Introduction to Spark in 50 minutes

Maria Dominguez, Xavier Tordoir

diSummit 2019

LUNATECH

SIMPLIFY YOUR IT

JVM development

Devops

ML & Big Data

Scala TensorFlow play Spark kafka amazon webservices DC/OS Java akka kubernetes cassandra ActiveMQ Kotlin

LUNATECH

105	27	Lots
Employees	Nationalities	Open source

Rotterdam NL

We welcome you to one of our offices

Paris FR

Amsterdam NL

LUNATECH

Outline

Spark

DataFrames & Datasets

Spark notebooks

ML concepts

Streaming

Hand-on set-up

VirtualBox installed:

https://www.virtualbox.org/wiki/Downloads

Download appliance (1.7GB):

https://xtordoirtmp.s3-eu-west-1.amazonaws.com/sparkintro.ova

Import Appliance in virtualBox:

- Menu “File” -> “Import Appliance”

Start VM “sparkintro”

Open http://localhost:9000 in browser

Spark history

2006

Batch processing

Trivial operations are difficult (filter, join)

Writing to disk

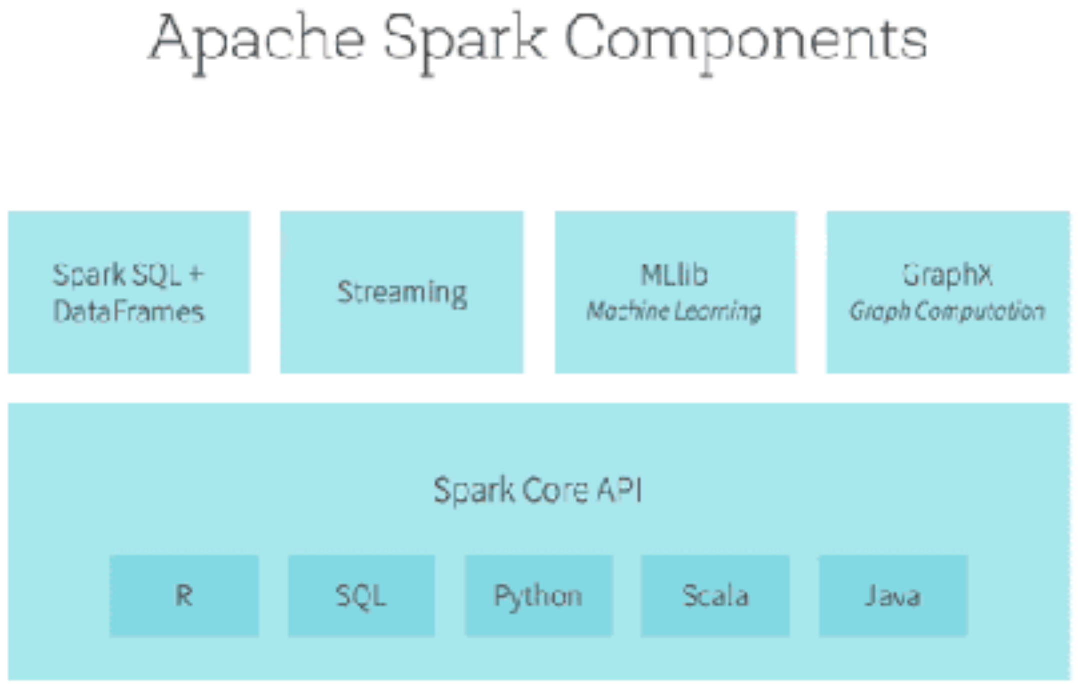
2009

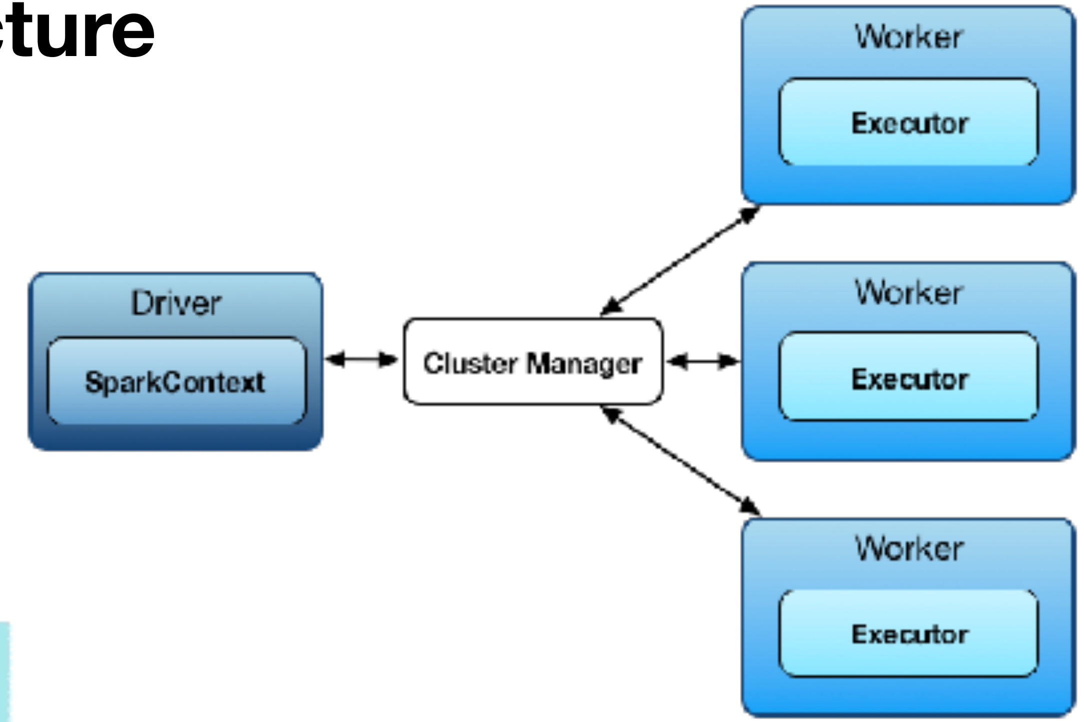
Batch and stream processing

Trivial operations are difficult (filter, join)

