MOONSHOT 2

Spark Enabled Real-time Streaming System

TEAM: ZOOTOPIA

TEAM MEMBERS:

1. Sanjeedha Sanofer Raja (010698805)
2. Meera Mali (010706579)
3. Ojas Milind Kale (010734672)
4. Diksha jain (010711116)
5. Shagun Juneja (010742875)

Name of the App: **TweetOViz**

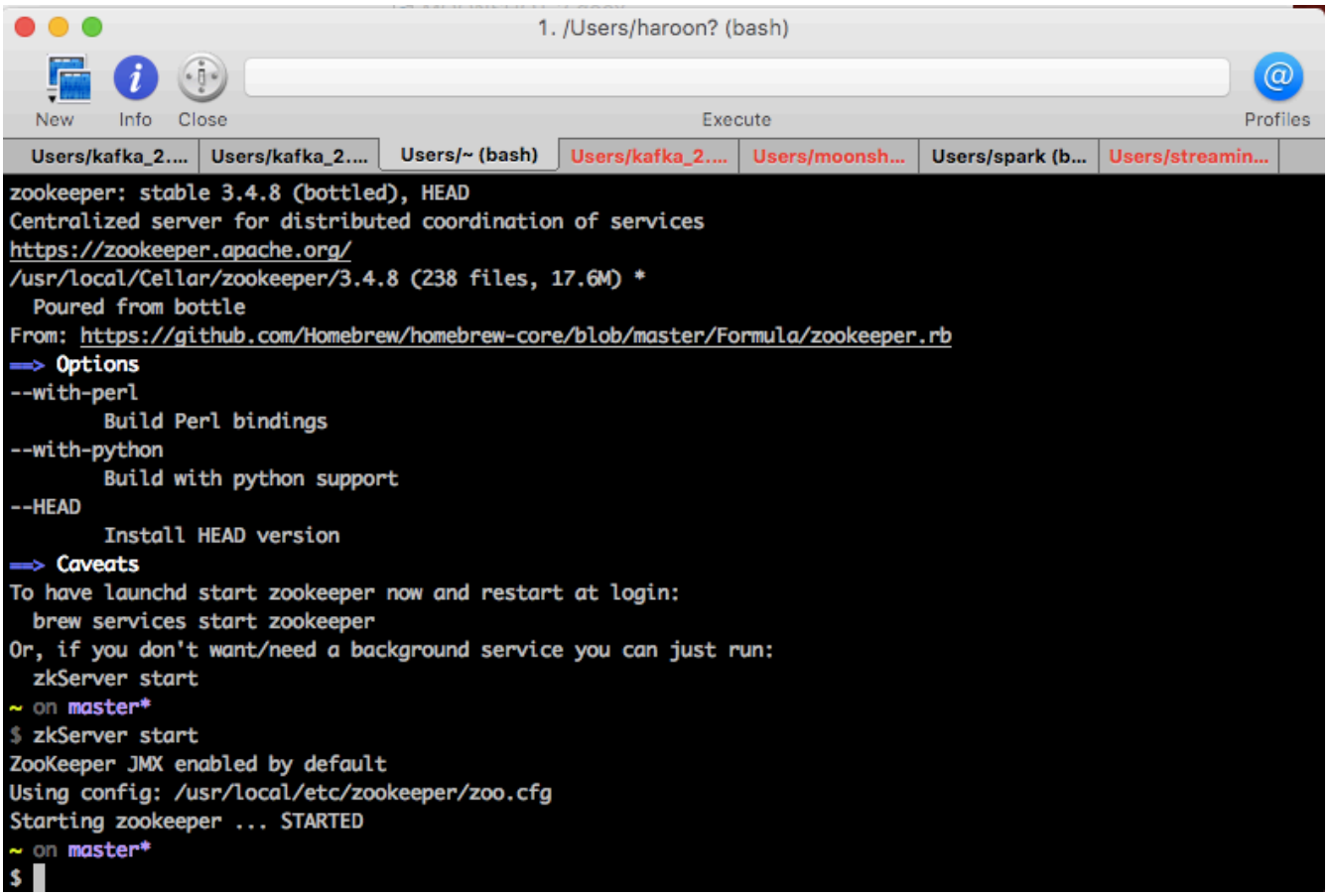
Real-time Streaming data: **Republicans vs Democrats**

Steps:

1) Create Twitter API account to get the API key and secret.

