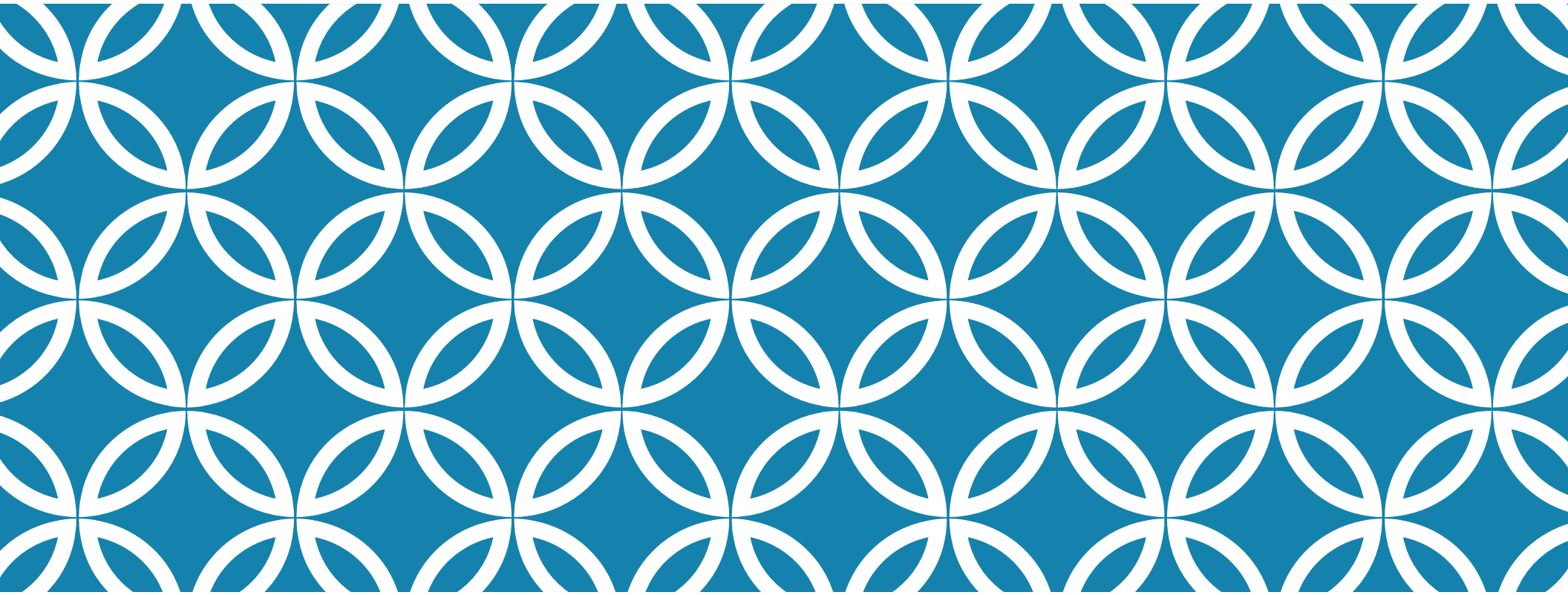


INTRODUCTION TO SPARK

WHY DO WE NEED A NEW PLATFORM?

Data analytic require the combination of distinct kinds of processing

- MapReduce-like code for data loading
- SQL-like queries
- Iterative machine learning

Ideally, they could be combined to build an application but

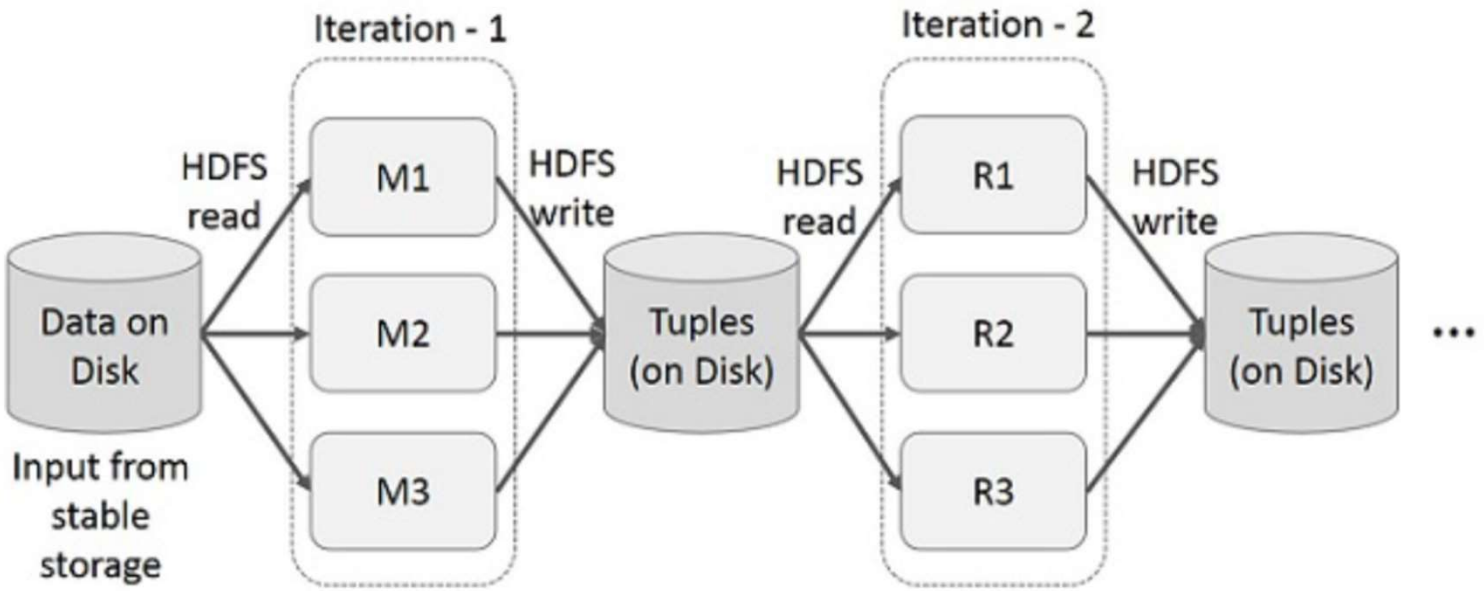
- Distinct kinds of processing impose distinct requirements on the computation engine

WHAT ABOUT HADOOP?

Hadoop offers a platform for distributed computing based on a simple programming model (MapReduce), which is scalable, flexible, and fault-tolerant

