream. Current time limit is 60
seconds as specified in the function definition.
        self.limit = time_limit
        super(StdOutListener, self).__init__()
    # defining the on_data function. This will tell the compiler what to
do whenever new data is received.
    def on_data(self, data):
        # setting the time limit checker. It will let the stream fetch
```

```
data as long as the time limit has not been reached.
        if (time.time() - self.start_time) < self.limit:
            # loading the fetched encoded json tweet into the decoded
variable.
            decoded = json.loads(data)
            try:
                if isinstance(decoded, dict):
                    # decoding the json tweet and loading it in decoded.
                    decoded = json.dumps(decoded)
                    # writing the decoded tweet into the output file.
                    outfile.write(decoded)
                    # adding a new line after the tweet into the output
file.
                    outfile.write('\n')
                    # printing the stuff we are writing into the output
file.
                    print decoded+'\n'
            # handling exceptions in case there is some error
            except (NameError, KeyError,AttributeError):
                pass
            return True
        # code will go here once the time limit is reached. The following
lines will disconnect the stream.
        else:
            time.sleep(10)
            stream.disconnect()
            return False
    # code will go here if there is any error while making connection with
the Twitter API.
    def on_error(self, status):
        # printing the received error.
        print status
# specifying the main method. This method is always called first when the
code starts so we do all the initializing here.
if __name__ == '__main__':
    # calling the listener class.
    l = StdOutListener()
    # specifying the authorization handler of the Twitter API and giving
it the variables we defined at the start.
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    # setting the access token with the tokens of our Twitter API.
    auth.set_access_token(access_key, access_secret)
    # creating a new output file and opening it in write back mode.
    with open(outputfilejson, 'wb') as outfile:
        print "Showing all new tweets for " + accountvar
```

```
# initializing the stream with the Twitter API authorization keys.
stream = tweepy.Stream(auth, l)
# searching for the required hashtag.
stream.filter(track=[str("#" + accountvar)])
```

## 3.2. Windows Scheduler

The tweet fetching and transfer has to be done every hour so we use a task scheduler to automate the process. It can be set up easily and made to run a .bat file.

NOTE: Read this entire section before following the steps

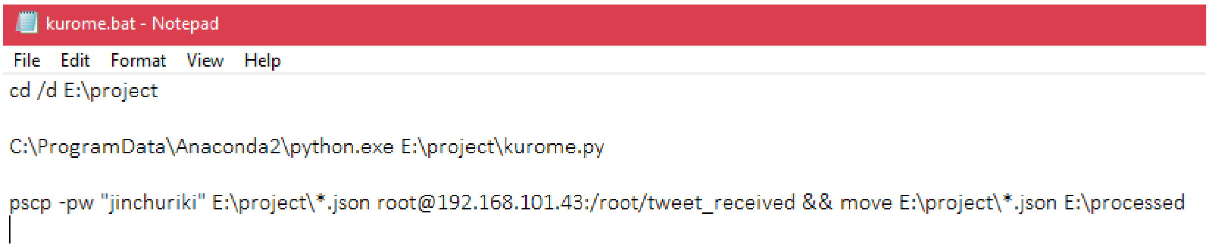
- Create a batch file in the folder where you placed your python file. Please make sure not to put them on the desktop. Open the batch file and paste the following code into it:

```
cd /d <path-of-your-python-file>
```

```
<path-to-your-python-executable>\python.exe <path-to-python-file>\<name-of-python-file.py>
```

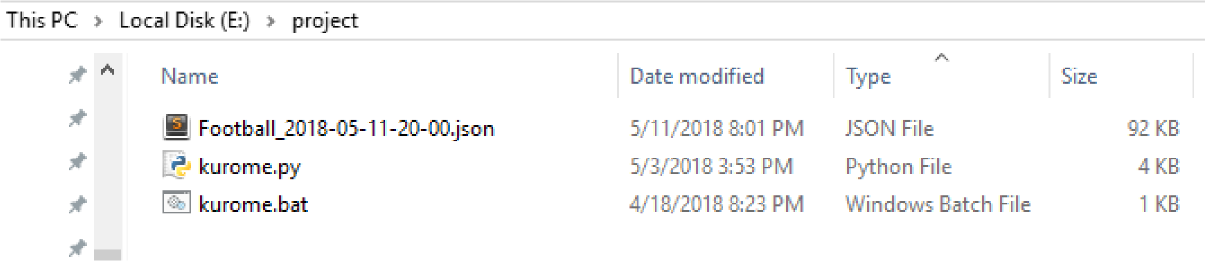
```
pscp -pw "<your-vm-root-password>" <path-to-where-your-json-files-are-created>\*.json root@<your-vm-ip-address>:/root/<folder-name-where-tweets-are received> && move <path-to-where-your-json-files-are-created>\*.json <path-to-folder-where-json-files-are-to-be-stored-after-processing>
```

For example my code would be:



