ALPEN-ADRIA UNIVERSITÄT KLAGENFURT

Apache Spark

Current Topics in Distributed Systems: Internet of Things and Cloud Computing

Narges Mehran, MSc.

SS2021

02.06.2021

Programming models in a Cloud environment

The growth of data volumes in industry and research poses tremendous opportunities, as well as tremendous computational challenges.

There are several programming models which focus on supporting batch or real-time data processing efficiently in a Cloud environment. For instance,

- MapReduce has become the de facto standard for batch data processing in the Apache Hadoop framework.
- Apache Spark is a distributed batch processing framework, but it also supports stream processing based on micro-batching.
- Apache Storm supports event-based stream processing.
- Apache Flink enables batch and stream processing with its unified APIs.

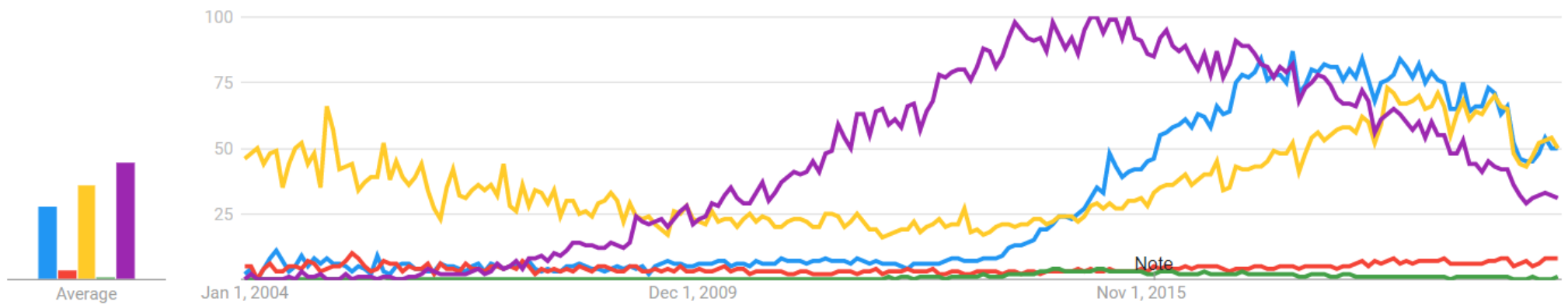
trends.google.com/trends/explore?date=all&q=%2Fm%2F0ndhxqz,%2Fg%2F11btzy2gtf,%2Fm%2F0zmynvd,%2Fm%2F0r4s2l1,%2Fm%2F0fdjtq

Google Trends | Compare

- Apache Spark — Software
- Apache Flink — Software
- Apache Kafka — Software
- Apache Storm — Computer program
- Apache Hadoop — Software

Worldwide | 2004 - present | All categories | Web Search

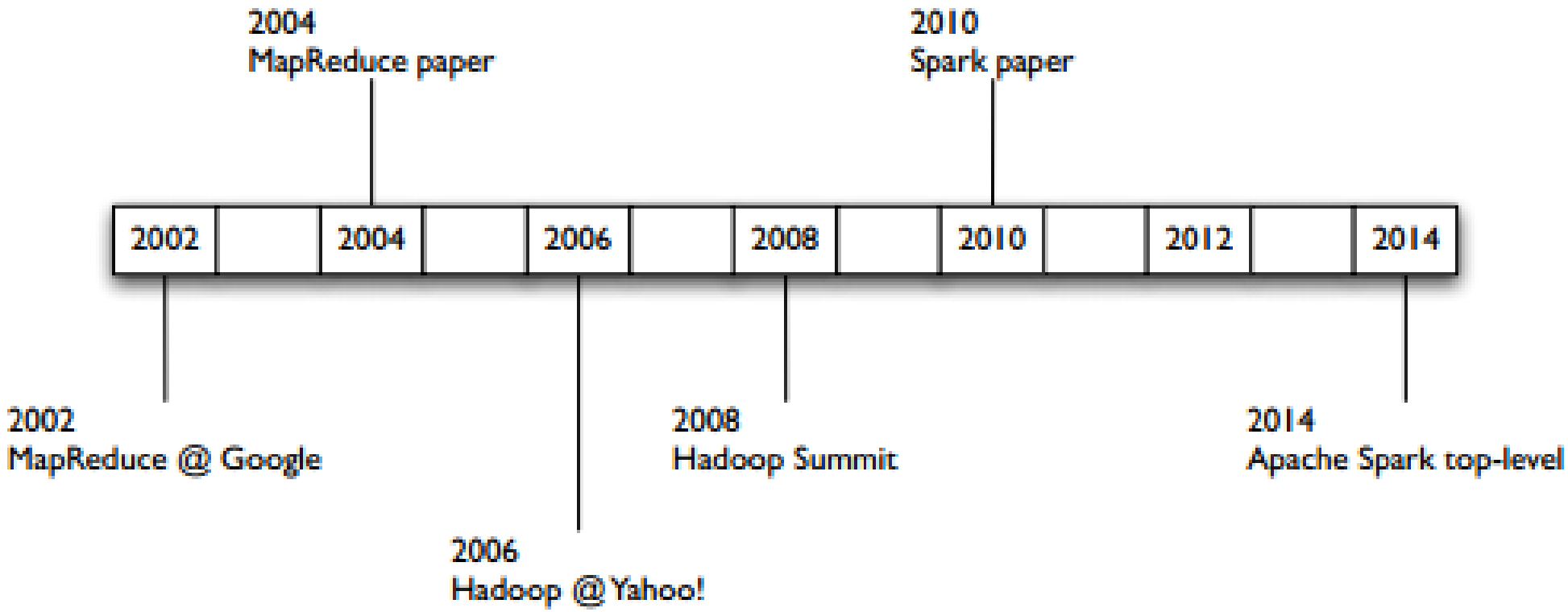
Interest over time



History

- Google invented a new methodology of processing data known as MapReduce.
- Doug Cutting and Mike Cafarella co-founded Hadoop to apply MapReduce concepts to an open-source software framework that supported the Nutch search engine project.
- Spark can process real-time data, such as real-time event streaming with a rate of millions of events/second, e.g., the live data streaming of Twitter, Facebook, Instagram.
  - But MapReduce fails as it cannot handle real-time data processing.
  - It can just perform batch processing on huge volumes of data.

Apache Platforms History



Introduction

- Apache Spark developed in 2009 at UC Berkeley AMPLab, then open-sourced in 2010, to be a general-purpose distributed computing system for big data analytics.
- Spark can complete jobs substantially faster than previous big data tools (i.e., Apache Hadoop) because of its in-memory caching, and optimized query execution.
- Spark provides development APIs in Python, Java, Scala, and R.
- On top of the main computing framework, Spark provides machine learning, SQL, graph analysis, and streaming libraries.
- Dealing with massive amounts of data often requires parallelization and cluster computing;
  - Apache Spark is an industry standard for doing this.

