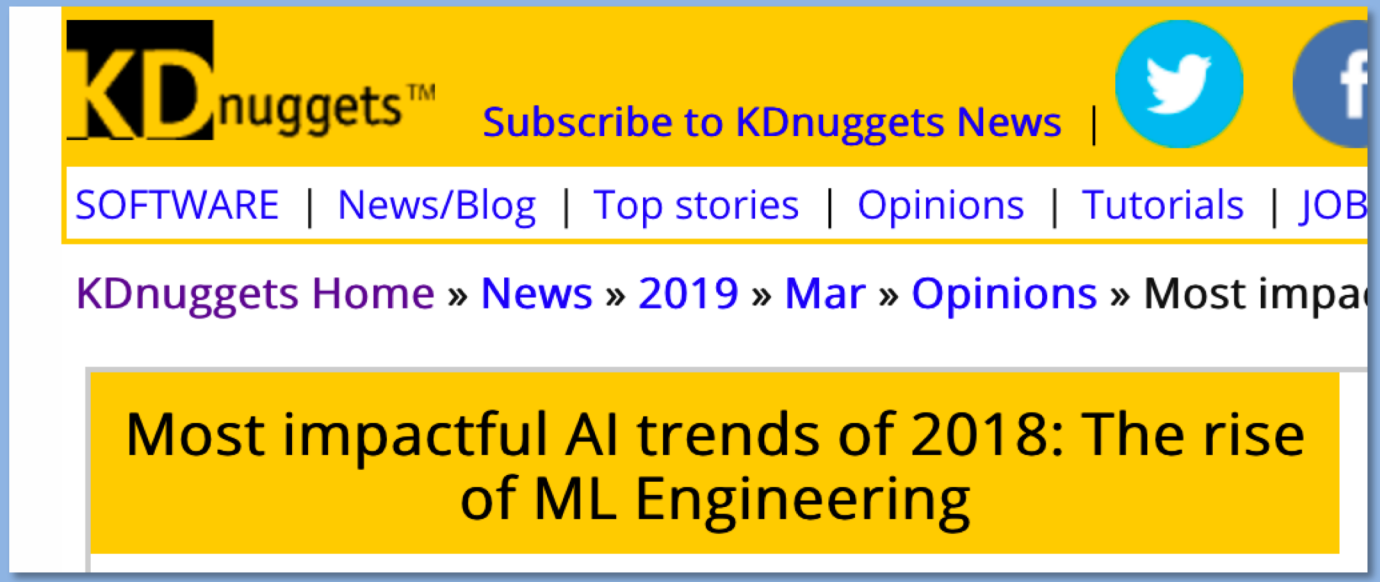
Construção de Data Pipelines em Apache Spark

Fabiane Bizinella Nardon
Chief Data Scientist
@fabianenardon

Data Science Pipeline



Three ways ML Engineering will grow in 2019

A common warning shared with aspiring Data Scientists is that 90% of the work is about **gathering and cleaning data**, or **validating, deploying, and monitoring models**. If that is the case, why are 90% of the frameworks and Github repositories (see this list for example)focused on model building?

A part of the job that demands so much of a practitioner's time should have proper tooling support.

https://www.kdnuggets.com/2019/03/most-impactful-ai-trends-2018-rise-ml-engineering.html