```
cd /d E:\project

C:\ProgramData\Anaconda2\python.exe E:\project\kurome.py

pscp -pw "jinchuriki" E:\project\*.json root@192.168.101.43:/root/tweet_received && move E:\project\*.json E:\processed
```

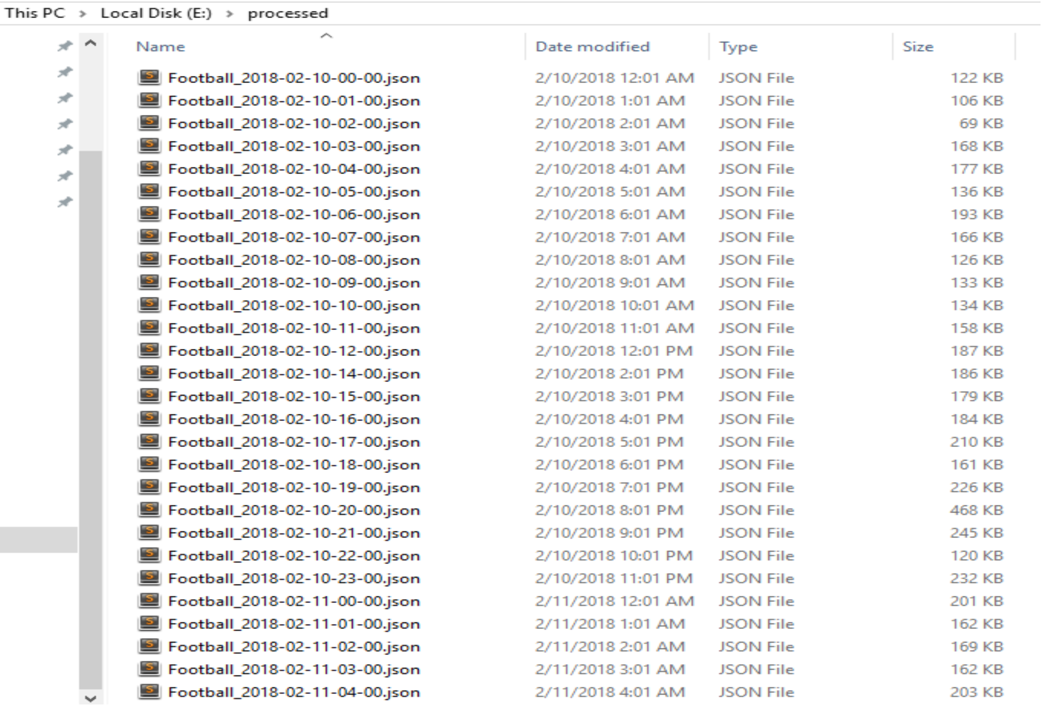
- Your json files will be created in the same folder as your python file or batch file. Create the folder where your json files will go after processing. The ETL pipeline on my laptop has been running for 3 months now so it looks like this:
  E:\project is where I’ve placed my python and batch file and where my json files are created



This PC > Local Disk (E:) > project

| Name | Date modified | Type | Size |
|---|---|---|---|
| Football_2018-05-11-20-00.json | 5/11/2018 8:01 PM | JSON File | 92 KB |
| kurome.py | 5/3/2018 3:53 PM | Python File | 4 KB |
| kurome.bat | 4/18/2018 8:23 PM | Windows Batch File | 1 KB |

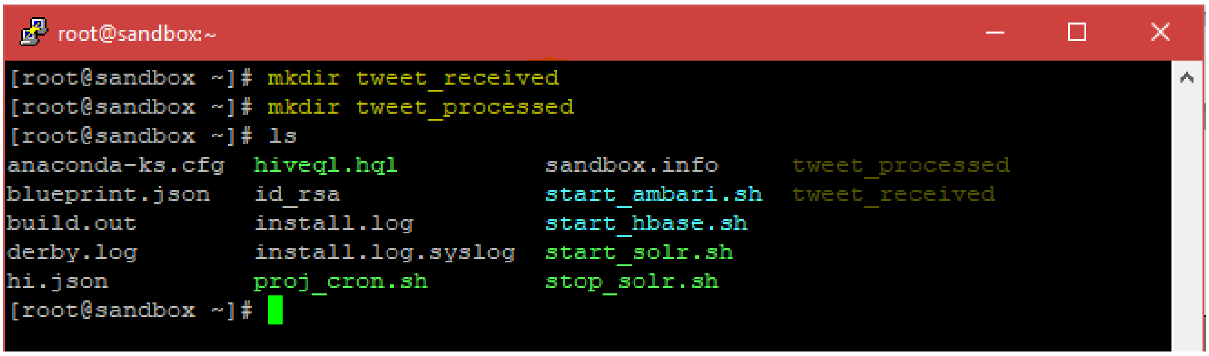
E:\processed is where my json files are stored after processing



This PC > Local Disk (E:) > processed

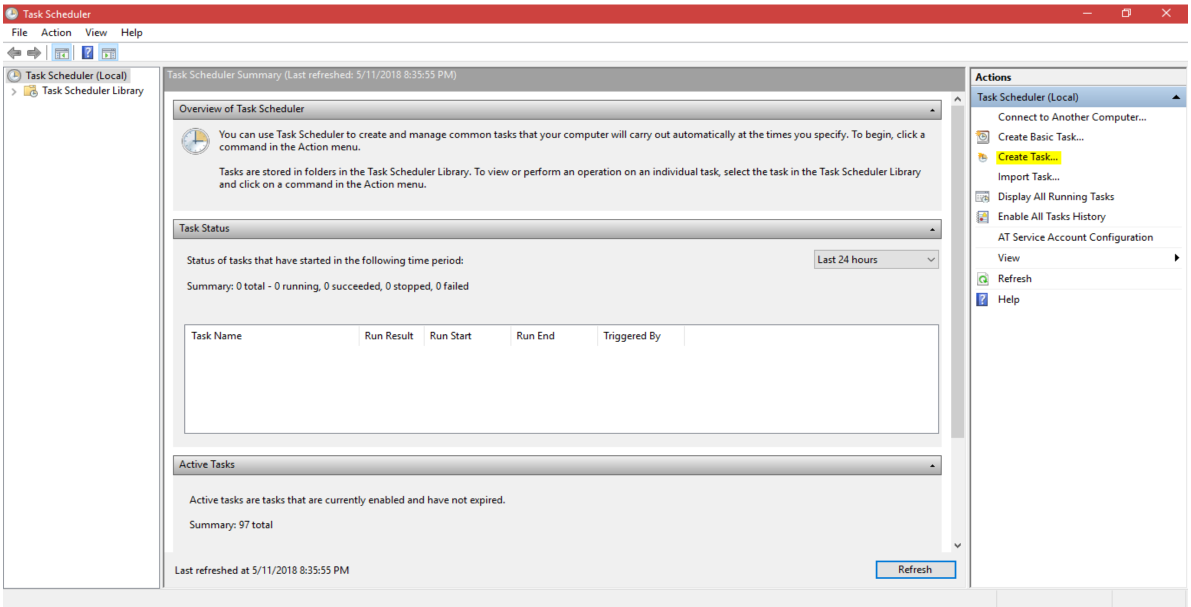
| Name | Date modified | Type | Size |
|---|---|---|---|
| Football_2018-02-10-00-00.json | 2/10/2018 12:01 AM | JSON File | 122 KB |
| Football_2018-02-10-01-00.json | 2/10/2018 1:01 AM | JSON File | 106 KB |
| Football_2018-02-10-02-00.json | 2/10/2018 2:01 AM | JSON File | 69 KB |
| Football_2018-02-10-03-00.json | 2/10/2018 3:01 AM | JSON File | 168 KB |
| Football_2018-02-10-04-00.json | 2/10/2018 4:01 AM | JSON File | 177 KB |
| Football_2018-02-10-05-00.json | 2/10/2018 5:01 AM | JSON File | 136 KB |
| Football_2018-02-10-06-00.json | 2/10/2018 6:01 AM | JSON File | 193 KB |
| Football_2018-02-10-07-00.json | 2/10/2018 7:01 AM | JSON File | 166 KB |
| Football_2018-02-10-08-00.json | 2/10/2018 8:01 AM | JSON File | 126 KB |
| Football_2018-02-10-09-00.json | 2/10/2018 9:01 AM | JSON File | 133 KB |
| Football_2018-02-10-10-00.json | 2/10/2018 10:01 AM | JSON File | 134 KB |
| Football_2018-02-10-11-00.json | 2/10/2018 11:01 AM | JSON File | 158 KB |
| Football_2018-02-10-12-00.json | 2/10/2018 12:01 PM | JSON File | 187 KB |
| Football_2018-02-10-14-00.json | 2/10/2018 2:01 PM | JSON File | 186 KB |
| Football_2018-02-10-15-00.json | 2/10/2018 3:01 PM | JSON File | 179 KB |
| Football_2018-02-10-16-00.json | 2/10/2018 4:01 PM | JSON File | 184 KB |
| Football_2018-02-10-17-00.json | 2/10/2018 5:01 PM | JSON File | 210 KB |
| Football_2018-02-10-18-00.json | 2/10/2018 6:01 PM | JSON File | 161 KB |
| Football_2018-02-10-19-00.json | 2/10/2018 7:01 PM | JSON File | 226 KB |
| Football_2018-02-10-20-00.json | 2/10/2018 8:01 PM | JSON File | 468 KB |
| Football_2018-02-10-21-00.json | 2/10/2018 9:01 PM | JSON File | 245 KB |
| Football_2018-02-10-22-00.json | 2/10/2018 10:01 PM | JSON File | 120 KB |
| Football_2018-02-10-23-00.json | 2/10/2018 11:01 PM | JSON File | 232 KB |
| Football_2018-02-11-00-00.json | 2/11/2018 12:01 AM | JSON File | 201 KB |
| Football_2018-02-11-01-00.json | 2/11/2018 1:01 AM | JSON File | 162 KB |
| Football_2018-02-11-02-00.json | 2/11/2018 2:01 AM | JSON File | 169 KB |
| Football_2018-02-11-03-00.json | 2/11/2018 3:01 AM | JSON File | 162 KB |
| Football_2018-02-11-04-00.json | 2/11/2018 4:01 AM | JSON File | 203 KB |

- After creating these two folders open the shell of your VM and create two folders there. One is for receiving json files from the laptop and the other one is for storing the json files after they have been sent from here to the hdfs (covered in the next section).



