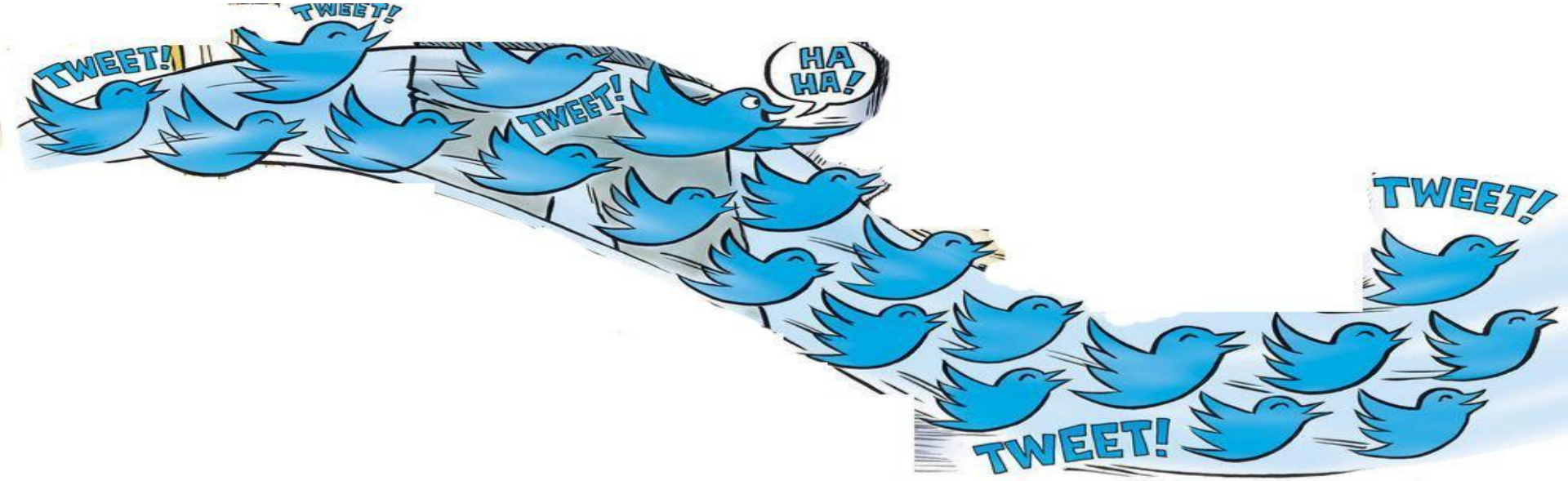
Big Data Pipeline

Pseudo distributed mode

Daniel Odiche
Tensaey Ayalew

Twitter as Input

Every second, on average, around **6,000** tweets are tweeted on Twitter, which corresponds to over **350,000** tweets sent per minute, 500 million tweets per day.



Ingestion

Hosebird + Kafka

```
65
66      private static Client createHoseBirdClient(){
67
68          Hosts hosebirdHosts = new HttpHosts(Constants.STREAM_HOST);
69          StatusesFilterEndpoint hosebirdEndpoint = new StatusesFilterEndpoint();
70
71          List<String> terms = Lists.newArrayList("twitter");
72          hosebirdEndpoint.trackTerms(terms);
73
74          Authentication hosebirdAuth = new OAuth1(AppConfig.Twitter.CUSTOMER_API_KEY, AppConfig.Twitter.CUSTOMER_API_SECRET, AppConfig.Twitter.AUTH_TOKEN, AppConf
75
76
77          ClientBuilder builder = new ClientBuilder()
78          .name("Inge")
79          .hosts(hosebirdHosts)
80          .authentication(hosebirdAuth)
81          .endpoint(hosebirdEndpoint)
82          .processor(new StringDelimitedProcessor(msgQueue));
83
84          return builder.build();
85      }
86
87      private Producer<String, String> createProducer(){
88
89          Properties props = new Properties();
90
91          props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, AppConfig.Kafka.KAFKA_BROKERS);
92          props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, AppConfig.Kafka.KEY_SERIALIZER_CLASS);
93          props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, AppConfig.Kafka.VALUE_SERIALIZER_CALSS);
94          props.put(ProducerConfig.ACKS_CONFIG, AppConfig.Kafka.ACKNOWLEDGEMENT_ALL);
95          props.put(ProducerConfig.RETRIES_CONFIG, 0);
96
97          Producer<String, String> producer = new KafkaProducer<String, String>(props);
98
99          return producer;
100     }
101
```