But…

HADOOP DOES DISK ORIENTED PROCESSING



https://www.tutorialspoint.com/apache_spark/apache_spark_rdd.htm

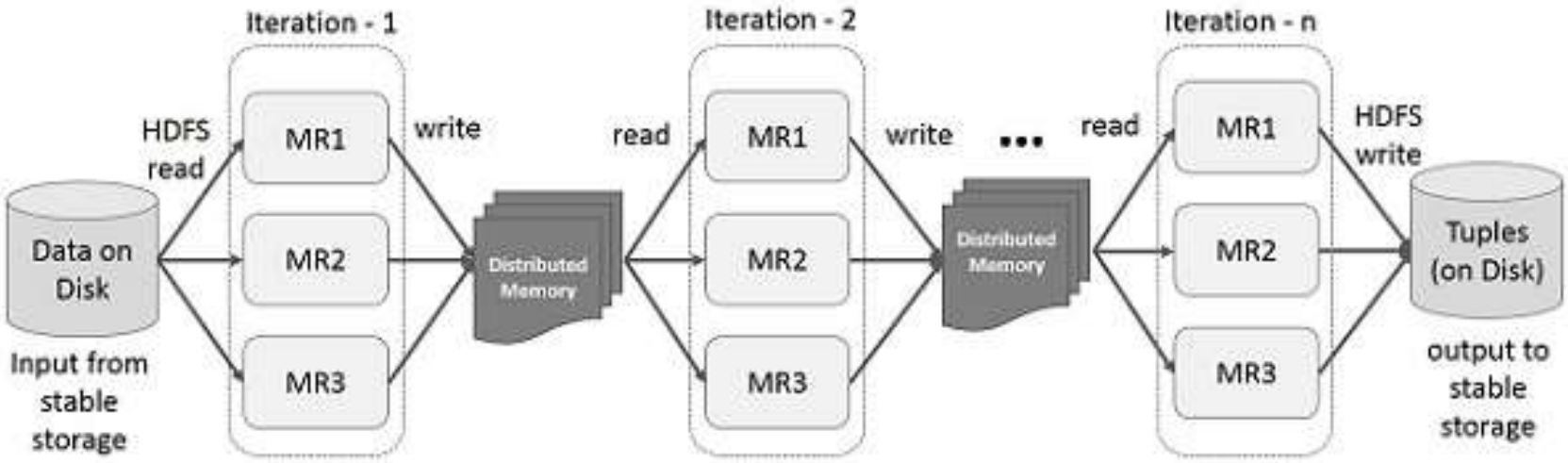
APACHE SPARK COMBINES

Computing engine for distributed computing with a simple programming model based on ***in-memory*** processing

Libraries

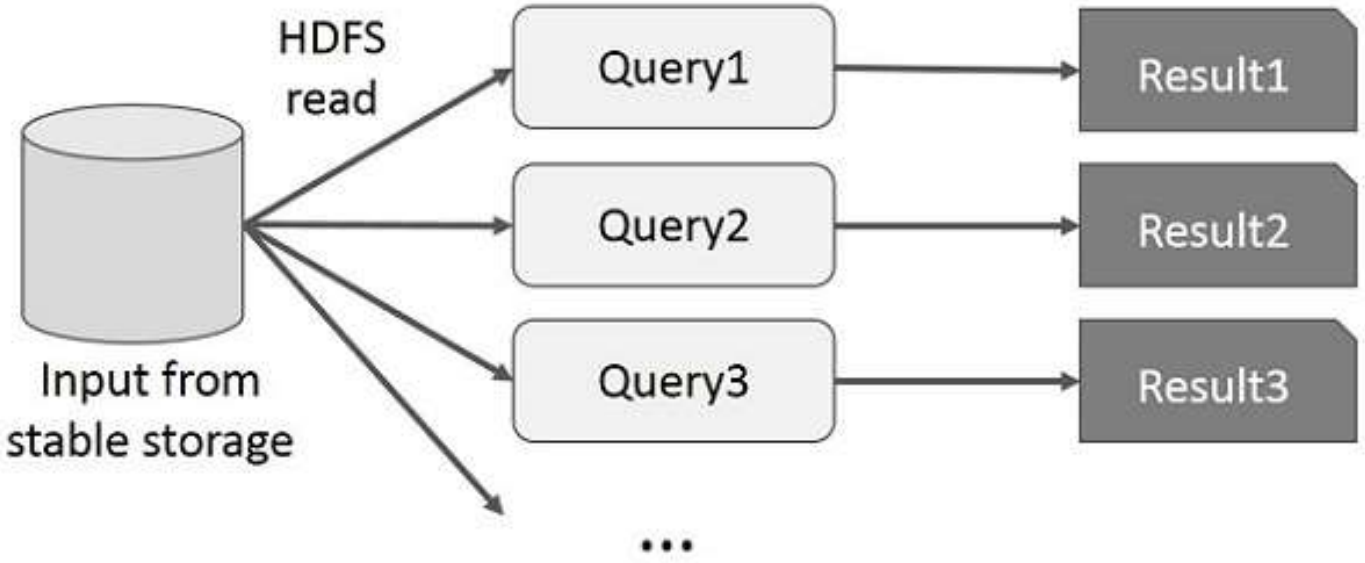
- SQL and structured data (Spark SQL)
- machine learning (MLlib)
- stream processing (Spark Streaming and the newer Structured Streaming)
- graph analytics (GraphX)

SPARK COMPUTATIONS



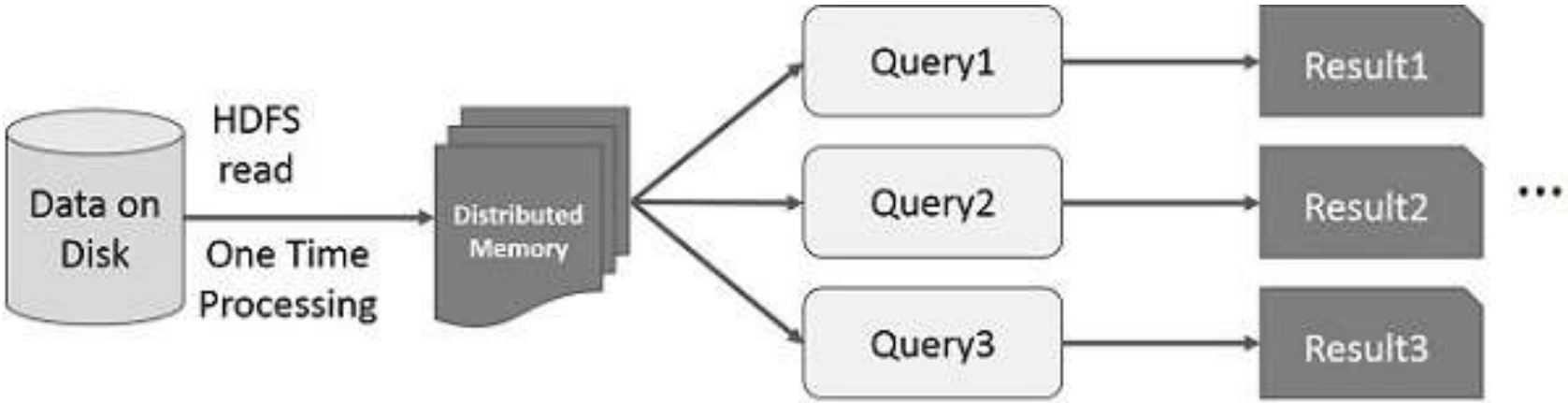
https://www.tutorialspoint.com/apache_spark/apache_spark_rdd.htm

ITERATIVE QUERIES IN HADOOP



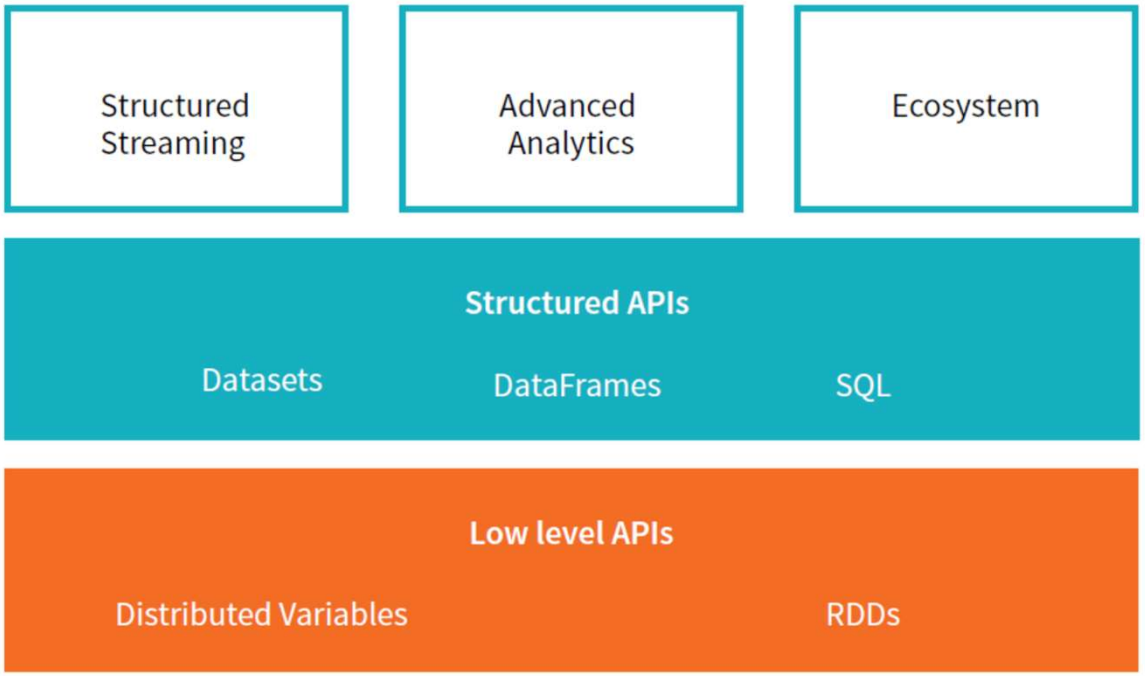
https://www.tutorialspoint.com/apache_spark/apache_spark_rdd.htm

ITERATIVE QUERIES IN SPARK



https://www.tutorialspoint.com/apache_spark/apache_spark_rdd.htm

WHAT IS SPARK?