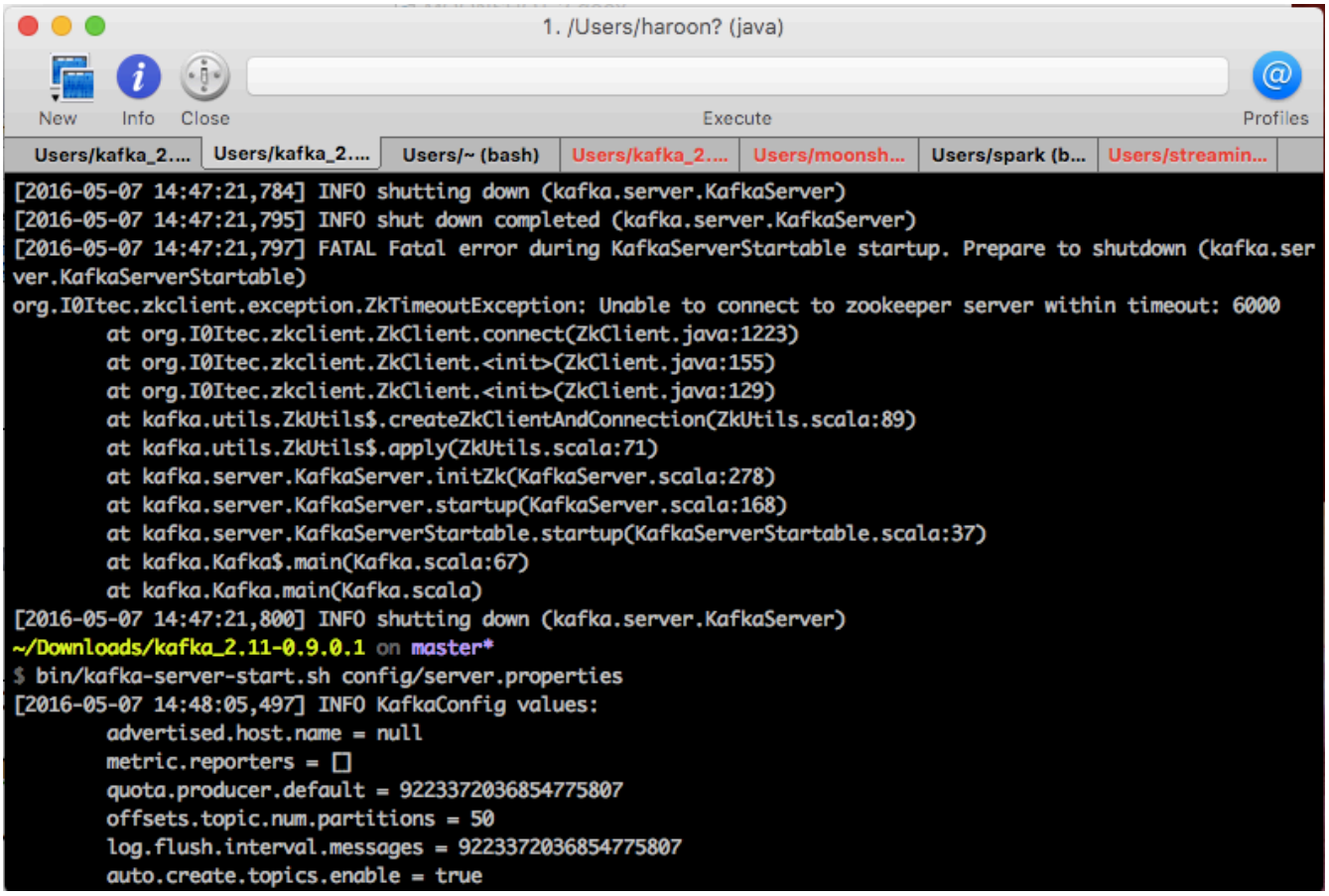
2) Start Zookeeper installed using brew

zkServer start



3) Start Kafka

bin/kafka-server-start.sh config/server.properties

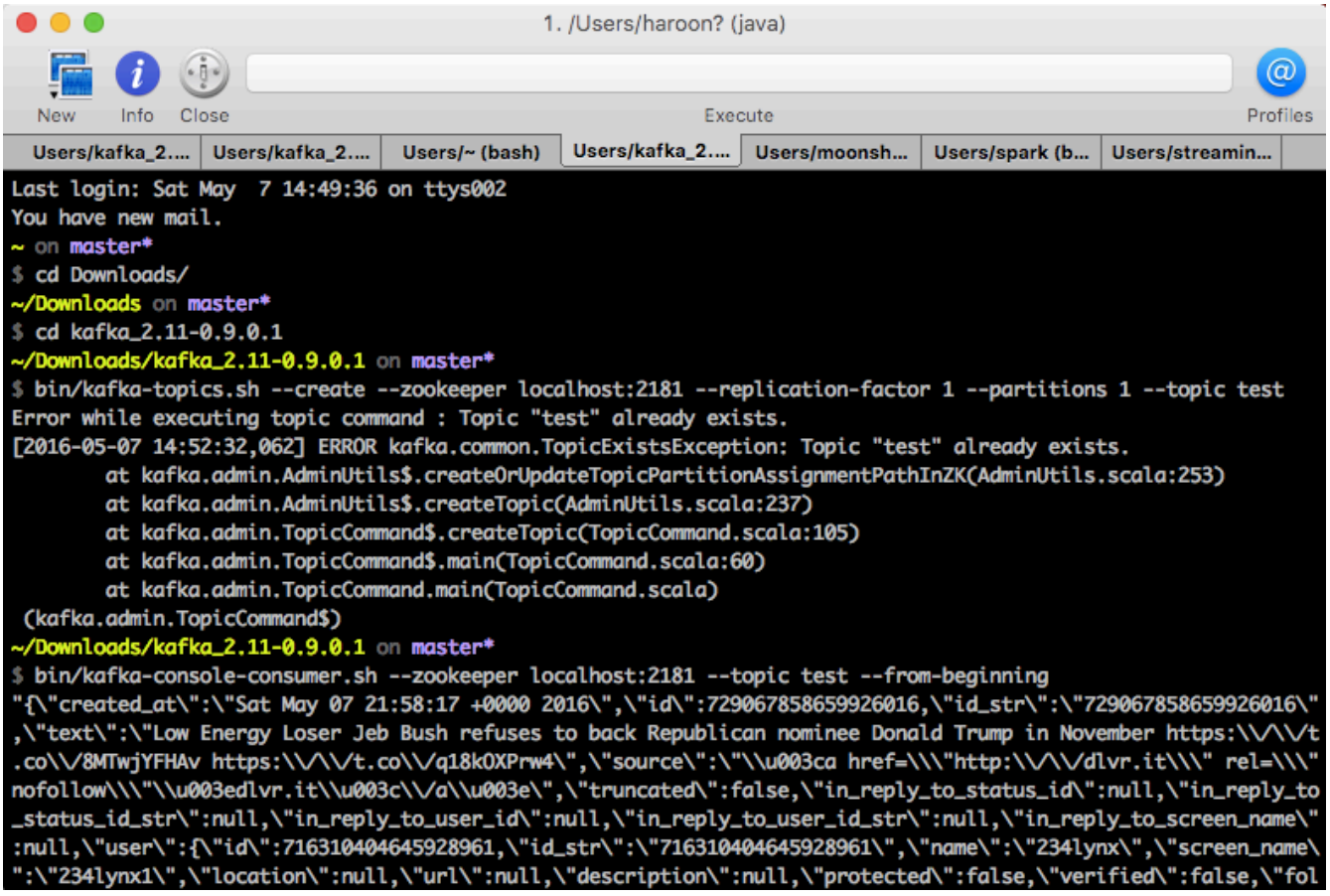


4) Create Kafka topic if necessary

bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test

5) Create kafka Consumer

bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic test --from-beginning



6) Run the python script to use twitter API and get tweets and send it to kafka broker