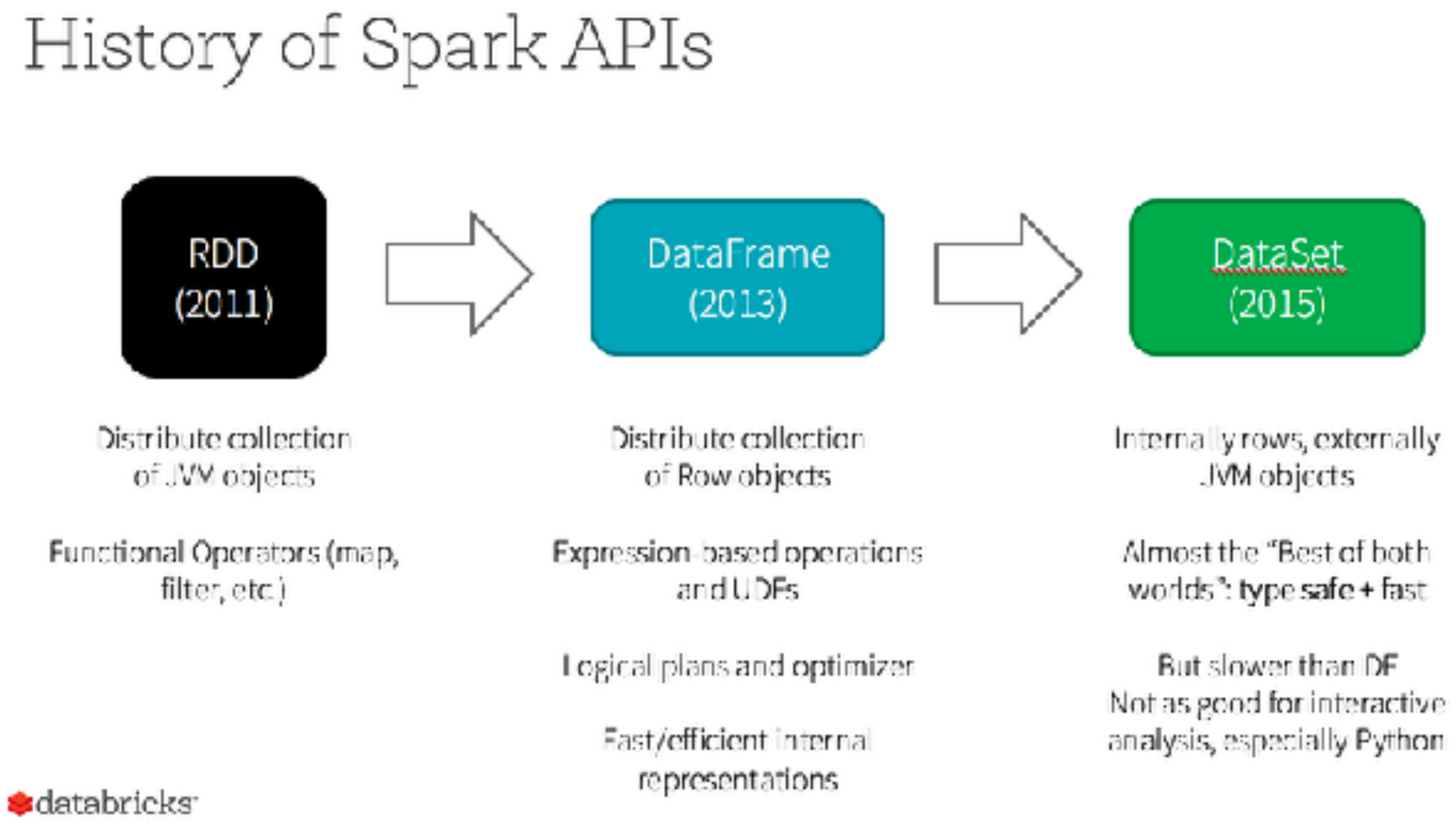
In memory computation

Spark components & architecture





Spark: From RDD to Dataset



History of Spark APIs

RDD (2011) → DataFrame (2013) → DataSet (2015)

RDD (2011)	DataFrame (2013)	DataSet (2015)
Distribute collection of JVM objects	Distribute collection of Row objects	Internally rows, externally JVM objects
Functional Operators (map, filter, etc.)	Expression-based operations and UDFs	Almost the "Best of both worlds": type safe + fast