KDnuggets™ Subscribe to KDnuggets News |

SOFTWARE | News/Blog | Top stories | Opinions | Tutorials | JOB

Your AI skills are worth less than you think

Jan 2019 PLATINUM

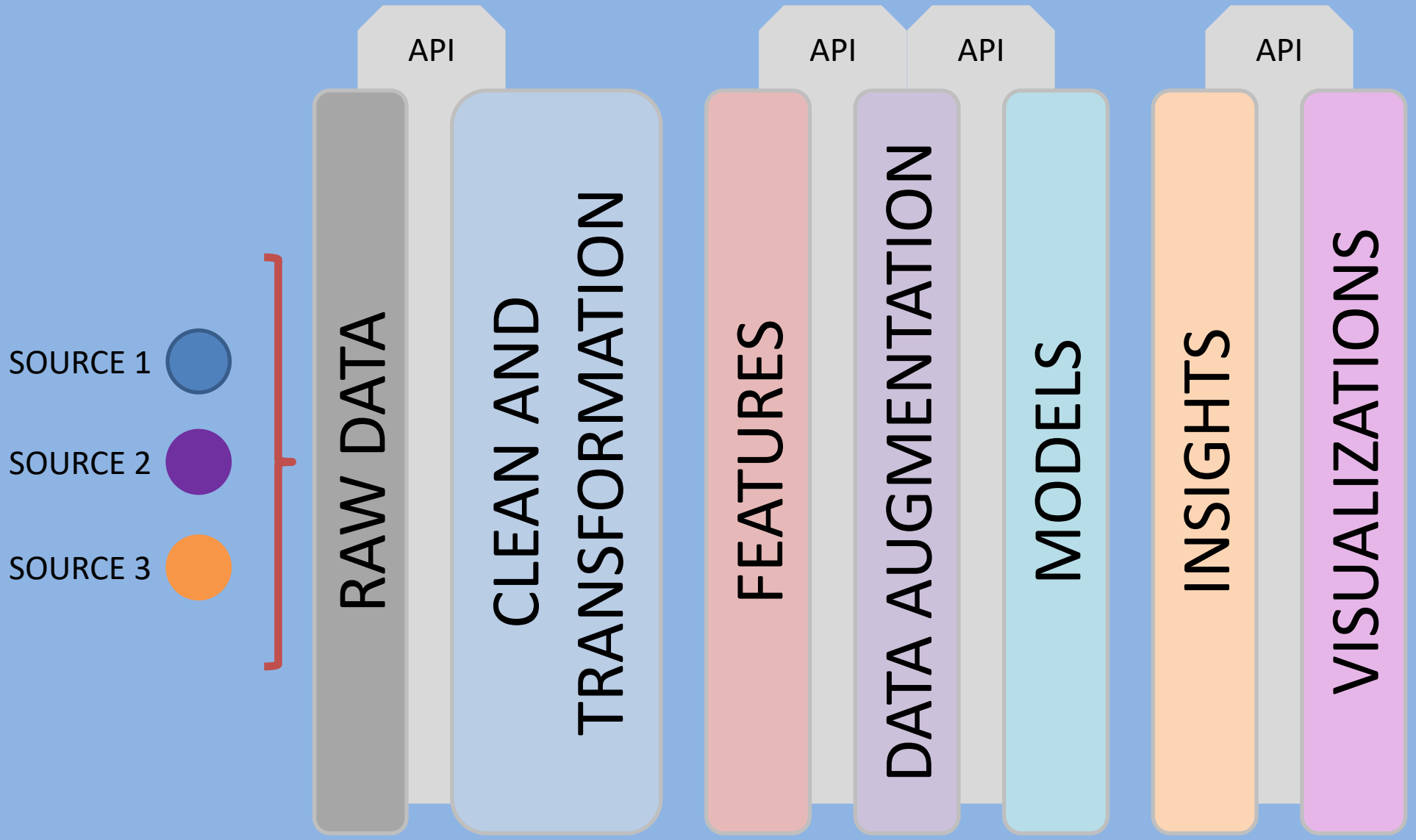
Data is more important than fancy AI architectures

Let's say that you have two AI startup founders, Alice and Bob. Their companies raised around the same amount of money, and are fiercely competing over the same market. Alice invests in the best engineers, PhDs with a good track record in AI research. Bob hires mediocre but competent engineers, and invests her ("Bob" is short for Roberta!) money in securing better data. On which company would you bet your money?

My money would be squarely on Bob. Why? At its essence, machine learning works by extracting information from a dataset and transferring it to the model weights. A better model is more more efficient at this process (in terms of time and/or overall quality), but assuming some baseline of adequacy (that is, the model is actually learning something) **better data will trump a better architecture**.