Source: A Gentle Introduction to Apache Spark

MOREOVER, SPARK SUPPORTS VARIOUS…

Development languages

- Java, Scala, Python, and R

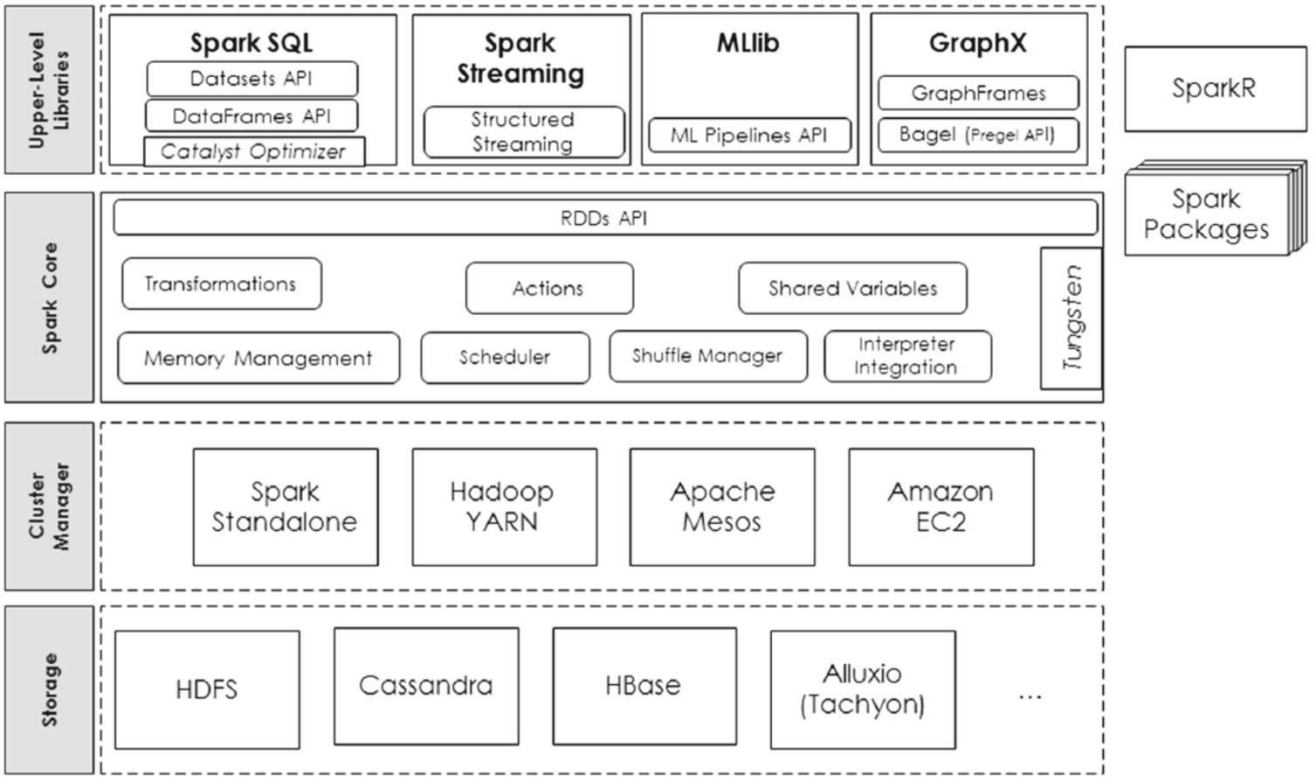
Cluster managers

- Hadoop Yarn, Mesos, Kubernets, or even stand alone

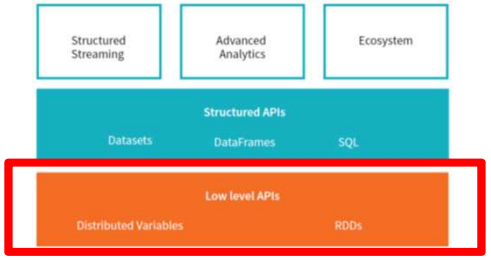
Various datasources

- files (e.g. HDFS, csv, parquet, avro)
- databases (e.g. JDBC, Cassandra)

A MORE FAIR REPRESENTATION



Source: Big data analytics on Apache Spark



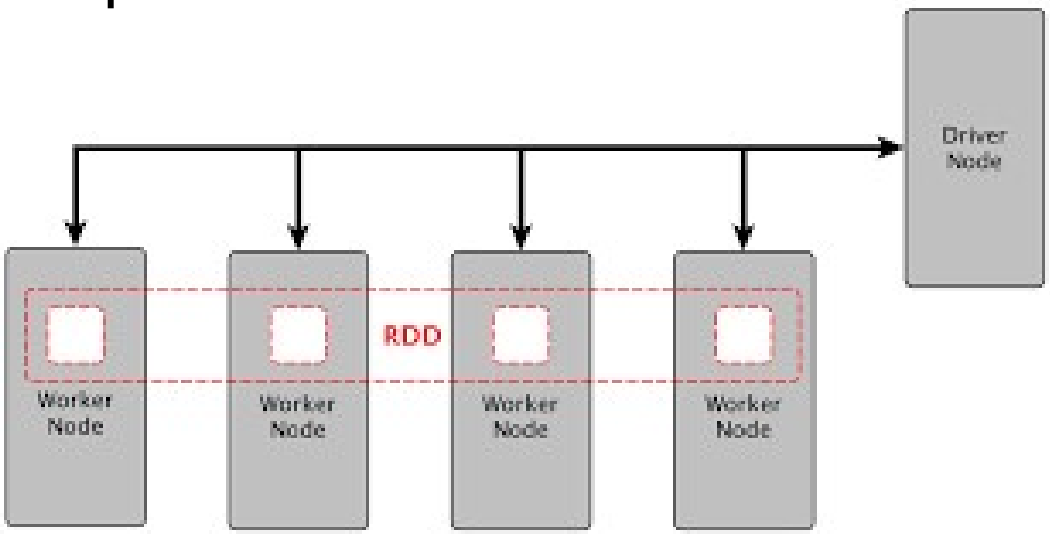
LOW LEVEL APIS

CORE

Programming model similar to MapReduce, but based on a new data abstraction called ***Resilient Distributed Dataset (RDD)***

RESILIENT DISTRIBUTED DATASET (RDD)

Immutable, partitioned collection of data that can be manipulated in parallel



https://medium.com/@lavishj77/spark-fundamentals-part-2-a2d1a78eff73

SPARK PROGRAMS

1. Create **RDDs**
2. Apply ***transformations*** on those RDDs
3. "*Persist*" intermadiate RDDs for reuse
4. Call ***actions*** to *efectivelly* launch parallel computations and get the results back to the program

