Using HDP and HDF to Harness the Power of Social Media

This lab will walk through the collection and analysis of Twitter data leveraging the Hortonworks Data Platform (HDP) and Hortonworks DataFlow (HDF), combining both data in motion and data at rest. While the initial analytics are simple, this pattern enables the building blocks for more advanced use cases.

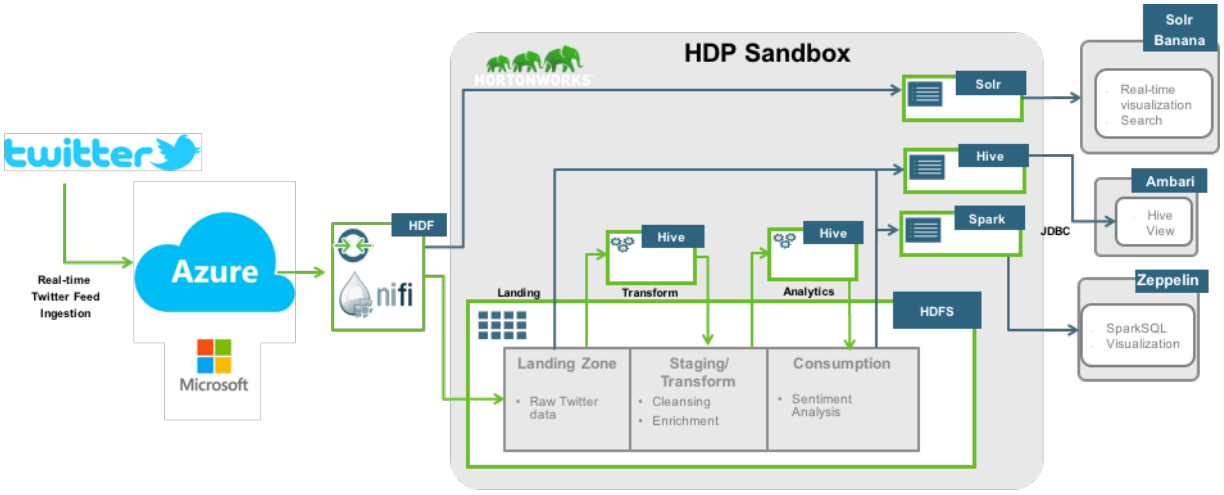
This lab goes through the preparation of the sandbox for the implementation of the sentiment analysis use case where participants will get to install and configure some components using the Ambari Administration console.

Participants will then be able to perform data ingestion, transformation and filtering using Apache NiFi.

Once the data is ingested, participants will be able to visualize data in real-time using a Banana dashboard. They will then use different analytics tools available on HDP such as Hive, SparkSQL and Zeppelin to analyze and visualize the data.

Overview

Below is a conceptual diagram of the implementation of the Twitter sentiment analysis performed in this lab using the Hortonworks Data Flow and the Hortonworks Data Platform Sandbox



Pre-requisites

Where possible, the following pre-requisites should be completed prior to start of the lab. Please follow the instructions in the Lab prerequisites document. Prerequisites steps included in the documents are:

- The creation of a Azure Pass subscription account.
- The creation of a Twitter developer account.
- The provisioning of the Hortonworks Sandbox 2.5 from the Microsoft Azure marketplace.

Once these prerequisites steps are done, please make sure your VM is started on Azure. If the VM is down, follow the same instructions that you used to navigate in Microsoft Azure in the prerequisites document to shut it down, but click on Start instead of Stop.

Once the VM is up, you should be able to open an ssh session as following:

The commands in this lab should be executed as the root user. In order to login to the sandbox, first ssh to the VM using the credentials (user & password) and the hostname you chose when you provisioned your sandbox from the Azure Marketplace. This will get you to the Docker host. From there, you need to ssh to the Docker container as following:

a. **ssh root@127.0.0.1 -p 2222**
b. Password is: **hadoop**
c. You will be prompted to change the password. Please use **Welcome2lab!**

Prepare the HDP 2.5 Sandbox for the lab

If not already completed, change the Ambari admin password. It is recommended to use the following password: Welcome2lab!

Change the Ambari admin password

```
# Reset the Ambari admin password
ambari-admin-password-reset

# Restart Ambari agent
service ambari-agent restart
```

Log into the Ambari Web Interface at http://<public ip>:8080/ using the new credentials. Note, it may take several minutes for the Ambari Web Interface to become available after restart.

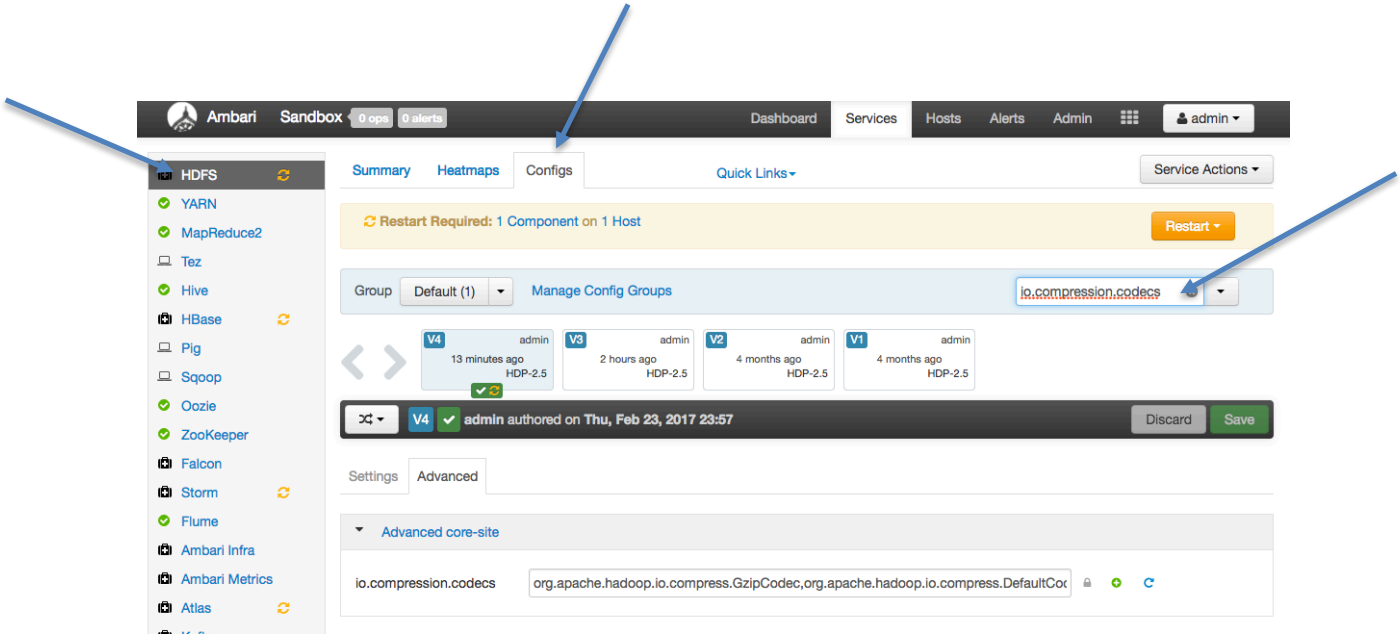
Install ntp on the sandbox

On the ssh session opened previously as root, please run the following commands:

```
yum install -y ntp
ntpdate pool.ntp.org
service ntpd start
```