https://www.kdnuggets.com/2019/01/your-ai-skills-worth-less-than-you-think.html

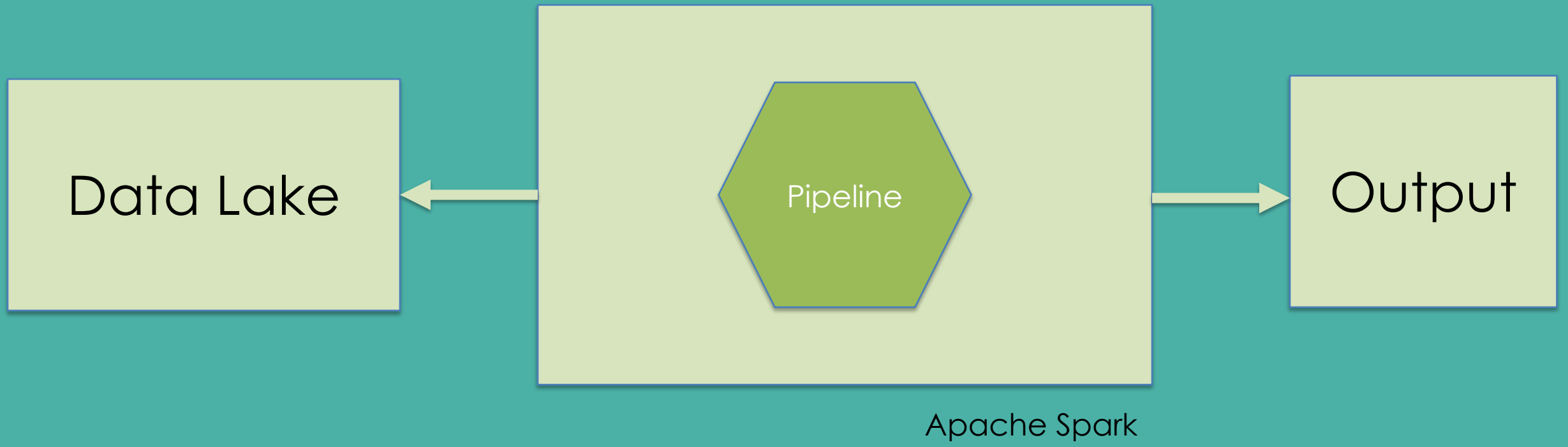
Data Science Pipeline



Escala

3.5 bilhões de novos registros
4.680 pipelines executados

em um dia

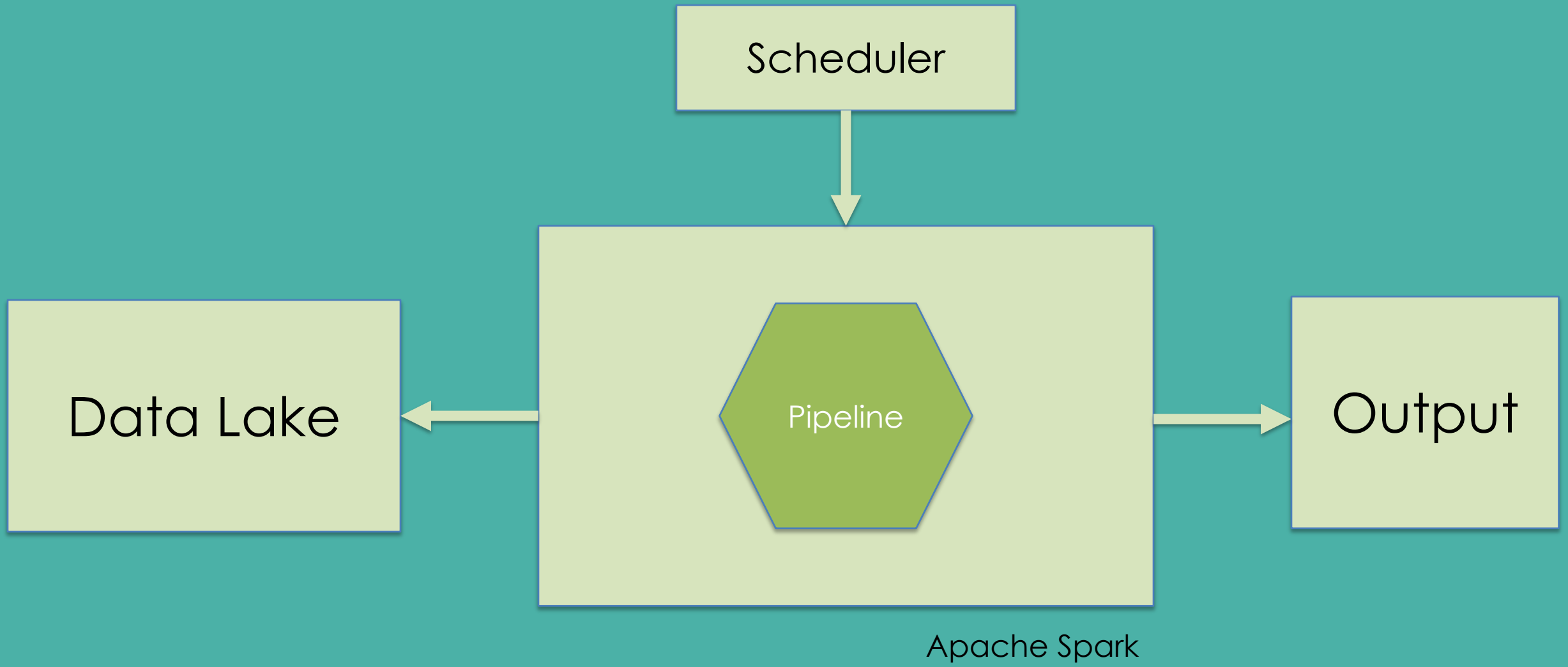

Data Lake
Pipeline
Output
Apache Spark

Spark SQL

```
val records = spark.read.option("header","true")
        .csv("example-database.csv").toDF

val filtered = records.filter("type = 'android'")

filtered.write
    .format("com.databricks.spark.csv")
    .option("header", "true")
    .save("newrecords.csv");
```

Spark SQL + meu código

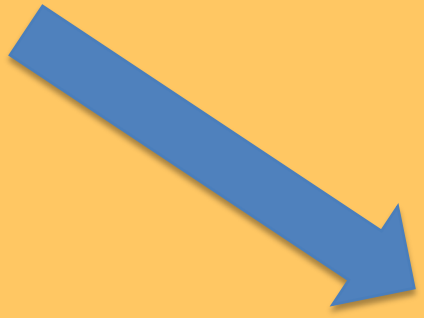
```
val records = spark.read.option("header","true")
        .csv("example-database.csv").toDF

val filtered = records.filter("type = 'android'")

val geoHash = new com.tailtarget.sparksqlgeohash.udf.SparkGeoHash();
spark.udf.register("geoHash", geoHash, DataTypes.StringType);

val recordsWithFunction = filtered.withColumn("generatedGeoHash",
      callUDF("geoHash", col("latitude"), col("longitude"), lit(12)))
      .select(col("id"), col("latitude"), col("longitude"),
              col("geo_hash"), col("generatedGeoHash"))

recordsWithFunction.write.format("com.databricks.spark.csv")
   .option("header", "true").save("newrecords.csv");
```


