Big Data – Spark
TD1

Course Esilv - Oct 2022

This document:

Objectives

1/ Install on your (Windows) PC a minimalist local SPARK

2/ Configure it, launch spark-shell

3/ Discover spark-shell scala> REPL

4/ Execute basic spark commands on DataSets

Step 1: Download Spark

Google

spark download

All | Images | Videos | Maps | News | More

About 341,000,000 results (0.51 seconds)

https://spark.apache.org › downloads

Downloads | Apache Spark

Download Apache **Spark**™. Choose a **Spark** release: 3.3.0 (Jun 16 2022) ...

Index of /dist/spark · 3.1.3 · Spark 3.3.0 released · Spark 3.1.3 released

You've visited this page 4 times. Last visit: 9/24/22

https://spark.apache.org/downloads.html

APACHE Spark™ | Download | Libraries | Documentation | Examples | Community | Developers

Download Apache Spark™

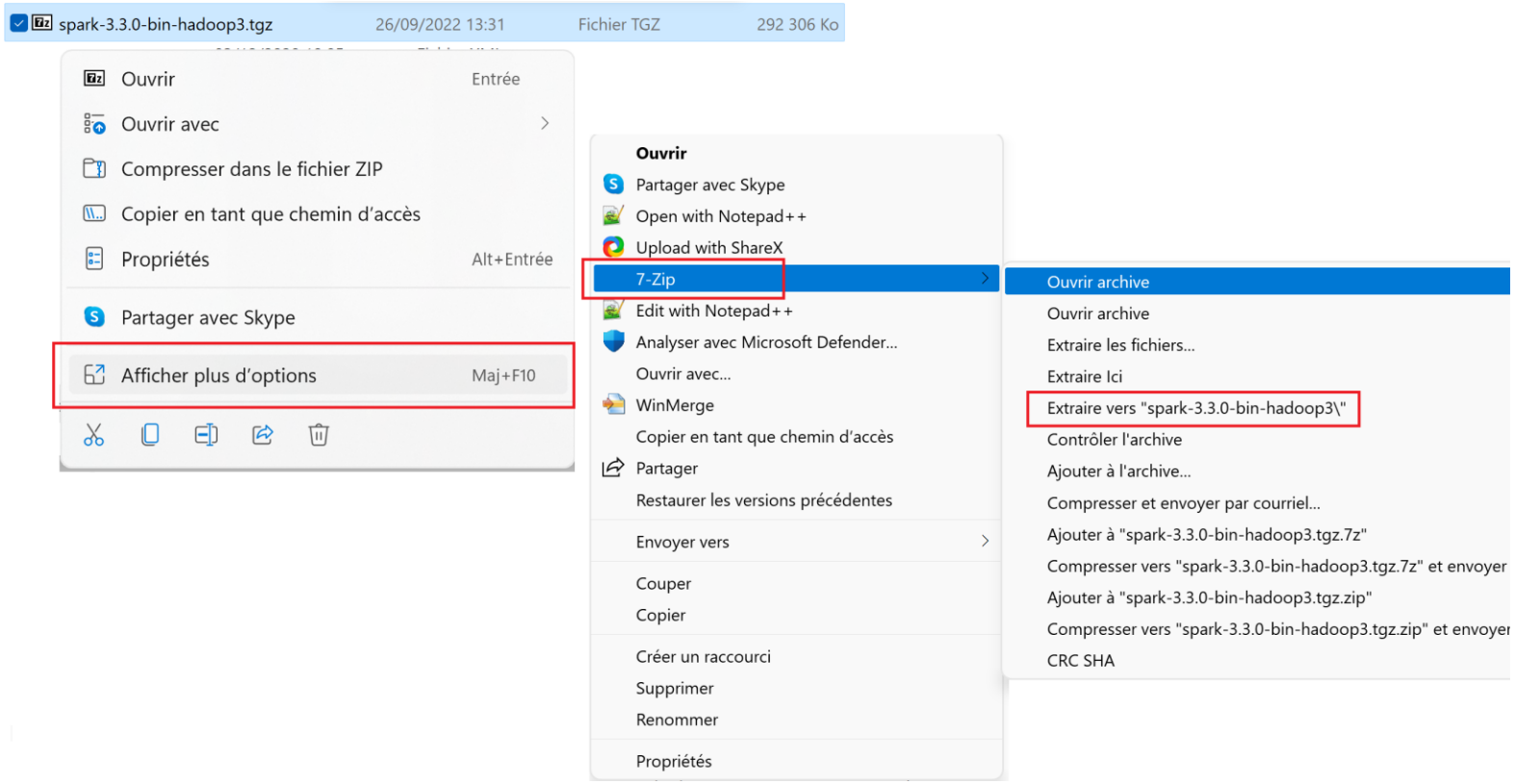
1. Choose a Spark release: 3.3.0 (Jun 16 2022)
2. Choose a package type: Pre-built for Apache Hadoop 3.3 and later
3. Download Spark: spark-3.3.0-bin-hadoop3.tgz