Customize io.compression.codecs configuration

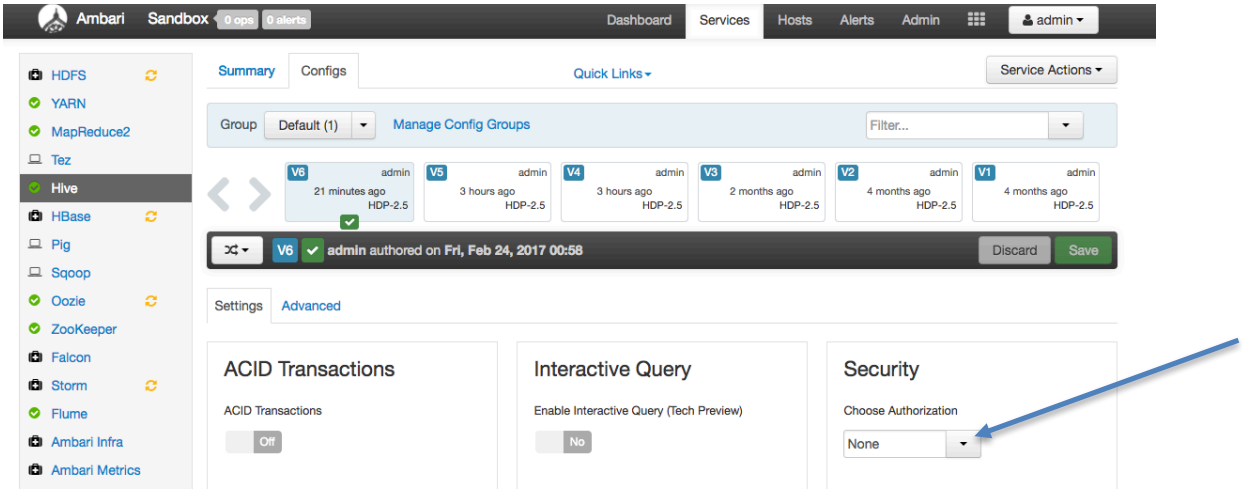
This configuration is a workaround to avoid a missing lzo compression library with Apache NiFi PutHDFS Processor. Click on HDFS, then on Configs tab, then look for io.compression.codecs parameter. Click on the configuration value and remove "`com.hadoop.compression.lzo.LzoCodec,com.hadoop.compression.lzo.LzopCodec`" from "`io.compression.codecs`" property
Save your configuration.



This configuration change requires restarting a few services to take effect. However, there are additional configuration changes at a step later which will restart the required services.

Disable Ranger Security for Hive

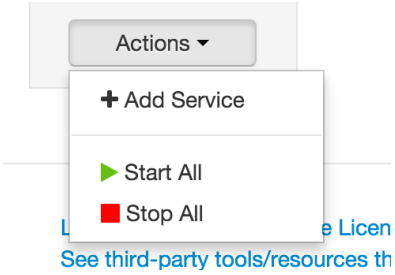
Click on Hive service, Config tab, and change Security setting “Choose Authorization” to None:



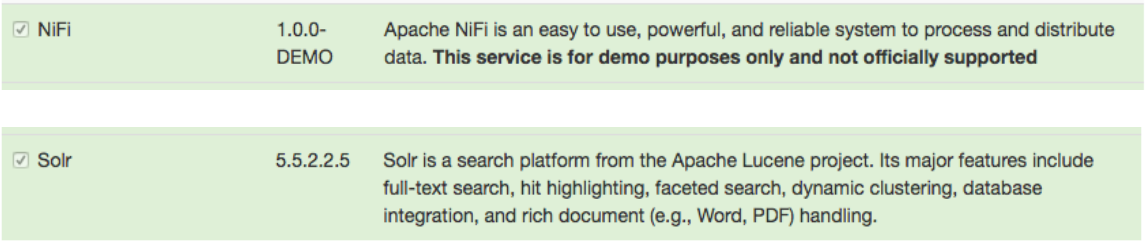
This configuration change requires restarting Hive to take effect. However, there are additional configuration changes at the next step which will restart Hive.

Install HDF (NiFi) and Solr via an Ambari Service

Open a browser and navigate to Ambari via <public ip>:8080. Login to Ambari as admin. On the bottom of the panel to the left, navigate to Actions -> Add Service.



On the Add Service wizard, ensure the checkbox next to NiFi and Solr is selected.



As this is a single node Sandbox, leave the defaults on the Assign Masters panel.

Finally, select Deploy and wait for Ambari to install and start HDF (NiFi) and Solr. Once the install completes, select Next and then Complete. Back to the Ambari Dashboard, make sure both NiFi and Solr are started.

Increase Memory Available for Hive, Tez, and YARN

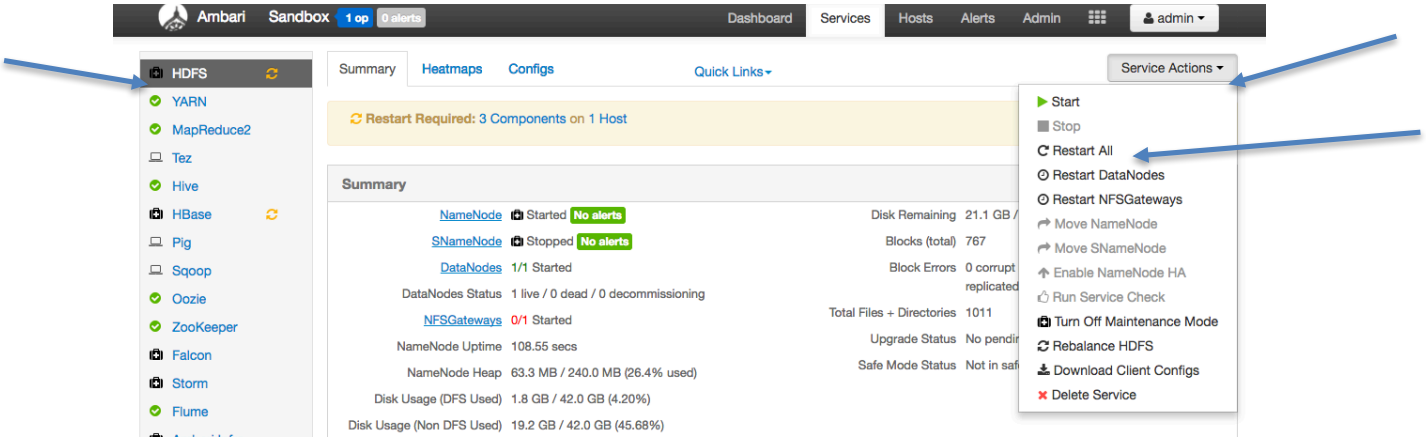
Sandbox is intended to run on minimal hardware requirements, as a result, the Hive, Tez, and YARN configurations are less than ideal out of the box. Run the following script to increase the memory available to Hive, Tez, and YARN.

Run the optimize_hive_config script:

```
# Increase Hive, Tez, and YARN memory
bash <(curl -s https://raw.githubusercontent.com/sakserv/twitter-nifi-lab-setup/master/optimize_hive_config.sh)
```

Restart HDFS and MapReduce2 Services

Identify HDFS and MapReduce2 services on the left-hand side of the Ambari console where all services appear (orange recycle icon) after services have been recycled by the previous script. For HDFS, click on the Service Actions button on the right-hand side, then select Restart All. Repeat the same process for MapReduce2 service.



Configure Solr

Solr enables the ability to search across large corpuses of information through specialized indexing techniques. Solr will be combined with Banana, a visualization tool, to create search driven dashboards of Twitter data.