CREATE RDD

Loding data from an external datasource

Parallelizing a collection of data

Applying *transformations* over an existing RDD

CREATE RDD

```
# import required libraries
from pyspark import SparkConf, SparkContext

# build a configuration
conf = SparkConf().setMaster("local").setAppName("FirstSparkApp")

# create a SparkContext using the configuration
sparkContext = SparkContext(conf = conf)

# create an RDD from file content
fileRdd = sparkContext.textFile("u.data")

# create an RDD from a list of ints
numbersRdd = sparkContext.parallelize(range(1,1000))
```

APPLY TRANSFORMATIONS

```
# applying a map that squares each value in the rdd
# and filtering the results to just consider those that are divisible by 4
squaredRdd = numbersRdd.map(lambda v : v ** 2)
filteredRdd = squaredRdd.filter(lambda a : a % 4 == 0)
```

APPLY TRANSFORMATIONS

```
# u.data is composed by lines
# following this structure:
# userid movieId rate    timestamp
# 196    242     3       881250949

# now, ratingsRdd contains just the ratings
ratingsRdd = fileRdd.map(lambda l : l.split()[2])

# and fiveStarsRdd, just '5's
fiveStarsRdd = ratingsRdd.filter(lambda r : r == '5')
```

APPLY TRANSFORMATIONS

```
logRDD = sc.textFile("log.txt")
errorsRDD = logRDD.filter(lambda x : "error" in x)
warningsRDD = logRDD.filter(lambda x : "warning" in x)
alertsRDD = errorsRDD.union(warningsRDD)
```

There are many others…

Transformation	Description
map(func)	Return a new distributed dataset formed by passing each element of the source through a function func.
flatMap(*func*)	Similar to map, but each input item can be mapped to 0 or more output items (so func should return a Seq rather than a single item).
filter(func)	Return a new dataset formed by selecting those elements of the source on which func returns true.
sample(withReplacement, fraction, seed)	Sample a fraction fraction of the data, with or without replacement, using a given random number generator seed.
union(otherDataset)	Return a new dataset that contains the union of the elements in the source dataset and the argument.
intersection(otherDataset)	Return a new RDD that contains the intersection of elements in the source dataset and the argument.
distinct([numPartitions]))	Return a new dataset that contains the distinct elements of the source dataset.
join(otherDataset, [numPartitions])	When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key. Outer joins are supported through leftOuterJoin, rightOuterJoin, and fullOuterJoin.

https://spark.apache.org/docs/latest/rdd-programming-guide.html#transformations

Transformation	Description
groupByKey([*numPartitions*])	When called on a dataset of (K, V) pairs, returns a dataset of (K, Iterable<V>) pairs.
reduceByKey(*func*, [*numPartitions*])	When called on a dataset of (K, V) pairs, returns a dataset of (K, V) pairs where the values for each key are aggregated using the given reduce function *func*, which must be of type (V,V) => V.
aggregateByKey(*zeroValue*)(*seqOp*, *combOp*, [*numPartitions*])	When called on a dataset of (K, V) pairs, returns a dataset of (K, U) pairs where the values for each key are aggregated using the given combine functions and a neutral "zero" value. Allows an aggregated value type that is different than the input value type
sortByKey([*ascending*], [*numPartitions*])	When called on a dataset of (K, V) pairs where K implements Ordered, returns a dataset of (K, V) pairs sorted by keys in ascending or descending order, as specified in the boolean ascending argument.

https://spark.apache.org/docs/latest/rdd-programming-guide.html#transformations

PERSIST/CACHE RDDS

```
# u.data is composed by lines
# following this structure:
# userid movieId rate    timestamp
# 196    242     3       881250949

# now, ratingsRdd contains just the ratings
ratingsRdd = fileRdd.map(lambda l : l.split()[2]).cache()

# and fiveStarsRdd, just '5's
fiveStarsRdd = ratingsRdd.filter(lambda r : r == '5')
print(fiveStarsRdd.count())

# and fourStarsRdd, just '4's
fourStarsRdd = ratingsRdd.filter(lambda r : r == '4')
print(fourStarsRdd.count())

fourStarsRdd.union(fiveStarsRdd).count()
```

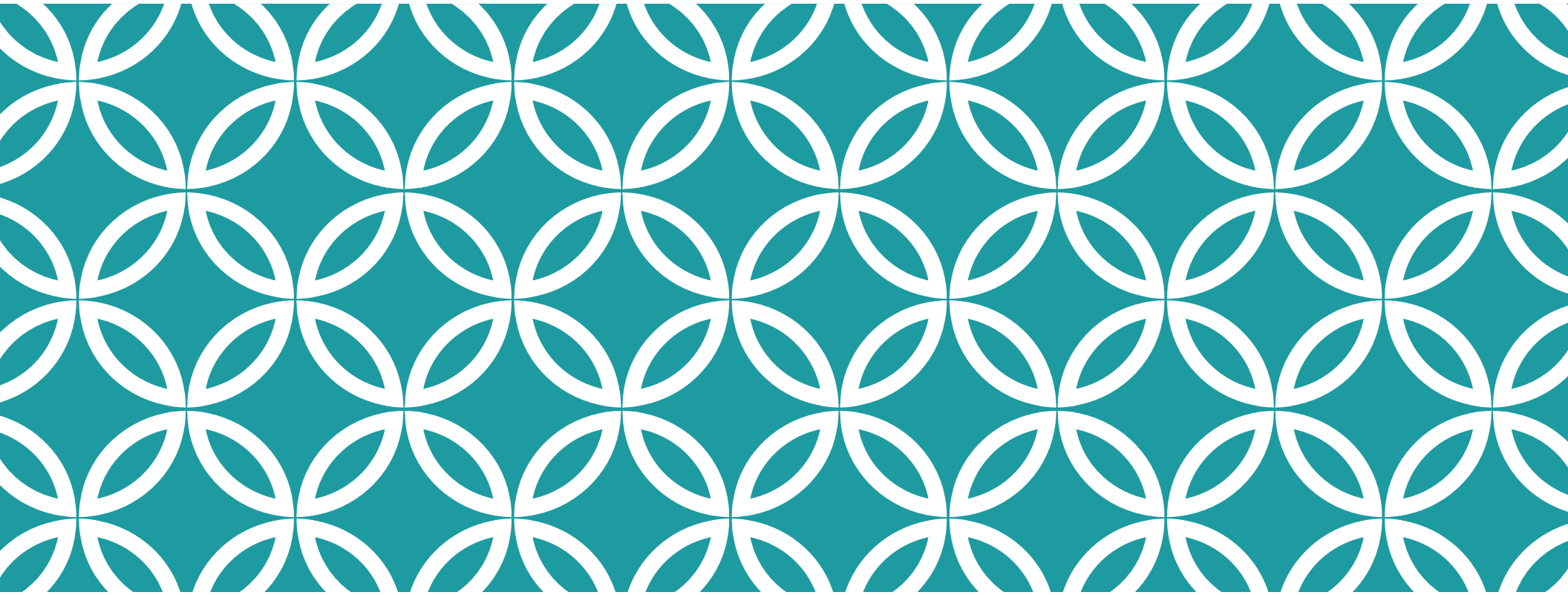
CALL ACTIONS TO GET THE RESULTS

```
# get the first element of an RDD
first = fileRdd.first()
print(first)

# get 5 elements
top5 = fileRdd.take(5)
print(top5)

# get the number elements
print(fileRdd.count())

# Be carefull, collect will bring all RDD's elements
# to memory at the driver program.
for idx, alert in enumerate(alertsRDD.collect()):
    print("Line %i is: %s" % (idx, alert))
```

CALL ACTIONS TO GET THE RESULTS

```
data = [1, 2, 3, 4, 5]
distData = sparkContext.parallelize(data)
distData.reduce(lambda a, b: a + b)
```

```
15
```