```
[root@sandbox ~]# mkdir tweet_received
[root@sandbox ~]# mkdir tweet_processed
[root@sandbox ~]# ls
anaconda-ks.cfg  hiveql.hql          sandbox.info     tweet_processed
blueprint.json   id_rsa              start_ambari.sh  tweet_received
build.out        install.log         start_hbase.sh
derby.log        install.log.syslog  start_solr.sh
hi.json          proj_cron.sh        stop_solr.sh
[root@sandbox ~]#
```

- Now get back to your laptop and open up Windows Task Scheduler. It is a default program of Windows and is already installed into your system. Click on Create Task on the right hand side of the window to create a new task.



Task Scheduler Summary (Last refreshed: 5/11/2018 8:35:55 PM)

Overview of Task Scheduler

You can use Task Scheduler to create and manage common tasks that your computer will carry out automatically at the times you specify. To begin, click a command in the Action menu.

Tasks are stored in folders in the Task Scheduler Library. To view or perform an operation on an individual task, select the task in the Task Scheduler Library and click on a command in the Action menu.

Task Status

Status of tasks that have started in the following time period: Last 24 hours

Summary: 0 total - 0 running, 0 succeeded, 0 stopped, 0 failed

| Task Name | Run Result | Run Start | Run End | Triggered By |
|---|---|---|---|---|

Active Tasks

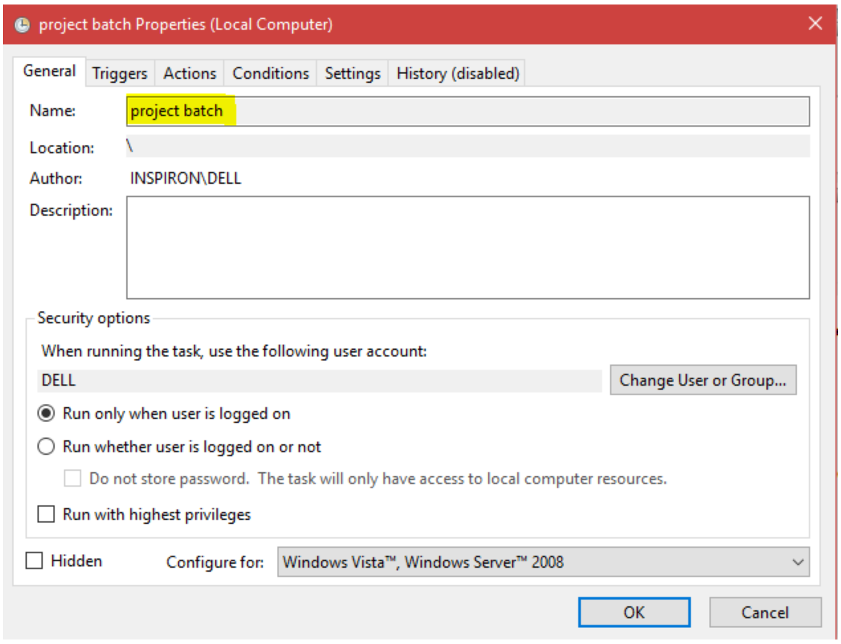
Active tasks are tasks that are currently enabled and have not expired.

Summary: 97 total

Last refreshed at 5/11/2018 8:35:55 PM

Refresh

- On the General tab enter the name you want to give to the task



project batch Properties (Local Computer)

Name: project batch

Location: \

Author: INSPIRON\DELL

Description:

Security options

When running the task, use the following user account:

DELL

● Run only when user is logged on

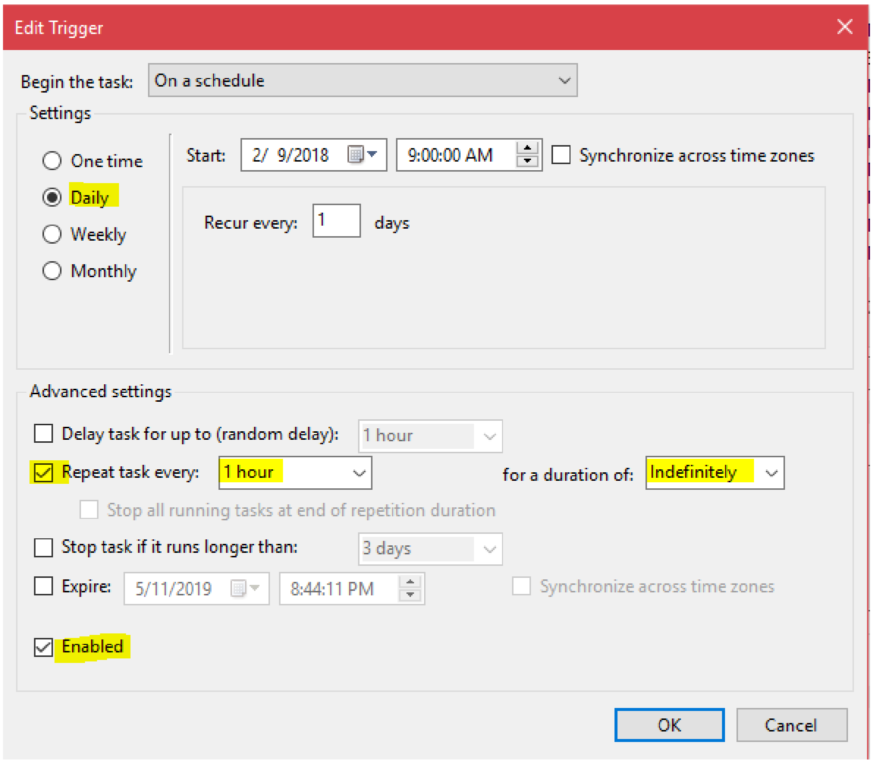
○ Run whether user is logged on or not

□ Do not store password. The task will only have access to local computer resources.

□ Run with highest privileges

□ Hidden Configure for: Windows Vista™, Windows Server™ 2008

- On the Triggers tab Click on New to create a new Trigger. Set scheduler to Daily, Repeat to one hour and indefinitely and check enabled as shown in the screenshot below. Click OK and the task will appear in the Triggers tab.



Edit Trigger

Begin the task: On a schedule

Settings

○ One time
● Daily
○ Weekly
○ Monthly

Start: 2/ 9/2018 9:00:00 AM □ Synchronize across time zones

Recur every: 1 days

Advanced settings

□ Delay task for up to (random delay): 1 hour

☑ Repeat task every: 1 hour for a duration of: Indefinitely

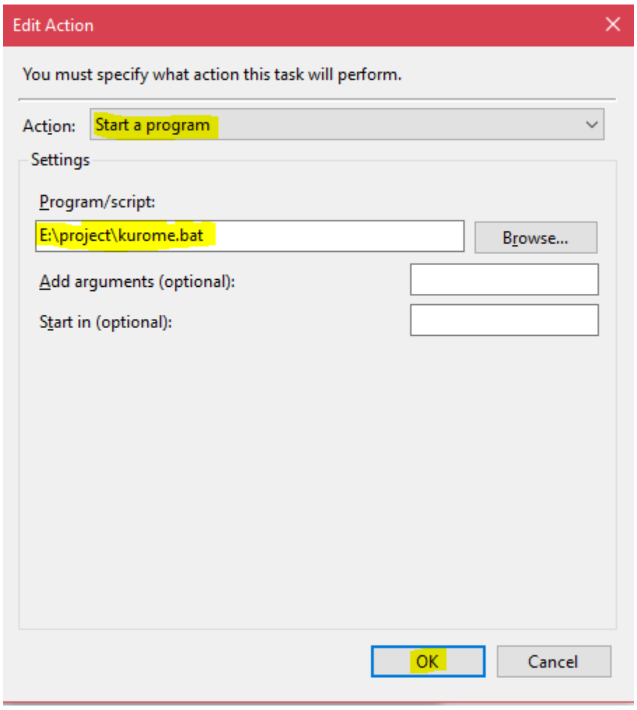
□ Stop all running tasks at end of repetition duration

□ Stop task if it runs longer than: 3 days

□ Expire: 5/11/2019 8:44:11 PM □ Synchronize across time zones

☑ Enabled

- Go to the Actions tab and Click New. Choose action as Start a Program and give the path to the batch file in the field specified. Click Ok to save it.



Edit Action

You must specify what action this task will perform.

Action: Start a program

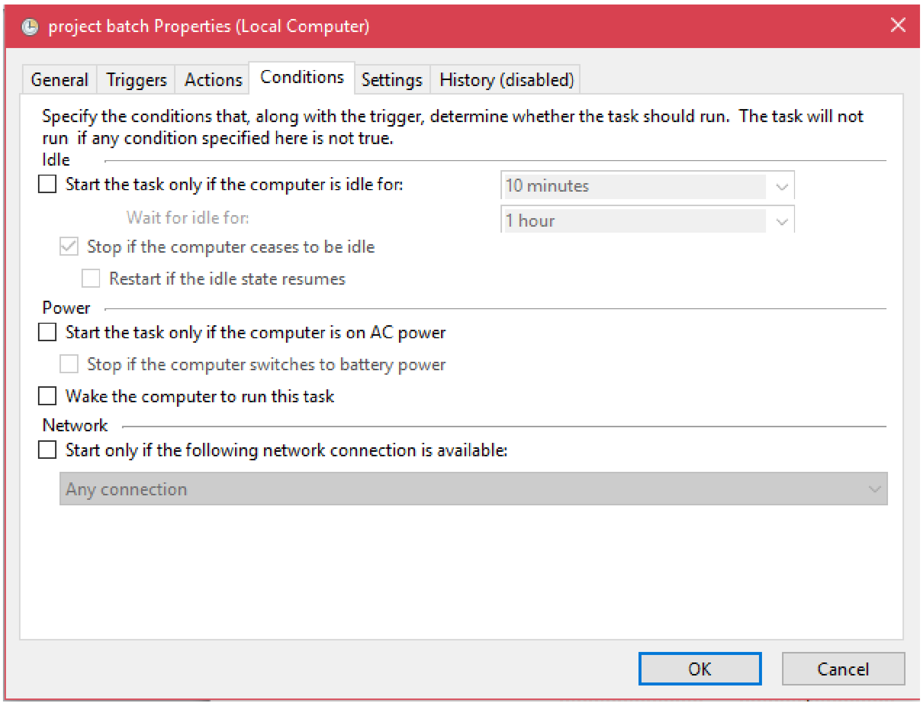
Settings

Program/script: E:\project\kurome.bat

Add arguments (optional):

Start in (optional):

- On the Conditions tab uncheck everything. Click OK to create the Task



project batch Properties (Local Computer)

Specify the conditions that, along with the trigger, determine whether the task should run. The task will not run if any condition specified here is not true.

Idle

□ Start the task only if the computer is idle for: 10 minutes

Wait for idle for: 1 hour

☑ Stop if the computer ceases to be idle

□ Restart if the idle state resumes

Power

□ Start the task only if the computer is on AC power

□ Stop if the computer switches to battery power

□ Wake the computer to run this task

Network

□ Start only if the following network connection is available:

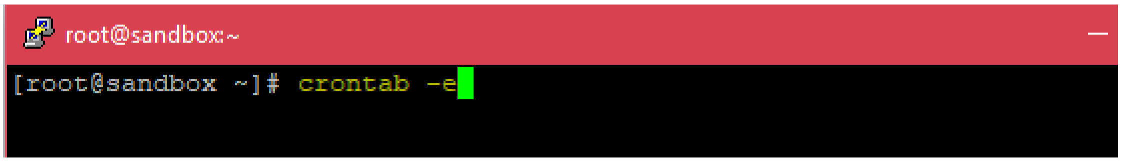
Any connection

According to the time you have set in the Triggers tab the task will activate and run the batch file. Make sure the IP address in the batch file is updated every time the VM is started as the IP address changes every time.

## 3.3. CRON jobs scheduler

Now that our json files are moving from the laptop to NDFS we now need to move the file from ndfs to hdfs and for that we will use the built-in CentOS scheduler known as CRON jobs.

- On the shell enter the command `crontab -e` to edit the cron list and add a new cronjob.

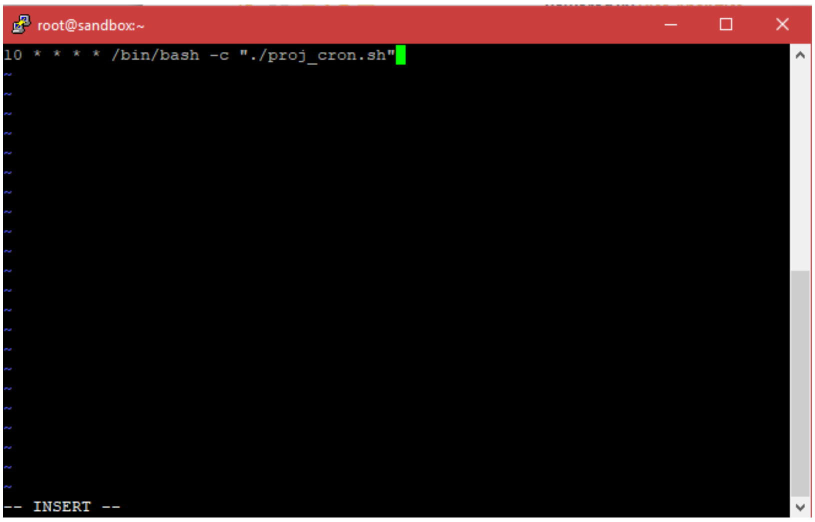


root@sandbox:~