Download the Banana Dashboard

Banana is a tool for creating dashboards on Solr data. Download and install the pre-built twitter dashboard.

```
# Download and install the Banana dashboard
wget https://raw.githubusercontent.com/abajwa-hw/ambari-nifi-service/master/demofiles/default.json -O /opt/lucidworks-hdpsearch/solr/server/solr-webapp/webapp/banana/app/dashboards/default.json
```

Download the Solr Configuration

Due to the timestamp used by Twitter, it is necessary to update the Solr configuration to support this additional timestamp format.

```
# Download and install the updated Solr configuration
wget https://raw.githubusercontent.com/sakserv/twitter-nifi-solrconfig/master/solrconfig.xml -O /opt/lucidworks-hdpsearch/solr/server/solr/configsets/data_driven_schema_configs/conf/solrconfig.xml
```

Create the Solr Collection

A Solr collection is the index and configuration associated with the data that will be stored in the search index. Also specified are the number of replicas and shards, as this is a single node cluster, only a single replica and shard are used.

```
# Create the Tweets Collection
/opt/lucidworks-hdpsearch/solr/bin/solr create -c tweets -d data_driven_schema_configs -s 1 -rf 1
```

Using Apache NiFi for ingesting Twitter Data

HDF (NiFi) has a powerful templating system to allow for easy sharing and consumption of data flows. The following imports the Twitter dataflow template and starts the collection of Tweets into Solr and HDFS.

Download the Twitter dataflow template

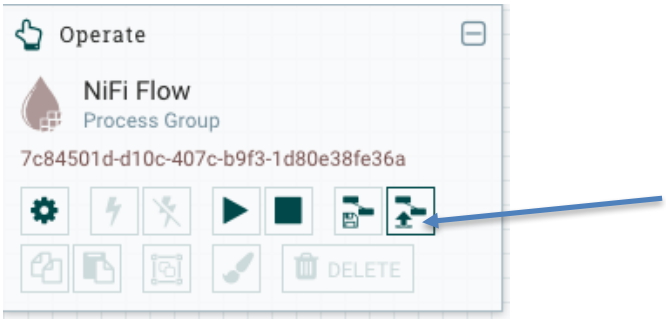
The Twitter dataflow template should be downloaded to the workstation accessing NiFi at the following link.

https://raw.githubusercontent.com/abajwa-hw/ambari-nifi-service/master/demofiles/Twitter_Dashboard.xml

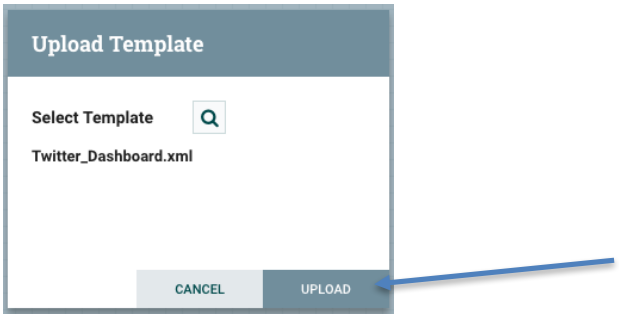
Import the Template

Navigate to the HDF (NiFi) web interface at *http://<public ip>:9090/nifi/* via a web browser.

Import the template by selecting the Import Template Icon found in the NiFi Flow Operate tile on the left hand side of the canvas.

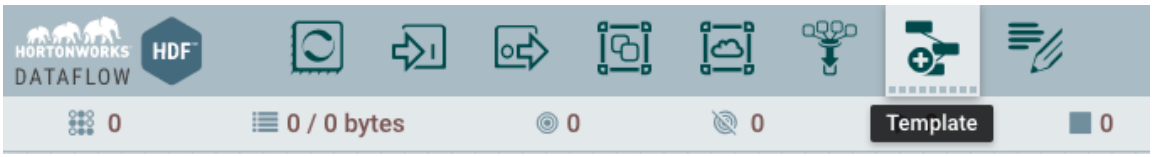


Click the Browse icon ( ) and navigate to where the Twitter_Dashboard.xml file was downloaded. Click on Upload.

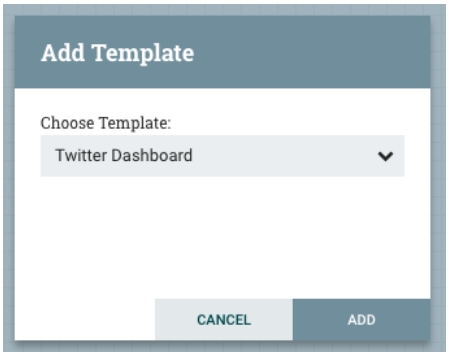


Instantiate the Template

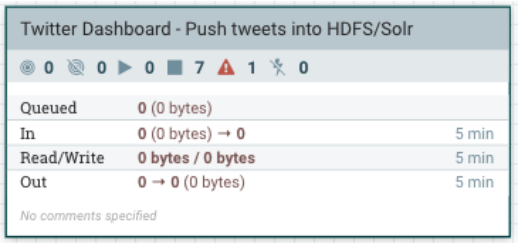
Drag and drop the template artifact onto the canvas.



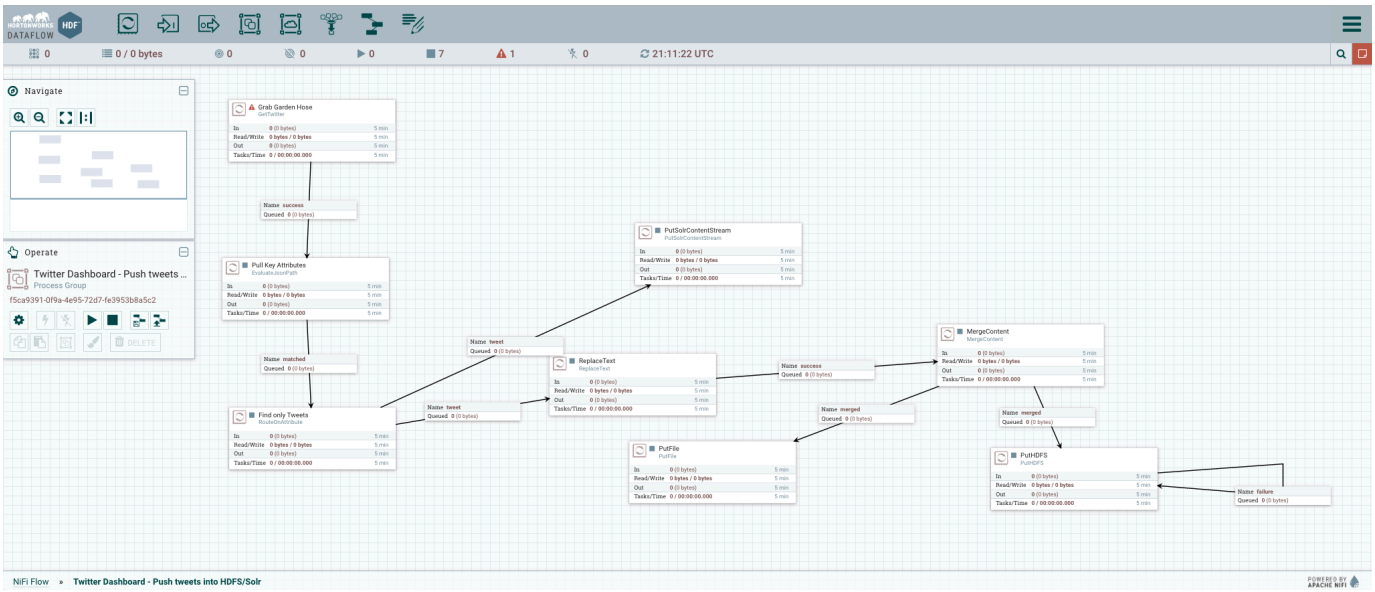
Select *Twitter Dashboard* and click *Add*



This will create a HDF (NiFi) *Process Group*.