SPARK ARCHITECTURE

SPARK'S ARCHITECTURE

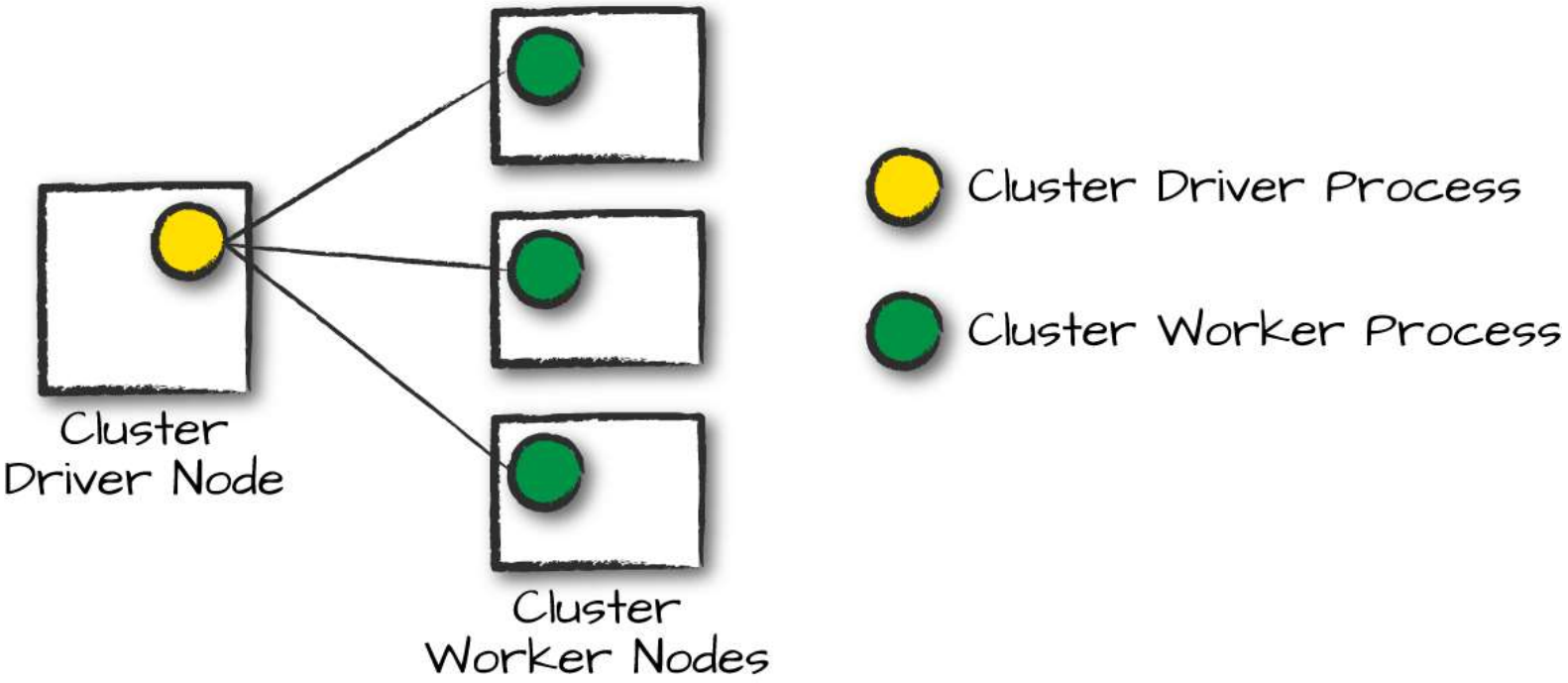
Driver process

- Maintain information about the application
- Distribute and schedule work (tasks) across executors

Executor processes

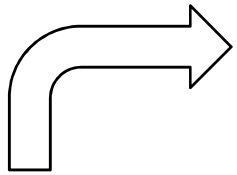
- Carry out the work received from the driver
- Report the state of the execution back to the driver

AND WHAT ABOUT THE CLUSTER?

