# IoT Weather Station

## Processing Data in Kafka with Spark Structured Streaming

**The Internet of Things: Big Data Processing and Analytics**

Instructor: Hinkmond Wong

Sheetal Gangakhedkar

November 21, 2017

# Introduction

Traditional weather stations have employed various instruments to measure the temperature, pressure, wind direction, wind speed, humidity, and rainfall. Now-a-days these weather stations are digital, so they can record all this data electronically, and now with the advent of Internet and connectivity of things (Internet of Things - IoT), these devices can transmit their monitored data to remote servers and exchange data with other connected devices and data processing services in the cloud.

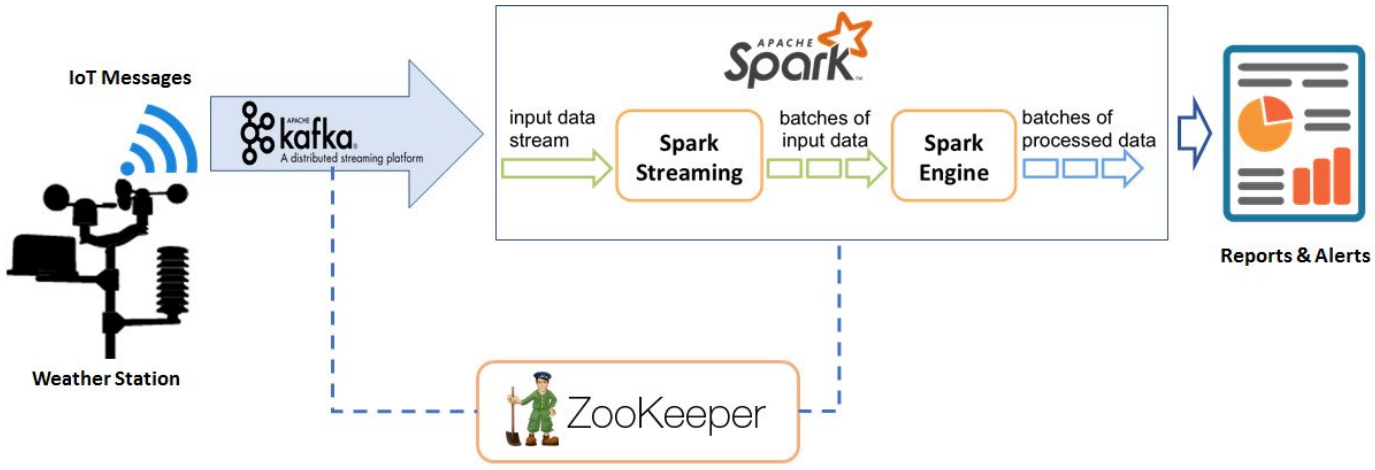
# Objective

Analyze temperature and wind-speed sensor "*weather station*" data sent hourly. This is a temperature and windspeed sensor device installed outside, it can transmit the current readings hourly to our Big Data analysis Spark Stream processing engine in the Cloud. The objective is to analyze this streaming data and generate reports, that includes the following:

- Average, Min, and Max daily temperatures
- Average, Min, and Max weekly temperatures
- Average, Min, and Max temperatures for the full batch of data received
- Average, Min, and Max daily wind-speeds
- Average, Min, and Max weekly wind-speeds
- Average, Min, and Max wind-speeds for the full batch of data received

The Spark 'iotmsgs' processing job can be further enhanced to generate alerts when the temperature or wind speed data values exceed a certain threshold. These alerts can be relayed back to some key personnel in the building to their mobile devices or to other connected IoT devices like room heaters, air-conditioners and sound alarms.

# Architecture



The schematic represents Processing IoT Sensor Data in Apache Kafka with Structured Streaming in Apache Spark. ZooKeeper is started in the VM to manage Kafka and the Spark jobs. Kafka Broker is started on port 9092 with 'iotmsgs' topic. IoT Sensor data is pushed to the Kafka producer, which is processed by the Kafka broker, and the Spark Streaming job as Kafka Consumer will be receive and process the IoT sensor data streaming on the 'iotmsgs' Kafka topic. Spark job is submitted to analyze the IoT sensor data streamed from the Kafka 'iotmsgs' topic queue.

# Design and Implementation

## Weather Station Simulator

For this project, we are employing a simulator for an IoT Weather Station that can be installed on buildings to monitor the outdoor weather. This simulator currently generates IoT messages that includes the GUID of the Weather Station, the latitude and longitude of where it is installed, and at regular intervals (every 1 hour) includes payload of the temperature sensor and the wind-speed sensor data. The simulator was modified to generate 'n' IoT messages that includes the weather station sensor data for temperature and wind speeds. The 'n' messages generated by the simulator are for the past 'n' hours of weather monitored. Also, the data generated uses the bounds for temperature and wind-speeds in San Jose, CA available as historical data at http://www.usclimatedata.com with GPS coordinates of a San Jose, CA building.

### Weather Station: Message Format

```
{
        "guid":"0-ZZZ12345678-01A",
        "destination":"0-AAA12345678",
        "location":[
                37.312125,
                -121.757265
        ],
        "eventTime":"2017-11-01T22:55:52.928972Z",
        "payload":{
                "format":"urn:example:sensor:temp",
                "data":{
                        "temperature":53.5,
                        "windspeed":1.9
                }
        }
}
```

### IoT Simulator Usage

```
# generate hourly temperature & wind-speed data for last 24 hrs
./iotsimulator.py 24
# generate hourly temperature & wind-speed data for last 10000 hrs
./iotsimulator.py 10000
```

## Kafka Producer

Data generated by the 'iotsimulator.py' is piped to the Kafka producer, which inserts the IoT messages into the Kafka queue with topic 'iotmsgs'. The producer pushes these messages to Kafka server running on localhost:9092. Enough memory needs to be allocated for the Kafka server, as it manages the 'iotmsgs' queue in memory. This is specified via KAFKA_HEAP_OPTS="-Xmx200M" environment setting.