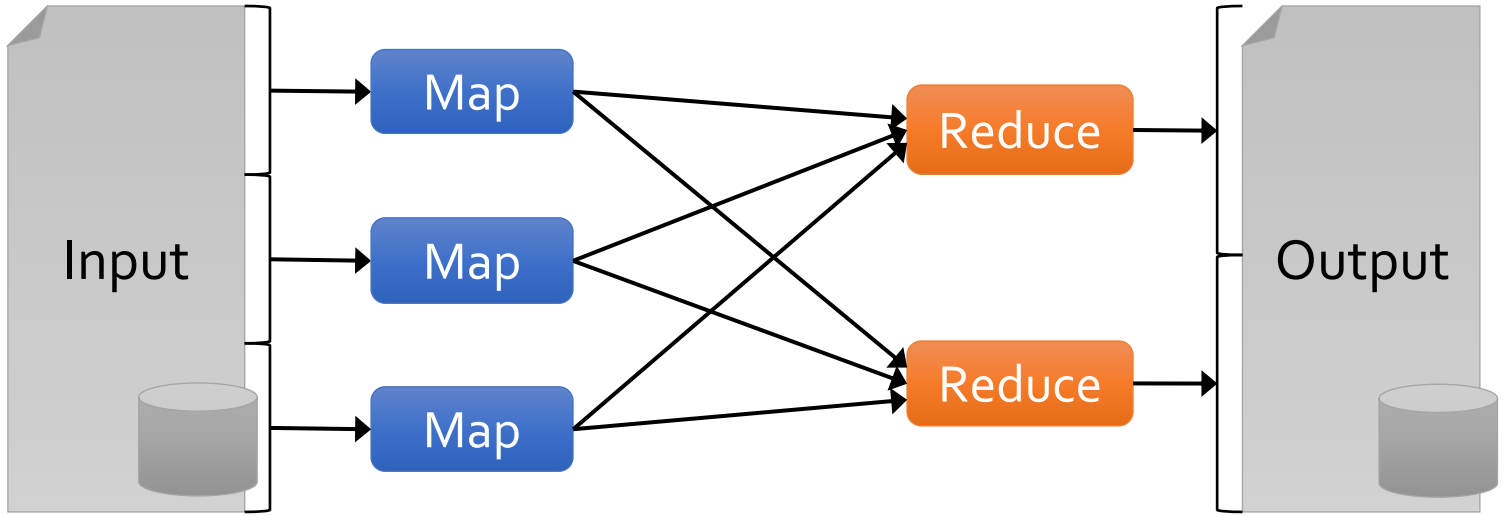
Goals

- Extend the MapReduce model to better support two common classes of analytics apps:
  - **Iterative** algorithms (machine learning, graphs)
  - **Interactive** data mining

- Enhance programmability:
  - Integrate into Scala programming language
  - Allow interactive use from Scala interpreter

Motivation (cont.)

- Most current cluster programming models are based on *acyclic data flow* from one stable storage to another stable storage



Motivation (cont.)

- Acyclic data flow is inefficient for applications that repeatedly reuse a *working set* of data:
  - **Iterative** algorithms (machine learning, graphs)
  - **Interactive** data mining tools (R, Excel, Python)
- With current frameworks, apps reload data from stable storage on each query

Solution: Resilient Distributed Datasets (RDDs)

- Allow apps to keep working-sets in memory for efficient reuse
- Retain the attractive properties of MapReduce
  - Fault tolerance, data locality, scalability
- Support a wide range of applications

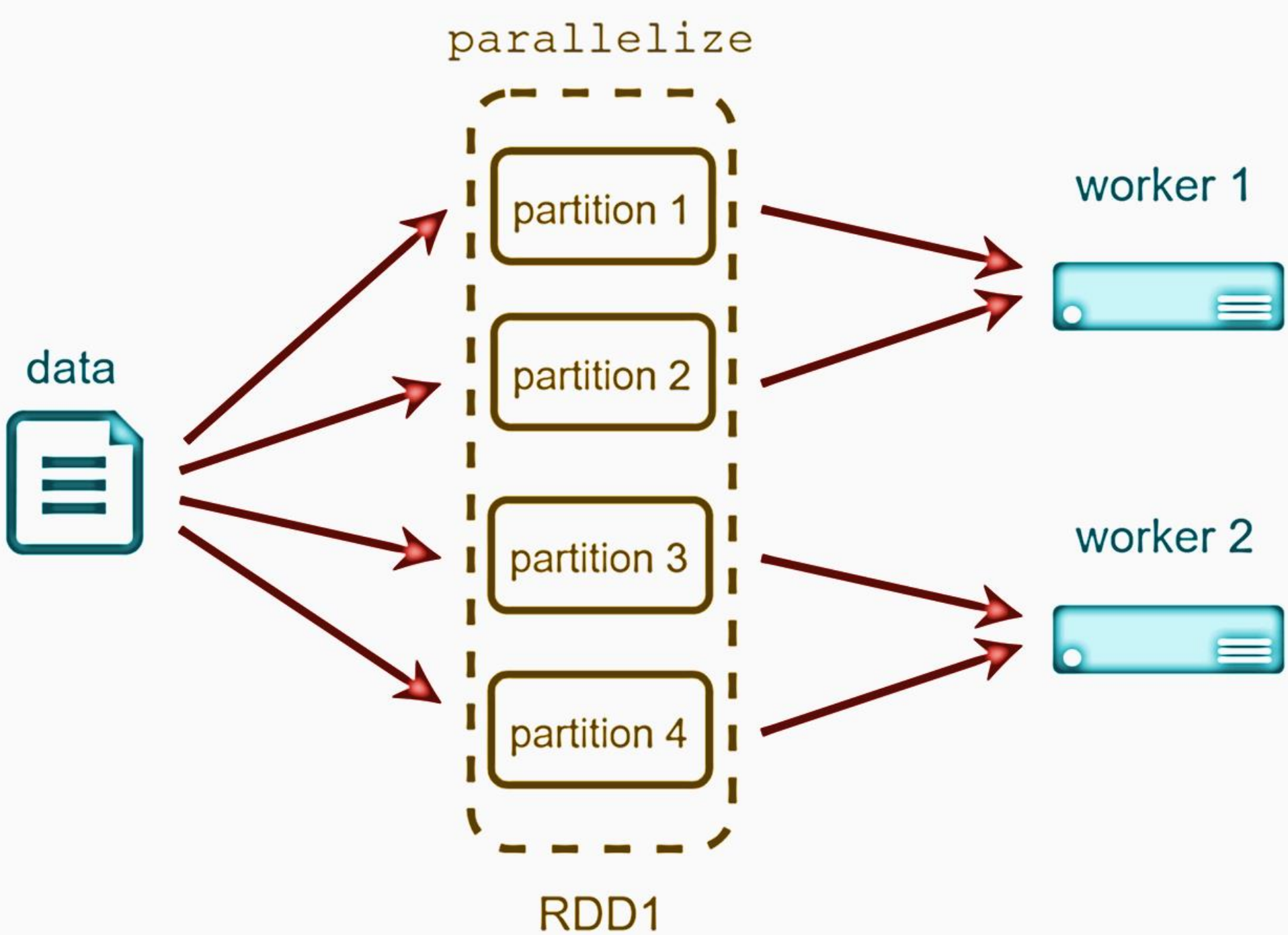
Resilient Distributed Dataset

An important building block of every Spark application is the "Resilient Distributed Dataset".