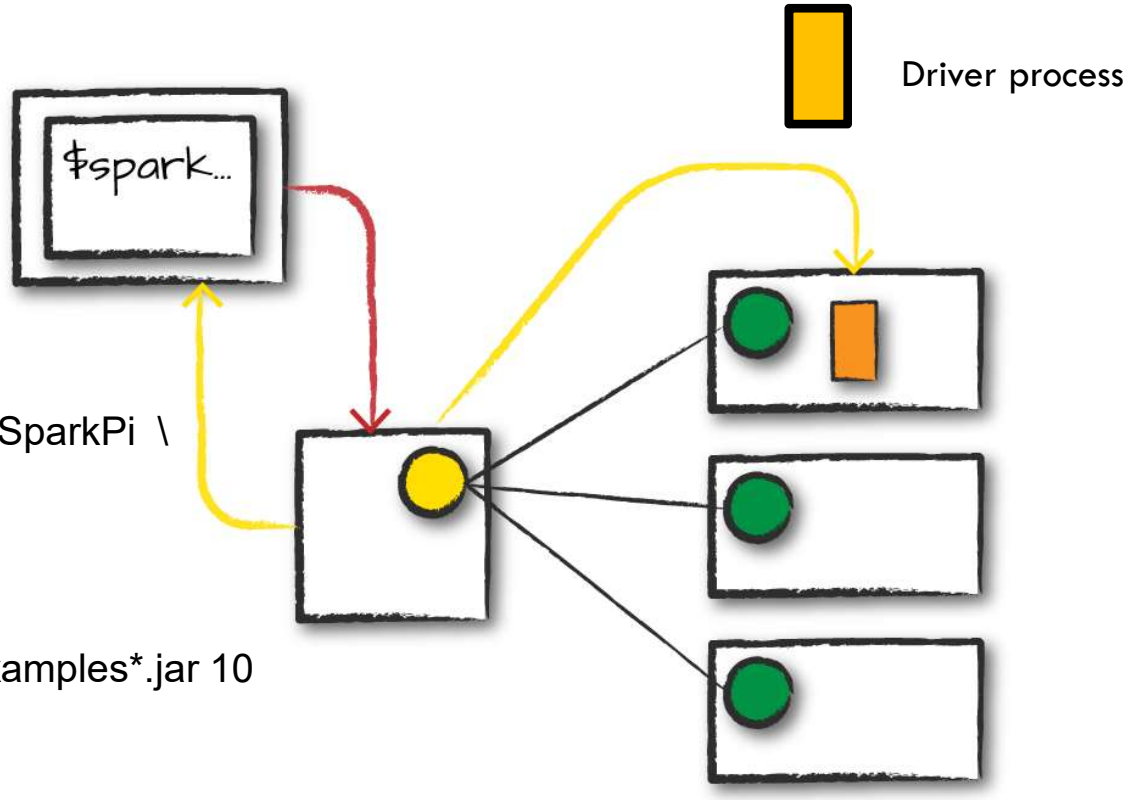


From: Spark: The Definitive Guide

CLIENT REQUEST

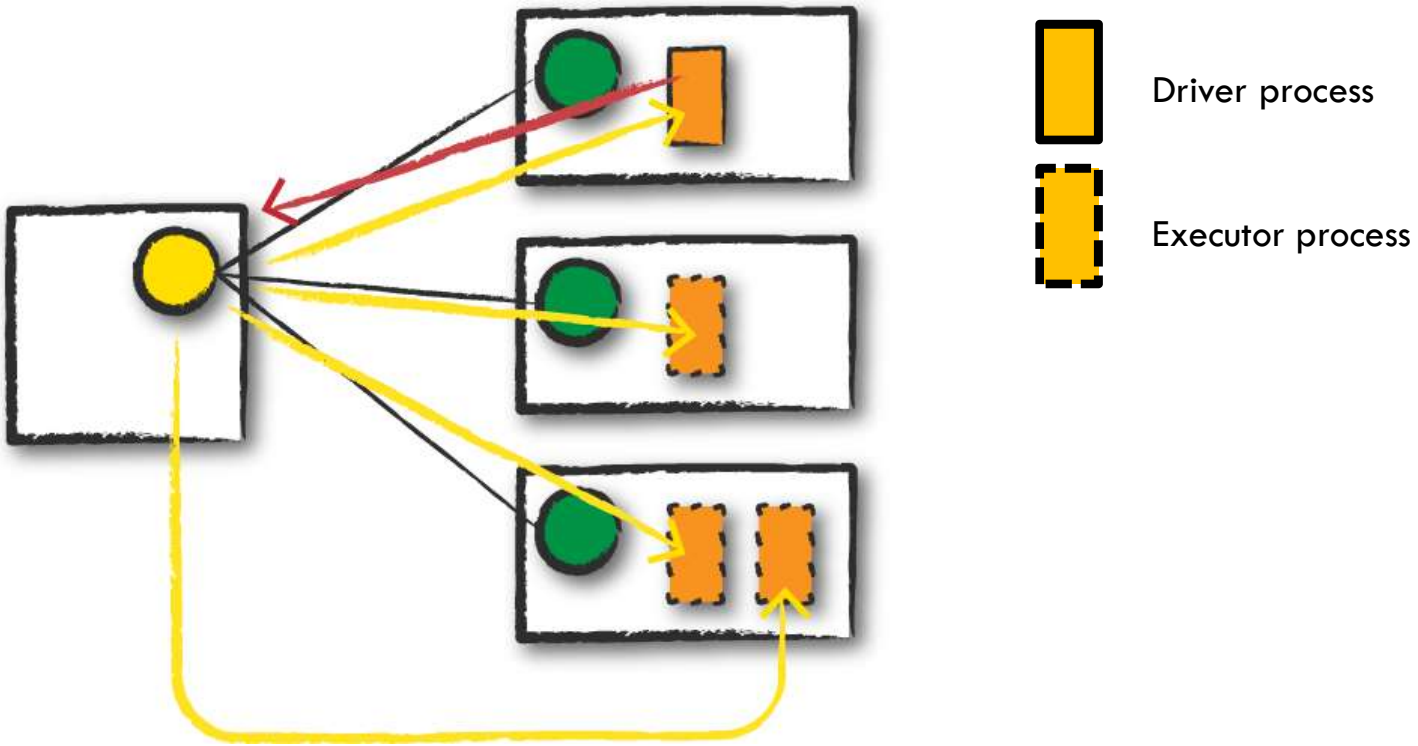


```
./bin/spark-submit \
      --class org.apache.spark.examples.JavaSparkPi  \
      --master yarn-cluster \
      --num-executors 3 \
      --driver-memory 10g \
       --executor-memory 10g \
      $SPARK_HOME/examples/jars/spark-examples*.jar 10
```