```
[root@sandbox ~]# crontab -e
```

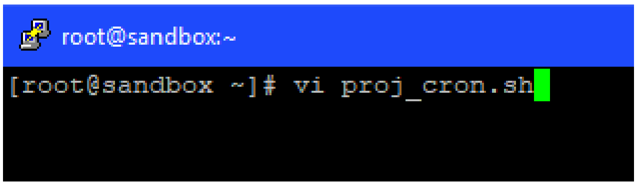
- The text editor will open up. Enter I to get into insert mode and add the following line:

```
10 * * * * /bin/bash -c "/<your-desired-file-name>.sh"
```



```
10 * * * * /bin/bash -c "./proj_cron.sh"
-- INSERT --
```

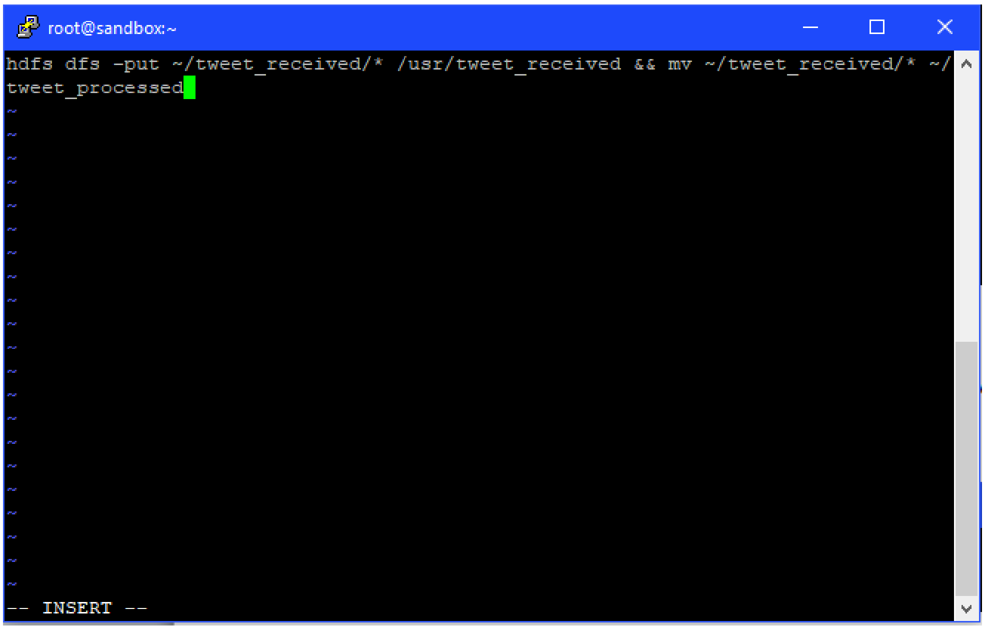
- **The above line shows that your scheduled job will run at 10th minute of every hour.**
- Press `Esc` and enter `:wq` to save the cron job. To view the installed cron job you can enter `crontab –l` in the shell.
- Now we need to create the .sh file we specified in the cron job above. In the root directory enter
  `vi <your-desired-file-name>.sh`



```
[root@sandbox ~]# vi proj_cron.sh
```

- A new text editor will open for editing the file. Press I to enter insert mode and add the following line:

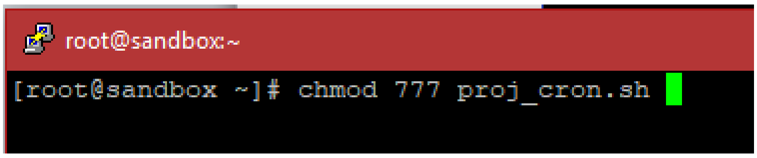
```
hdfs dfs -put ./<folder-where-json-files-are-received-from-laptop>/*
/<destination-folder-in-hdfs> && mv ~<folder-where-json-files-are-
received-from-laptop>/* ~/<folder-to-put-files-after-processing>
```



```
hdfs dfs -put ~/tweet_received/* /usr/tweet_received && mv ~/tweet_received/* ~/
tweet_processed
-- INSERT --
```

- Press `Esc` and enter `:wq` to save the file. Now you need to give execution permissions to the file. Enter the following line in the shell for that:

```
chmod 777 <shell-file-name>.sh
```



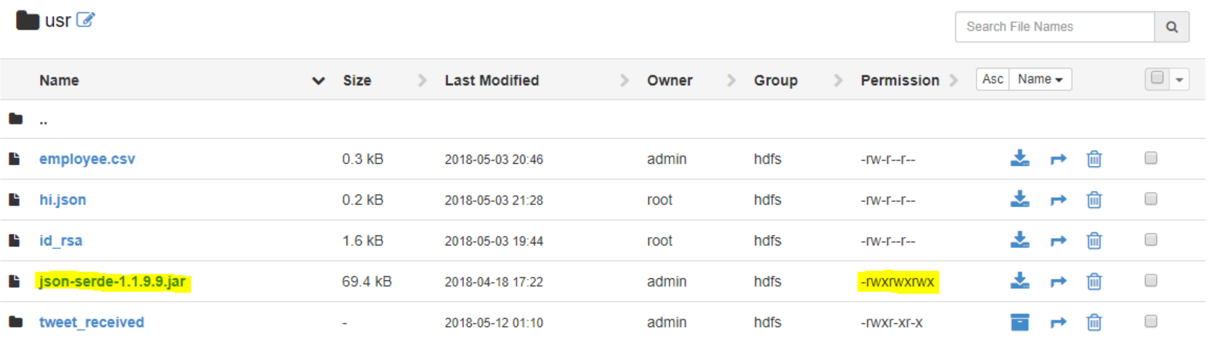
```
[root@sandbox ~]# chmod 777 proj_cron.sh
```

- Now we need to create the folder in hdfs where the json files would be stored. We have already specified the folder name and location in the shell command above. Simply create a folder using Ambari GUI with the required name at the required location.

## 3.4. Add SERDE (Skip this section (3.4) if you're using Hive's native JSON SERDE)

At this point we are receiving the json files in HDFS and hence we have data to load. We now need to load this data into a hive table. The first step in loading the data is to get a serde for json. You are already provided with the serde needed in the Whatsapp group.

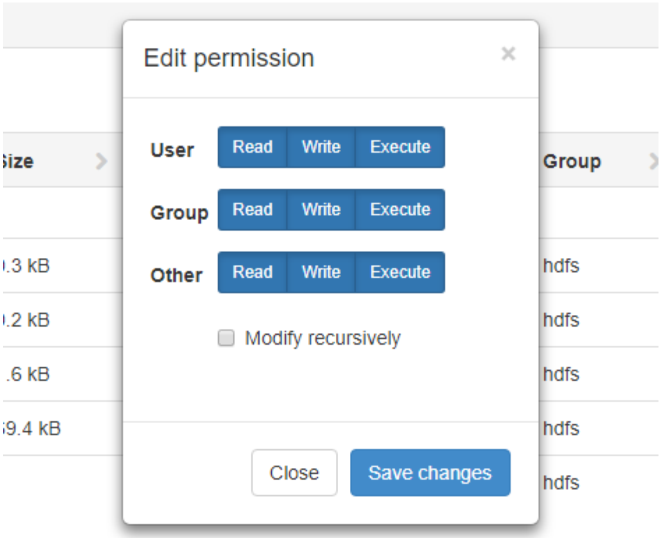
- Simply place the serde into a location in the HDFS using the Ambari GUI and give it full permissions.



usr

Search File Names

| Name | Size | Last Modified | Owner | Group | Permission |
|---|---|---|---|---|---|
| .. | | | | | |
| employee.csv | 0.3 kB | 2018-05-03 20:46 | admin | hdfs | -rw-r--r-- |
| hi.json | 0.2 kB | 2018-05-03 21:28 | root | hdfs | -rw-r--r-- |
| id_rsa | 1.6 kB | 2018-05-03 19:44 | root | hdfs | -rw-r--r-- |
| json-serde-1.1.9.9.jar | 69.4 kB | 2018-04-18 17:22 | admin | hdfs | -rwxrwxrwx |
| tweet_received | - | 2018-05-12 01:10 | admin | hdfs | -rwxr-xr-x |



Edit permission

User Read Write Execute

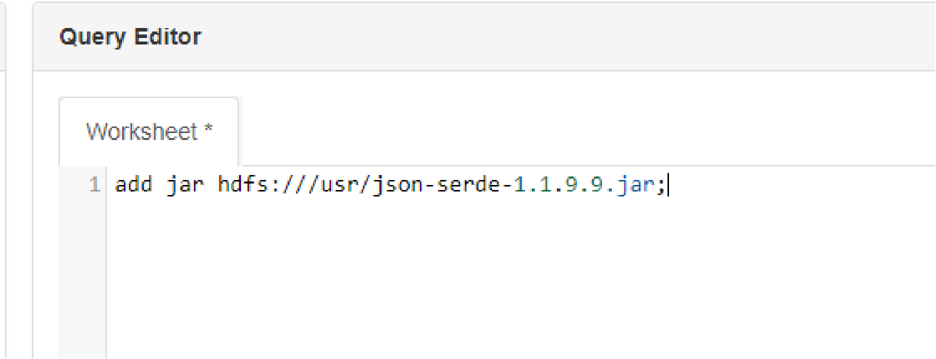
Group Read Write Execute

Other Read Write Execute

□ Modify recursively

- Now go to Hive view and enter the following command to add the serde to the worksheet.

```
add jar hdfs:///<path-to-serde>/<name-of-serde>.jar;
```



Query Editor

Worksheet *

```
add jar hdfs:///usr/json-serde-1.1.9.9.jar;
```

NOTE: Remember that using this method the serde will only be added for the duration that the worksheet is active. Once the Hive view is closed the serde will automatically be removed and will have to be added next time.

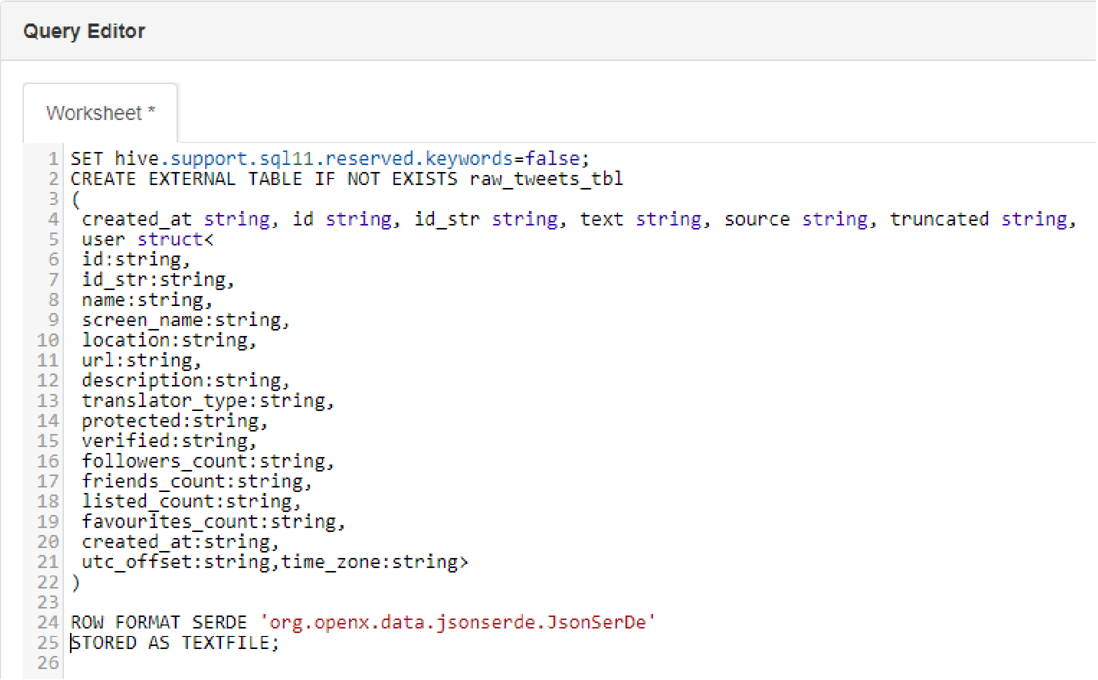
## 3.5. Create Hive table

Now we can create the hive table to load tweets in it. In the query below I have only selected some columns from the json file to show that we can just load the columns we require later in the process. Use the query or modify it to create the table where tweets will be loaded.

