New Data Lake/Tutorial 5 Working with Spark and Maps

Tutorial 5: Working Spark and Maps in Zeppelin

This tutorial was built for BDCS-CE version 17.4.1 as part of the New Data Lake User Journey: here (https://github.com/oracle/learning-library/tree/master/workshops/journey2-new-data-lake). Questions and feedback about the tutorial: david.bayard@oracle.com (mailto:david.bayard@oracle.com)

```
Be sure to have completed the previous Tutorial: "Working with Spark Interpreter"
```

This tutorial will illustrate how to display Spark data on Maps inside your Zeppelin Notebook.

Contents

As a reminder, the documentation for BDCS-CE can be found here (https://docs.oracle.com/cloud/latest/big-data-compute-cloud/index.html)

About Displaying Maps in Zeppelin

This example will plot our top bike pickup/dropoff stations on a map. To do so,

- We'll use Spark code to gather the data that we will use to define specific markers to add to our map. We will bind this marker data to a named angular variable. Angular is the framework that Zeppelin uses for its UI. here (https://angularjs.org)
- Then we will create a %angular paragraph to hold some custom HTML/javascript code. This is the paragraph that will display the map. Our code will use the javascript Leaflet project (here (http://leafletjs.com/examples/quick-start/)) to create an html/javascript map object. As part of our javascript, we'll instruct it to listen for changes to our named angular variable that we are populating in our spark paragraph. When the angular variable changes, our code will re-draw the markers on the map.

This approach is derived from examples on the internet, such as here (https://gist.github.com/granturing/a09aed4a302a7367be92) or here (https://community.hortonworks.com/articles/90320/add-leaflet-map-to-zeppelin-notebook.html), that are all based on a similar approach.

Our First Map - Top 10 Pickup Stations

To get started, we will plot the 10 top pickup stations on a map. We will have 1 paragraph which will use Spark and Spark SQL to query our bike_trips table and save the results into an angular variable. And we will have 1 paragraph that contains the html/javascript to display our map.

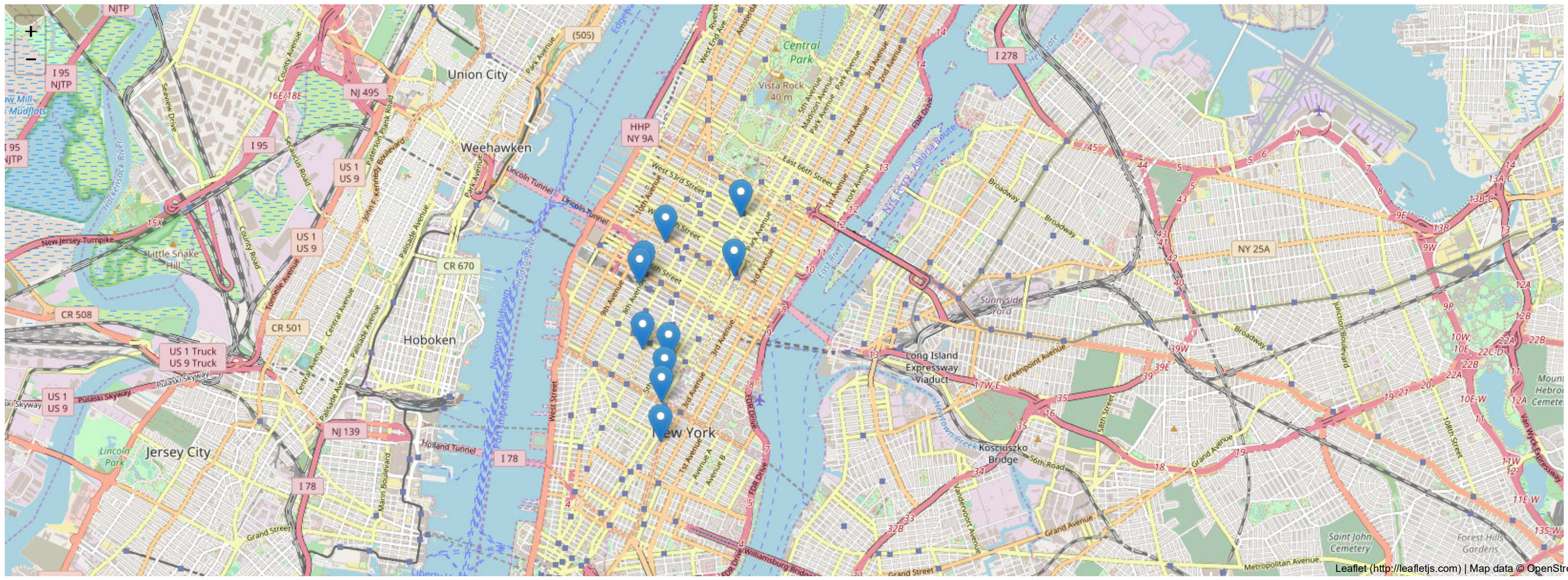
- Run the "Spark code for our first map" to gather the data for our map,
- Then run the "HTML/Javascript for our first map" to display the map with the selected data