Scheduler
Data Lake
Pipeline
Output
Apache Spark

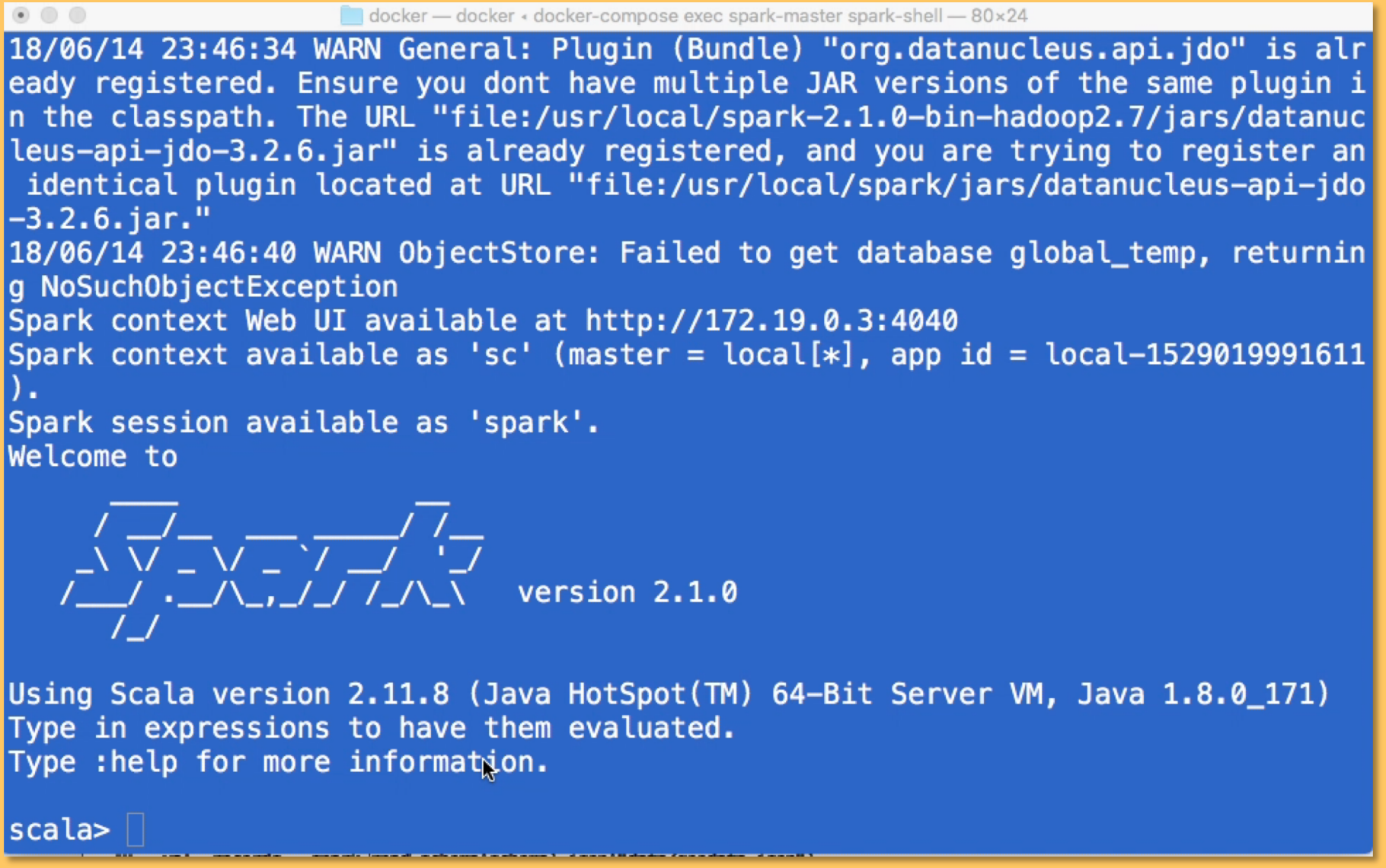
Experimentação vs. Produção

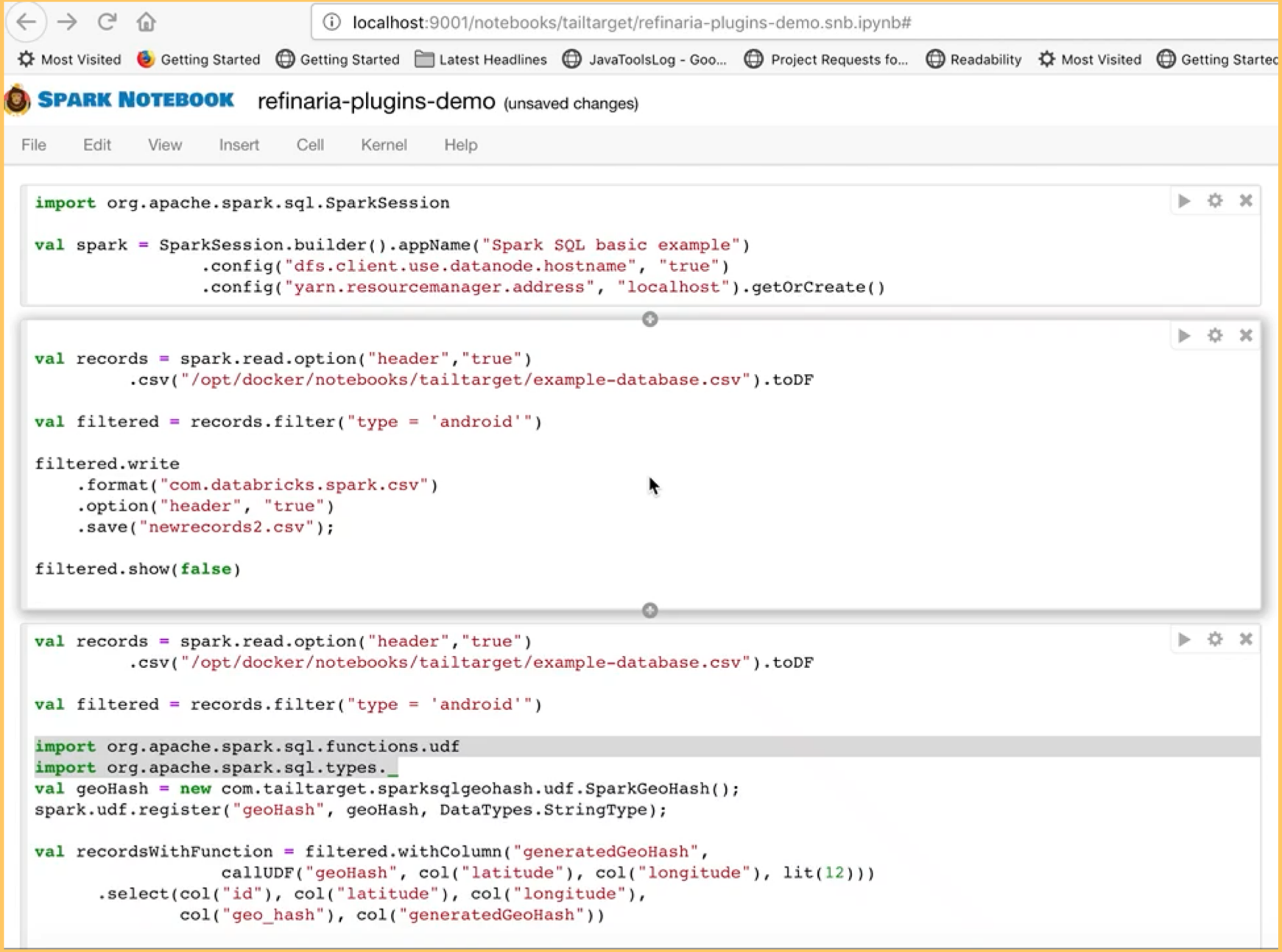
Experimentação

- Amostras
- Interativo



Spark Shell
Notebooks


docker — docker ◂ docker-compose exec spark-master spark-shell — 80×24
18/06/14 23:46:34 WARN General: Plugin (Bundle) "org.datanucleus.api.jdo" is already registered. Ensure you dont have multiple JAR versions of the same plugin in the classpath. The URL "file:/usr/local/spark-2.1.0-bin-hadoop2.7/jars/datanucleus-api-jdo-3.2.6.jar" is already registered, and you are trying to register an identical plugin located at URL "file:/usr/local/spark/jars/datanucleus-api-jdo-3.2.6.jar."
18/06/14 23:46:40 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
Spark context Web UI available at http://172.19.0.3:4040
Spark context available as 'sc' (master = local[*], app id = local-1529019991611).
Spark session available as 'spark'.
Welcome to
version 2.1.0
Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_171)
Type in expressions to have them evaluated.
Type :help for more information.
scala>


localhost:9001/notebooks/tailtarget/refinaria-plugins-demo.snb.ipynb#
Most Visited
Getting Started
Getting Started
Latest Headlines
JavaToolsLog - Goo...
Project Requests fo...
Readability
Most Visited
Getting Started
SPARK NOTEBOOK
