Implementing Real-Time Analysis with Hadoop in Azure HDInsight

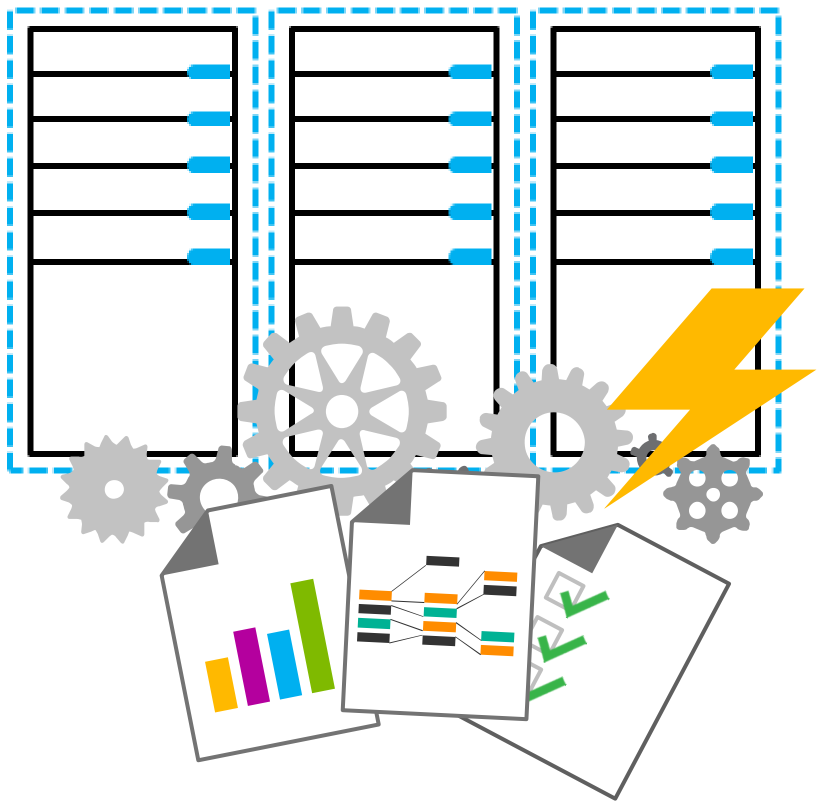
03 | Using Spark for Interactive Data Analysis

Graeme Malcolm | Snr Content Developer, Microsoft

- What is Apache Spark?
- How is Spark supported in Azure HDInsight?
- How do I work with data in Spark?
- How do I write Spark programs?
- What are Notebooks?
- How do I query data in Spark using SQL?
- What is Spark Streaming?

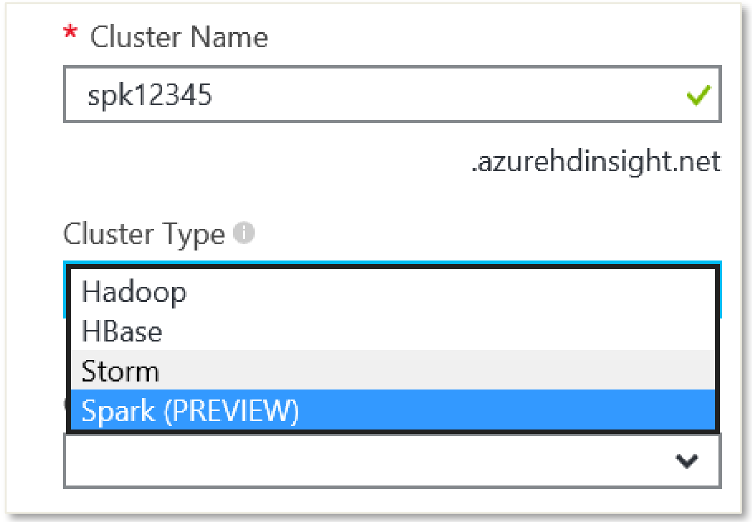
What is Apache Spark?