python twitter_streaming.py

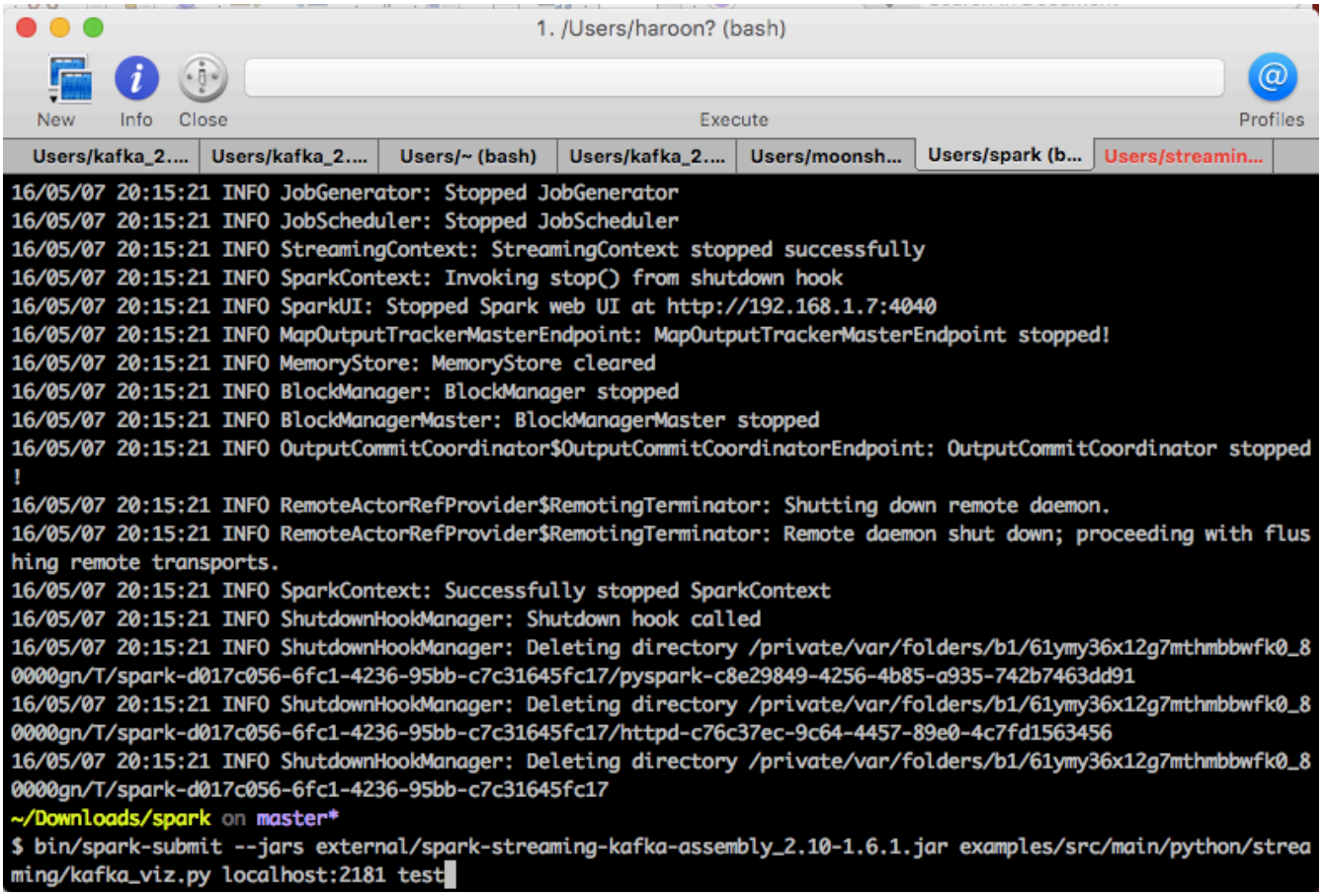
twitter_streaming.py

```
#Import the necessary methods from tweepy library
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
from kafka import KafkaProducer
import json

producer = KafkaProducer(value_serializer=lambda v: json.dumps(v).encode('utf-8'))

#Variables that contains the user credentials to access Twitter API
access_token = "4176278119-OkdJTZgtdJh6qZYz483HTdMjcDAVbJQL45CoSgZ"
access_token_secret = "Al6PWC6EXWNPUvvmP964kVDfZivlCyVEEdIUaCQYoecxx"
consumer_key = "j19KmpBmWHDZVbORuUgXj3aMF"
consumer_secret = "MrzG1HCftg9lgEu93QMnjtPfo4AxFb8YkzEXe4HvZZGbFcm4NO"


#This is a basic listener that just prints received tweets to stdout.
class StdOutListener(StreamListener):

    def on_data(self, data):
        producer.send('test', data)
        return True

    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: 'python', 'javascript', 'ruby'
```

stream.filter(track=['democrat', 'republican'])