~ 285 Mo

Step 2: Unzip
for example in C:\apps\hadoop\spark

From cygwin / Linux ? => simply type « tar zxf spark*.tgz »

From Windows …



Step 2… check extract « .tar » file!

Ce PC › Windows-SSD (C:) › apps › hadoop › spark-3.3.0-bin-hadoop3

Nom	Modifié le	Type	Taille
spark-3.3.0-bin-hadoop3.tar	26/09/2022 13:31	Fichier TAR	329 940 Ko

SSD (C:) › apps › hadoop › spark-3.3.0

Nom	Modifié le	Type	Taille
bin	09/06/2022 22:37	Dossier de fichiers	
conf	09/06/2022 22:37	Dossier de fichiers	
data	09/06/2022 22:37	Dossier de fichiers	
examples	09/06/2022 22:37	Dossier de fichiers	
jars	09/06/2022 22:37	Dossier de fichiers	
kubernetes	09/06/2022 22:37	Dossier de fichiers	
licenses	09/06/2022 22:37	Dossier de fichiers	
python	09/06/2022 22:37	Dossier de fichiers	
R	09/06/2022 22:37	Dossier de fichiers	
sbin	09/06/2022 22:37	Dossier de fichiers	
yarn	09/06/2022 22:37	Dossier de fichiers	
LICENSE	09/06/2022 22:37	Fichier	23 Ko
NOTICE	09/06/2022 22:37	Fichier	57 Ko
README.md	09/06/2022 22:37	Fichier MD	5 Ko
RELEASE	09/06/2022 22:37	Fichier	1 Ko

Simplify… move and rename sub-sub dir to « c:\apps\hadoop\spark-3.3.0 »

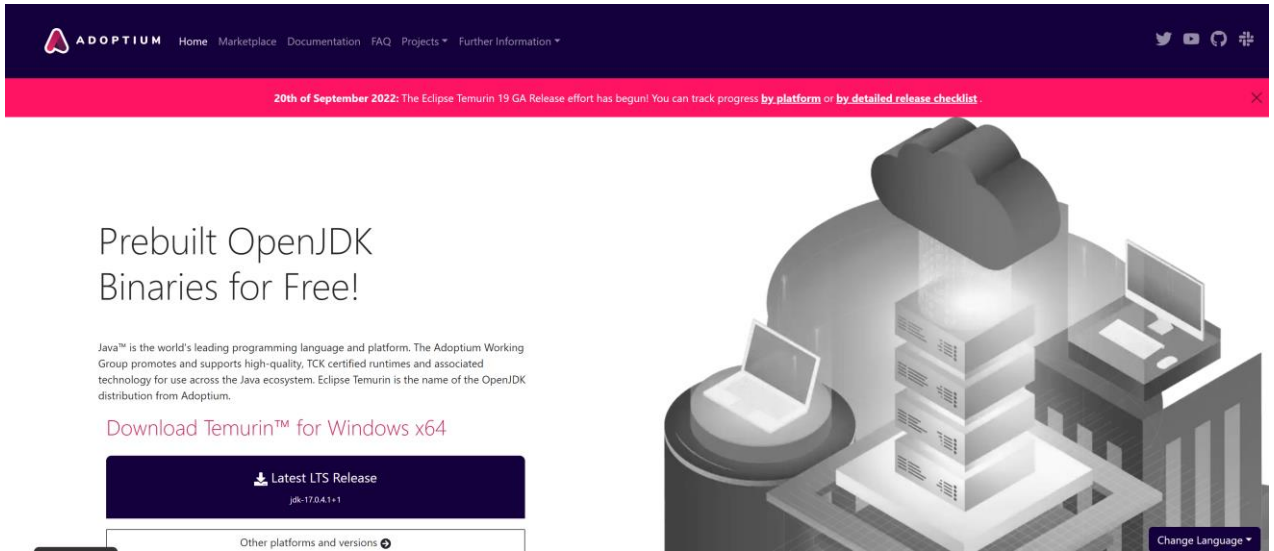
Step 3: Pre-requisite
Install JDK 8 (or more recent)