- A fast, general purpose computation engine that supports in-memory operations
- A unified stack for interactive, streaming, and predictive analysis
- Can run in Hadoop clusters



How is Spark supported in Azure HDInsight?

- HDInsight supports an **Spark** cluster type
  - Choose Cluster Type in the Azure Portal
- Can be provisioned in a virtual network


* Cluster Name

spk12345

.azurehdinsight.net

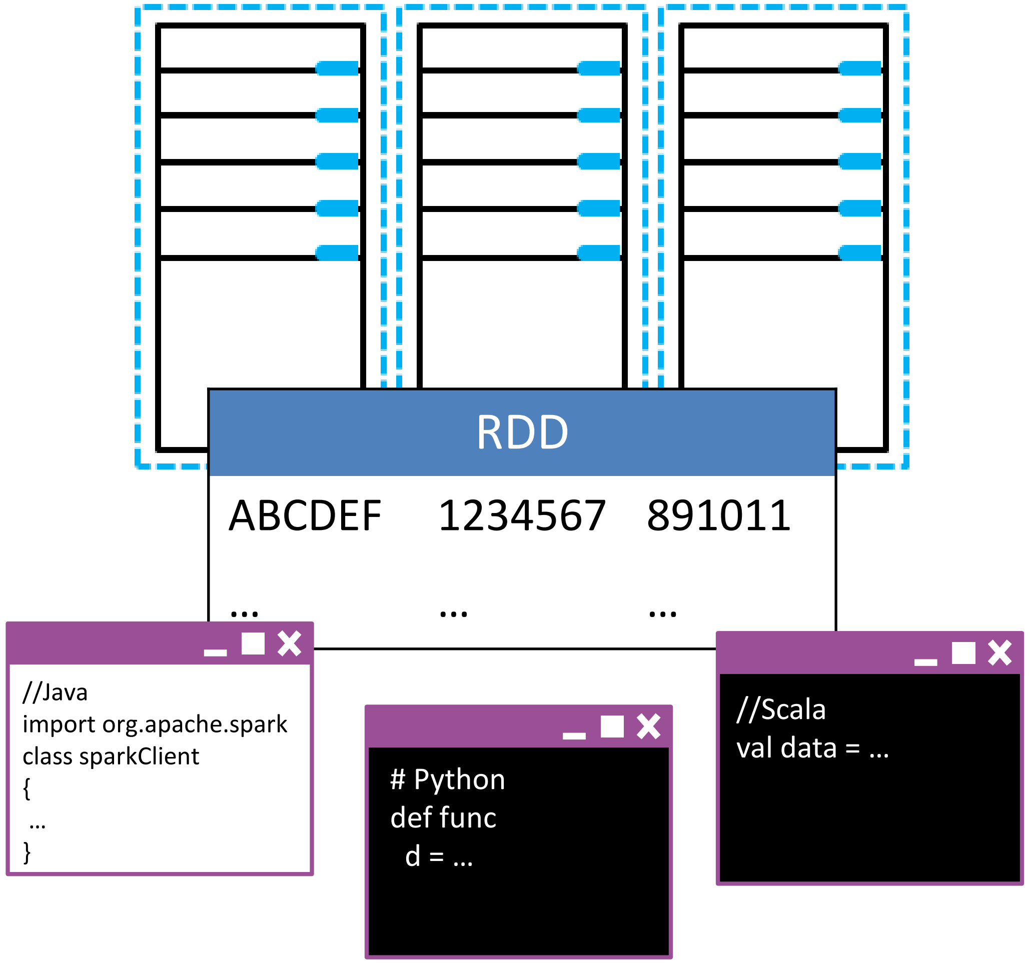
Cluster Type

DEMO

Provisioning a Spark Cluster

How do I work with data in Spark?