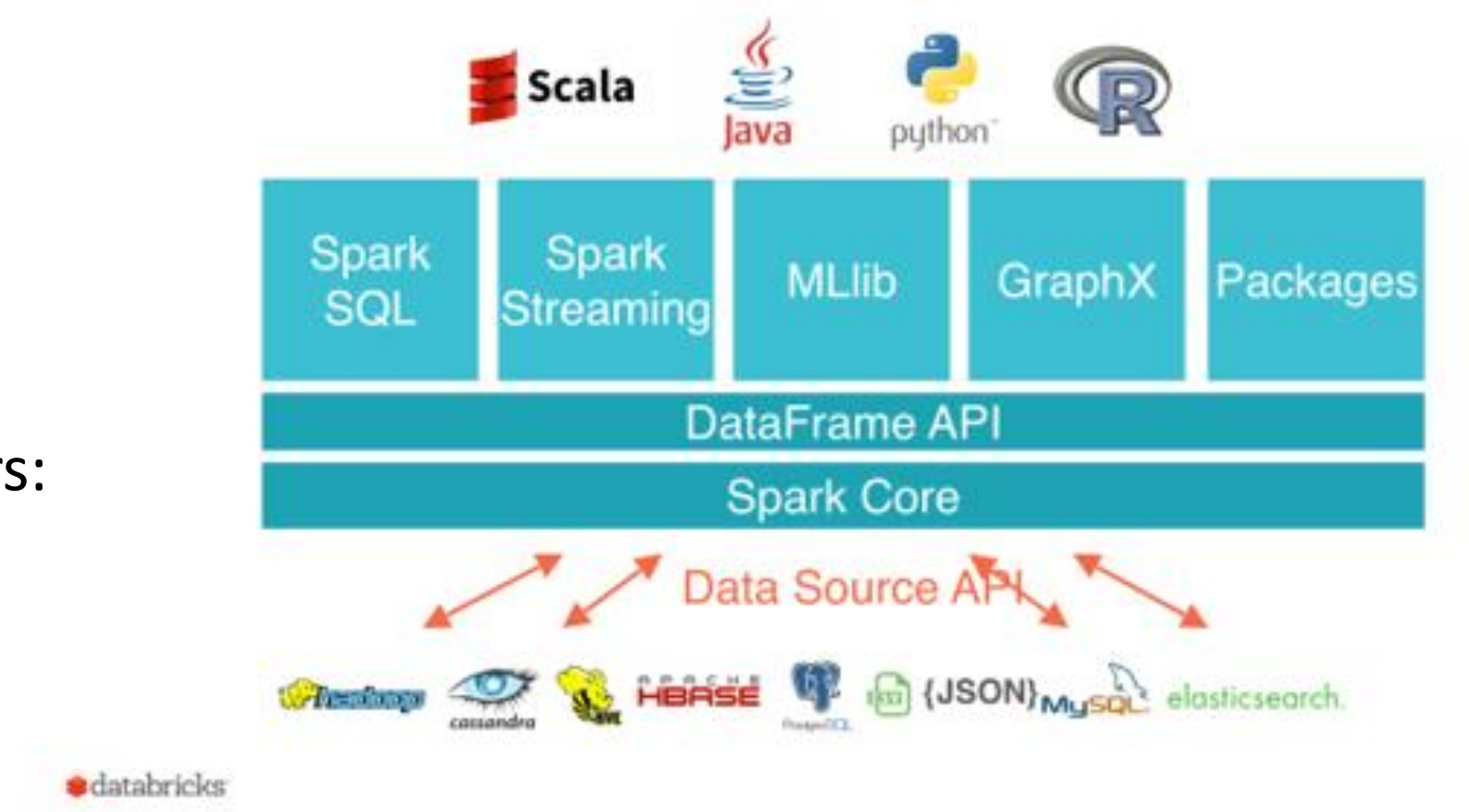
- "Resilient" means fault-tolerant and can rebuild data upon failure.
- Distributed with data residing on multiple nodes in a cluster. The RDD keeps track of data distribution across the workers.

Resilient Distributed Dataset (cont.)

- Dataset is a collection of partitioned data with primitive values or values of values, e.g., tuples or other objects (that represent records of the data with which you work).
- Users write applications that feed data into RDDs and each worker will work on a part of an RDD.


parallelize
data
partition 1
partition 2
partition 3
partition 4
worker 1
worker 2
RDD1

- You can code in Scala, Java, Python, R
- Interactive interpreters: Scala & Python



https://annefou.github.io/pyspark/03-pyspark_context/

Computation Distribution

Spark Core contains the basic functionalities of Spark, such as the APIs that define RDDs, the operations and the actions (e.g., map, filter, reduce, etc.)