	Logical plans and optimizer	But slower than DF Not as good for interactive analysis, especially Python
	Fast/efficient internal representations	

databricks

Spark processing

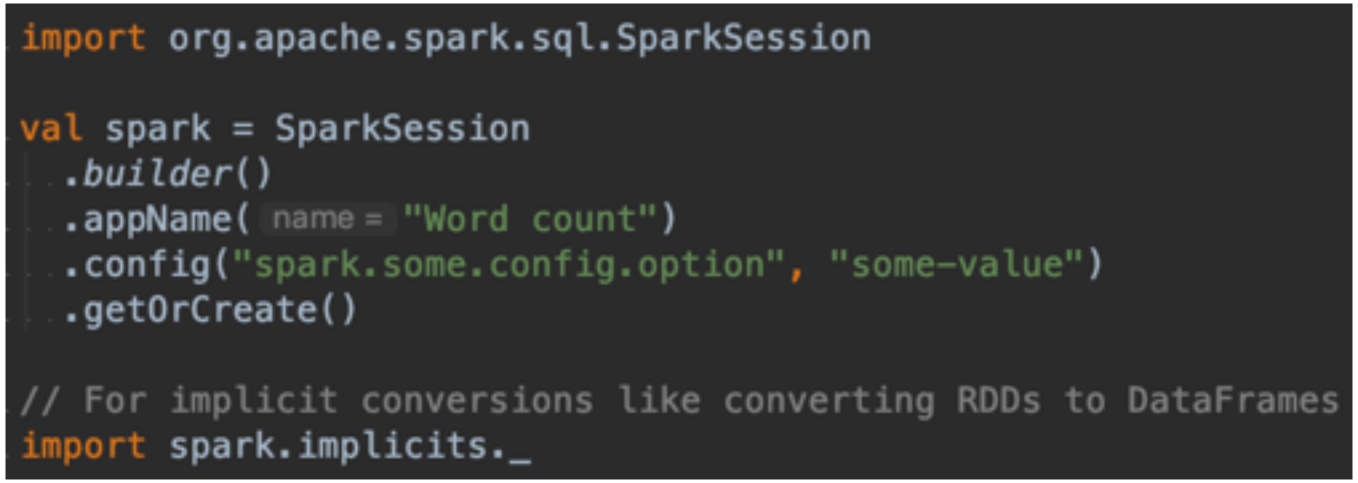
Batch processing	Real time processing
Large group of data processed in a single run	Instantaneously data (events) processing
Entire data pre-selected and fed to the application	Stringent constrains in response time
Eg: Training data model	Eg: Prediction making