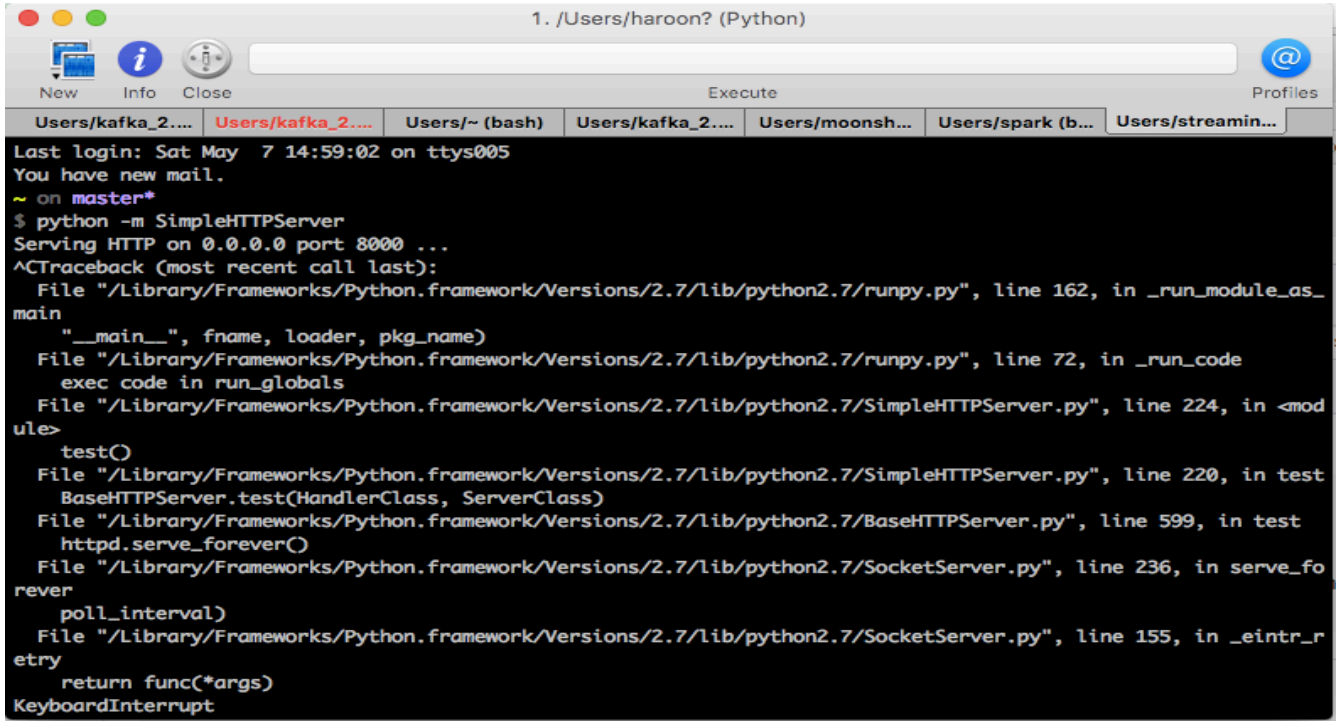
7) Run the spark kafka script to listen to kafka broker, receive the tweets from kafka, process the tweets and count the number of tweets for democrats and republicans. Save the information in a JSON file.

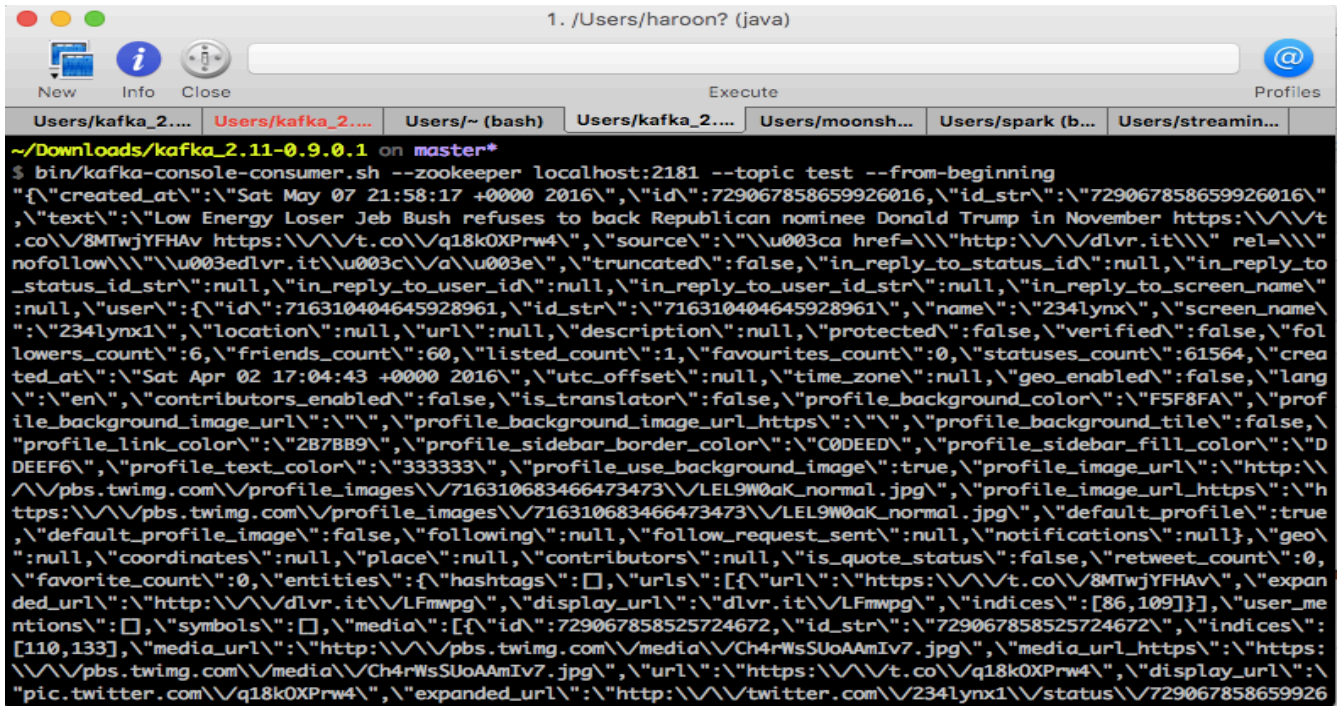
bin/spark-submit --jars external/spark-streaming-kafka-assembly_2.10-1.6.1.jar examples/src/main/python/streaming/kafka_viz.py localhost:2181 test