Consumption

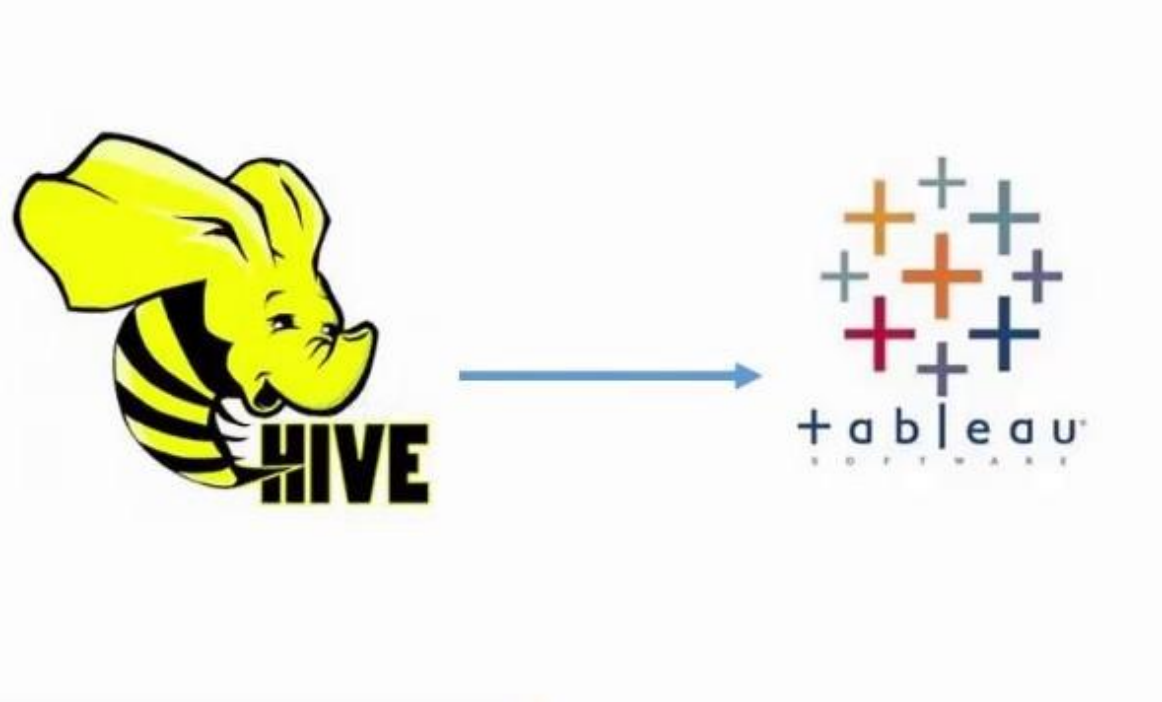
Kafka Consumer - to accept the data

Spark Streaming - Selective operations

```
66      // Create a Java Spark Context
67      SparkConf sparkConf = new SparkConf();
68      sparkConf.setAppName("SparkKafka");
69      sparkConf.setMaster("local");
70
71      JavaStreamingContext streamingContext = new JavaStreamingContext(sparkConf, Durations.seconds(1));
72
73
74      Map<String, Object> kafkaParams = new HashMap<>();
75      kafkaParams.put("bootstrap.servers", "localhost:9092");
76      kafkaParams.put("key.deserializer", StringDeserializer.class.getName());
77      kafkaParams.put("value.deserializer", StringDeserializer.class.getName());
78      kafkaParams.put("group.id", "use_a_separate_group_id_for_each_stream");
79      kafkaParams.put("auto.offset.reset", "latest");
80      kafkaParams.put("enable.auto.commit", false);
81
82      Collection<String> topics = Arrays.asList("sessiondata");
83
84      JavaInputDStream<ConsumerRecord<String, String>> stream =
85                KafkaUtils.createDirectStream(
86                  streamingContext,
87                  LocationStrategies.PreferConsistent(),
88                  ConsumerStrategies.<String, String>Subscribe(topics, kafkaParams)
89                );
90
91
92      JavaDStream<String> dstream = stream.map(st -> convertTweetToJSON(st.value()))
93      .filter(tweet -> tweet != null)
94      .filter(tweet -> !tweet.isEmpty());
95
96
97      dstream.foreachRDD(rdd -> {
98
99          if(!rdd.isEmpty()){
100             rdd.saveAsTextFile(SAVE_LOCATION);
101             System.out.println("saving rdd ...###" );
102
103     }
104     });
105
106     streamingContext.start();
107     streamingContext.awaitTermination();
108
```

Storage and Visualization

org.openx.data.jsonserde.JsonSerDe



```java
public class TableCreator {

    private static String driverName = "org.apache.hive.jdbc.HiveDriver";

    public static void main(String[] args) throws ClassNotFoundException,
            SQLException {

        Class.forName(driverName);

        Connection con = DriverManager.getConnection("jdbc:hive2://localhost:10000/default", "cloudera", "cloudera");
        Statement stmt = con.createStatement();

        stmt.execute("DROP TABLE tweets");
        stmt.execute("CREATE EXTERNAL TABLE tweets(created_at string, text string, name string, "
                + "followers_count int, friends_count int, retweet_count int, reply_count int) "
                + "ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe' "
                + "LOCATION 'hdfs://quickstart.cloudera:8020/cloudera/home/tweets'");

        System.out.println("Table tweets created.");

        con.close();

    }
}
```

Spark SQL

```
SparkConf sparkConf = new SparkConf();
sparkConf.setAppName("Spark Sql Hive");
sparkConf.setMaster("local");

sparkConf.set("hive.metastore.uris", "thrift://localhost:9083");

SparkSession sparkSession = SparkSession.builder().appName("Spark SQL-Hive").config(sparkConf)
        .config("spark.sql.hive.hiveserver2.jdbc.url", "jdbc:hive2://localhost:10000")
        .config("hive.metastore.warehouse.external.dir", "/user/cloudera/tweets")
        .enableHiveSupport().getOrCreate();

Dataset<Row>  tabledata = sparkSession.sql("SELECT * FROM tweets LIMIT 10");
tabledata.show();
```

Overall Architecture

kafka

Spark Streaming

tableau

Spark SQL

HIVE

Demo Time