Spark SQL

Structured data processing
Extra optimisation by Spark: tungsten (memory management) + catalyst (query optimiser)

- SQL API
- Dataset API

Starting point: **`SparkSession`** *(Already available in the notebooks/spark-shell as:* `spark` *)*



```
import org.apache.spark.sql.SparkSession

val spark = SparkSession
  .builder()
  .appName( name = "Word count")
  .config("spark.some.config.option", "some-value")
  .getOrCreate()

// For implicit conversions like converting RDDs to DataFrames
import spark.implicits._
```

Spark Datasets

Distributed collection of data

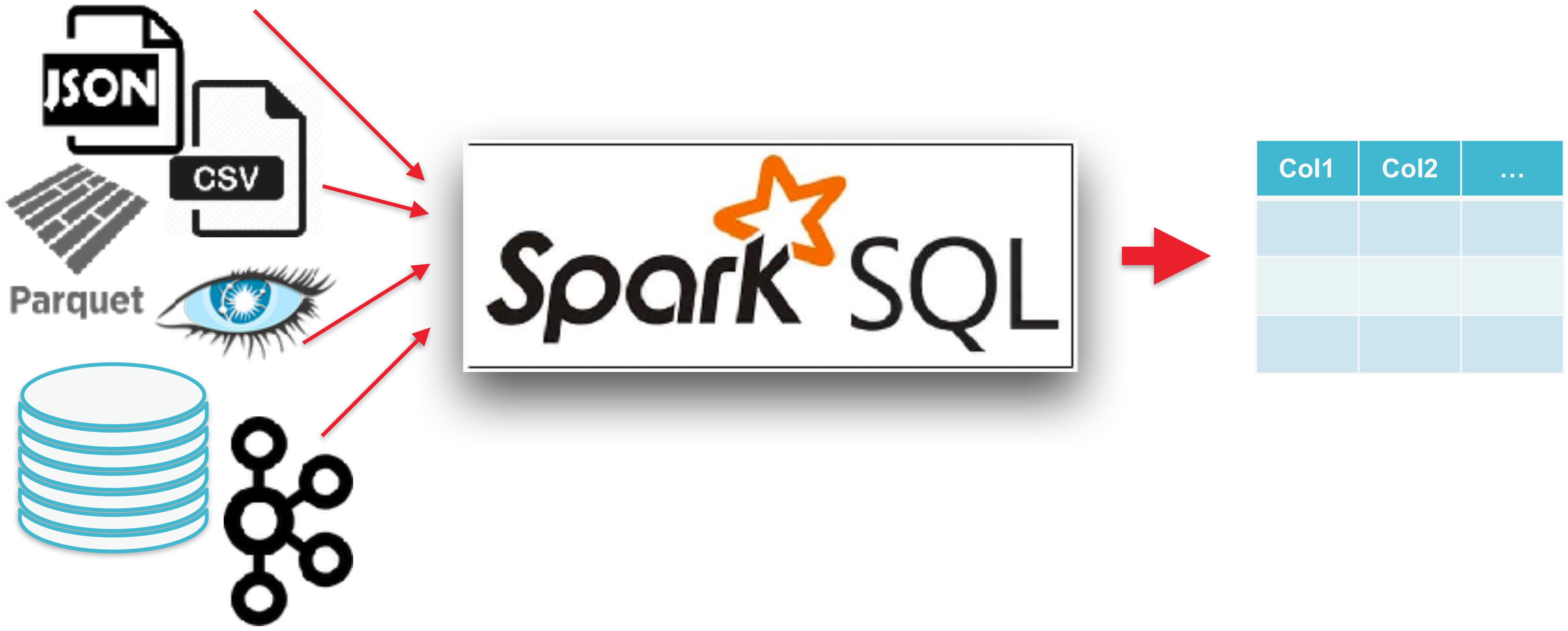
Strongly typed

A Dataset can be constructed from JVM objects and then manipulated using functional transformations (`map`, `flatMap`, `filter`)

Encoders

API in Scala/Java

Spark Datasets



Spark DataFrames / SQL

```
DataFrame == Dataset[Row]

val df = spark.read.json("people.json")
df.printSchema()
```