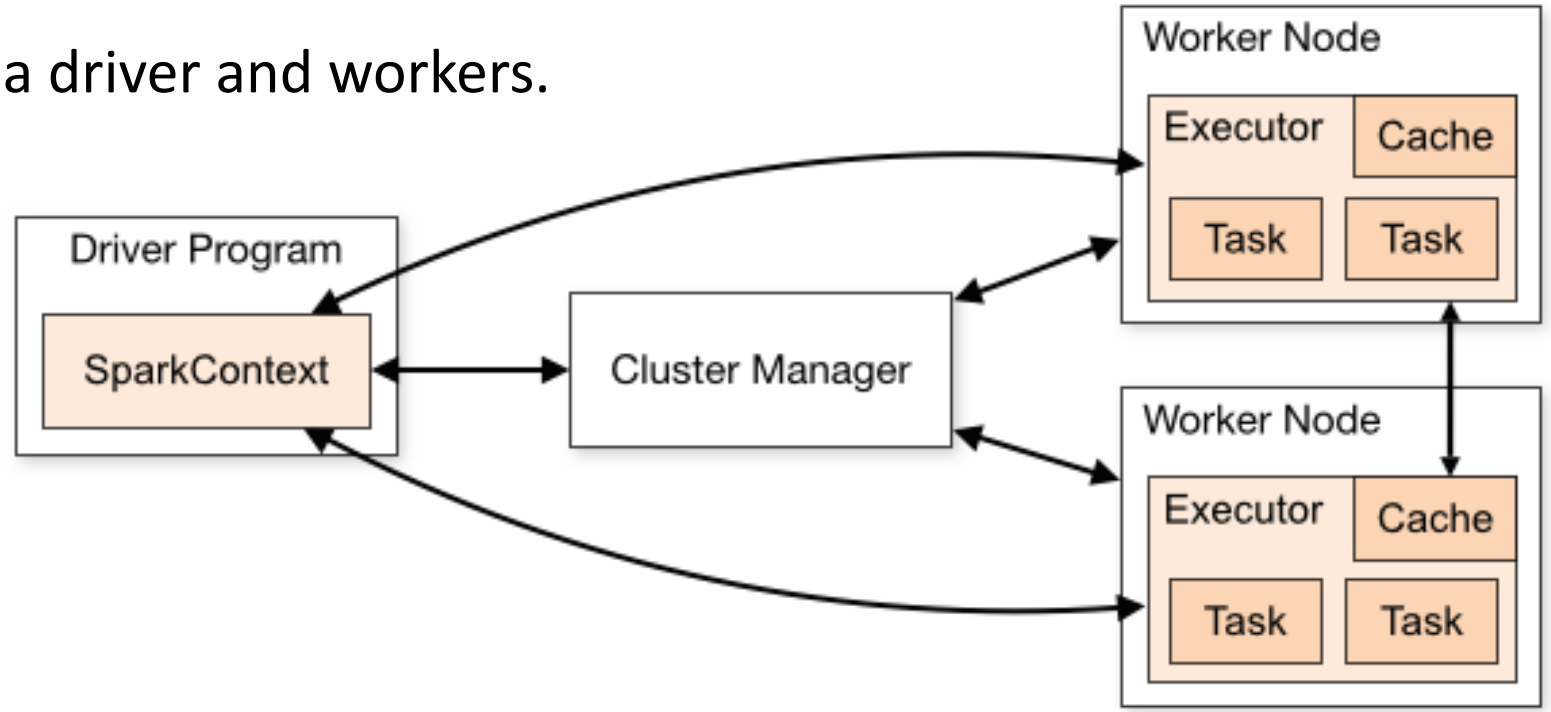
The rest of Spark's libraries are built on top of the RDD and Spark Core:

- **Spark SQL** for SQL and structured data processing. Every database table is represented as an RDD and Spark SQL queries are transformed into Spark operations.
- **MLlib** is a library of common machine learning algorithms implemented as Spark operations on RDDs. This library contains scalable learning algorithms like classifications, regressions, etc. that require iterative operations across large data sets. ML provides higher-level API built on top of DataFrames for constructing ML pipelines. The pipeline concept is inspired by the scikit-learn project.
- **GraphX** is a collection of algorithms and tools for manipulating graphs and performing parallel graph operations and computations.
- **Spark Streaming** for scalable, high-throughput, fault-tolerant stream processing of real-time data.

Spark Arch

The goal is to be able to process your data in parallel by the Spark runtime system.

It consists of a driver and workers.



There are other components such as schedulers, memory manager, but those are details.

Data storage

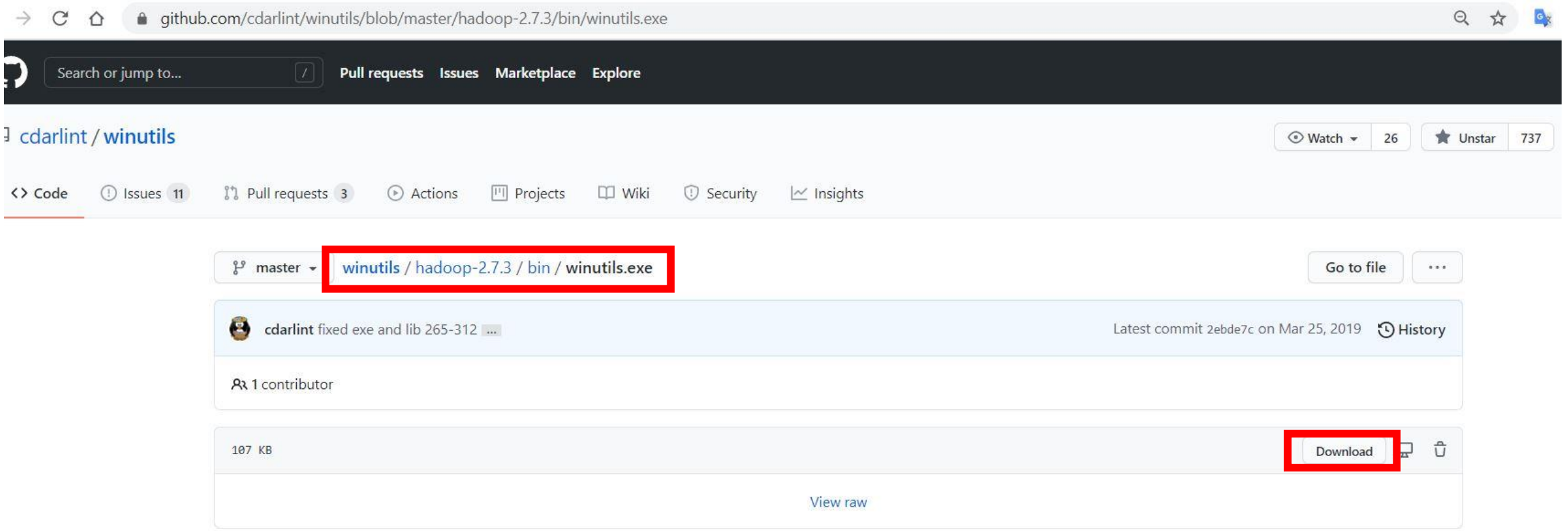
- Spark uses a variety of data storages:

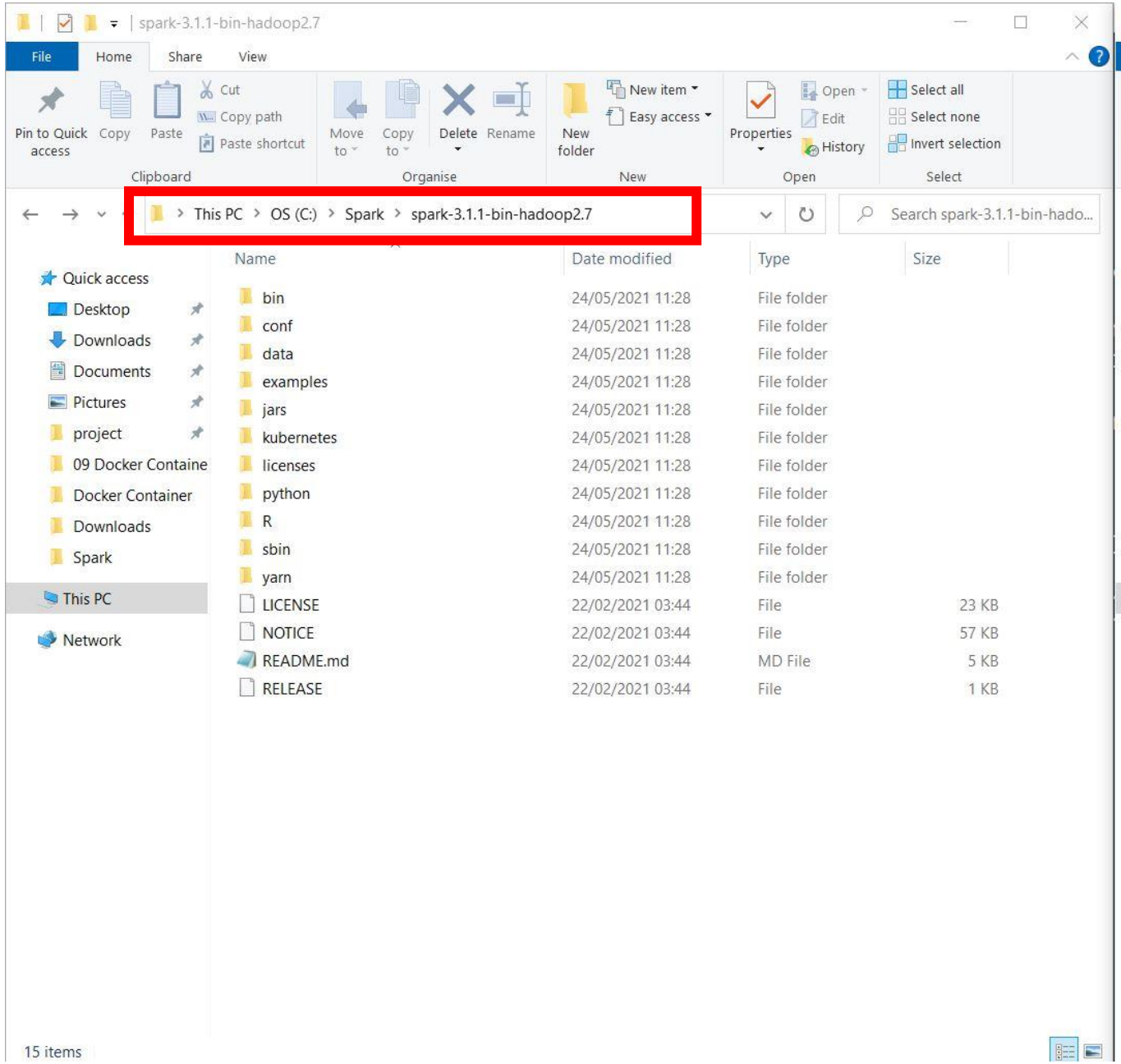
  - ✓ HadoopHDFS
  - ✓ Apache Hbase
  - ✓ Cassandra
  - ✓ MapR-DB
  - ✓ MongoDB
  - ✓ Amazon S3