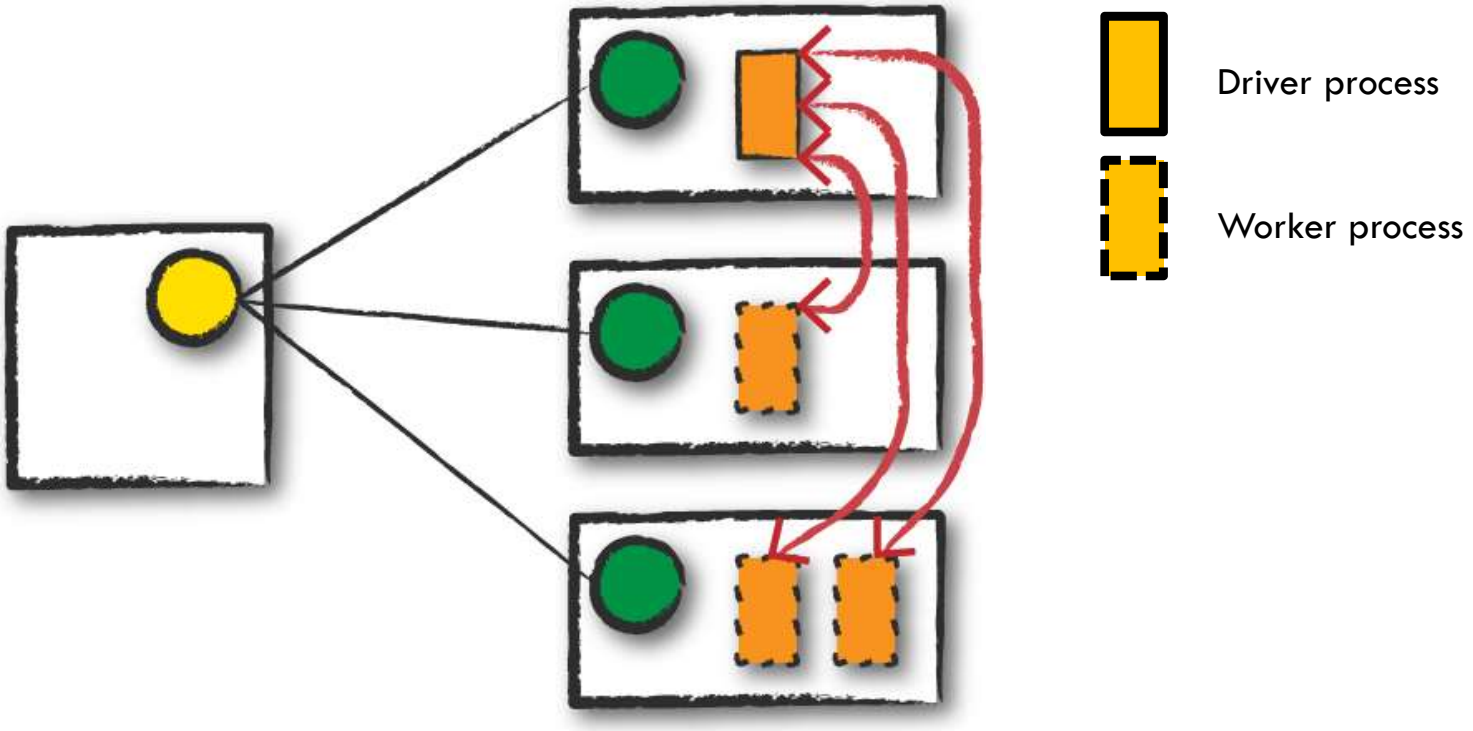
From:Spark: The Definitive Guide

LAUNCH



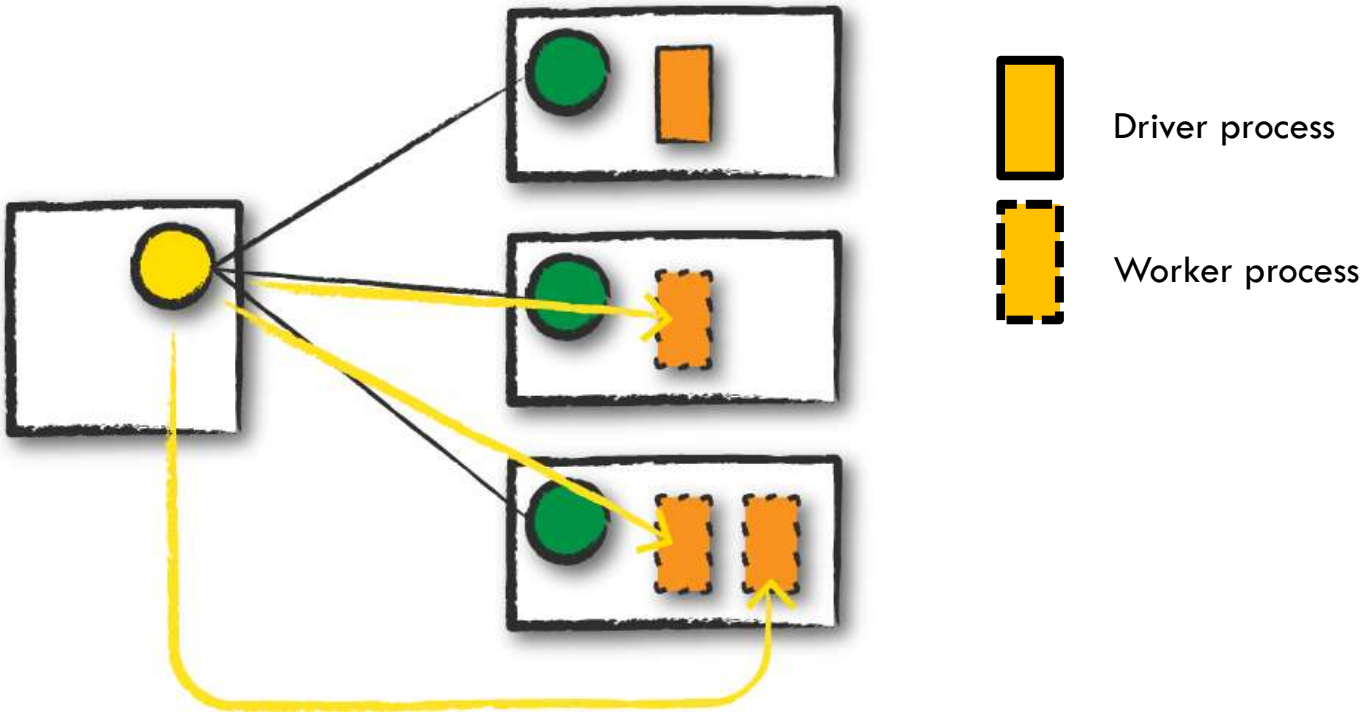
From: Spark: The Definitive Guide

EXECUTION

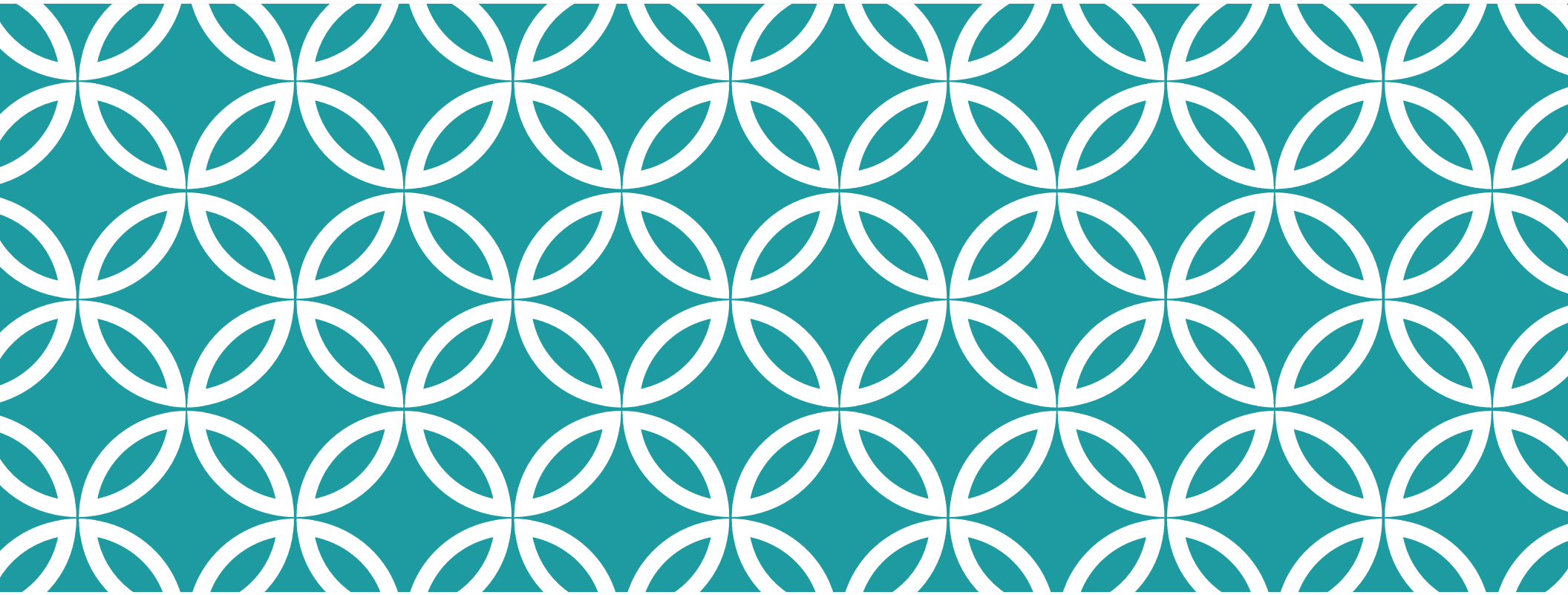


From: Spark: The Definitive Guide

COMPLETION



From: Spark: The Definitive Guide



ENVIRONMENT

APPLICATIONS

Development

- IDEs (e.g. VSCode, Canopy)

Execution

- *spark-submit* script

INTERACTIVE EXPLORATION

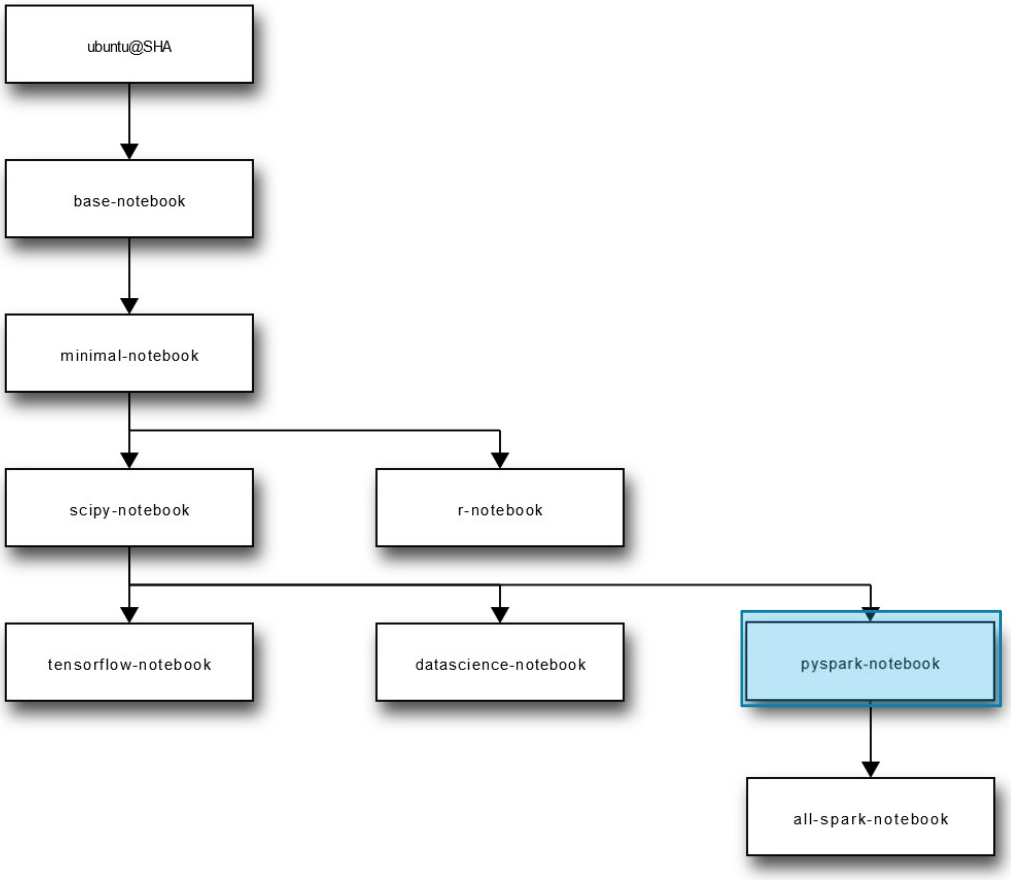
Consoles

- pyspark, spark-shell, spark-sql

Notebooks

- Local
  - Docker + Jupyter Docker Stacks
- Remote
  - Databricks Community Edition