DataFrame API

```
df.select($"name").show()
```

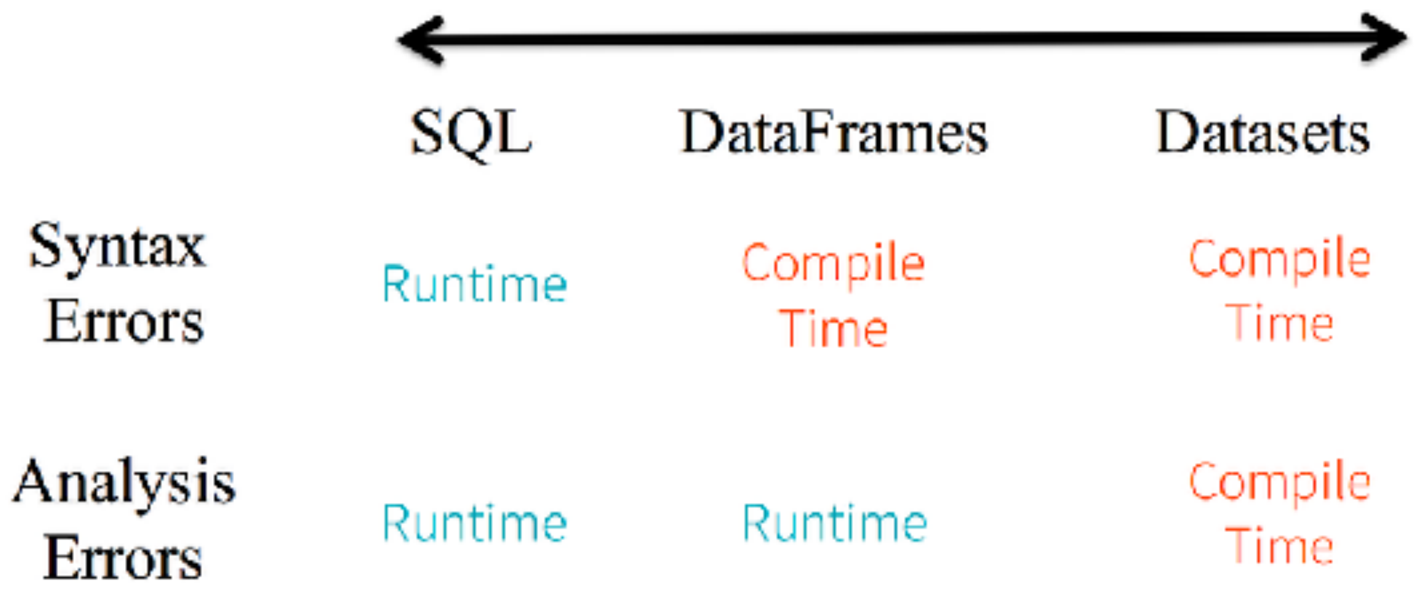
SQL API

```
df.createOrReplaceTempView("people")
spark.sql("SELECT name FROM people").show()
```

Spark DataFrame

Infer/Programatically define the Schema

Untyped (Dataset[Row])



	SQL	DataFrames	Datasets
Syntax Errors	Runtime	Compile Time	Compile Time
Analysis Errors	Runtime	Runtime	Compile Time

Spark notebooks

Interactive Spark shell in a browser using http://spark-notebook.io/

00_read_raw_history: Read historical data, do analysis, preprocessing and save results

01_train_model: Train linear model using preprocessed data

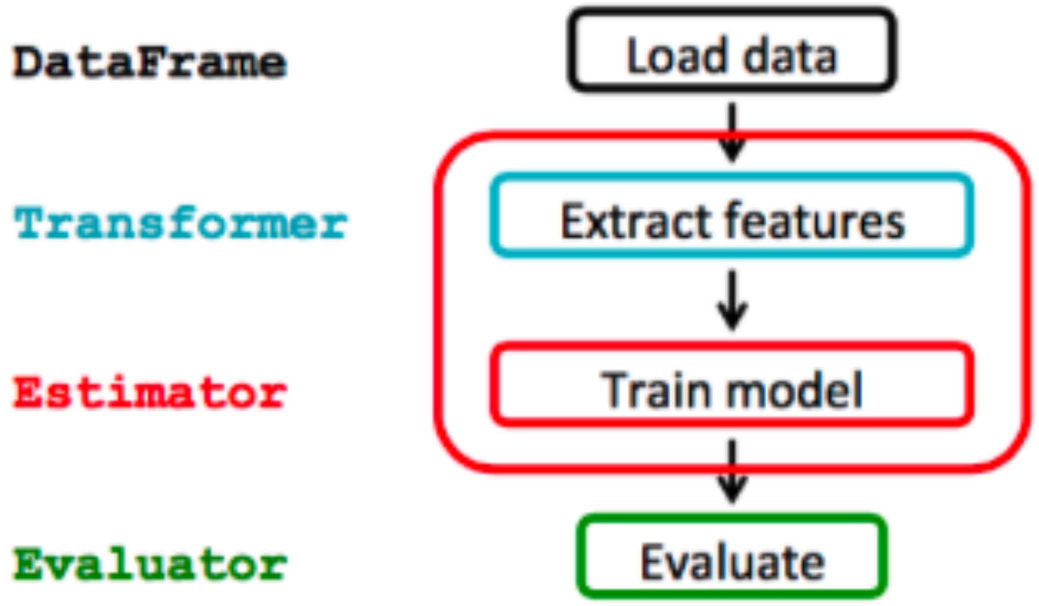
02_publish_stream: Generate a stream of data flowing in Kafka