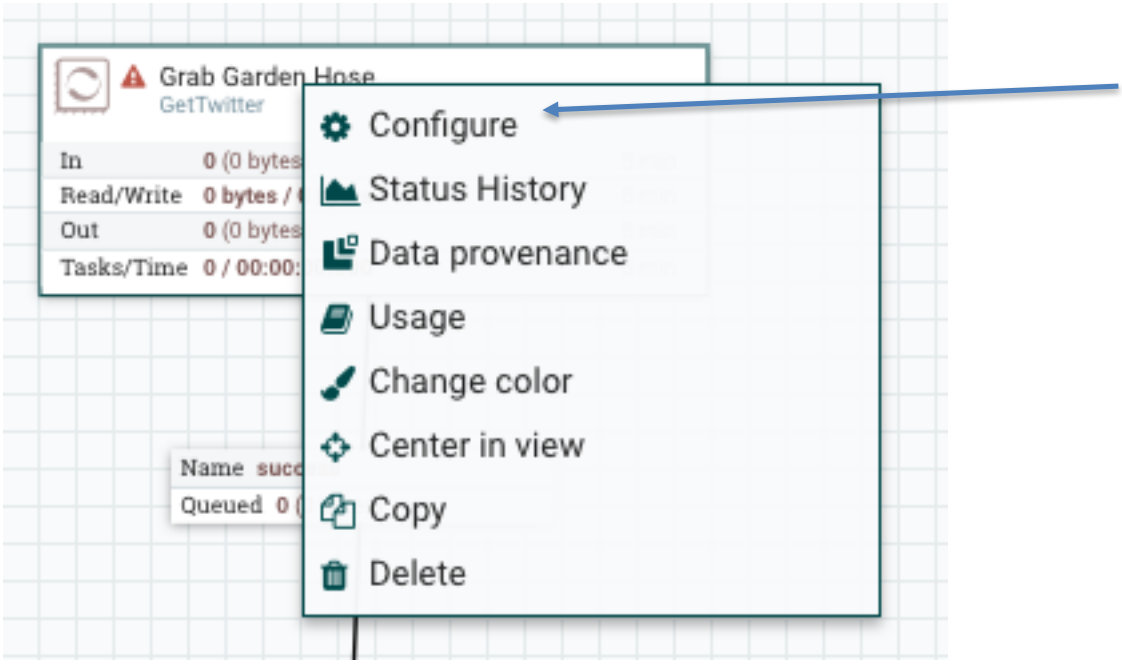


Double click on the Process Group just added to your canvas to drill down into the flow:



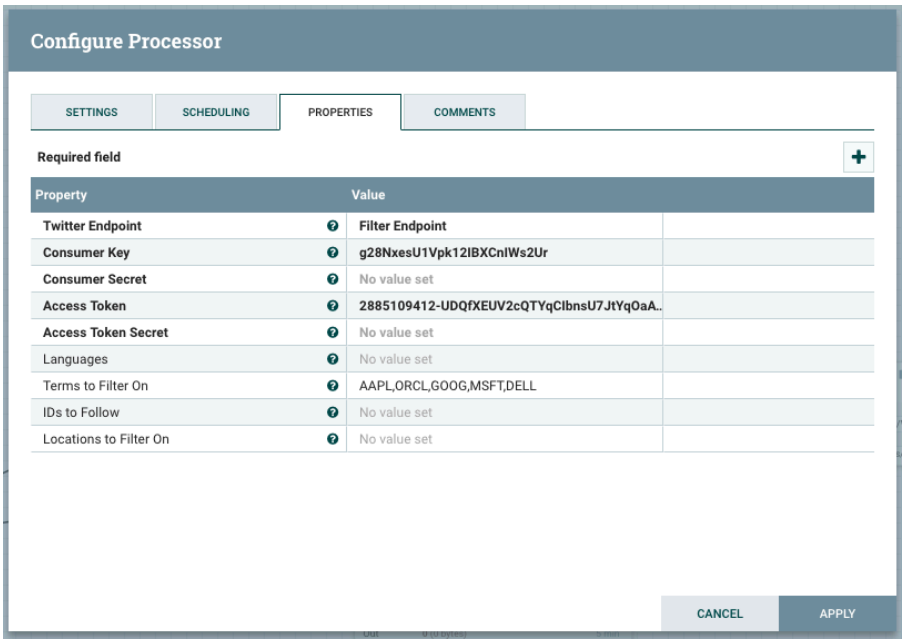
Configure the Twitter Grab Garden Hose Processor

Next, configure the Twitter Grab Garden Hose Processer with the Twitter API information. Right click the processor and select Configure.



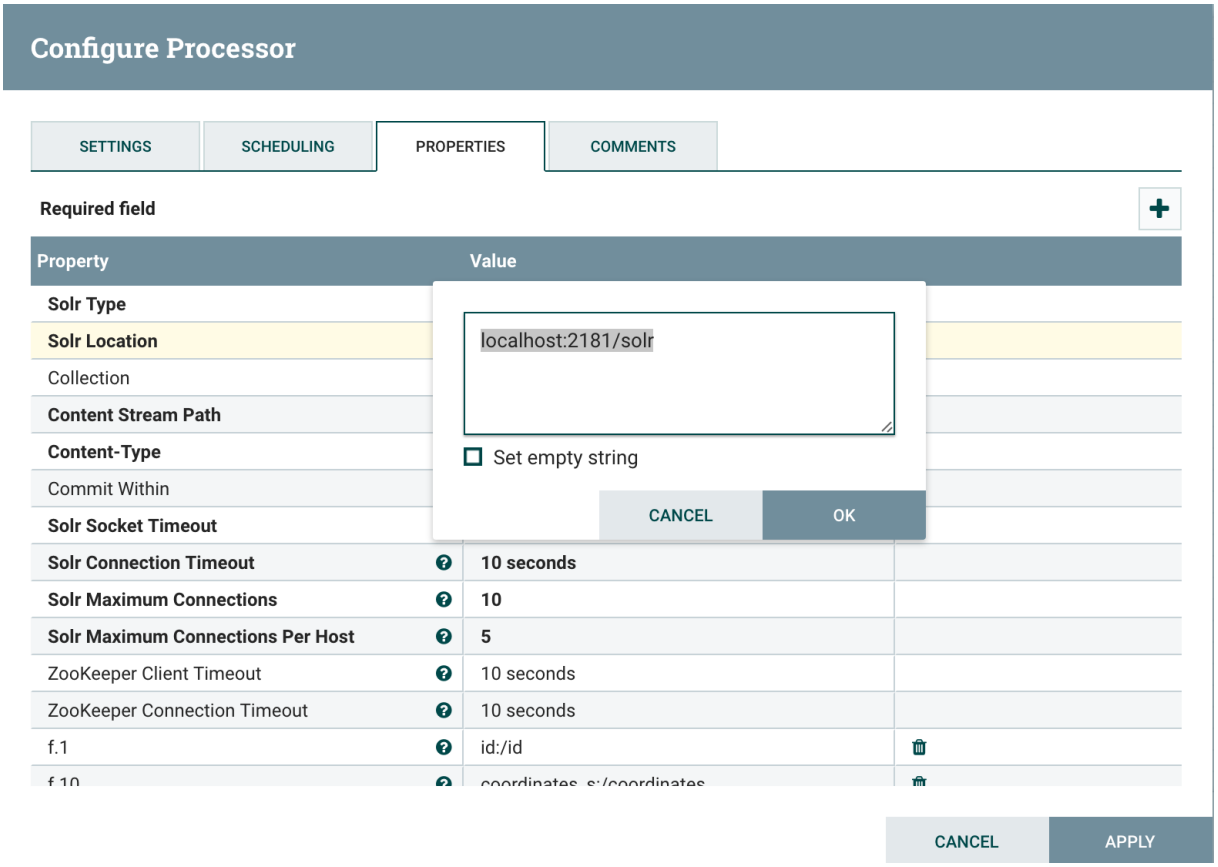
Fill in the following mandatory properties (in bold):