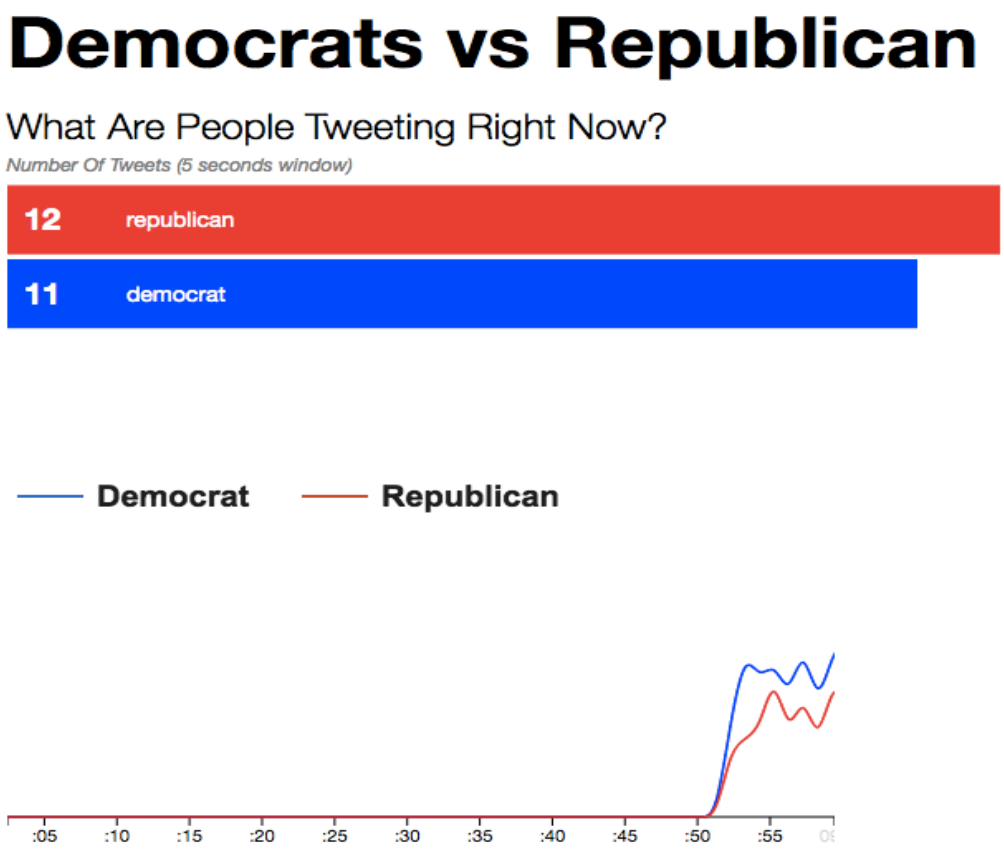
8) Use d3.js to read the JSON file in step 7 at regular intervals and plot the data in a bar chart and time series chart.