Notice: most of the Hadoop ecosystem still use Java 8

JDK is Open-Source, but pre-built binaries may be licenced if downloading from oracle.com

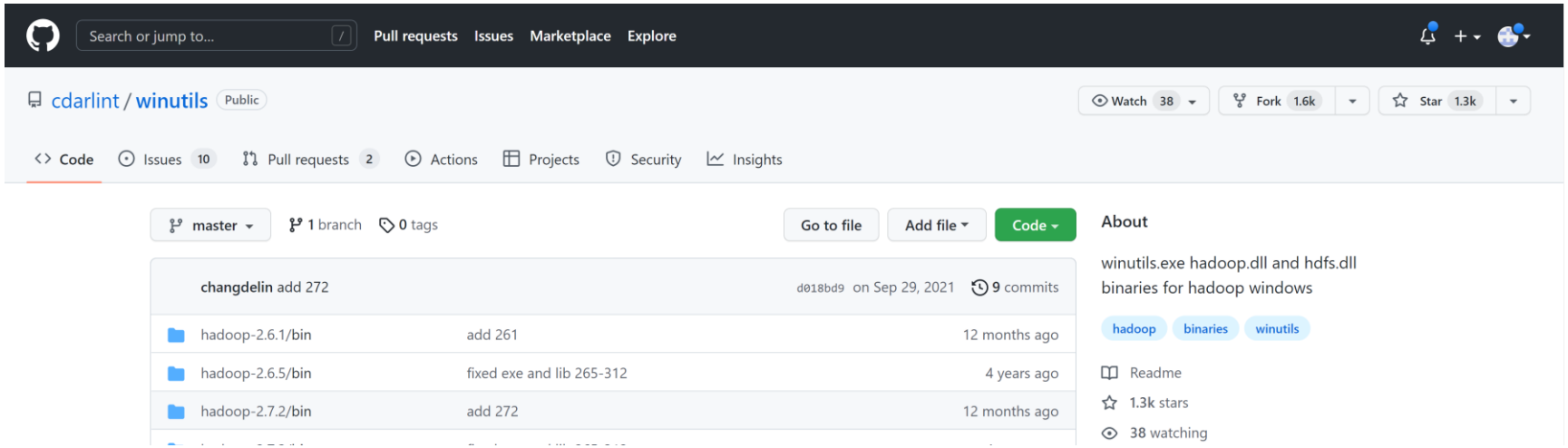
Download from « adoptium » (previously « adoptOpenJdk »)

https://adoptium.net/



Step 4: on Windows only
Download « WinUtils »

https://github.com/cdarlint/winutils



Step 4… Copy winutils.exe, hadoop.dll to you Spark (or Hadoop) \bin\

master winutils / hadoop-3.2.2 / bin / Go to file Add file ···

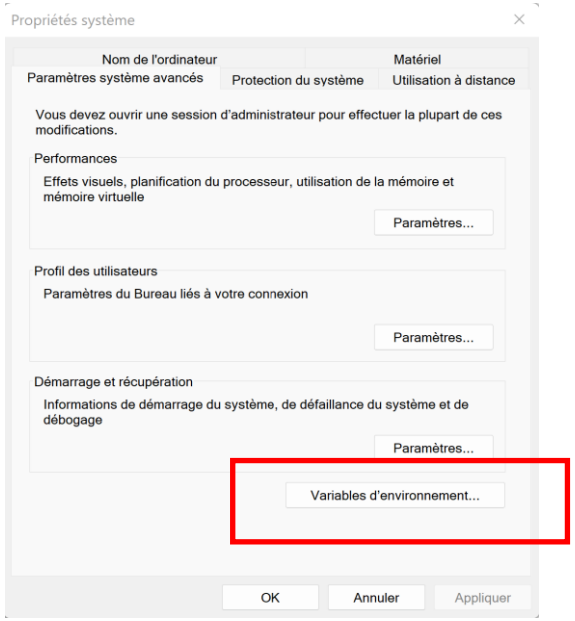
jarieshan compile hadoop-3.2.2 2def4b7 on Apr 13, 2021 History