03_serve_model_stream: Read data from Kafka and make predictions using our model

Spark ML concepts

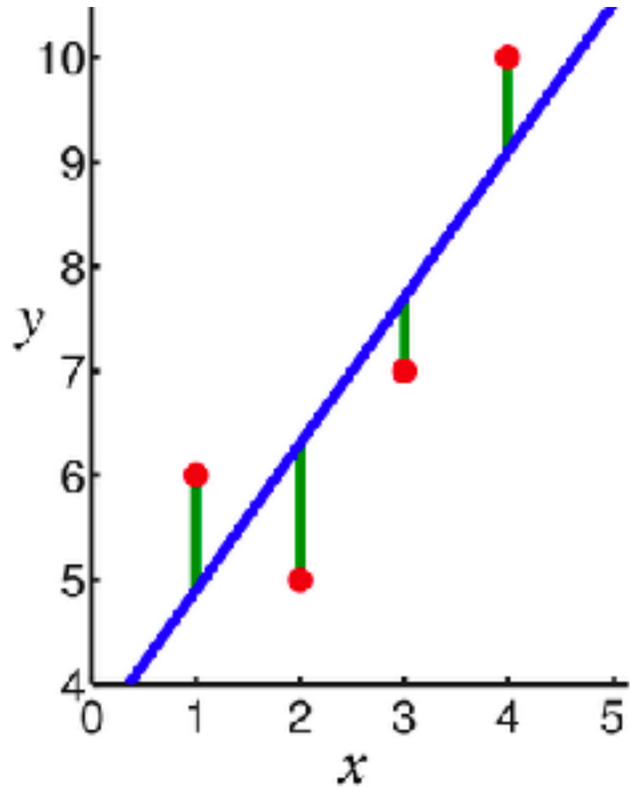
Pipeline (Sequence of `PipelineStages`):

- **Transformers**: Read a DataFrame, select a column, map it into a new column. Output is a new DataFrame with the mapped column appended.
- **Estimators**: Produce a Model from a given DataFrame (Transformer)



Linear Regression

In linear regression, the observations (**red**)
are assumed to be the result of random deviations (**green**)
from an underlying relationship (**blue**)
between a dependent variable (*y*)
and an independent variable (*x*).



Spark notebooks

Interactive Spark shell in a browser using http://spark-notebook.io/

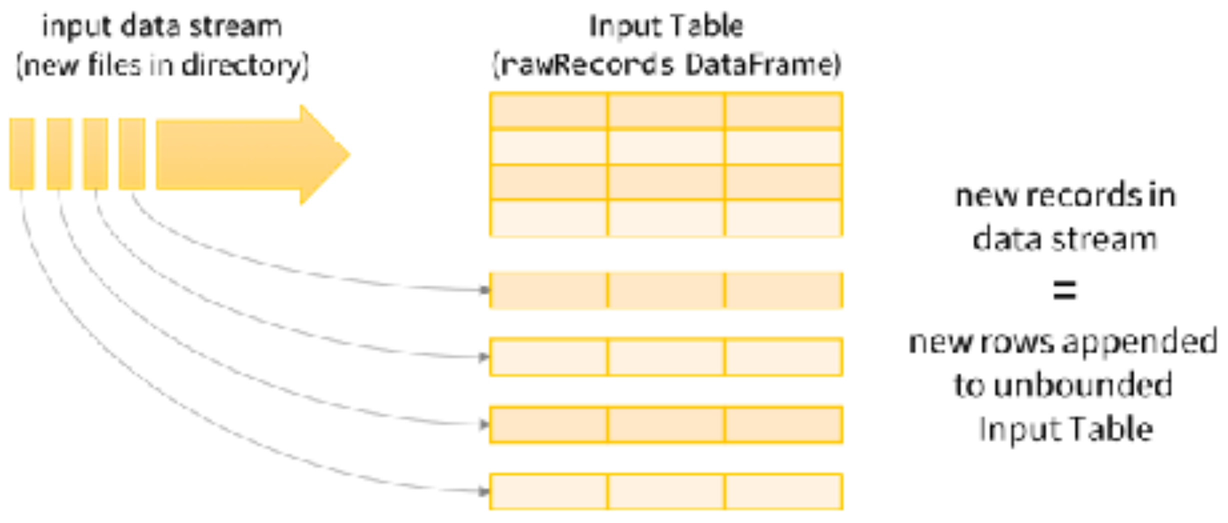
00_read_raw_history: Read historical data, do analysis, preprocessing and save results