JUPYTER DOCKER STACKS



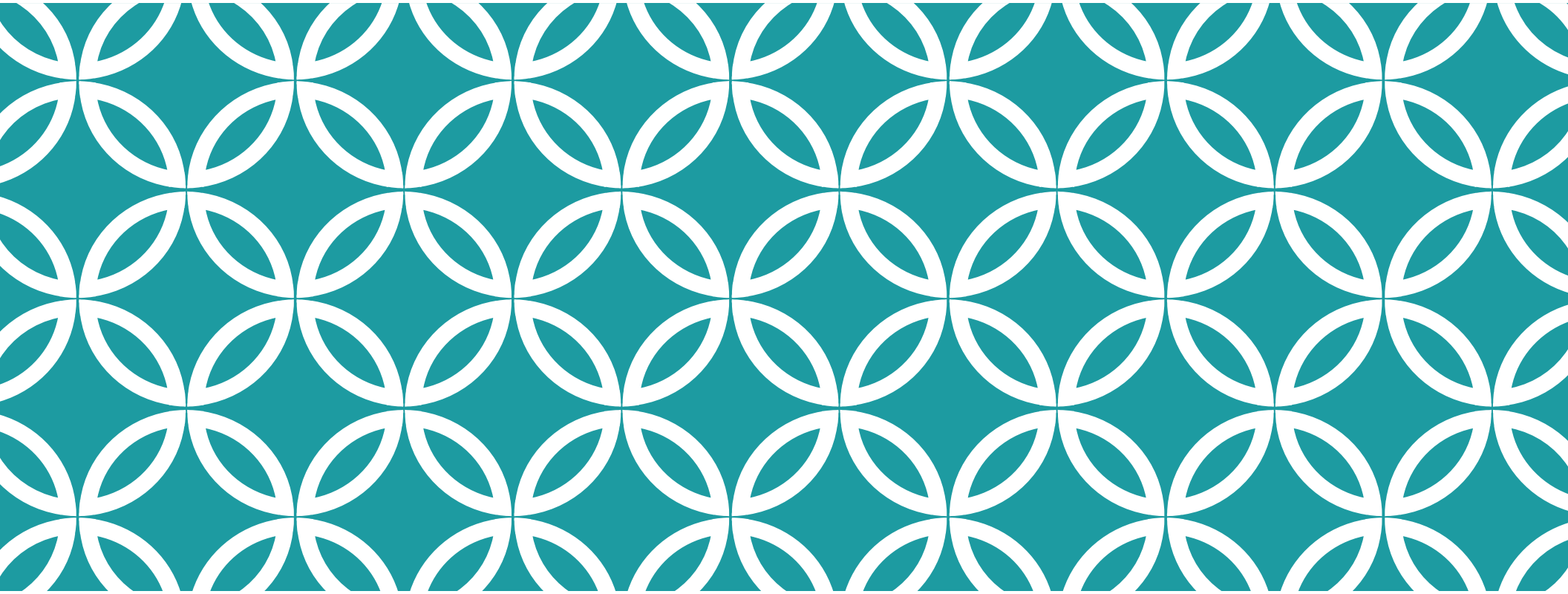
JUPYTER DOCKER STACKS

Which Docker image do you wish to use?

- https://jupyter-docker-stacks.readthedocs.io/en/latest/using/selecting.html

How can you run containers from that image?

- https://jupyter-docker-stacks.readthedocs.io/en/latest/using/running.html
- PowerShell
  - $ docker run -p 8888:8888 -p 4040:4040 -v ${PWD}:/home/jovyan/work --name jupyter **jupyter/pyspark-notebook:**1386e2046833



DEMO

Hello, Spark!

SCENARIO 01: PYTHON'S INTERACTIVE CONSOLE

1. Go to the spark's installation directory
2. Start a Power Shell and execute:
   - $.\bin\pyspark
3. Create a RDD with contentes of the README.md file
   - >>> fileRdd = sc.textFile("README.md")
4. Count the number of lines in this file
   - >>> fileRdd.count()

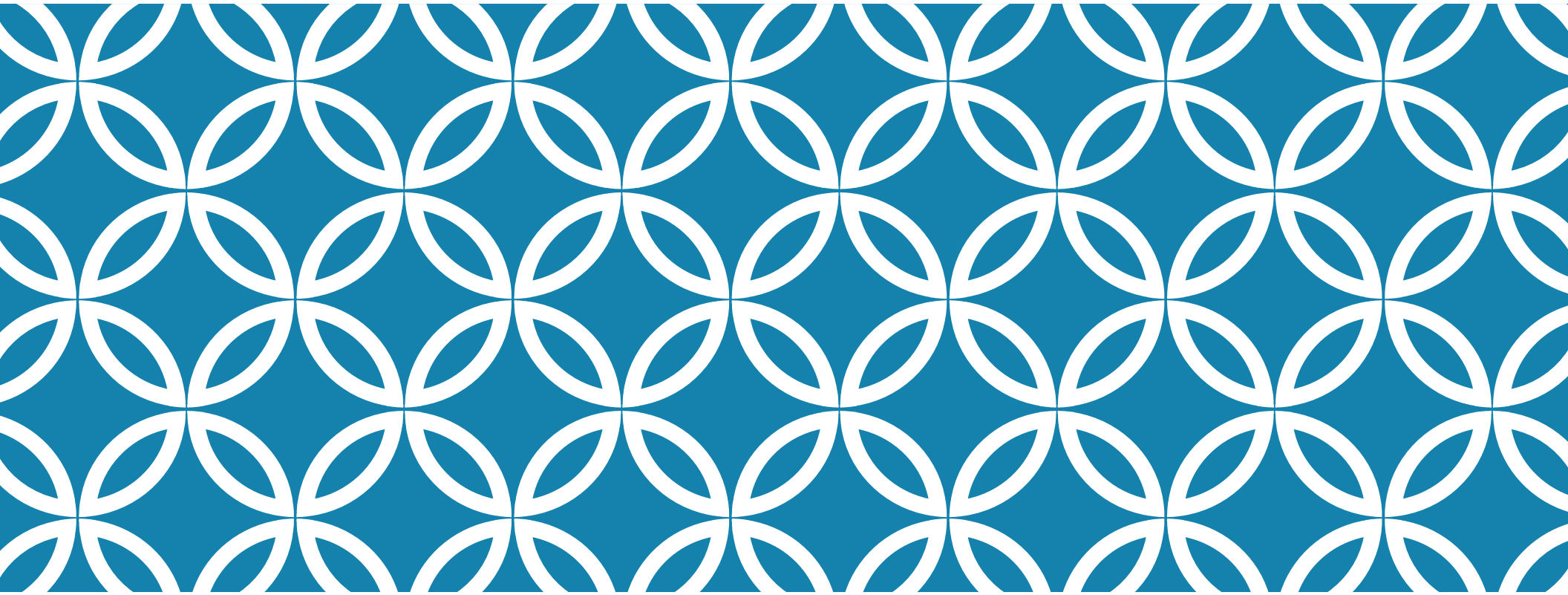
SCENARIO 02: PYTHON'S INTERACTIVE CONSOLE

1. Go to the spark's installation directory
2. Take a look at examples\src\main\python\pi.py
3. Run pi.py example using spark-submit command
   - spark-submit --master local[*] examples\src\main\python\pi.py 20

SCENARIO 03: JUPYTER NOTEBOOK

1. Verify docker configuration to confirm that hard drives are visible inside the container
2. Create a directory to contain your projects and inside it a directory called example01
3. Move to that directory and copy a simple text file there
4. Open powershell and start a container running jupyter notebook

```
$ docker run -p 8888:8888 -p 4040:4040 -v
${PWD}:/home/jovyan/work --name jupyter jupyter/pyspark-
notebook:1386e2046833
```



QUESTIONS???