- Consumer Key
- Consumer Secret
- Access Token
- Access Token Secret



Configure PutContentSolrStream processor

Since we installed Solr via Ambari, you will need to append /solr to the ZK string in the 'Solr Location':



Review the remaining Processors

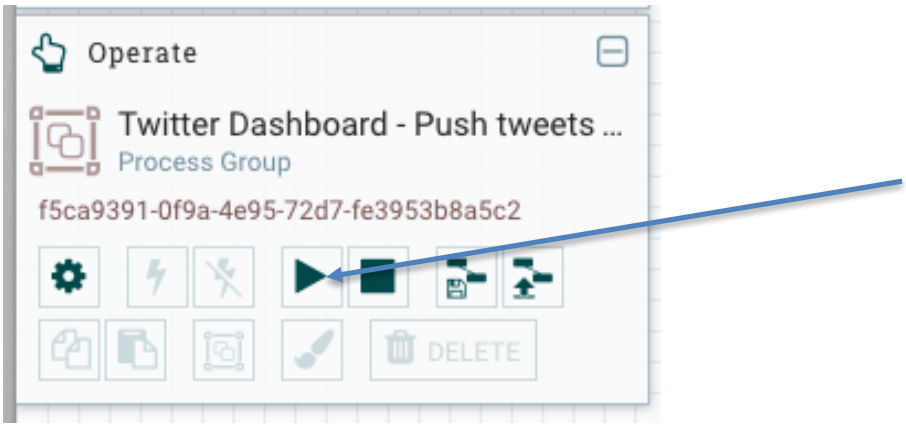
Review the other processors and modify properties as needed:

- **EvaluateJsonPath**: Pulls out attributes of tweets
- **RouteonAttribute**: Ensures only tweets with non-empty messages are processed
- **PutSolrContentStream**: Writes the selected attributes to Solr. In this case, assuming Solr is running in cloud mode with a collection 'tweets'
- **ReplaceText**: Formats each tweet as pipe (|) delimited line entry e.g. tweet_id|unixtime|humantime|user_handle|message|full_tweet
- **MergeContent**: Merges tweets into a single file (either 20 tweets or 120s, whichever comes first) to avoid having a large number of small files in HDFS. These values can be configured.
- **PutFile**: writes tweets to local disk under /tmp/tweets/
- **PutHDFS**: writes tweets to HDFS under /tmp/tweets_staging

Start the Flow

If the processors are correctly configured, all processors should have a red stop symbol in the upper left. If a yellow caution symbol is present, this indicates required configuration is missing.

Start the flow by selecting an empty area of the canvas and clicking the start icon.



Visualize and check the data

Now that the Twitter dataflow is running, the matching tweets will be persisted to Solr and HDFS. Validate this is successfully occurring.

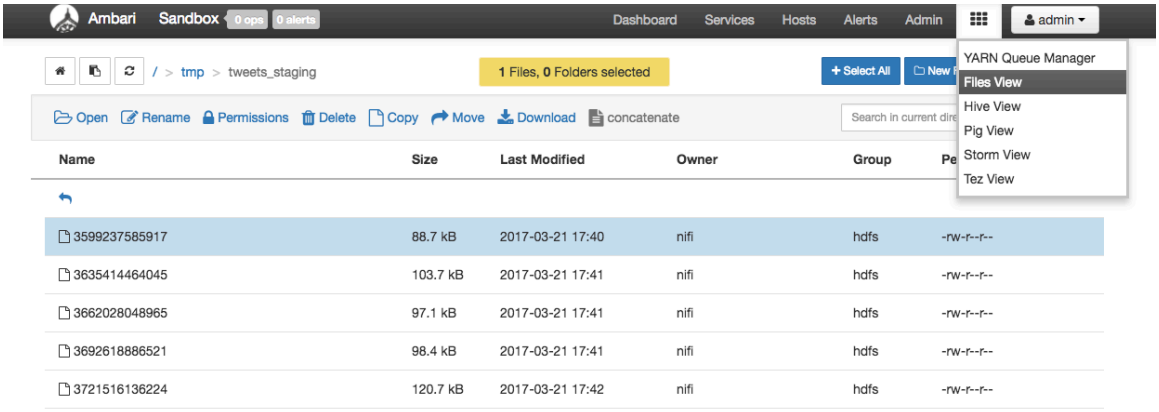
Visualize the data in real-time using the Banana Dashboard

Navigate to the banana dashboard and ensue the dashboard is populated.

- http://<public ip>:8983/solr/banana/index.html#/dashboard

View Files in HDFS

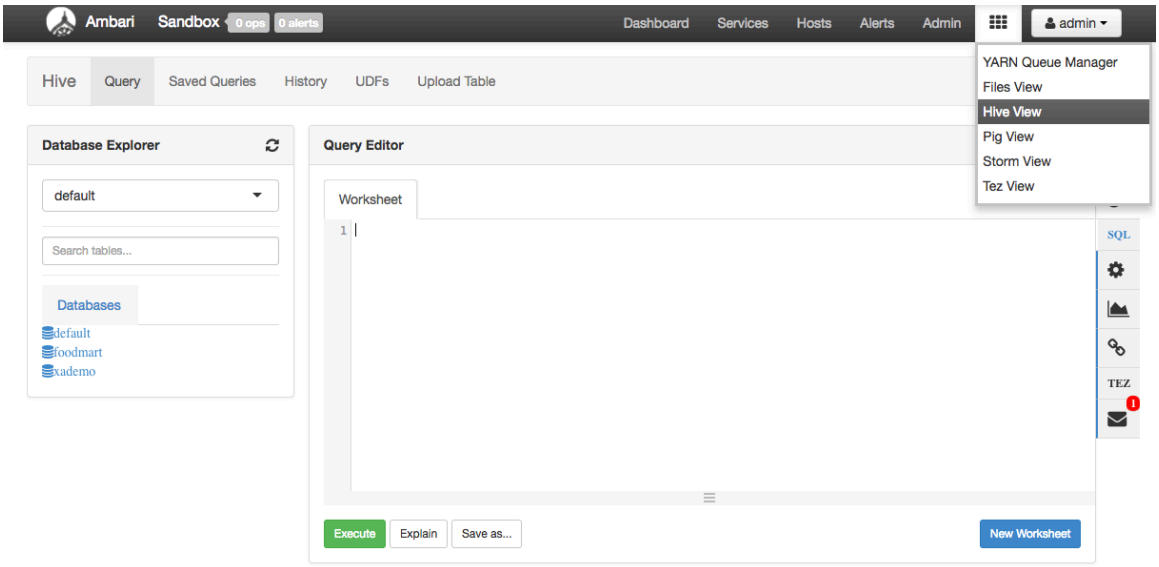
Tweets appear under /tmp/tweets_staging dir in HDFS. You can see this via Files view in Ambari:



Data Analysis with Hive

After persisting the tweets in HDFS, the opportunity to perform analytics against the data is nearly unlimited. Leveraging Hive, SQL can be used to analyze the Tweets. The following sections will create the Hive table and run some basic queries against the tweets.