Spark code for our first map

```
%spark


//a previous tutorial placed the csv file into your Object Store citibike container
//notice the use of the swift://CONTAINER.default/ syntax
val Container = "journeyC"
val Directory = "citibike"

//We will use the bdfs (alluxio) cached file system to access our object store data...
val df = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").option("inferSchema","true").load("bdfs://localhost:19998/"+Directory+"/raw/201612-citibike-tripdata.csv")

// If you get this error message:
// java.lang.IllegalStateException: Cannot call methods on a stopped SparkContext.
// Then go to the Settings tab, then click on Notebook.  Then restart the Notebook.  This will restart your SparkContext


df.createOrReplaceTempView("bike_trips_temp")

//Define a class for the structure of the data we will be passing to the map javascript code
case class Stations(ridetype: String, station: String, trips: String, lat: Double, lon: Double)

//Unbind angular variable in case it already exists from previous run
z.angularUnbind("topstations")

//Define a new dataframe based off a query
val topstationsDF = sqlContext.sql(s"""select "Start" ridetype, `Start Station Name` station,`Start Station Latitude` lat,`Start Station Longitude` lon, count(*) trips from bike_trips_temp
group by `Start Station Name`,`Start Station Latitude`,`Start Station Longitude`
order by count(*) desc limit 10""")
```

```
Container: String = journeyC
Directory: String = citibike
df: org.apache.spark.sql.DataFrame = [Trip Duration: int, Start Time: timestamp ... 13 more fields]
defined class Stations
topstationsDF: org.apache.spark.sql.DataFrame = [ridetype: string, station: string ... 3 more fields]
items: Array[Stations] = Array(Stations(Start,Pershing Square North,Pickups:9642,40.751873,-73.977706), Stations(Start,W 21 St & 6 Ave,Pickups:5990,40.74173969,-73.99415556), Stations(Start,E 17 St & Broadway,Pickups:5802,40.73704984,-73.99009296), Stations(Start,Broadway & E 22 St,
Pickups:5420,40.7403432,-73.98955109), Stations(Start,Broadway & E 14 St,Pickups:5342,40.73454567,-73.99074142), Stations(Start,8 Ave & W 33 St,Pickups:5282,40.751551,-73.993934), Stations(Start,W 41 St & 8 Ave,Pickups:4963,40.75640548,-73.9900262), Stations(Start,W 52 St & 5 Ave,Pi
ckups:4822,40.75992262,-73.97648516), Stations(Start,Cooper Square & E 7 St,Pickups:4644,40.72923649910006,-73.99086803197861), Stations(Start,8 Ave & W 31 St,Pickups:4622,40.7505853470215,-73.9946848154068))
..
done
```

HTML/Javascript for our first map - Top 10 Pickup Stations



Leaflet (http://leafletjs.com) | Map data © OpenStreetMap (http://openstreetmap.org) contributors

A more complex example - Top Pickup and Dropoff locations for certain times of day

Now that we have shown a simple example of a map, we will show a more complex example using a parameter-driven query with multiple map markers and layers.