refinaria-plugins-demo (unsaved changes)
File
Edit
View
Insert
Cell
Kernel
Help
import org.apache.spark.sql.SparkSession
val spark = SparkSession.builder().appName("Spark SQL basic example")
.config("dfs.client.use.datanode.hostname", "true")
.config("yarn.resourcemanager.address", "localhost").getOrCreate()
val records = spark.read.option("header","true")
.csv("/opt/docker/notebooks/tailtarget/example-database.csv").toDF
val filtered = records.filter("type = 'android'")
filtered.write
.format("com.databricks.spark.csv")
.option("header", "true")
.save("newrecords2.csv");
filtered.show(false)
val records = spark.read.option("header","true")
.csv("/opt/docker/notebooks/tailtarget/example-database.csv").toDF
val filtered = records.filter("type = 'android'")
import org.apache.spark.sql.functions.udf
import org.apache.spark.sql.types._
val geoHash = new com.tailtarget.sparksqlgeohash.udf.SparkGeoHash();
spark.udf.register("geoHash", geoHash, DataTypes.StringType);
val recordsWithFunction = filtered.withColumn("generatedGeoHash",
callUDF("geoHash", col("latitude"), col("longitude"), lit(12)))
.select(col("id"), col("latitude"), col("longitude"),
col("geo_hash"), col("generatedGeoHash"))

Produtividade

Amostras prontas para os datasets do Data Lake

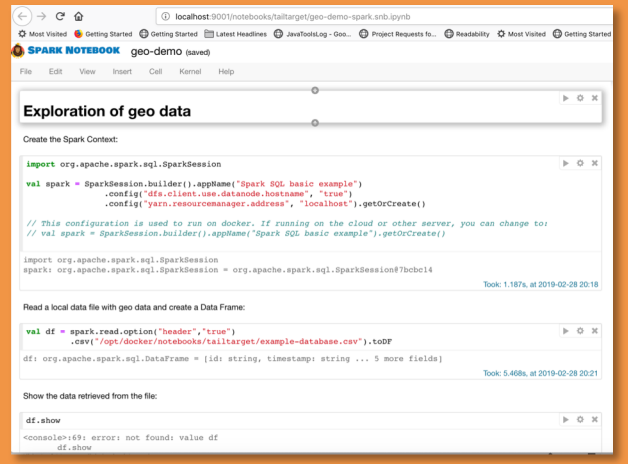
Bibliotecas unificadas para tarefas comuns (conversão de dados, filtros, lógica de negócio, etc)