- The core abstraction for data in Spark is the *resilient distributed dataset* (RDD)
- An RDD represents a collection of items that can be distributed across compute nodes
- APIs for working with RDDs are provided for Java, Python, and Scala
  - HDInsight distribution includes Python and Scala shells

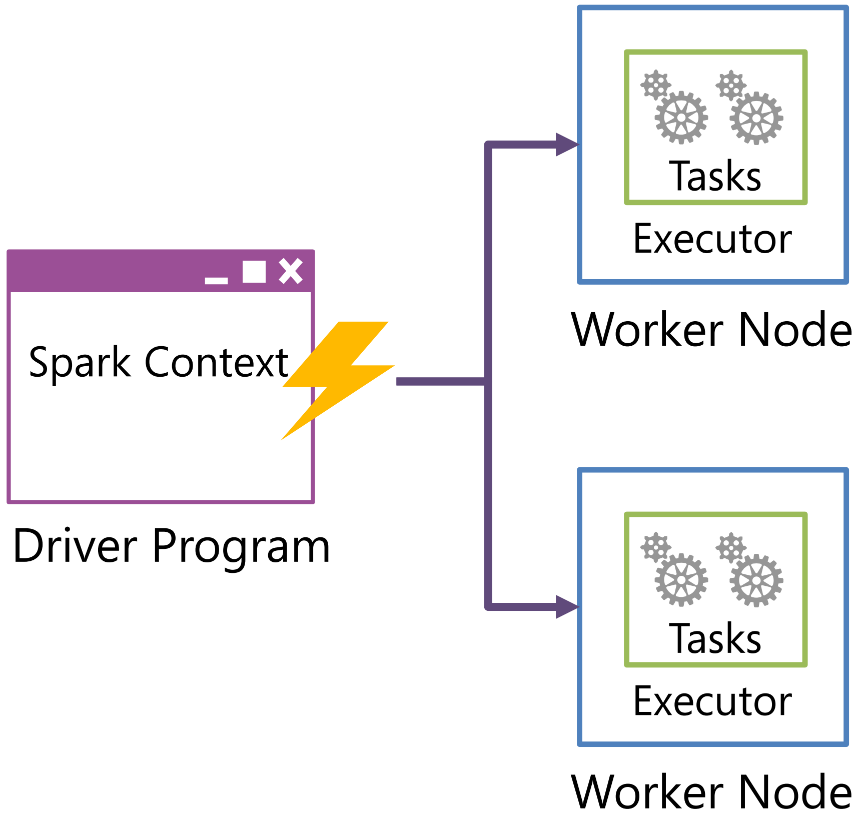


```
//Java
import org.apache.spark
class sparkClient
{
 …
}
```

```
# Python
def func
 d = …
```

```
//Scala
val data = …
```

- Distributed processing architecture consists of:
  - A *driver program*
  - One or more *worker nodes*
- The driver program uses a spark context to connect to the cluster…
- …and uses worker nodes to perform operations on RDDs



- To create a Spark Context:
  1. Create a configuration for your cluster and application
  2. Use the configuration to create a context

  (Spark shells have one pre-created)

Cluster URL

```
cfg = SparkConf().setMaster("local").setAppName("App")
sc = SparkContext(conf = cfg)
```

Your application name

- To create an RDD
  - Load from a source
    - Text file, JSON, XML, etc.
  - Parallelize a collection

Path to file
(default text delimiter is newline)

```
txtRDD = sc.textFile("/data/tweets.txt")
```

```
lstRDD = sc.parallelize(["A", "B", "C"])
```

List

- RDD operations include:
  - *Transformations*
    - Create a new RDD by transforming an existing one
  - *Actions*
    - Return results to the driver program or an output file
- Spark uses *Lazy Evaluation*
  - No execution occurs until an action
  - RDDs are recomputed with each action
    - Use **persist** action to retain in memory

Inline function

```
msTwts = txtRDD.filter(lambda t: "#ms" in t)
```

```
msTwts.count()
```

```
msTwts.persist()
```

DEMO

Working with Data in Spark

How do I write Spark programs?