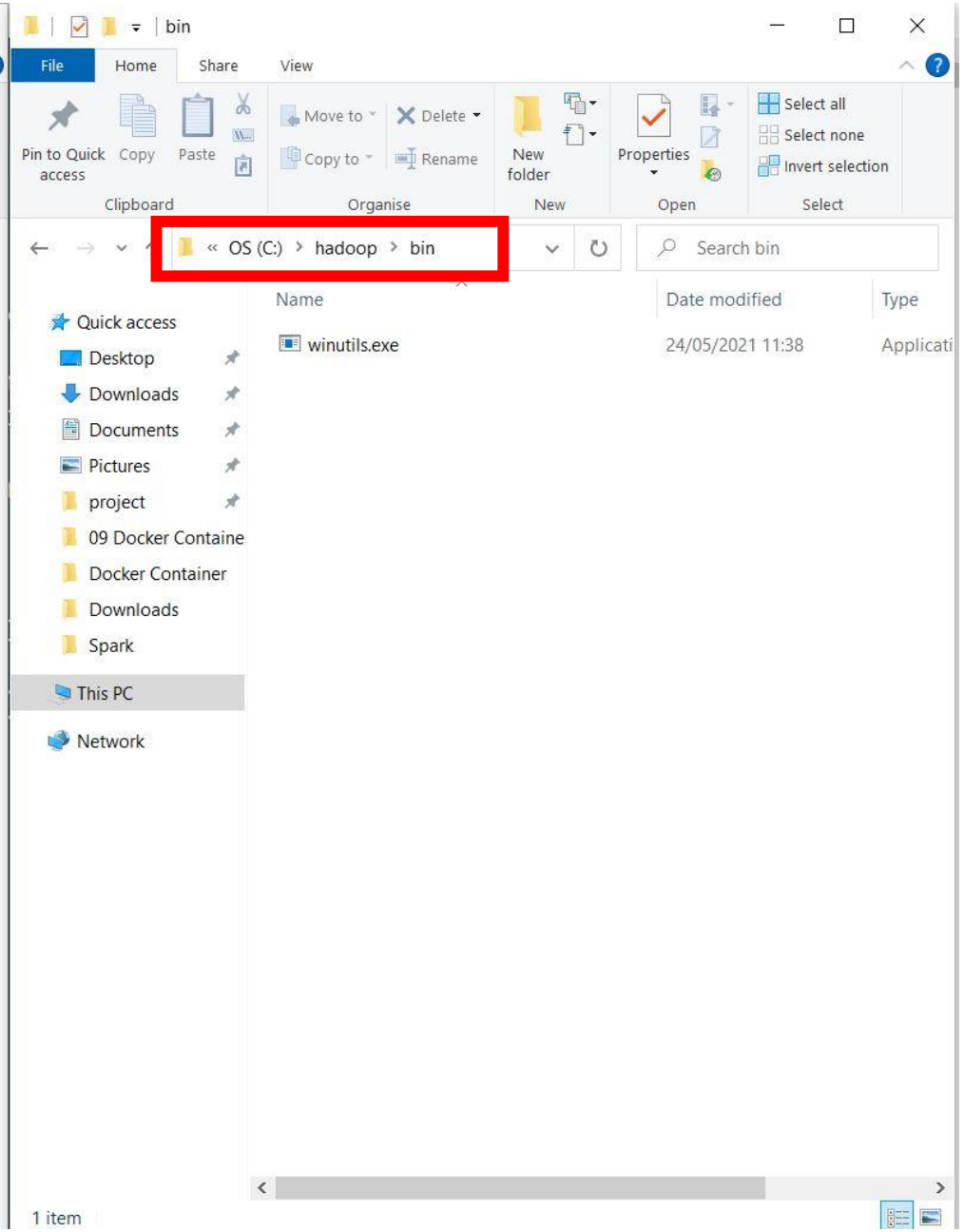
1st example

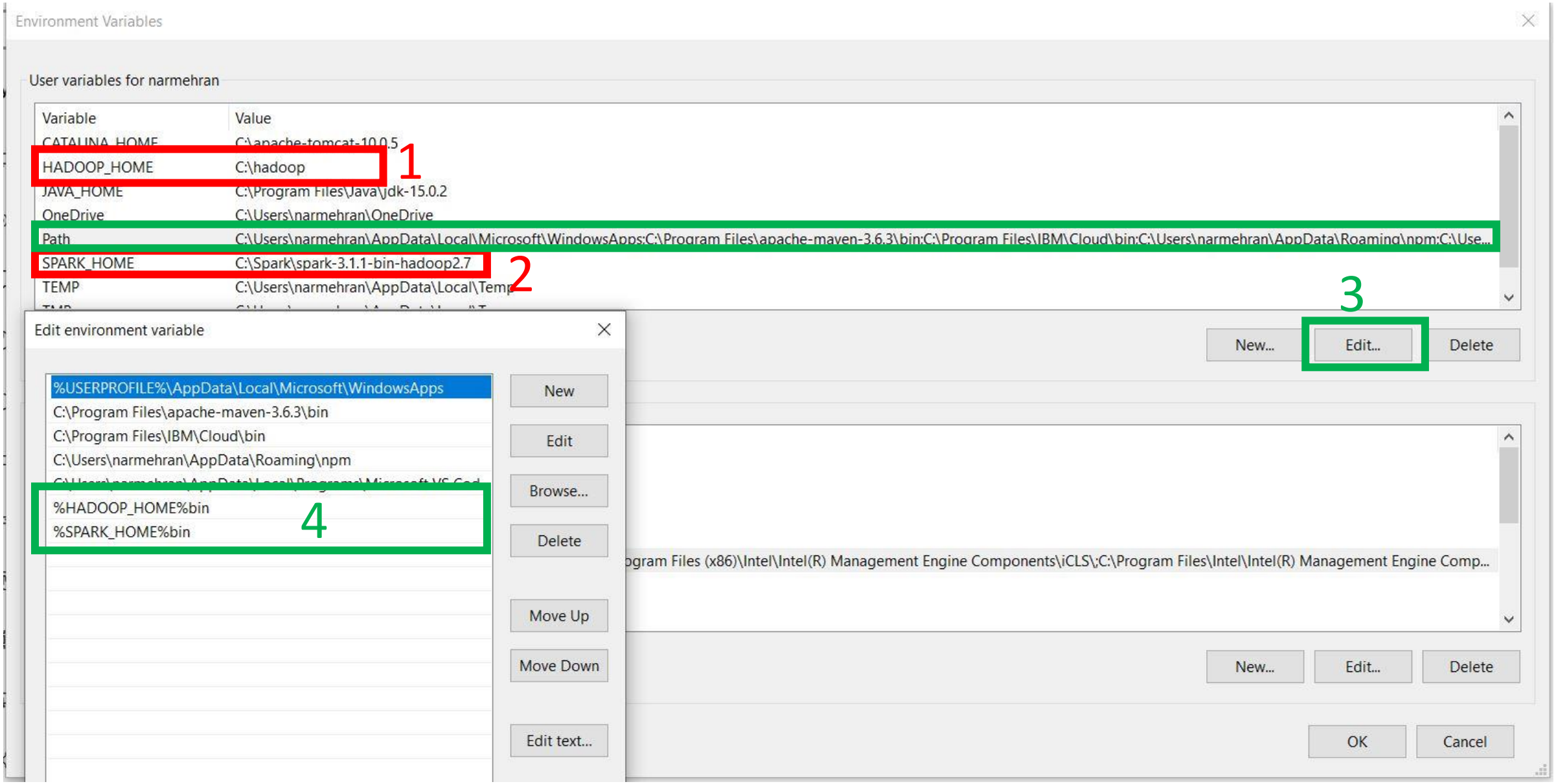
Download Spark

Download *winutils.exe* file for the Hadoop version




spark-3.1.1-bin-hadoop2.7
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organise
New item
Easy access
New folder
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
This PC
OS (C:)
Spark
spark-3.1.1-bin-hadoop2.7
Search spark-3.1.1-bin-hado...
Quick access
Desktop
Downloads
Documents
Pictures
project
09 Docker Containe
Docker Container
Downloads
Spark
This PC
Network
Name
Date modified
Type
Size
bin 24/05/2021 11:28 File folder
conf 24/05/2021 11:28 File folder
data 24/05/2021 11:28 File folder
examples 24/05/2021 11:28 File folder
jars 24/05/2021 11:28 File folder
kubernetes 24/05/2021 11:28 File folder
licenses 24/05/2021 11:28 File folder
python 24/05/2021 11:28 File folder
R 24/05/2021 11:28 File folder
sbin 24/05/2021 11:28 File folder
yarn 24/05/2021 11:28 File folder
LICENSE 22/02/2021 03:44 File 23 KB
NOTICE 22/02/2021 03:44 File 57 KB
README.md 22/02/2021 03:44 MD File 5 KB
RELEASE 22/02/2021 03:44 File 1 KB
15 items


bin
File
Home
Share
View
Pin to Quick access
Copy
Paste
Clipboard
Move to
Copy to
Delete
Rename
Organise
New folder
New
Properties
Open
Select all
Select none
Invert selection
Select
OS (C:)
hadoop
bin
Search bin
Quick access
Desktop
Downloads
Documents
Pictures
project
09 Docker Containe
Docker Container
Downloads
Spark
This PC
Network
Name
Date modified
Type
winutils.exe 24/05/2021 11:38 Applicati
1 item


Environment Variables
User variables for narmehran
Variable
Value
HADOOP_HOME
C:\hadoop
1
JAVA_HOME
C:\Program Files\Java\jdk-15.0.2
OneDrive
C:\Users\narmehran\OneDrive
Path
SPARK_HOME
C:\Spark\spark-3.1.1-bin-hadoop2.7
2
TEMP
C:\Users\narmehran\AppData\Local\Temp
3
New...
Edit...
Delete
Edit environment variable
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Program Files\apache-maven-3.6.3\bin
C:\Program Files\IBM\Cloud\bin
C:\Users\narmehran\AppData\Roaming\npm
%HADOOP_HOME%bin
%SPARK_HOME%bin
4
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel

To run Scala Spark interpreter

```
PS C:\WINDOWS\system32> spark-shell.cmd
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/C:/Spark/spark-3.1.1-bin-hadoop2.7/jars/spark-unsafe_2.12
-3.1.1.jar) to constructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
21/05/24 13:39:27 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where app
licable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://DESKTOP-UHNUR97.mshome.net:4040
Spark context available as 'sc' (master = local[*], app id = local-1621859373501).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.1.1
      /_/

Using Scala version 2.12.10 (Java HotSpot(TM) 64-Bit Server VM, Java 15.0.2)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

To run Scala Spark interpreter

```
scala> spark.range(1000 * 1000 * 1000).count()
res1: Long = 1000000000
```

```
scala> val x =sc.textFile("C:\\Spark\\spark-3.1.1-bin-hadoop2.7\\README.md")
x: org.apache.spark.rdd.RDD[String] = C:\Spark\spark-3.1.1-bin-hadoop2.7\README.md MapPartitionsRDD[8] at textFile at <console>:24
```

```
scala> x.take(11).foreach(println)
# Apache Spark

Spark is a unified analytics engine for large-scale data processing. It provides
high-level APIs in Scala, Java, Python, and R, and an optimized engine that
supports general computation graphs for data analysis. It also supports a
rich set of higher-level tools including Spark SQL for SQL and DataFrames,
MLlib for machine learning, GraphX for graph processing,
and Structured Streaming for stream processing.

<https://spark.apache.org/>
```

To run Scala Spark interpreter

```
scala> val y = x.map(_.reverse)
y: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[9] at map at <console>:25
```

```
scala> y.take(11).foreach(println)
krapS ehcapA #

sedivorp tI .gnissecorp atad elacs-egral rof enigne scitylana deifinu a si krapS
taht enigne dezimitpo na dna ,R dna ,nohtyP ,avaJ ,alacS ni sIPA level-hgih
a stroppus osla tI .sisylana atad rof shparg noitatupmoc lareneg stroppus
,semarFataD dna LQS rof LQS krapS gnidulcni sloot level-rehgih fo tes hcir
,gnissecorp hparg rof XhparG ,gninrael enihcam rof bilLM
.gnissecorp maerts rof gnimaertS derutcurtS dna

>/gro.ehcapa.kraps//:sptth<
```

2nd example

Spark installation in Ubuntu

~$ pip3 install pyspark

~$ wget https://www.apache.org/dyn/closer.lua/spark/spark-3.0.1/spark-3.0.1.tgz

~$ tar xf spark-3.0.1.tgz

~$ cd spark-3.0.1/

~/spark-3.0.1$ export SPARK_HOME="/home/narges/spark-3.0.1"

~/spark-3.0.1$ export PATH="$SPARK_HOME/bin:$PATH"

~/spark-3.0.1$ export PYTHONPATH="$SPARK_HOME/python:$PYTHONPATH"

~/spark-3.0.1$ export PYSPARK_PYTHON=python3

~/spark-3.0.1$ spark-shell

~/spark-3.0.1$ pyspark

PySpark running?

```
narges@ThinkCentreM910s:~/spark-3.0.1$ pyspark
Python 3.6.9 (default, Oct  8 2020, 12:12:24)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
20/12/10 17:51:02 WARN Utils: Your hostname, ThinkCentreM910s resolves to a loopback address: 127.0.1.1; using 143.205.122.147 instead (on interface enp0s31f6)
20/12/10 17:51:02 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/home/narges/spark-3.0.1/assembly/target/scala-2.12/jars/spark-unsafe_2.12-3.0.1.jar) to constructor java.nio.DirectByteBuffer(lo
ng,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
20/12/10 17:51:03 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.0.1
      /_/

Using Python version 3.6.9 (default, Oct  8 2020 12:12:24)
SparkSession available as 'spark'.
>>> nums = sc.parallelize([1,2,3,4])
>>> nums.map(lambda x: x*x).collect()
[1, 4, 9, 16]
>>>
```

PySpark wordcount example

~/spark-3.0.1/examples/src/main/python$ python3 wordcount.py ~/spark-3.0.1/AUTHORS.txt

```
narges@ThinkCentreM910s:~/spark-3.0.1/examples/src/main/python$ python3 wordcount.py ~/spark-3.0.1/AUTHORS.txt
20/12/10 18:36:45 WARN Utils: Your hostname, ThinkCentreM910s resolves to a loopback address: 127.0.1.1; using 143.205.122.147 instead (on interface enp0s31f6)
20/12/10 18:36:45 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/home/narges/spark-3.0.1/assembly/target/scala-2.12/jars/spark-unsafe_2.12-3.0.1.jar) to constructor java.nio.DirectByteBuffer(lo
ng,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
20/12/10 18:36:45 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
[redacted] 1
[redacted]1
narges@ThinkCentreM910s:~/spark-3.0.1/examples/src/main/python$
```

Lambda function in Python

- The general structure of a lambda function is:

  lambda <args>: <expr>

- For instance, “a python function to double the value of a scalar” is:

  lambda x: x**2

Basic Transformations

```
# Create a simple RDD by simply passing some data such as a file or a list
>nums = sc.parallelize([1, 2, 3])

# Pass each element through a function
>squares = nums.map(lambda x: x*x)   // {1, 4, 9}

# Keep elements passing a predicate
>even = squares.filter(lambda x: x % 2 == 0) // {4}

# Read a text file and count the number of lines containing error
>lines = sc.textFile(“file.log”)
>lines.filter(lambda s: “ERROR” in s).count()
```

Basic Actions

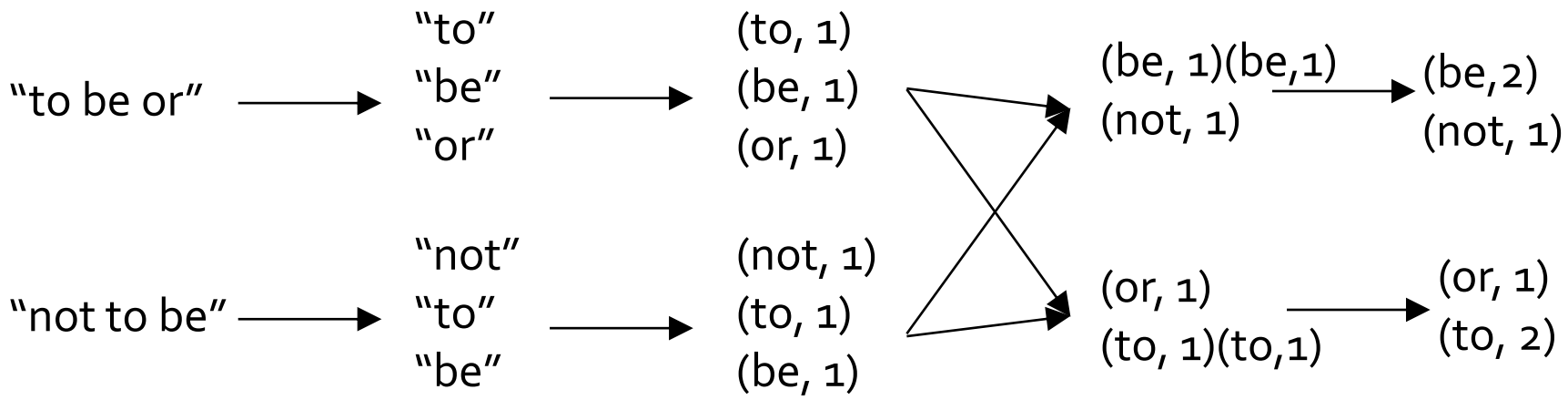
```
> nums = sc.parallelize([1, 2, 3])