To run this example:

- Use the Query Parameters paragraph to choose the time frame for the query as well as how many of the Top pickup and dropoff stations to show on the map.
- Then run the Query Parameters paragraph. This will query the desired data (by automatically running the "Spark code for the more complex map" paragraph)
- Once the data has been queried (takes about 20 seconds), the "More Complex Map" will be updated. If the map is not showing, then play the "More Complex Map" paragraph to make it visible.
  – Pickups are drawn in green. Dropoffs are in red.
  – Use the layers control in the upper right of the map to show/hide pickup and dropoff stations.
  – The size of the circle on the map represents the number of pickups/dropoffs for that station during the selected time frame.

Query Parameters (change these drop-downs, then run the paragraph and it will automatically requery the data)

dayOfWeek

Mon-Fri

hourOfDay

7am to 10am

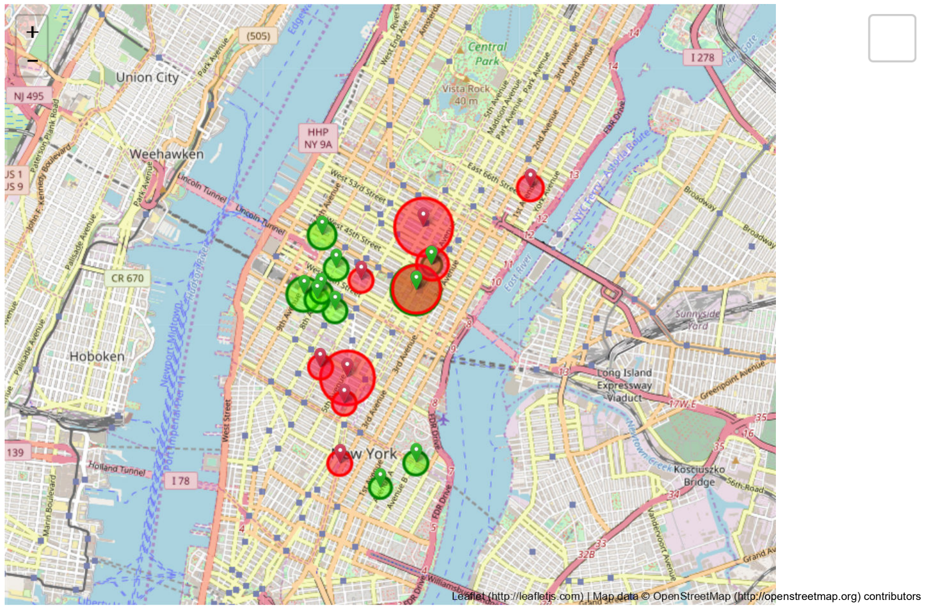
Top N

10

Spark code for the more complex map

```
defined class Stations
WHERE STRING:  where 1=1  AND date_format(`Start Time`,"E") in ("Mon","Tue","Wed","Thu","Fri")  AND date_format(`Start Time`,"H") in ("7","
8","9","10")
done
```

More Complex Map


Leaflet (http://leafletjs.com) | Map data © OpenStreetMap (http://openstreetmap.org) contributors
Union City
Weehawken
Hoboken
Central Park
New York
Long Island Expressway Viaduct
Kosciuszko Bridge

Before you continue

Experiment with the drop-down choices above. For instance, compare Mon-Fri/7am-10am to Mon-Fri/4pm-7pm. Do you see a difference between where bikes are being picked up (green) and dropped off (red)?

Or compare Mon-Fri/7am-10am to Weekend/7am-10am? Do you see a difference in the quantity of bike trips (the size of the circle under the pin)?

Next Steps

In our next part of the demonstration, we will use Spark Streaming and Oracle Event Hub Cloud Service to work with bike trip data in "real-time".

Change Log

September 7, 2017 - Confirmed it works with 17.3.5-20. Switched to bdfs file paths
August 23, 2017 - Minor tweaks
August 13, 2017 - Confirmed it works with 17.3.3-20
August 11, 2017 - Journey v2. Confirmed it works with Spark 2.1
July 28, 2017 - Confirmed it works with 17.3.1-20

```
%md
```