Aproximar experimentação da produção

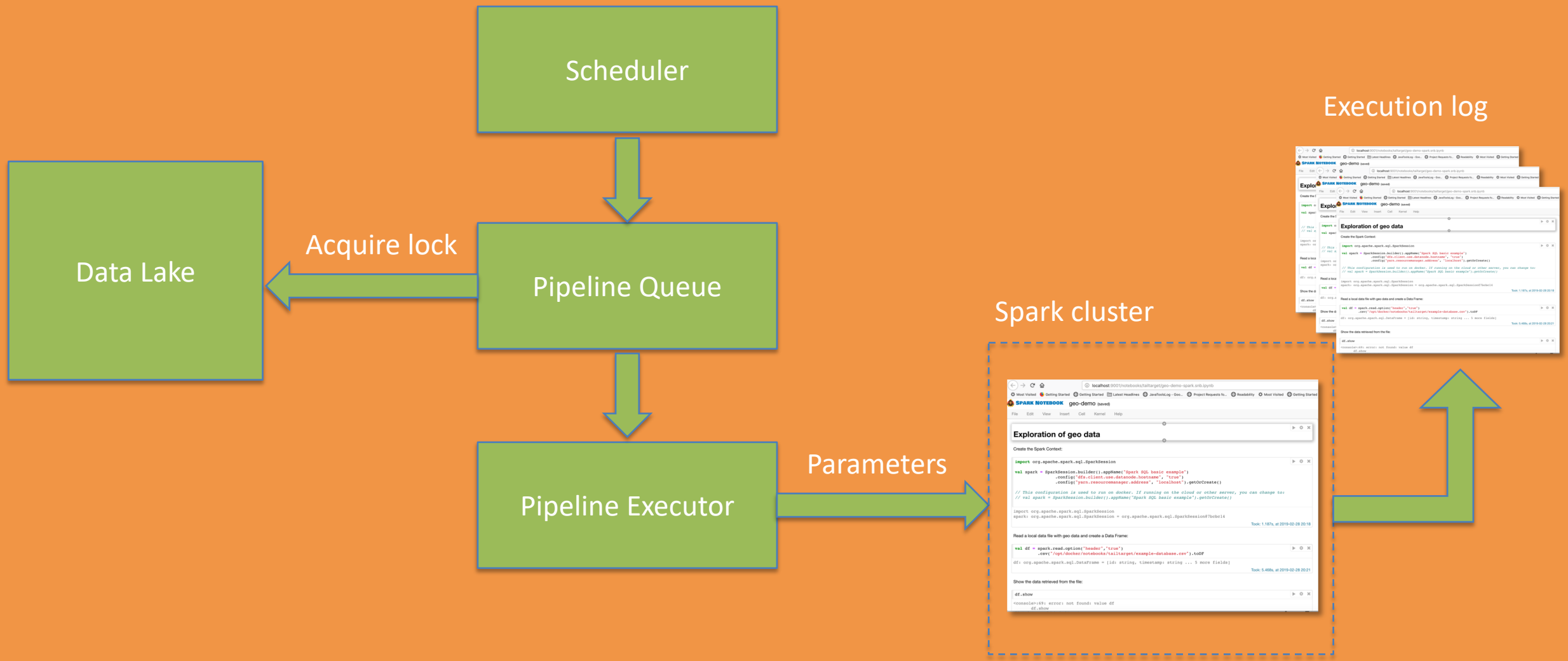
Experimentação

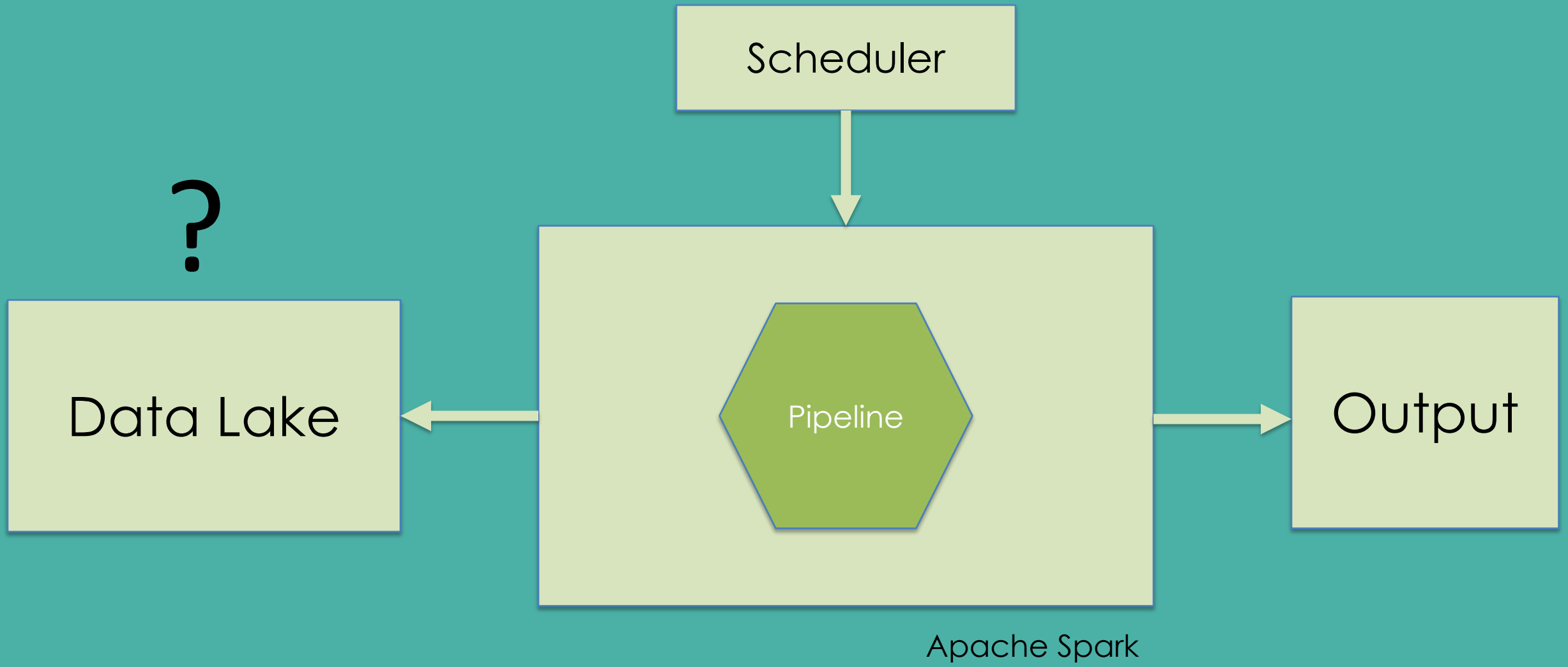
Spark cluster

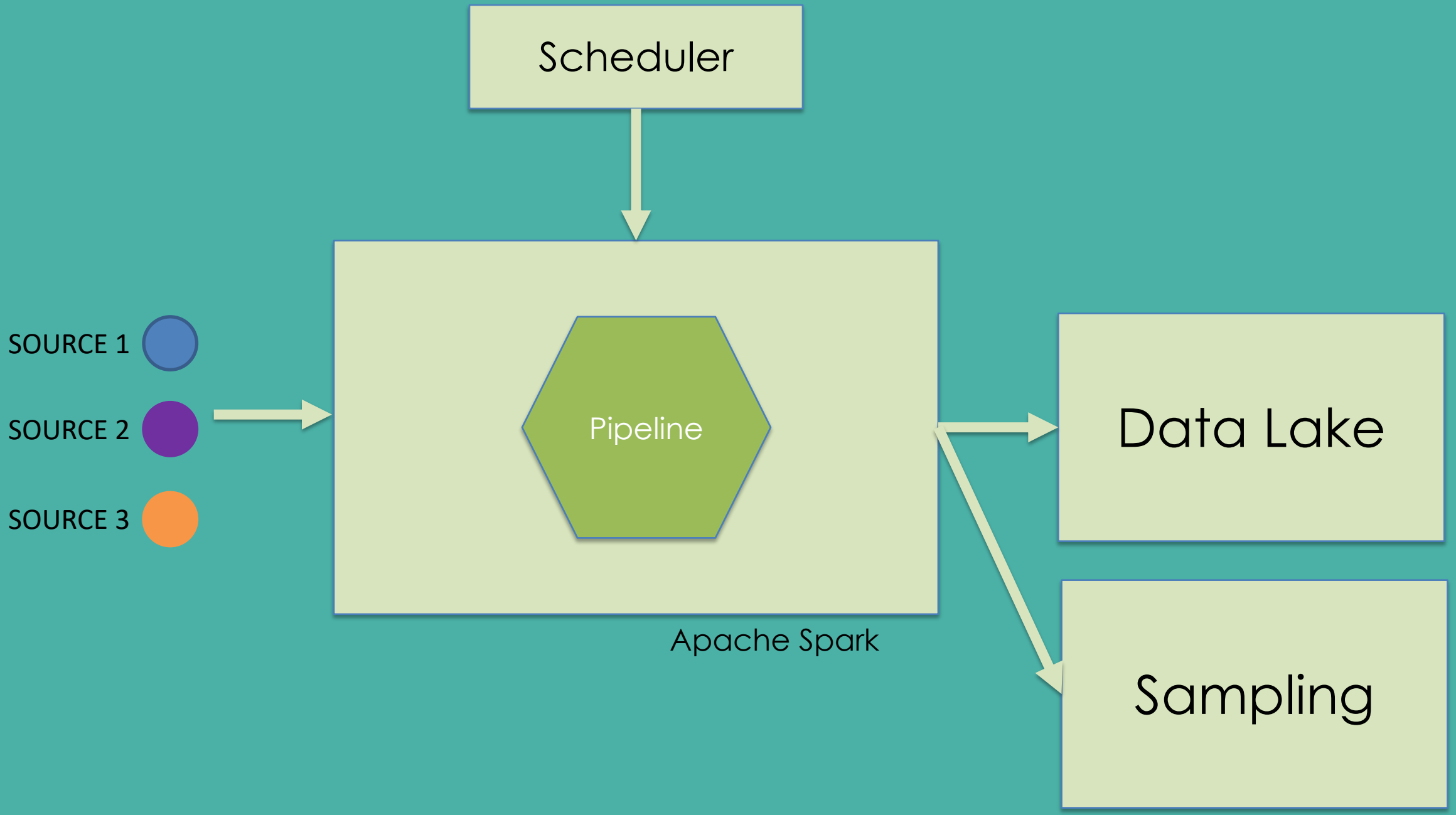
Data Lake
Sampling

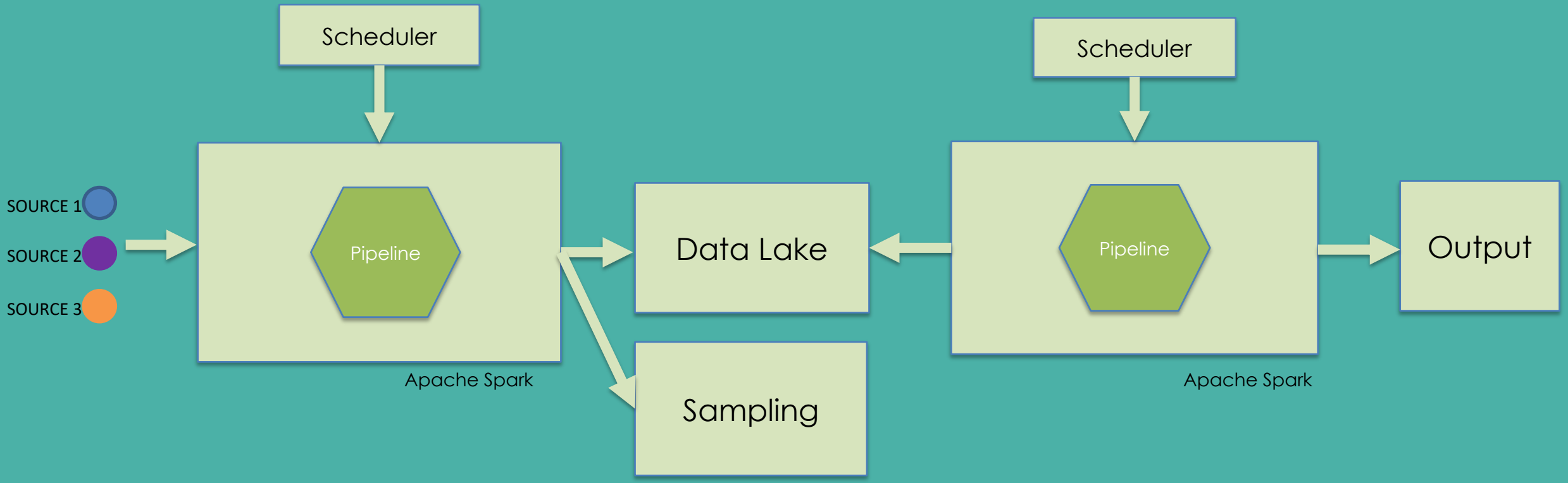


Produção


Scheduler
Data Lake
Acquire lock
Pipeline Queue
Pipeline Executor
Parameters
Spark cluster
Execution log


Scheduler
?
Data Lake
Pipeline
Output
Apache Spark


Scheduler
SOURCE 1
SOURCE 2
SOURCE 3
Pipeline
Apache Spark
Data Lake
Sampling


Scheduler
SOURCE 1
SOURCE 2
SOURCE 3
Pipeline
Apache Spark
Data Lake
Sampling
Scheduler
Pipeline
Apache Spark
Output

Code Lake

Plugins

Tipos de dados com semântica
Transformações
Agregações
Filtros

Tipos de dados com semântica

```
public class CpfSparkType implements SparkType<String, CpfType> {

    @Override
    public String serialize(CpfType dataType) {
        return dataType.value();
    }

    @Override
    public CpfType deserialize(String sparkData) {
        return new CpfType(sparkData);
    }