- Most operations involve passing a function to a transformation or action
- Functions can be:
  - Explicitly declared
  - Passed inline
    - Python uses **lambda** keyword
    - Scala uses **=>** syntax
    - Java uses function classes or lambdas (Java 8)

```
RDD.filter(function)
```

```
def containsMSTag(txt):
    return "#ms" in txt

msTwts = txtRDD.filter(containsMSTag)
```

```
#Python
msTwts = txtRDD.filter(lambda txt: "#ms" in txt)
```

```
//Scala
val msTwts = txtRDD.filter(txt => txt.contains("#ms"))
```

Common Transformations:

- **filter**: Creates a filtered RDD
- **flatMap**: Applies a function to each element that returns multiple elements into a new RDD
- **map**: Applies a function to each element that returns an element in a new RDD
- **reduceByKey**: Aggregates values for each key in a key-value pair RDD

```
txt = sc.parallelize(["the owl and the pussycat",
                      "went to sea"])
```

{["the owl and the pussycat"], ["went to sea"]}

```
owlTxt = txt.filter(lambda t: "owl" in t)
```

{["the owl and the pussycat"]}

```
words = owlTxt.flatMap(lambda t: t.split(" "))
```

{["the"], ["owl"], ["and"], ["the"], ["pussycat"]}

```
kv = words.map(lambda key: (key, 1))
```

{["the",1], ["owl",1], ["and",1], ["the",1], ["pussycat",1]}

```
counts = kv.reduceByKey(lambda a, b: a + b)
```

{["the",2], ["owl",1], ["and",1], ["pussycat",1]}

Common Actions:

- **reduce**: Aggregates the elements of an RDD using a function that takes two arguments
- **count**: Returns the number of elements in the RDD
- **first**: Returns the first element in the RDD
- **collect**: Returns the RDD as an array to the driver program
- **saveAsTextFile**: Saves the RDD as a text file in the specified path

```
nums = sc.parallelize([1, 2, 3, 4])
```

{[1], [2], [3], [4]}

```
nums.reduce(lambda x, y: x + y)
```

10

```
nums.count()
```

4

```
nums.first()
```

1

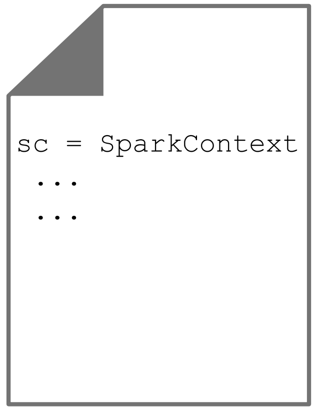
```
nums.collect()
```

[1, 2, 3, 4]

```
nums.saveAsTextFile("/results")
```

/results/part-00000

- To create a standalone application:
  - Create a Python script
  - Use Maven to build Scala or Java apps
  - Include code to create Spark context
- To run a standalone application:
  - Use the **spark-submit** script



```
sc = SparkContext
 ...
 ...
```