```
SET hive.support.sql11.reserved.keywords=false;
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS <your-table-name>(created_at string,
id string, id_str string, text string, source string, truncated string,
user struct< id:string, id_str:string, name:string, screen_name:string,
location:string, url:string,description:string, translator_type:string,
protected:string, verified:string, followers_count:string,
friends_count:string, listed_count:string, favourites_count:string,
created_at:string, utc_offset:string,time_zone:string>)
```

```
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe' STORED AS TEXTFILE;
```



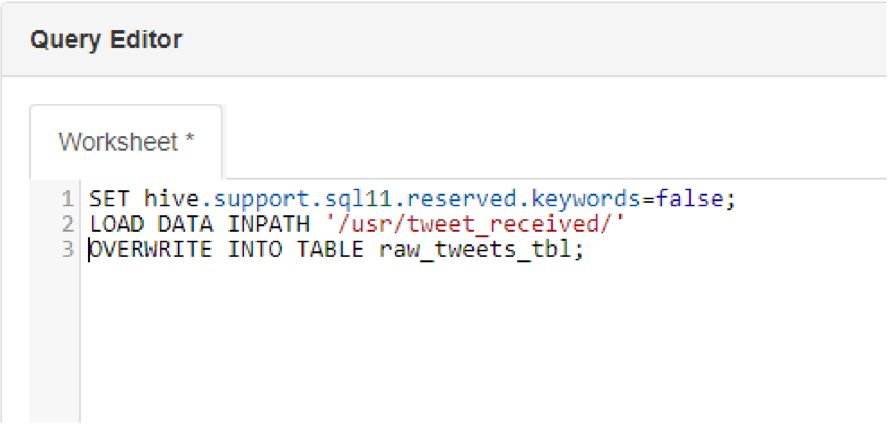
```
SET hive.support.sql11.reserved.keywords=false;
CREATE EXTERNAL TABLE IF NOT EXISTS raw_tweets_tbl
(
 created_at string, id string, id_str string, text string, source string, truncated string,
 user struct<
 id:string,
 id_str:string,
 name:string,
 screen_name:string,
 location:string,
 url:string,
 description:string,
 translator_type:string,
 protected:string,
 verified:string,
 followers_count:string,
 friends_count:string,
 listed_count:string,
 favourites_count:string,
 created_at:string,
 utc_offset:string,time_zone:string>
)

ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
STORED AS TEXTFILE;

```

## 3.6. Load Data into Hive Table

Now we will load the tweets into the hive table we created above. Use the following command to load the tweets.

```
SET hive.support.sql11.reserved.keywords=false;
```

```
LOAD DATA INPATH '/<path-to-tweets-folder/' OVERWRITE INTO TABLE <your-
table-name>;
```

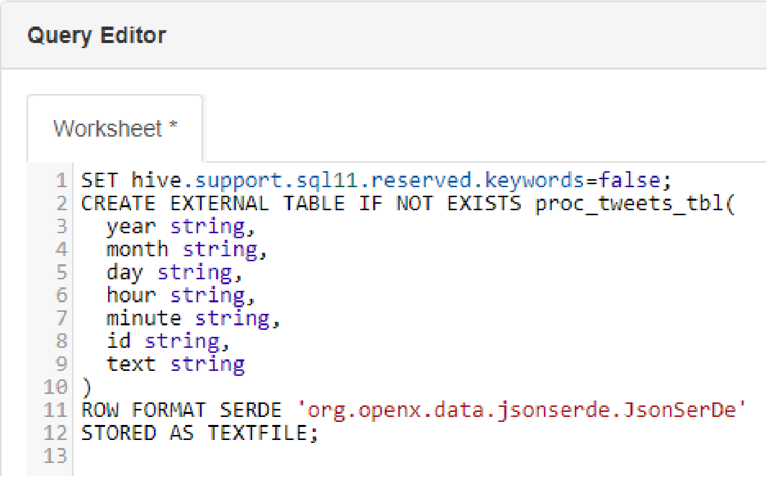


```
SET hive.support.sql11.reserved.keywords=false;
LOAD DATA INPATH '/usr/tweet_received/'
OVERWRITE INTO TABLE raw_tweets_tbl;
```

## 3.7. Create processed table

The next step of the process is to load the table we just created into Pig and extract year, month, day, hour, minute, id and text. We will then store these into a new hive table which we need to create before going to Pig as we need a place to store the output of Pig. Use the following lines to create the table.

```
SET hive.support.sql11.reserved.keywords=false;

CREATE EXTERNAL TABLE IF NOT EXISTS <your-table-name>(year string, month
string, day string, hour string, minute string, id string, text string)

ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe' STORED AS TEXTFILE;
```



```
SET hive.support.sql11.reserved.keywords=false;
CREATE EXTERNAL TABLE IF NOT EXISTS proc_tweets_tbl(
  year string,
  month string,
  day string,
  hour string,
  minute string,
  id string,
  text string
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
STORED AS TEXTFILE;

```

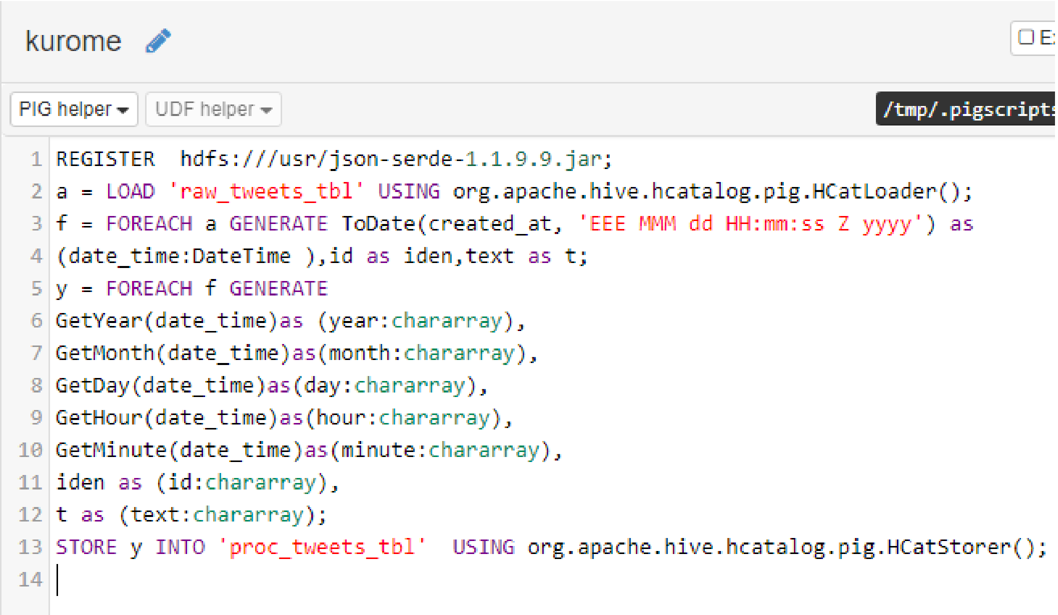
## 3.8. Load Hive Table to Pig and Process it

Next we upload the hive table into pig and process it. We will extract only the data we need from the tweets and store it in the new table we just created. In addition, we will extract the created_at column and split it to get 5 separate columns of year, month, date, hour and minutes.

- Open the Pig View and create a new script. Give it the name you want and create it.

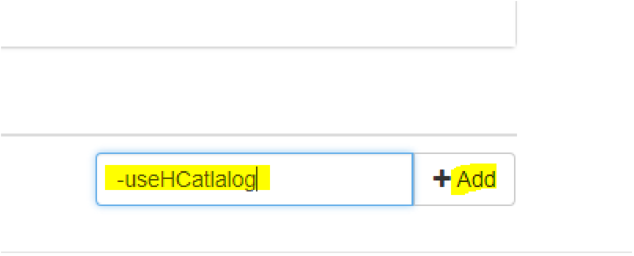
- Open the script and paste the following code in it changing the code where required.

```
REGISTER hdfs:///<serde-path>/<serde-name>.jar;

a = LOAD '<tweets-table-name>' USING
org.apache.hive.hcatalog.pig.HCatLoader();
f = FOREACH a GENERATE ToDate(created_at, 'EEE MMM dd HH:mm:ss Z yyyy') as
(date_time:DateTime ),id as iden,text as t;

y = FOREACH f GENERATE
GetYear(date_time)as (year:chararray),
GetMonth(date_time)as(month:chararray),
GetDay(date_time)as(day:chararray),
GetHour(date_time)as(hour:chararray),
GetMinute(date_time)as(minute:chararray),
iden as id,
t as text;

STORE y INTO '<processed-table-name>'  USING
org.apache.hive.hcatalog.pig.HCatStorer();
```

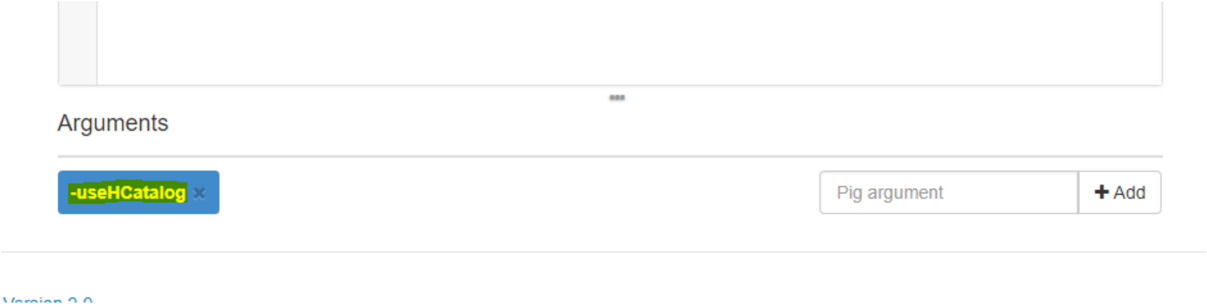