    @Override
    public DataType sqlType() {
        return DataTypes.StringType;
    }
}
```

Tipos de dados com semântica

```
public class CpfSparkType implements SparkType<String, CpfType> {

    @Override
    public String serialize(CpfType dataType) {
        return da
    }

    @Override
    public CpfTyp
        return ne
    }

    @Override
    public DataTy
        return Da
    }
}
```

```
val schema = StructType(Array(
    StructField("id",StringType,true),
    StructField("cpf", new CpfSparkType().sqlType(),true)
    )
)

val records = spark.read.schema(schema)
                        .option("header","true")
                        .csv("cpfs.csv")
```

Transformações

```
public class SparkGeoHash
      implements UDF3<Double, Double, Integer, String> {

    public String call(Double latitude, Double longitude,
                Integer numberOfCharacters){

        return GeoHash.withCharacterPrecision(latitude,
              longitude, numberOfCharacters).toBase32();
    }
}
```

Transformações

```
public class SparkGeoHash
      implements

    public String
                I

        return Ge
               lon
    }
}
```

```
val geoHash = new SparkGeoHash();
spark.udf.register("geoHash", geoHash, DataTypes.StringType);

val newRecords = dataFrame.withColumn("generatedGeoHash",
                   callUDF("geoHash", col("latitude"),
                           col("longitude"), lit(12)))
                 .select(col("id"), col("latitude"),
                   col("longitude"),
                   col("geo_hash"), col("generatedGeoHash"))
```

Agregações

```
public class NewestRecord extends UserDefinedAggregateFunction {

    …

}
```

Agregações

```
public class NewestReco

    …

}
```

```
val newestRecord = new NewestRecord()

dataFrame.groupBy(”id").agg(
            newestRecord(dataFrame.col(”timestamp")).as(”last_record")
).show()
```

Data Science

Construção de Data Pipelines em Apache Spark

mecontrata@tailtarget.com

Fabiane Bizinella Nardon
Chief Data Scientist
@fabianenardon