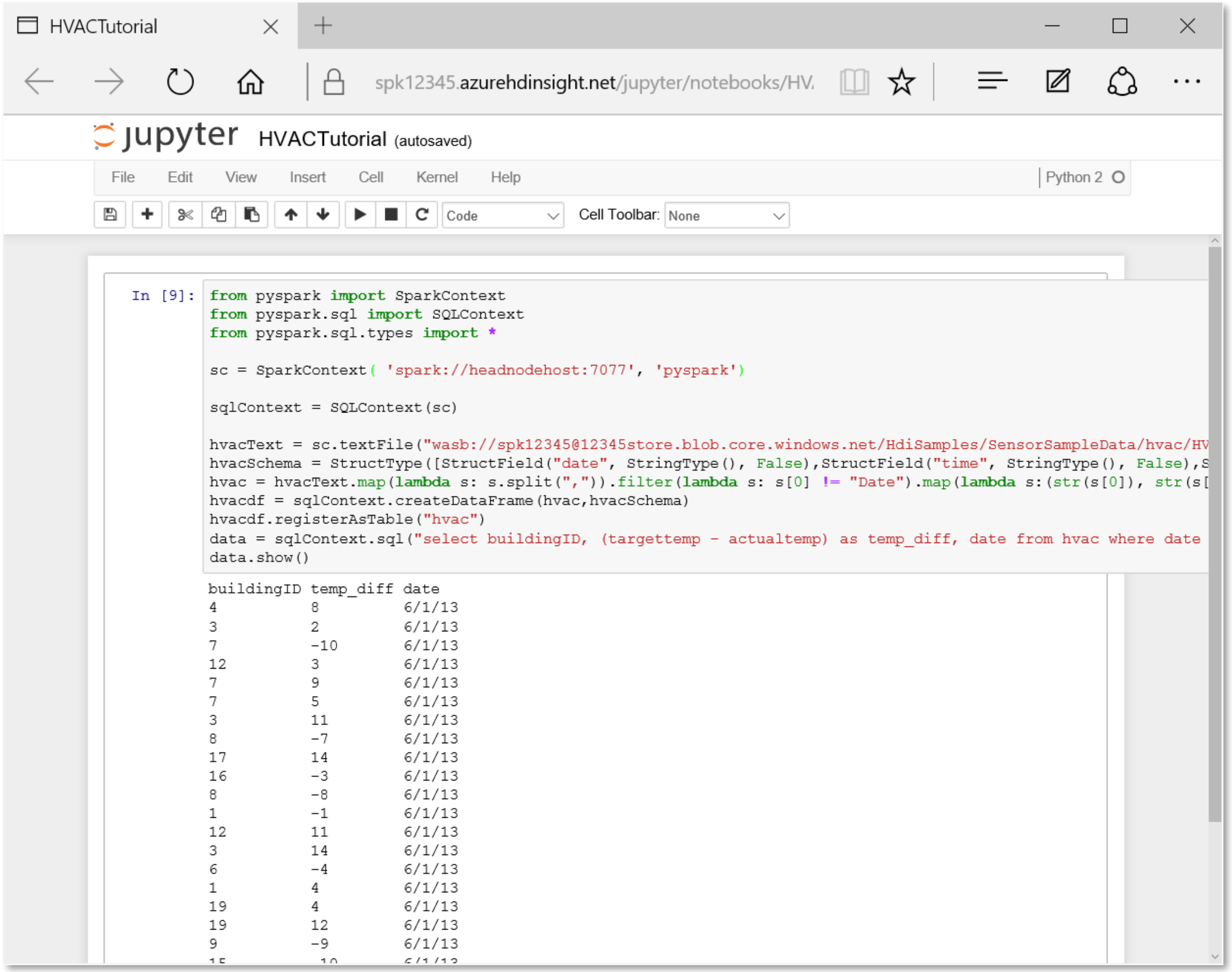
```
spark-submit myscript.py
```

DEMO

Submitting a Standalone Python Script

What are Notebooks?

- Web-based interactive consoles for
  - Experimentation
  - Collaboration
- Spark HDInsight clusters include Jupyter
  - Interactive Python
  - Interactive Scala


HVACTutorial
spk12345.azurehdinsight.net/jupyter/notebooks/HV
jupyter HVACTutorial (autosaved)
File Edit View Insert Cell Kernel Help
Python 2
Code
Cell Toolbar: None
In [9]:

```
from pyspark import SparkContext
from pyspark.sql import SQLContext
from pyspark.sql.types import *

sc = SparkContext( 'spark://headnodehost:7077', 'pyspark')

sqlContext = SQLContext(sc)

hvacText = sc.textFile("wasb://spk12345@12345store.blob.core.windows.net/HdiSamples/SensorSampleData/hvac/HV
hvacSchema = StructType([StructField("date", StringType(), False),StructField("time", StringType(), False),S
hvac = hvacText.map(lambda s: s.split(",")).filter(lambda s: s[0] != "Date").map(lambda s:(str(s[0]), str(s[
hvacdf = sqlContext.createDataFrame(hvac,hvacSchema)
hvacdf.registerAsTable("hvac")
data = sqlContext.sql("select buildingID, (targettemp - actualtemp) as temp_diff, date from hvac where date
data.show()
```