..		
hadoop	compile hadoop-3.2.2	2 years ago
hadoop.cmd	compile hadoop-3.2.2	2 years ago
hadoop.dll	compile hadoop-3.2.2	2 years ago
hadoop.exp	compile hadoop-3.2.2	2 years ago
hadoop.lib	compile hadoop-3.2.2	2 years ago
hadoop.pdb	compile hadoop-3.2.2	2 years ago
hdfs	compile hadoop-3.2.2	2 years ago
hdfs.cmd	compile hadoop-3.2.2	2 years ago
libwinutils.lib	compile hadoop-3.2.2	2 years ago
mapred	compile hadoop-3.2.2	2 years ago
mapred.cmd	compile hadoop-3.2.2	2 years ago
winutils.exe	compile hadoop-3.2.2	2 years ago
winutils.pdb	compile hadoop-3.2.2	2 years ago
yarn	compile hadoop-3.2.2	2 years ago
yarn.cmd	compile hadoop-3.2.2	2 years ago

Step 5 : Configure Environment Variables

Prefer edit + executing specific « c:\apps\setenv-xyz-version-123.bat »

Rather then Edit Windows System



Step 5: set JAVA_HOME, SPARK_HOME, HADOOP_HOME, PATH

setenv-spark-td1.cmd

```
@echo off
@echo ... executing setenv-spark-td1.cmd

set JAVA_HOME=C:\apps\jdk\jdk-8
set PATH=%JAVA_HOME%\bin;%PATH%

set SPARK_HOME=C:\apps\hadoop\spark-3.3.0-hadoop-3.3
set PATH=%SPARK_HOME%\bin;%PATH%

set HADOOP_HOME=C:\apps\hadoop\spark-3.3.0-hadoop-3.3

@echo .. using JAVA_HOME: %JAVA_HOME%
@echo .. using SPARK_HOME: %SPARK_HOME%
```

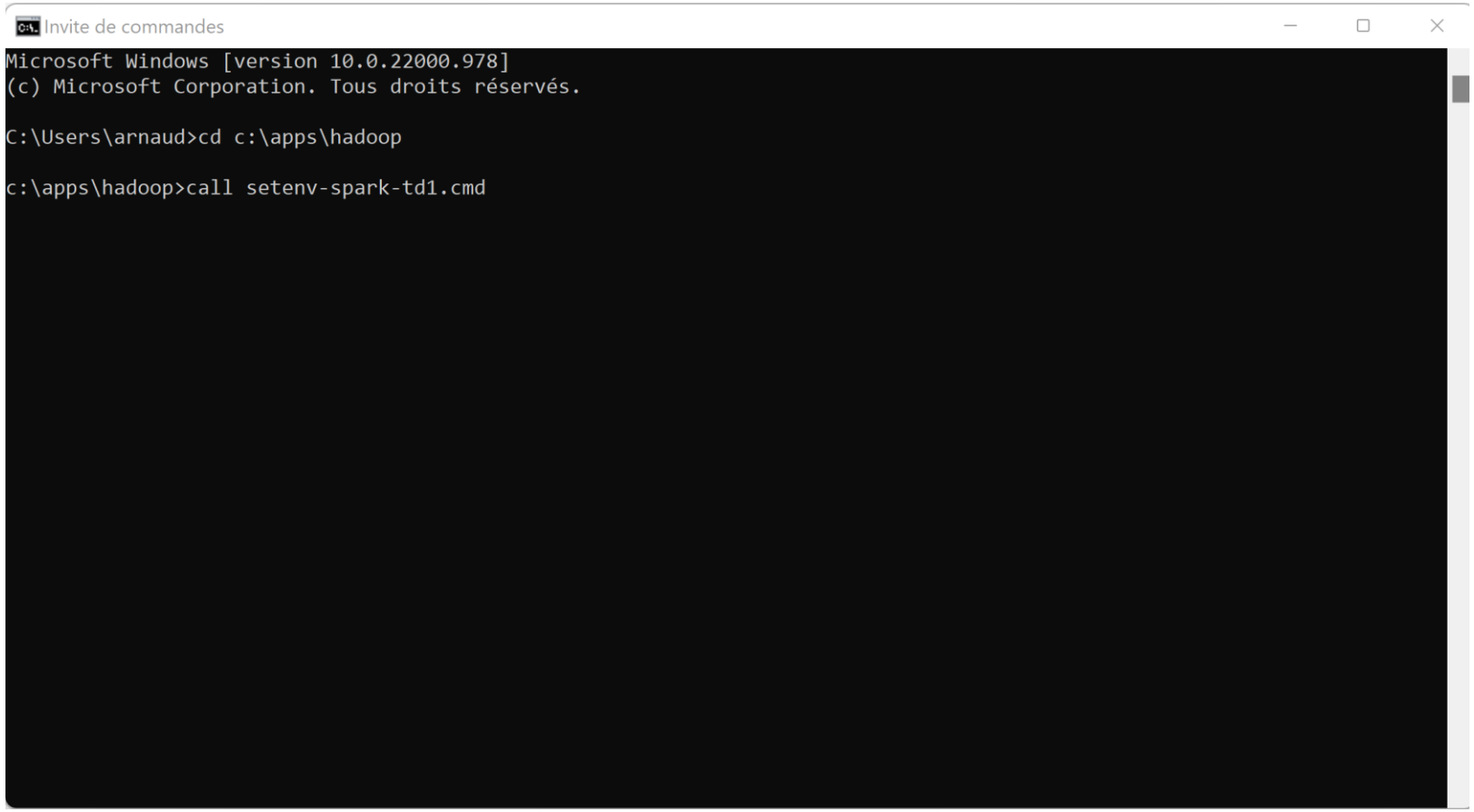
Step 5 ? : if using explicit HADOOP_HOME ?