python -m SimpleHTTPServer





9) Run the URL for visualizing the flow.
http://localhost:8000/streaming.html



streaming.html

```
<!DOCTYPE html>

<html>
<head>
 <script src="http://d3js.org/d3.v3.min.js" charset="utf-8"></script>
 <link rel="stylesheet" href="style.css">
 <style type="text/css">
   html, body, div, span, applet, object, iframe,
   h1, h2, h3, h4, h5, h6, p, blockquote, pre,
   a, abbr, acronym, address, big, cite, code,
   del, dfn, em, img, ins, kbd, q, s, samp,
   small, strike, strong, sub, sup, tt, var,
   b, u, i, center,
   dl, dt, dd, ol, ul, li,
```

```
  fieldset, form, label, legend,
  table, caption, tbody, tfoot, thead, tr, th, td,
  article, aside, canvas, details, embed,
  figure, figcaption, footer, header, hgroup,
  menu, nav, output, ruby, section, summary,
  time, mark, audio, video {
    margin: 0;
    padding: 0;
    border: 0;
    font: inherit;
    font-size: 100%;
    vertical-align: baseline;
     font-family: "Helvetica Neue", Helvetica, Arial, "Lucida Grande", sans-serif;
     font-weight: inherit;
  }

  h1 {
    font-size:300%;
     font-family: "HelveticaNeue-Light", "Helvetica Neue Light", "Helvetica Neue", Helvetica, Arial,
"Lucida Grande", sans-serif;
     font-weight: 600;
    text-align: left;
    margin-top: 15px;
    margin-bottom: 15px;
  }

  h2 {
    font-size:150%;
     font-family: "HelveticaNeue-Light", "Helvetica Neue Light", "Helvetica Neue", Helvetica, Arial,
"Lucida Grande", sans-serif;
     font-weight: 300;
    text-align: left;
    margin-top: 5px;
    margin-bottom: 5px;
  }
  h3 {
    font-size: 12px;
    font-style: italic;
    color: gray;
    margin-top:5px;
    margin-bottom:5px;
  }
  #everything{
    width:600px;
    margin-top:200px;
    margin-left: auto;
    margin-right: auto;
```

```
  }
  #chart{
    width:800px;
    height:200px;
  }
  .bar{
    fill:#eaeaea;
  }
  text.label{
    fill: white;
    color: white;
    font-size: 20px;
    font-weight: bold;
  }
  text.category{
    fill: white;
    font-size: 14px;
  }

  .graph .axis {
        stroke-width: 1;
    }

    .graph .axis .tick line {
        stroke: black;
    }

    .graph .axis .tick text {
        fill: black;
        font-size: 0.7em;
    }

    .graph .axis .domain {
        fill: none;
        stroke: black;
    }

    .graph .group {
        fill: none;
        stroke: black;
        stroke-width: 1.5;
    }
 </style>
</head>

<body>

 <div id="everything">
```

```
   <h1>Democrats vs Republican</h1>
   <h2>What Are People Tweeting Right Now?</h2>
   <h3>Number Of Tweets (5 seconds window)</h3>
   <div id="chart"></div>

   <img src="legend.png">
   <div class="graph"></div>
  </div>


  <script>

//updatingBarChart.js

var setup = function(targetID){
  //Set size of svg element and chart
  var margin = {top: 0, right: 0, bottom: 0, left: 0},
    width = 600 - margin.left - margin.right,
    height = 100 - margin.top - margin.bottom,
    categoryIndent = 4*15 + 5,
    defaultBarWidth = 2000;

  //Set up scales
  var x = d3.scale.linear()
    .domain([0,defaultBarWidth])
    .range([0,width]);

  var y = d3.scale.ordinal()
    .rangeRoundBands([0, height], 0.1, 0);

  //Create SVG element
  d3.select(targetID).selectAll("svg").remove()
  var svg = d3.select(targetID).append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g")
    .attr("transform", "translate(" + margin.left + "," + margin.top + ")");

  //Package and export settings
  var settings = {
    margin:margin, width:width, height:height, categoryIndent:categoryIndent,
    svg:svg, x:x, y:y
  }
  return settings;
}

var redrawChart = function(targetID, newdata) {
```

```
 //Import settings
 var margin=settings.margin, width=settings.width, height=settings.height,
categoryIndent=settings.categoryIndent,
 svg=settings.svg, x=settings.x, y=settings.y;

 //Reset domains
 y.domain(newdata.sort(function(a,b){
  return b.value - a.value;
 })
  .map(function(d) { return d.key; }));
 var barmax = d3.max(newdata, function(e) {
  return e.value;
 });
 x.domain([0,barmax]);

 /////////
 //ENTER//
 /////////

 //Bind new data to chart rows

 //Create chart row and move to below the bottom of the chart
 var chartRow = svg.selectAll("g.chartRow")
  .data(newdata, function(d){ return d.key});
 var newRow = chartRow
  .enter()
  .append("g")
  .attr("class", "chartRow")
  .attr("transform", "translate(0," + height + margin.top + margin.bottom + ")");

 //Add rectangles
 newRow.insert("rect")
  .attr("class","bar")
  .attr("x", 0)
  .attr("opacity",0)
  .attr("height", y.rangeBand())
  .style("fill", function(d) { return d.color; })
  .attr("width", function(d) { return x(d.value);})

 //Add value labels
 newRow.append("text")
  .attr("class","label")
  .attr("y", y.rangeBand()/2)
  .attr("x",0)
  .attr("opacity",0)
  .attr("dy",".35em")
  .attr("dx","0.5em")
  .text(function(d){return d.value;});
```

```
//Add Headlines
newRow.append("text")
  .attr("class","category")
  .attr("text-overflow","ellipsis")
  .attr("y", y.rangeBand()/2)
  .attr("x",categoryIndent)
  .attr("opacity",0)
  .attr("dy",".35em")
  .attr("dx","0.5em")
  .text(function(d){return d.key});