## Kafka Consumer

This is a Spark job, that creates a Kafka direct stream as a consumer of IoT messages on 'localhost:9092' with 'iotmsgs' Kafka topic. Kafka consumer fires the IoT message processing engine, this is the weather station data Spark stream processing engine. The processing engine gets called with the RDD that includes the latest batch of weather station IoT messages. It parses the RDD for the JSON messages. It creates Spark DataFrames for a sliding time window of 1 week first, and another DataFrame for 1 day. Then we process these DataFrames by extracting insight from the weather data IoT messages. The Spark job computes daily and weekly average, min and max of temperatures and wind speeds seen in data from a IoT weather station device.

It also computes the average, min, and max of temperatures and wind speeds in the whole RDD received, irrespective of the streaming windows defined, using Spark SQL queries on the RDD JSON records stored in a temporary DB 'iotmsgsTable'.

## Setting up IoT Spark Stream Processing Pipeline

```
# SSH AWS ec2-user Shell-1
cd ~/kafka_*
KAFKA_HEAP_OPTS="-Xmx32M" \
./bin/zookeeper-server-start.sh config/zookeeper.properties > \
 /tmp/zookeeper.log &

KAFKA_HEAP_OPTS="-Xmx200M" \
./bin/kafka-server-start.sh config/server.properties > \
 /tmp/kafka.log 2>&1 &

cd ~/iotsim
KAFKA_HEAP_OPTS="-Xmx1M" spark-submit --jars \
  spark-streaming-kafka-0-8-assembly_2.11-2.0.0-preview.jar \
  ./kafka-direct-iotmsg9.py localhost:9092 iotmsgs

# SSH AWS ec2-user Shell-2
KAFKA_HEAP_OPTS="-Xmx20M" \
  ./iotsimulator.py 100 | ~/kafka_2.11-0.10.1.0/bin/kafka-console-producer.sh \
  --broker-list localhost:9092 --topic iotmsgs
```

The zookeeper and Kafka server are started first in a t2micro AWS Linux VM ec2-user shell. Then the Spark stream processing job that consumes Kafka messages is started on '`localhost:9092`' with Kafka topic '`iotmsgs`'. In a different shell connected to the same ec2-user Linux VM, the IoT Simulator is

started and the data generated is piped to the Kafka producer, which pushes the messages to the Kafka server running on 'localhost:9092' with topic 'iotmsgs'.

## Results

```
Show Weekly IoT Sensor Stats
+----------------+--------------------+---------------------------+---------------------------------
--+
|     guid       | window | min(temp)| max(temp)| avg(temp)| min(windspeed)| max(windspeed)|
avg(windspeed)|
+----------------+--------------------+---------------------------+---------------------------------
--+
|0-ZZZ12345678-01A|[2017-11-16 00:00...|   44.3   |   65.0   |   50.75   |   0.1   |   10.9  |   5.789
|
+----------------+--------------------+---------------------------+---------------------------+-----
--+

Show Daily IoT Sensor Stats
+----------------+--------------------+---------------------------+---------------------------------
--+
|     guid       | window | min(temp)| max(temp)| avg(temp)| min(windspeed)| max(windspeed)|
avg(windspeed)|
+----------------+--------------------+---------------------------+---------------------------------
--+
|0-ZZZ12345678-01A|[2017-11-16 00:00...|   45.2   |   64.4   |    53.47  |   1.3   |  10.9   |  4.922
|
|0-ZZZ12345678-01A|[2017-11-17 00:00...|   45.4   |   57.6   |    49.99  |   0.3   |  10.8   |  5.637
|
|0-ZZZ12345678-01A|[2017-11-18 00:00...|   44.3   |   63.6   |    51.61  |   0.2   |  10.8   |  6.312
|
|0-ZZZ12345678-01A|[2017-11-19 00:00...|   44.7   |   65.0   |    50.84  |   0.1   |  10.9   |  5.245
|
|0-ZZZ12345678-01A|[2017-11-20 00:00...|   45.0   |   59.8   |    49.19  |   0.8   |  10.2   |  6.415
|
+----------------+--------------------+---------------------------+---------------------------------
--+

Show SUMMARY of IoT Sensor temperature data
+-------+-----------------+
|summary|      temperature|
+-------+-----------------+
|  count|              100|
|   mean|            50.75|
| stddev|4.797168777813462|
|    min|             44.3|
|    max|             65.0|
+-------+-----------------+

Show SUMMARY of IoT Sensor windspeed data
+-------+------------------+
|summary|         windspeed|
+-------+------------------+
|  count|               100|
|   mean| 5.789000000000002|
| stddev|3.4867684961084726|
|    min|               0.1|
|    max|              10.9|
+-------+------------------+
```

## Challenges

Spark Structured Stream processing does not allow the consumer to set up event time windows longer than a month, the limit is upto 4 weeks. The challenge was to learn to use Structured Streaming Spark programming concepts to implement the objective of doing the data analysis on daily and weekly event windows. The second challenge was to deal with the AWS t2micro Linux VM, it constantly ran out of Java heap space, even 700MB free space was getting consumed quickly in one run, when I send even 100 JSON IoT Messages via Kafka topic. I worked around this development hurdle by using Databricks Community edition account to test the PySpark job on a 10000 JSON IoT messages file.