```
REGISTER  hdfs:///usr/json-serde-1.1.9.9.jar;
a = LOAD 'raw_tweets_tbl' USING org.apache.hive.hcatalog.pig.HCatLoader();
f = FOREACH a GENERATE ToDate(created_at, 'EEE MMM dd HH:mm:ss Z yyyy') as
(date_time:DateTime ),id as iden,text as t;
y = FOREACH f GENERATE
GetYear(date_time)as (year:chararray),
GetMonth(date_time)as(month:chararray),
GetDay(date_time)as(day:chararray),
GetHour(date_time)as(hour:chararray),
GetMinute(date_time)as(minute:chararray),
iden as (id:chararray),
t as (text:chararray);
STORE y INTO 'proc_tweets_tbl'  USING org.apache.hive.hcatalog.pig.HCatStorer();

```

- Scroll to the end of the page and add the argument -`useHCatalog` otherwise the HCatalog service will not be accessible.



-useHCatlalog

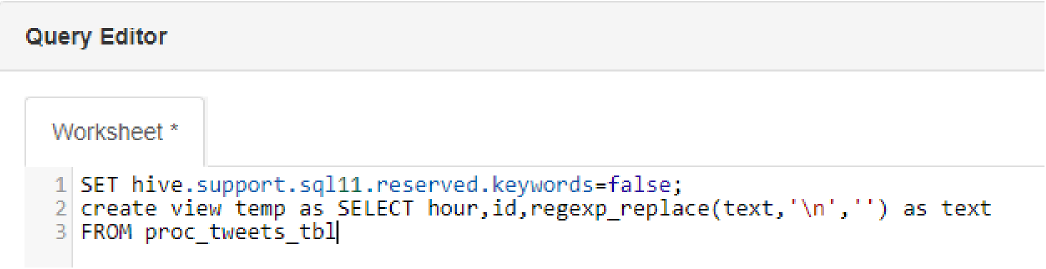


- After this execute the script. You can check Execute on Tez to use the Tez execution engine. Otherwise it will run on MapReduce.

## 3.9. Remove new line characters in hive

The next step is to use Hive's explode function on the text column (explode function is explained in the next section). Before that, however, we need to remove all newline characters in the text column as they can cause unwanted behavior. Use the following query in Hive View to remove newline characters. **Make sure the Serde is added to the worksheet before executing this query.**

```
SET hive.support.sql11.reserved.keywords=false;

create view <view-name> as SELECT hour,id,regexp_replace(text,'\n','') as text

FROM <processed-table-name>;
```



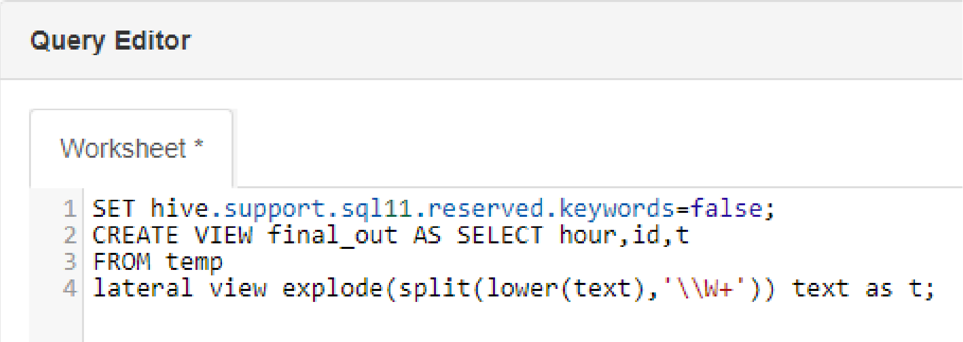
```
SET hive.support.sql11.reserved.keywords=false;
create view temp as SELECT hour,id,regexp_replace(text,'\n','') as text
FROM proc_tweets_tbl
```

## 3.10. Hive Explode Function

Next we will use hive's explode function. The explode function separate the words in a sentence into multiple rows. For example, if a row contains a sentence that has 5 words then then the explode function will create 5 rows with each word in the sentence in a different row. The other columns in the table will be replicated as they are in the 5 rows.

Use the following query to execute the explode function

```
SET hive.support.sql11.reserved.keywords=false;

CREATE VIEW <view-name> AS SELECT hour,id,t

FROM <view-name-created-in-last-step>

lateral view explode(split(lower(text),'\\W+')) text as t;
```



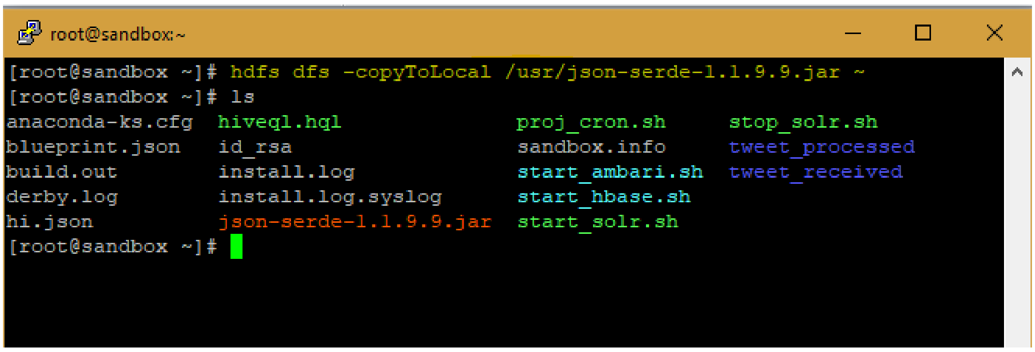
```
SET hive.support.sql11.reserved.keywords=false;
CREATE VIEW final_out AS SELECT hour,id,t
FROM temp
lateral view explode(split(lower(text),'\\W+')) text as t;
```

## 3.11. Spark Group By Function

After separating the words we will now take the data into spark and get insights from the data. Right now the following query will get us the **number of words used per hour** in these tweets. We can write more complex queries in spark and even create graphs from this data using Spark's GraphX library but that will be for another time.

- Go to the shell and enter the following command to get the serde into the ndfs. This step is done because the Spark only adds jars from the ndfs so we need to put the jar file in the ndfs.

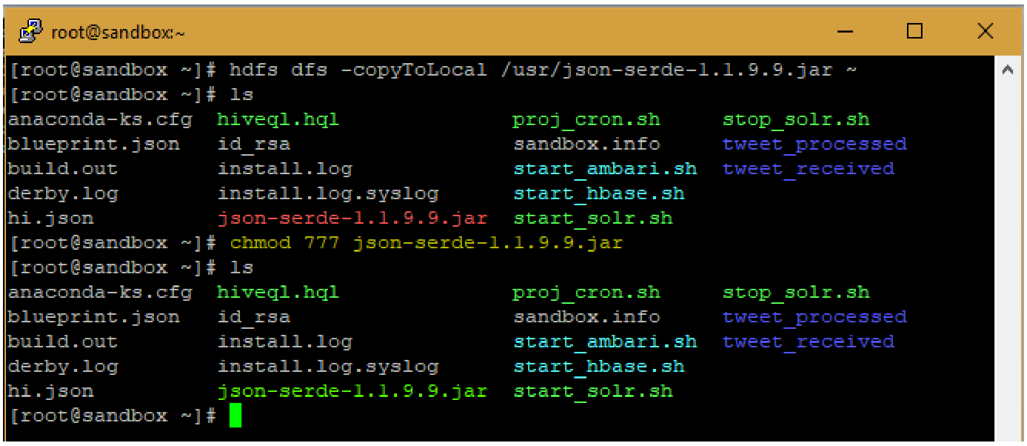
```
hdfs dfs -copyToLocal /usr/json-serde-1.1.9.9.jar ~
```