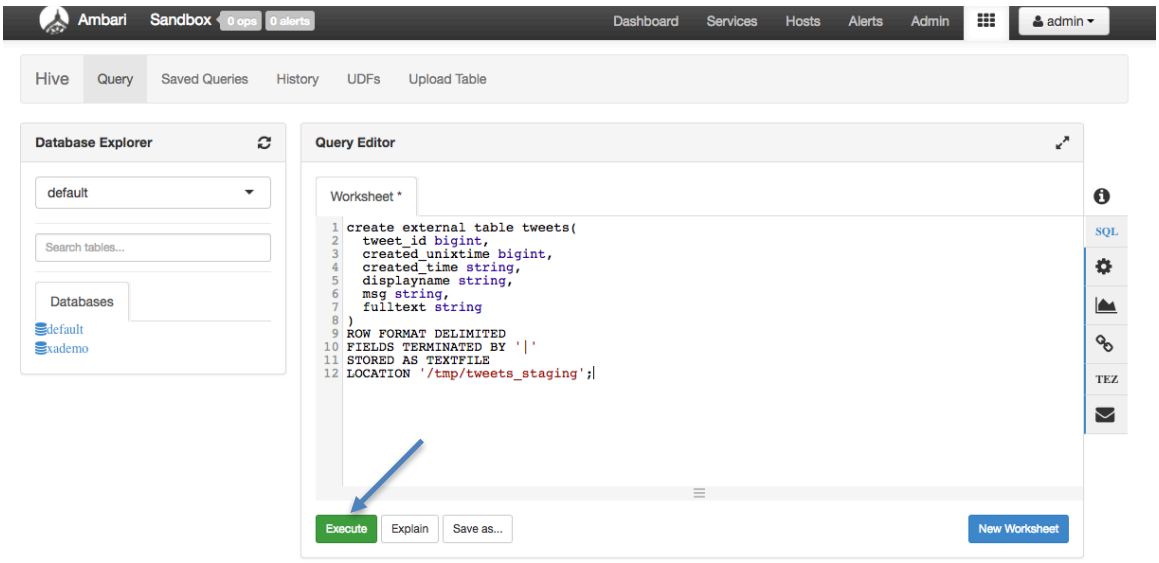
You can access Hive via the Hive view available on Ambari:



Create the Hive Table

Create the hive table to allow querying the tweets by pasting the following query onto the worksheet, and click on Execute button:

```
-- Create the tweets table

create external table tweets(
  tweet_id bigint,
  created_unixtime bigint,
  created_time string,
  displayname string,
  msg string,
  fulltext string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION '/tmp/tweets_staging';
```

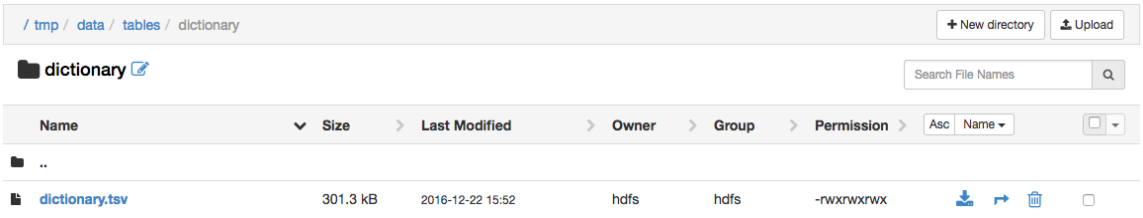


Sentiment Analysis with Hive

To do sentiment analysis, we need to import a data dictionary which will help to evaluate the sentiment polarity of tweets. The steps below will help in creating the required tables Dictionary

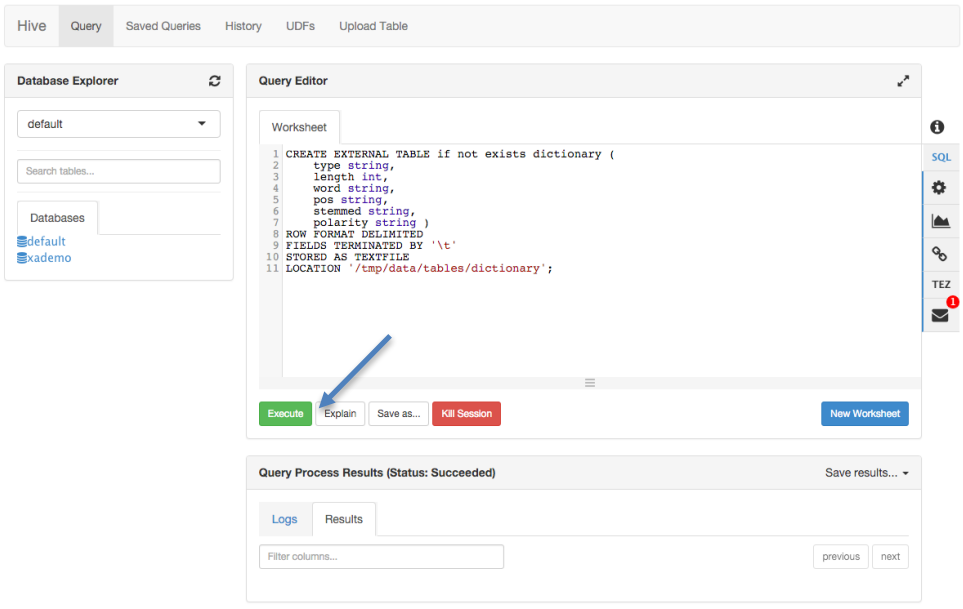
```
# Import data dictionary

su – hdfs

wget https://raw.githubusercontent.com/hortonworks/tutorials/hdp-
2.5/assets/nifi-sentiment-analytics/assets/dictionary.tsv

# Create directory /tmp/data/tables/dictionary

hdfs dfs -mkdir /tmp/data
hdfs dfs -mkdir /tmp/data/tables
hdfs dfs -mkdir /tmp/data/tables/dictionary

# Copy data sets onto hdfs

hdfs dfs -copyFromLocal dictionary.tsv /tmp/data/tables/dictionary

# Open up permissions on hdfs

hdfs dfs -chmod -R 777 /tmp/data/tables
```

You can view the file uploaded on the HDFS Files View. By clicking on the file name, you can preview its contents:



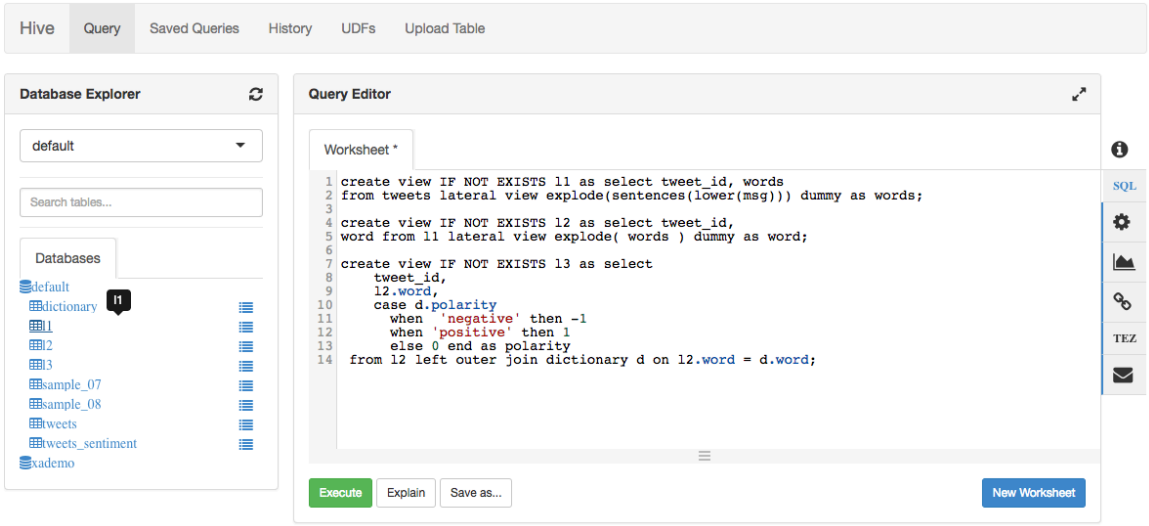
You can now create the dictionary table using the Hive view:

```
-- Create the dictionary table

CREATE EXTERNAL TABLE if not exists dictionary (
       type string,
       length int,
       word string,
       pos string,
       stemmed string,
       polarity string )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
LOCATION '/tmp/data/tables/dictionary';
```