# Retrieve RDD contents as a local collection
> nums.collect() # => [1, 2, 3]

# Return first K elements
> nums.take(2)   # => [1, 2]

# Count number of elements
> nums.count()   # => 3

# Merge elements with an associative function
> nums.reduce(lambda x, y: x + y)  # => 6

# Write elements to a text file
> nums.saveAsTextFile("hdfs://file.txt")
```

Several Key-Value Operations

```
> pets = sc.parallelize([(“cat”, 1), (“dog”, 1), (“cat”, 2)])

> pets.reduceByKey(lambda x, y: x + y)
                    # => {(cat, 3), (dog, 1)}

> pets.groupByKey() # => {(cat, [1, 2]), (dog, [1])}

> pets.sortByKey()  # => {(cat, 1), (cat, 2), (dog, 1)}
```

Example: Word Count

```
> lines = sc.textFile("hamlet.txt")
> counts = lines.flatMap(lambda line: line.split(" "))
                .map(lambda word: (word, 1))
                .reduceByKey(lambda x, y: x + y)
```



Spark Operations

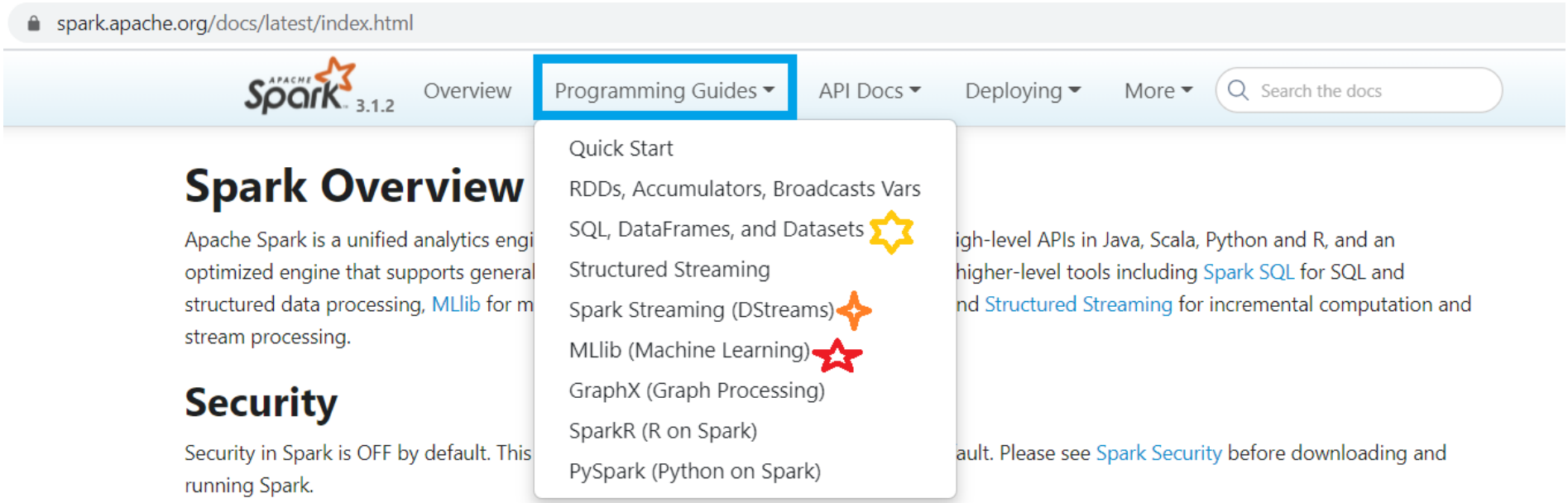
Transformations (define a new RDD) https://spark.apache.org/docs/latest/rdd-programming-guide.html#transformations	map filter sample groupByKey reduceByKey sortByKey	flatMap union join cogroup cross mapValues
Actions (return a result to driver program) https://spark.apache.org/docs/latest/rdd-programming-guide.html#actions	collect reduce count save lookupKey	

Running PySpark with Jupyter in Docker Containers

- Browse DockerHub and,
  - ✓search for the image of jupyter/pyspark-notebook to start your trip with PySpark in Jupyter,
- or:
  - ✓https://github.com/jupyter/docker-stacks/tree/master/pyspark-notebook
- https://realpython.com/pyspark-intro/

Assignment 11

- Each group can be responsible to consider one of the Spark features such as Spark Streaming, SQL or ML,



- Explain about this feature of Spark, prepare an example of it, execute it and get prepared to explain it line by line during the class.

Some links

- https://www.youtube.com/watch?v=3bWdJB96EF4
- https://www.youtube.com/watch?v=5dARTeE6OpU
- https://spark.incubator.apache.org/docs/latest/api/python/index.html
- https://spark.apache.org/docs/latest/api/python/getting_started/quickstart.html
- https://github.com/StephenHarrington/spark
- https://www.tutorialspoint.com/pyspark/pyspark_mllib.htm
- https://www.datacamp.com/community/tutorials/apache-spark-tutorial-machine-learning

References

- https://spark.apache.org/downloads.html
- https://github.com/cdarlint/winutils
- https://phoenixnap.com/kb/install-spark-on-windows-10
- https://runawayhorse001.github.io/LearningApacheSpark/fnn.html
- Zaharia, M., et al. "Apache spark: a unified engine for big data processing." Communications of the ACM 59.11 (2016): 56-65.
- Zaharia, M. An Architecture for Fast and General Data Processing on Large Clusters. Ph.D. thesis, Electrical Engineering and Computer Sciences Department, University of California, Berkeley, 2014; https://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-12.pdf
- https://github.com/himank/K-Means/blob/master/src/KMeans.java