```
[root@sandbox ~]# hdfs dfs -copyToLocal /usr/json-serde-1.1.9.9.jar ~
[root@sandbox ~]# ls
anaconda-ks.cfg  hiveql.hql              proj_cron.sh     stop_solr.sh
blueprint.json   id_rsa                  sandbox.info     tweet_processed
build.out        install.log             start_ambari.sh  tweet_received
derby.log        install.log.syslog      start_hbase.sh
hi.json          json-serde-1.1.9.9.jar  start_solr.sh
[root@sandbox ~]#
```

- Give execution permission to the serde

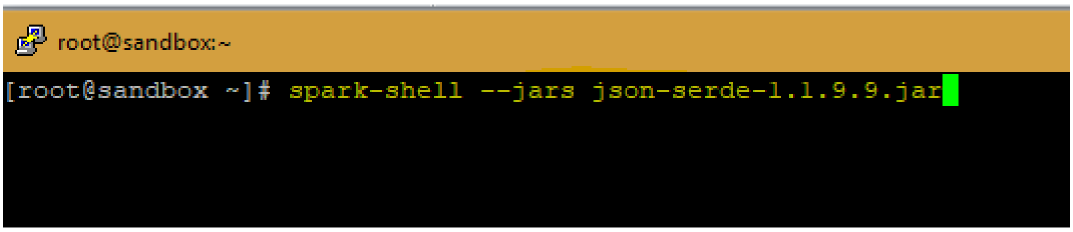
```
chmod 777 json-serde-1.1.9.9.jar
```



```
[root@sandbox ~]# hdfs dfs -copyToLocal /usr/json-serde-1.1.9.9.jar ~
[root@sandbox ~]# ls
anaconda-ks.cfg  hiveql.hql             proj_cron.sh     stop_solr.sh
blueprint.json   id_rsa                 sandbox.info     tweet_processed
build.out        install.log            start_ambari.sh  tweet_received
derby.log        install.log.syslog     start_hbase.sh
hi.json          json-serde-1.1.9.9.jar  start_solr.sh
[root@sandbox ~]# chmod 777 json-serde-1.1.9.9.jar
[root@sandbox ~]# ls
anaconda-ks.cfg  hiveql.hql             proj_cron.sh     stop_solr.sh
blueprint.json   id_rsa                 sandbox.info     tweet_processed
build.out        install.log            start_ambari.sh  tweet_received
derby.log        install.log.syslog     start_hbase.sh
hi.json          json-serde-1.1.9.9.jar  start_solr.sh
[root@sandbox ~]#
```

- Start the spark shell with the following command to add the serde to the spark classpath

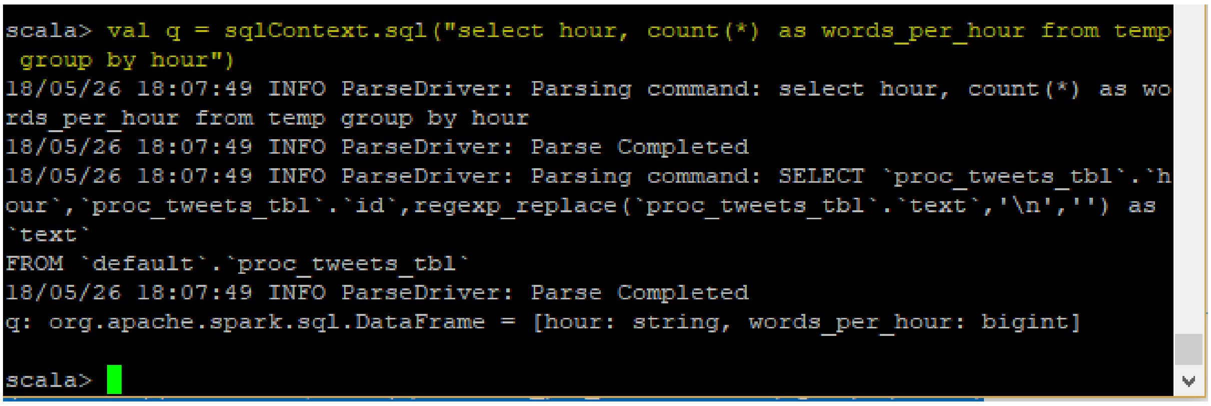
```
spark-shell --jars json-serde-1.1.9.9.jar
```



```
[root@sandbox ~]# spark-shell --jars json-serde-1.1.9.9.jar
```

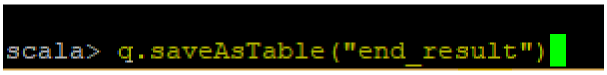
- After the shell has started enter the following commands to load the hive view, process it and store the result in a new hive table.

```
val q = sqlContext.sql("select hour, count(*) as words_per_hour from
final_out group by hour")
```



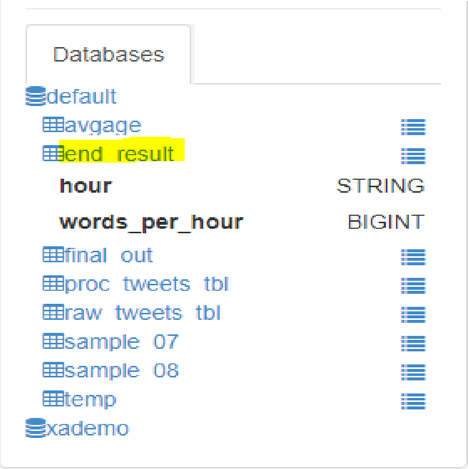
```
scala> val q = sqlContext.sql("select hour, count(*) as words_per_hour from temp
 group by hour")
18/05/26 18:07:49 INFO ParseDriver: Parsing command: select hour, count(*) as wo
rds_per_hour from temp group by hour
18/05/26 18:07:49 INFO ParseDriver: Parse Completed
18/05/26 18:07:49 INFO ParseDriver: Parsing command: SELECT `proc_tweets_tbl`.`h
our`,`proc_tweets_tbl`.`id`,regexp_replace(`proc_tweets_tbl`.`text`,'\n','') as
`text`
FROM `default`.`proc_tweets_tbl`
18/05/26 18:07:49 INFO ParseDriver: Parse Completed
q: org.apache.spark.sql.DataFrame = [hour: string, words_per_hour: bigint]

scala>
```

```
q.saveAsTable("end_result")
```



```
scala> q.saveAsTable("end_result")
```

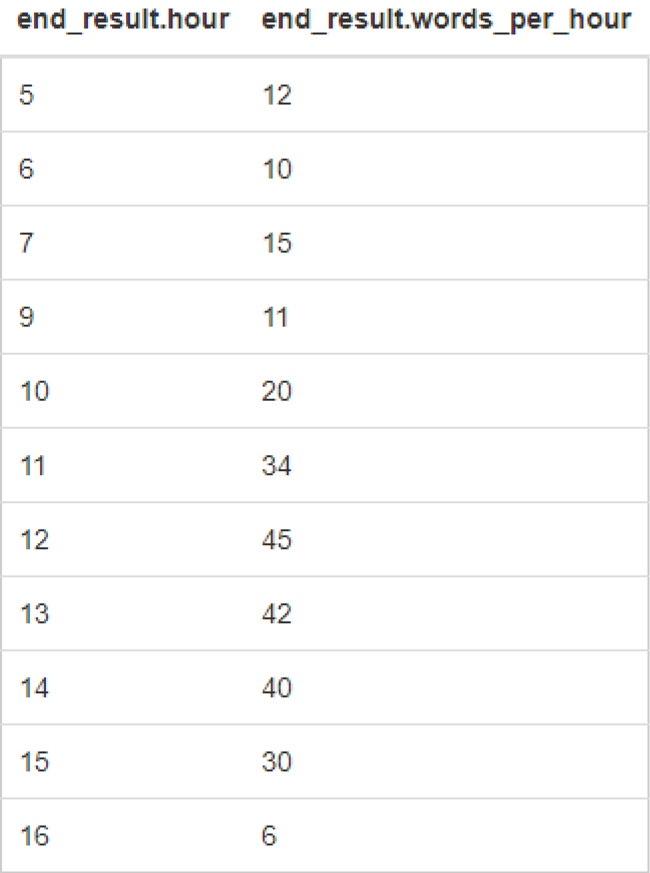
- After the table has been saved go to the Hive view through Ambari and see the created table.



Databases

- default
  - avgage
  - end result
    - hour STRING
    - words_per_hour BIGINT
  - final out
  - proc tweets tbl
  - raw tweets tbl
  - sample 07
  - sample 08
  - temp
- xademo

- Contents of the table:



| end_result.hour | end_result.words_per_hour |
|---|---|
| 5 | 12 |
| 6 | 10 |
| 7 | 15 |
| 9 | 11 |
| 10 | 20 |
| 11 | 34 |
| 12 | 45 |
| 13 | 42 |
| 14 | 40 |
| 15 | 30 |
| 16 | 6 |