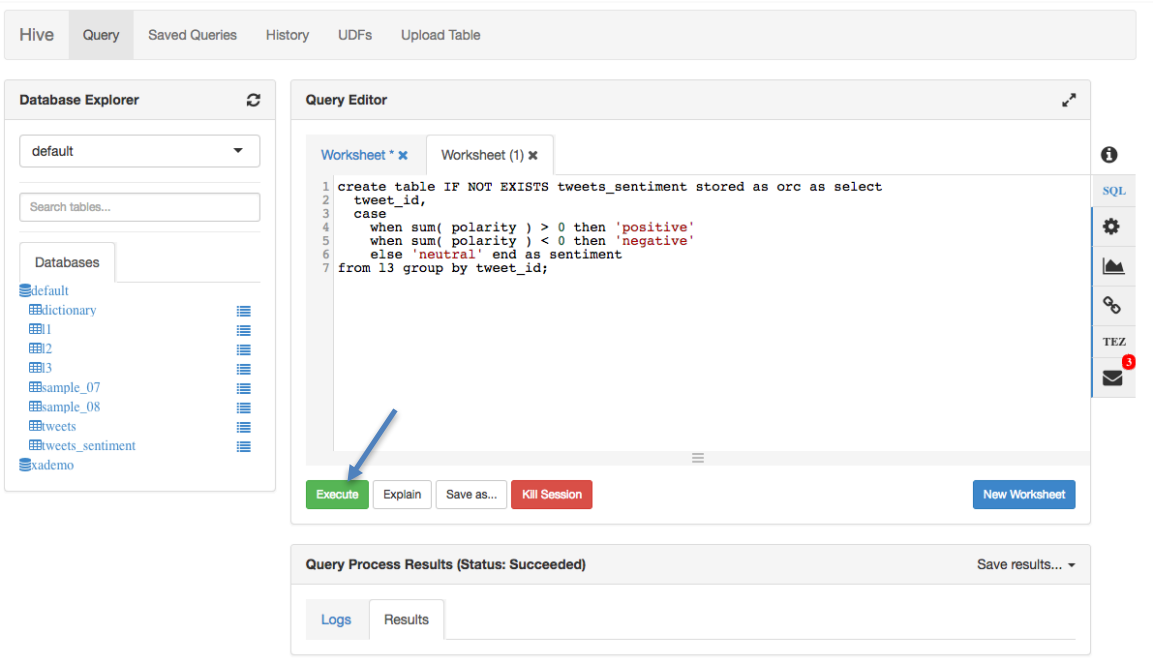
The next step is to create some views to compute the sentiment using the Hive view as following:

```
-- Compute the sentiment

create view IF NOT EXISTS l1 as select tweet_id, words from tweets lateral
view explode(sentences(lower(msg))) dummy as words;
create view IF NOT EXISTS l2 as select tweet_id, word from l1 lateral view
explode( words ) dummy as word;
create view IF NOT EXISTS l3 as select tweet_id, l2.word, case d.polarity when
'negative' then -1 when 'positive' then 1 else 0 end as polarity from l2 left
outer join dictionary d on l2.word = d.word;
```

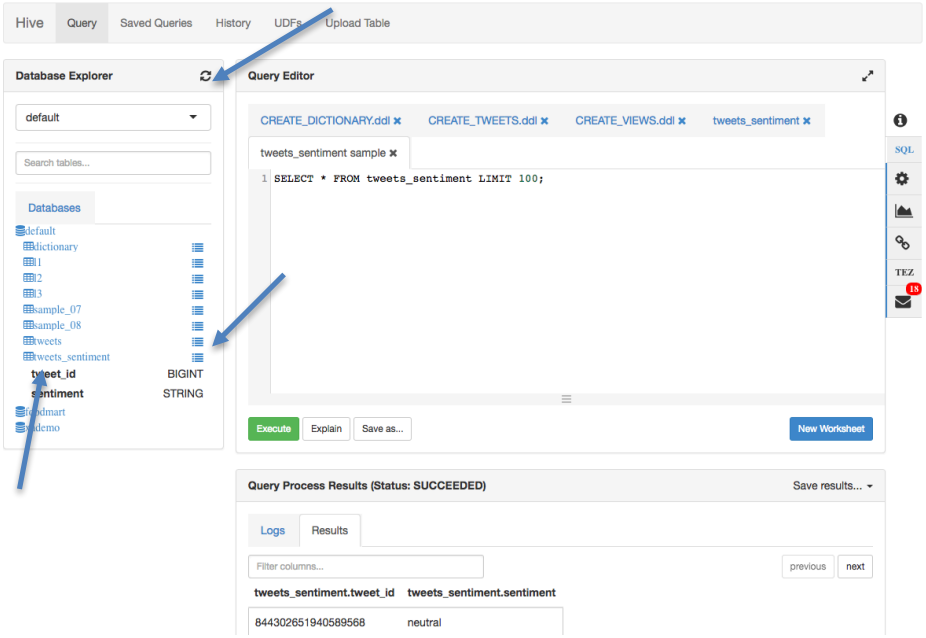


We can now compute the sentiment and materialize the results as an ORC table using the following query:

```
-- Compute and persist the tweet sentiment

create table IF NOT EXISTS tweets_sentiment stored as orc as select
  tweet_id,
  case
    when sum( polarity ) > 0 then 'positive'
    when sum( polarity ) < 0 then 'negative'
    else 'neutral' end as sentiment
from l3 group by tweet_id;
```

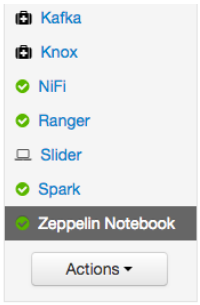


You can look at this Hive table containing the computing of the sentiment based on a simple data dictionary. Refresh the Database Explorer by clicking the refresh button. The tweets_sentiment table will appear in the list of tables. Click on the table name to expand the table definition, then click the icon on the right side of the table to display a sample of the data.



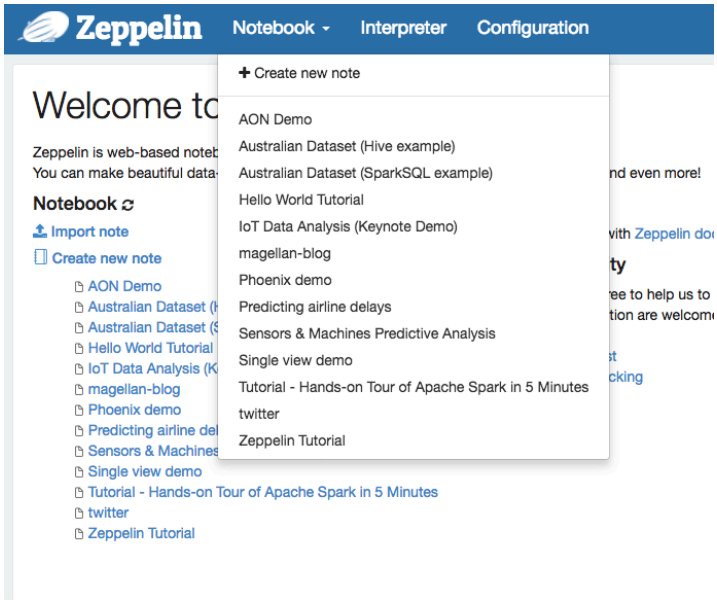
Data Visualization with Zeppelin

In this section, we are going to visualize the sentiment analysis data using Zeppelin. SparkSQL will be used to perform data analysis. Make sure the Zeppelin service is started in Ambari:



Navigate to Zeppelin using the following URL: http://<public-ip>:9995/#/

Use the Notebook drop down menu to create a new note called “Twitter Analysis”



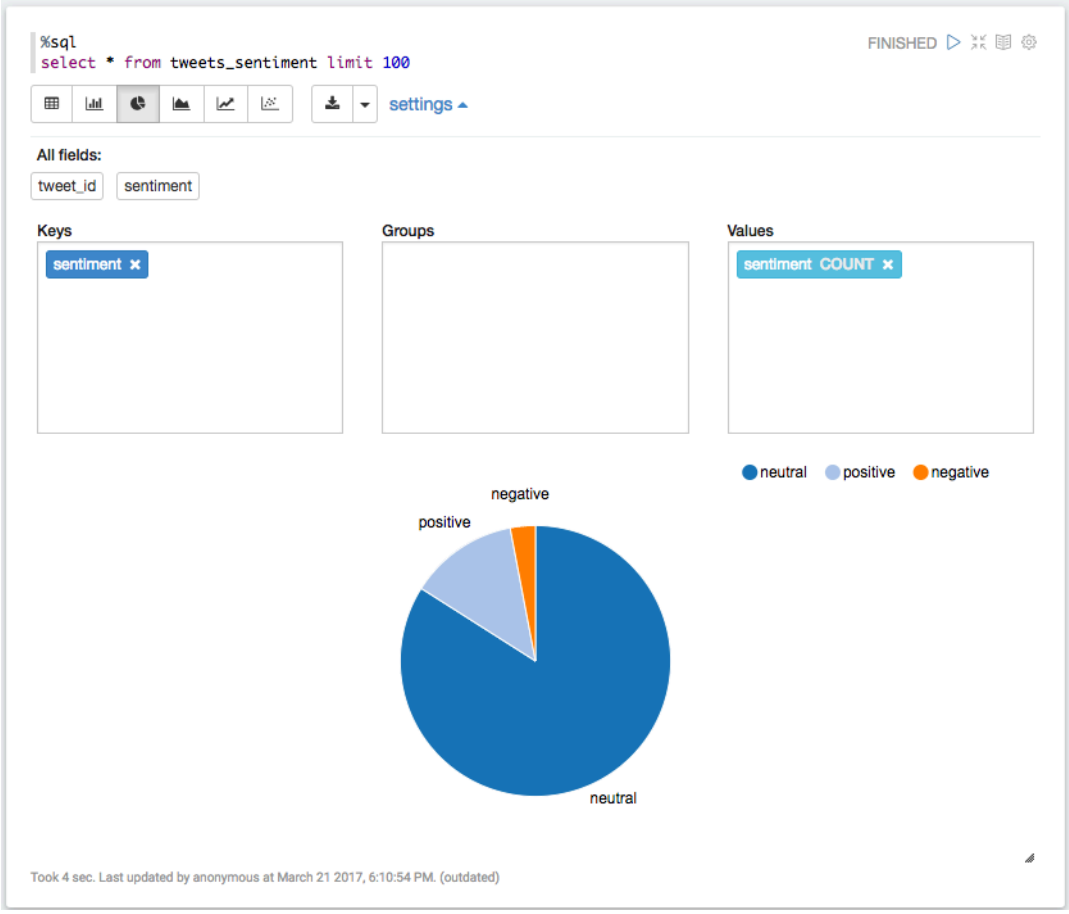
After creating the note, open it up to the blank Notebook screen and type the following command. `%sql` is the SparkSQL interpreter.

```
%sql
select * from tweets_sentiment limit 100
```

Arrange your results so that your chart is a pie chart:

- The tweets.sentiment column is a key and the tweets.sentiment as the value.
- Make sure that sentiment is labeled as COUNT.
- Run the query by clicking the arrow on the right-hand side, or by pressing Shift+Enter.

Your results should look like the following:



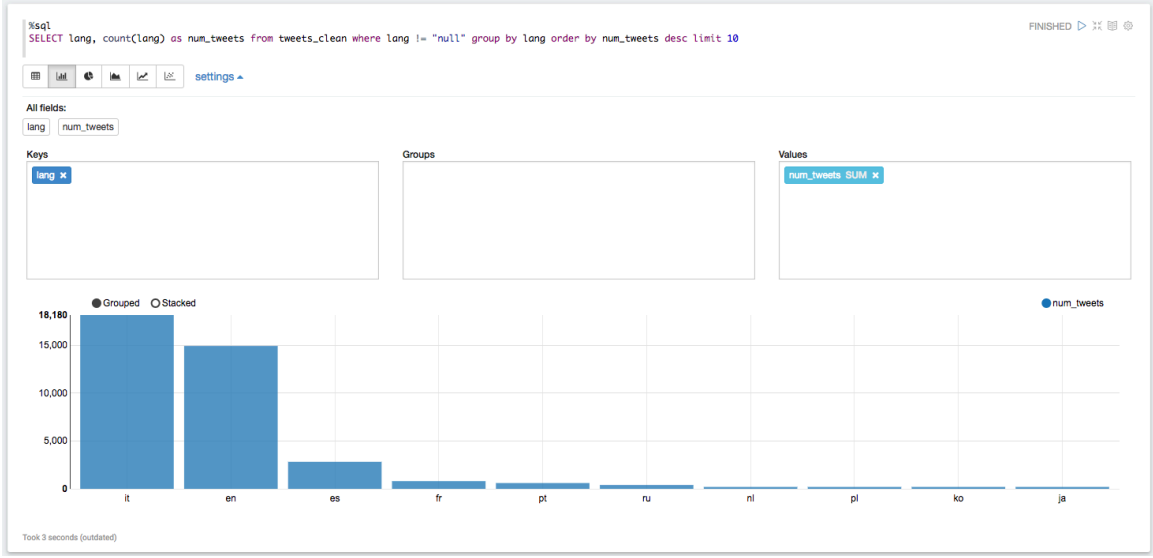
We can further cleanse and enrich the tweets table by creating a new tweets_clean table. To do this additional processing, we can extract further attributes from the tweets in json format using SparkSQL built-in functions, and converting the timestamp in a Hive timestamp format:

```
%sql
CREATE TABLE tweets_clean
AS
WITH TweetInfo as ( SELECT v1.id, v2.lang, v2.time_zone,v2.location,
v2.followers_count FROM tweets
LATERAL VIEW json_tuple(tweets.fulltext,'id','user') v1
as id, user
LATERAL VIEW
json_tuple(v1.user,'lang','time_zone','location','followers_count') v2
as lang, time_zone, location,followers_count)
SELECT
  t.tweet_id,
  cast ( from_unixtime( unix_timestamp(concat( '2016 ',
substring(t.created_time,5,15)), 'yyyy MMM dd hh:mm:ss')) as timestamp)
create_ts,
  s.sentiment,
  t.displayname,
  TweetInfo.lang,
  TweetInfo.time_zone,
  TweetInfo.location,
  cast(TweetInfo.followers_count as bigint) followers_count,
  t.msg
FROM tweets t LEFT OUTER JOIN tweets_sentiment s on t.tweet_id =
s.tweet_id,TweetInfo
```

We can now do further analysis on the enriched tweets table for example to count the top 10 languages used in the tweets:

```
%sql
SELECT lang, count(lang) as num_tweets from tweets_clean where lang !=
"null" group by lang order by num_tweets desc limit 10
```

The data can be visualized in a histogram as below:



We can list the top 10 polarized tweets (either negative or positive) by the most influential authors (measured in number of followers) with the following query:

```
%sql
SELECT displayname,followers_count,sentiment,msg from tweets_clean where
sentiment != "neutral" order by followers_count desc limit 10
```