01_train_model: Train linear model using preprocessed data

02_publish_stream: Generate a stream of data flowing in Kafka

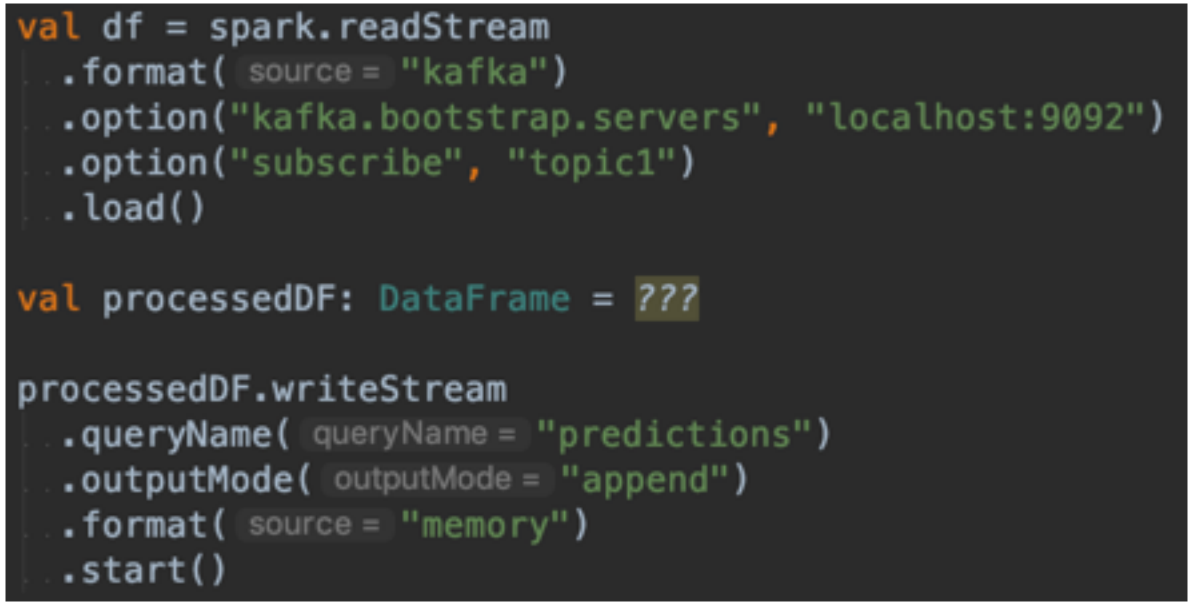
03_serve_model_stream: Read data from Kafka and make predictions using our model

Spark Structured Streaming



Structured Streaming Model
treat data streams as unbounded tables

Spark Structured Streaming



```
val df = spark.readStream
    .format( source = "kafka")
    .option("kafka.bootstrap.servers", "localhost:9092")
    .option("subscribe", "topic1")
    .load()

val processedDF: DataFrame = ???

processedDF.writeStream
    .queryName( queryName = "predictions")
    .outputMode( outputMode = "append")
    .format( source = "memory")
    .start()
```

Spark notebooks

Interactive Spark shell in a browser using http://spark-notebook.io/

00_read_raw_history: Read historical data, do analysis, preprocessing and save results

01_train_model: Train linear model using preprocessed data

02_publish_stream: Generate a stream of data flowing in Kafka

03_serve_model_stream: Read data from Kafka and make predictions using our model