//////////
//UPDATE//
//////////

//Update bar widths
chartRow.select(".bar").transition()
  .duration(300)
  .attr("width", function(d) { return x(d.value);})
  .attr("opacity",1);

//Update data labels
chartRow.select(".label").transition()
  .duration(300)
  .attr("opacity",1)
  .tween("text", function(d) {
  var i = d3.interpolate(+this.textContent.replace(/\,/g,''), +d.value);
  return function(t) {
    this.textContent = Math.round(i(t));
  };
  });

//Fade in categories
chartRow.select(".category").transition()
  .duration(300)
  .attr("opacity",1);


////////
//EXIT//
////////

//Fade out and remove exit elements
chartRow.exit().transition()
  .style("opacity","0")
  .attr("transform", "translate(0," + (height + margin.top + margin.bottom) + ")")
```

```
   .remove();


  ////////////////
  //REORDER ROWS//
  ////////////////

  var delay = function(d, i) { return 200 + i * 10; };

  chartRow.transition()
    .delay(delay)
    .duration(500)
    .attr("transform", function(d){ return "translate(0," + y(d.key) + ")"; });
};



//Pulls data
//Since our data is fake, adds some random changes to simulate a data stream.
//Uses a callback because d3.json loading is asynchronous
var pullData = function(settings,callback){
  d3.json("testdata.json", function (err, data){
    if (err) return console.warn(err);

    var newData = data;
    data.forEach(function(d,i){
      var newValue = d.value + Math.floor((Math.random()*10) - 5)
      newData[i].value = newValue <= 0 ? 10 : newValue
    })

    newData = formatData(newData);

    callback(settings,newData);
  })
}

//Sort data in descending order and take the top 10 values
var formatData = function(data){
    return data.sort(function (a, b) {
        return b.value - a.value;
      })
    .slice(0, 10);
}

//I like to call it what it does
var redraw = function(settings){
  pullData(settings,redrawChart)
}
```

```
//setup (includes first draw)
var settings = setup('#chart');
redraw(settings)

//Repeat every 3 seconds
setInterval(function(){
  redraw(settings)
}, 5000);

  </script>

<script src="http://d3js.org/d3.v3.min.js"></script>
      <script>
      var limit = 60 * 1,
         duration = 1000,
         now = new Date(Date.now() - duration)

      var width = 500,
         height = 200

      var groups = {
         democrat: {
            value: 0,
            color: 'blue',
            data: d3.range(limit).map(function() {
               return 0
            })
         },
         republican: {
            value: 0,
            color: 'red',
            data: d3.range(limit).map(function() {
               return 0
            })
         }
      }

      var xx = d3.time.scale()
         .domain([now - (limit - 2), now - duration])
         .range([0, width])

      var yy = d3.scale.linear()
         .domain([0, 30])
         .range([height, 0])

      var line = d3.svg.line()
         .interpolate('basis')
```

```
    .x(function(d, i) {
      return xx(now - (limit - 1 - i) * duration)
    })
    .y(function(d) {
      return yy(d)
    })

var svg2 = d3.select('.graph').append('svg')
    .attr('class', 'chart')
    .attr('width', width)
    .attr('height', height + 50)

var axis = svg2.append('g')
    .attr('class', 'xx axis')
    .attr('transform', 'translate(0,' + height + ')')
    .call(xx.axis = d3.svg.axis().scale(xx).orient('bottom'))

var paths = svg2.append('g')

for (var name in groups) {
    var group = groups[name]
    group.path = paths.append('path')
        .data([group.data])
        .attr('class', name + ' group')
}

function tick() {

    now = new Date()

    d3.json("testdata.json", function (err, data){
        if (err) return console.warn(err);

        var newData = data;
        data.forEach(function(d,i){
          var newValue = d.value + Math.floor((Math.random()*10) - 5)
          newData[i].value = newValue <= 0 ? 10 : newValue
        });

        var obj = {
            'democrat': 0,
            'republican': 0
        };

        for (var j = 0; j < newData.length; j++) {
            obj[newData[j].key] = newData[j].value;
        }
```

```
        // Add new values
        for (var name in groups) {
          var group = groups[name]
          //group.data.push(group.value) // Real values arrive at irregular intervals
          group.data.push(obj[name])
          group.path.style('stroke', group.color)
          group.path.attr('d', line)
        }

        // Shift domain
        xx.domain([now - (limit - 2) * duration, now - duration])

        // Slide xx-axis left
        axis.transition()
          .duration(duration)
          .ease('linear')
          .call(xx.axis)

        // Slide paths left
        paths.attr('transform', null)
          .transition()
          .duration(duration)
          .ease('linear')
          .attr('transform', 'translate(' + xx(now - (limit - 1) * duration) + ')')
          .each('end', tick)

        // Remove oldest data point from each group
        for (var name in groups) {
          var group = groups[name]
          group.data.shift()
        }


      });

    }

    tick()
    </script>

</body>
```