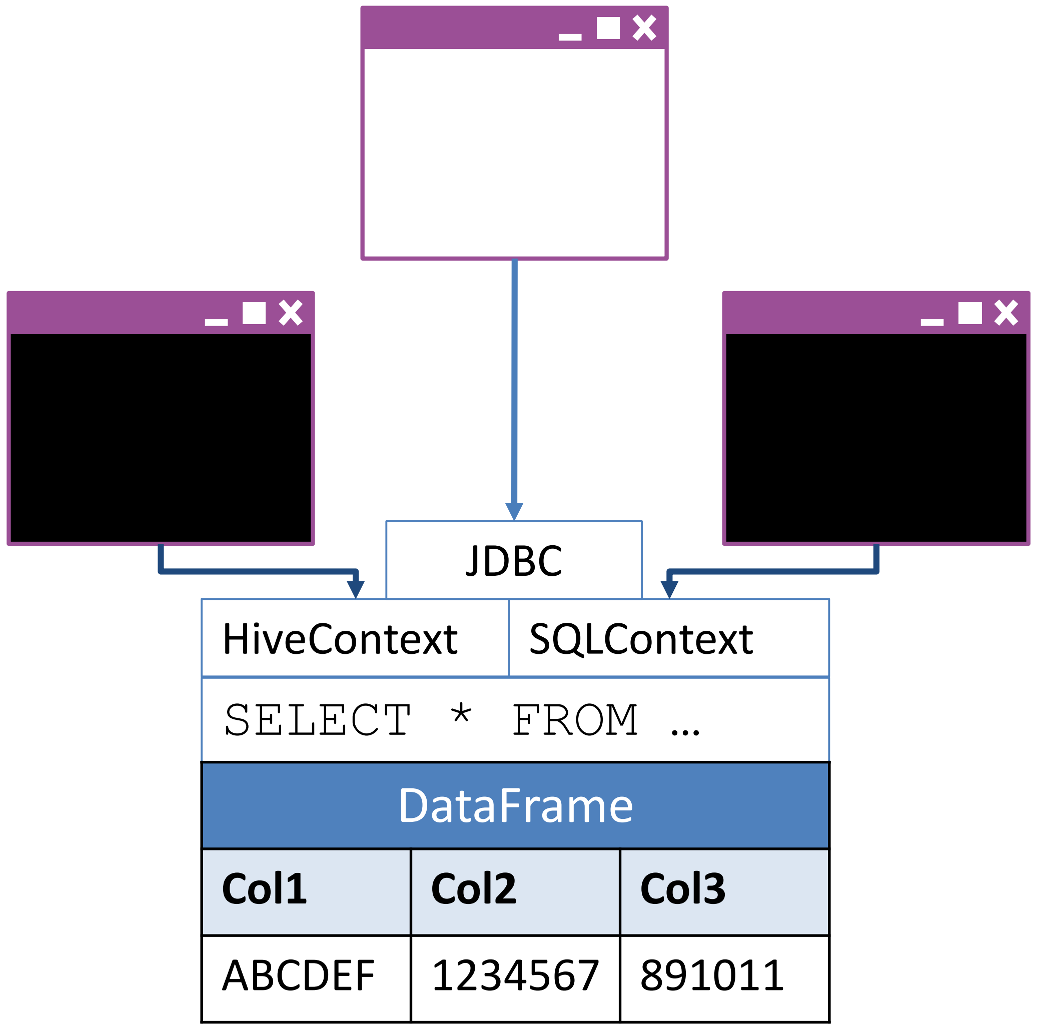
```
buildingID temp_diff date
4          8         6/1/13
3          2         6/1/13
7          -10       6/1/13
12         3         6/1/13
7          9         6/1/13
7          5         6/1/13
3          11        6/1/13
8          -7        6/1/13
17         14        6/1/13
16         -3        6/1/13
8          -8        6/1/13
1          -1        6/1/13
12         11        6/1/13
3          14        6/1/13
6          -4        6/1/13
1          4         6/1/13
19         4         6/1/13
19         12        6/1/13
9          -9        6/1/13
```

DEMO

Using Notebooks

How do I query data in Spark using SQL?

- Spark SQL provides a query interface for structured data
- DataFrames are used to abstract RDDs and define a schema
- There are two API entry points:
  - HiveContext
  - SQLContext
- Client applications can connect to Spark SQL using JDBC



- Infer the schema of an RDD by using Reflection

```
# Python
rows = txtRDD.map(lambda c: Row(name=c[0], email=c[1]))
contacts = sqlContext.inferSchema(rows)
```

```
// Scala
case class Contact (name: String, email: String)
val contacts = txtRDD.map(c => Contact(c(0), c(1)).toDF()
```

- Specify the schema programmatically

```
# Python
fields = [StructField("name", StringType(), False), StructField("email", StringType(), False)]
schma = StructType(fields)
contacts = sqlContext.createDataFrame(txtRDD, schma)
```

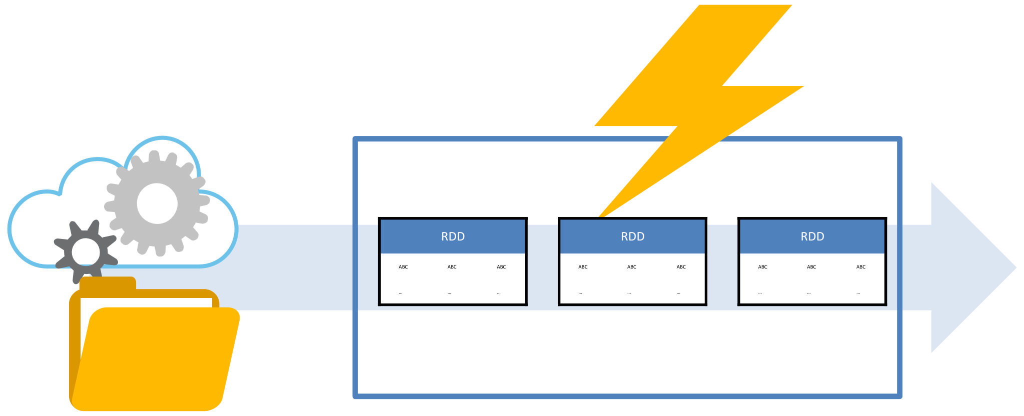
```
// Scala
val schemaString = "name,email"
val schma = StructType(schemaString.split(",").map(fName => StructField(fName, StringType, true))
val contacts = sqlContext.createDataFrame(rowRDD, schma)
```

DEMO

Using Spark SQL

What is Spark Streaming?

- Streaming module built on Spark
- Data is presented as a sequence of RDDs in discretized streams (*DStreams*)
- Many sources supported:
  - TCP Socket
  - File
  - Kafka
  - Flume
  - Azure Event Hubs



1. Create a streaming context
2. Create an RDD from a streaming source
3. Perform operations on the RDD
   - Regular RDD operations
   - Streaming-specific operations
4. Start the streaming context

```
ssc = StreamingContext(sc, 1)

r = ssc.socketTextStream("localhost", 77)

words = r.flatMap(...)
pairs = words.map(...)
counts = pairs.reduceByKeyAndWindow(...)
ssc.start()
ssc.awaitTermination()
```

DEMO

Using Spark Streaming

- What is Apache Spark?
- How is Spark supported in Azure HDInsight?
- How do I work with data in Spark?
- How do I write Spark programs?
- What are Notebooks?
- What are Dataframes?
- How do I query data in Spark using SQL?
- What is Spark Streaming?

Microsoft