setenv-spark-td1.cmd

```
@echo off
@echo ... executing setenv-spark-td1.cmd

set JAVA_HOME=C:\apps\jdk\jdk-8
set PATH=%JAVA_HOME%\bin;%PATH%

set HADOOP_HOME=C:\apps\hadoop\hadoop-3.3.4
set PATH=%HADOOP_HOME%\bin;%PATH%

set SPARK_HOME=C:\apps\hadoop\spark-3.3.0
set PATH=%SPARK_HOME%\bin;%PATH%


@echo .. using JAVA_HOME: %JAVA_HOME%
@echo .. using HADOOP_HOME: %HADOOP_HOME%
@echo .. using SPARK_HOME: %SPARK_HOME%
```

Step 6: Sanity Checks …

Windows > cmd
cd <yourdir>
call <your-setenv>.cmd



Sanity Check: Java

Invite de commandes

```
c:\apps\hadoop>java -version
openjdk version "1.8.0_345"
OpenJDK Runtime Environment (Temurin)(build 1.8.0_345-b01)
OpenJDK 64-Bit Server VM (Temurin)(build 25.345-b01, mixed mode)

c:\apps\hadoop>
```

Sanity check: WinUtils

Invite de commandes

```
c:\apps\hadoop>
c:\apps\hadoop>
c:\apps\hadoop>winutils
Usage: winutils [command] ...
Provide basic command line utilities for Hadoop on Windows.

The available commands and their usages are:

chmod          Change file mode bits.

Usage: chmod [OPTION] OCTAL-MODE [FILE]
   or: chmod [OPTION] MODE [FILE]
Change the mode of the FILE to MODE.

   -R: change files and directories recursively

Each MODE is of the form '[ugoa]*([-+=]([rwxX]*|[ugo]))+'.


chown          Change file owner.

Usage: chown [OWNER][:[GROUP]] [FILE]
Change the owner and/or group of the FILE to OWNER and/or GROUP.

Note:
On Linux, if a colon but no group name follows the user name, the group of
the files is changed to that user's login group. Windows has no concept of
a user's login group. So we do not change the group owner in this case.
```

Sanity Check « spark-shell --version »

Invite de commandes

```
Microsoft Windows [version 10.0.22000.978]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\arnaud>cd c:\apps\hadoop

c:\apps\hadoop>call setenv-spark-td1.cmd
... executing setenv-spark-td1.cmd
.. using JAVA_HOME: C:\apps\jdk\jdk-8
.. using SPARK_HOME: C:\apps\hadoop\spark-3.3.0-hadoop-3.3
c:\apps\hadoop>
c:\apps\hadoop>
c:\apps\hadoop>spark-shell --version
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.3.0
      /_/

Using Scala version 2.12.15, OpenJDK 64-Bit Server VM, 1.8.0_345
Branch HEAD
Compiled by user ubuntu on 2022-06-09T19:58:58Z
Revision f74867bddfbcdd4d08076db36851e88b15e66556
Url https://github.com/apache/spark
Type --help for more information.

c:\apps\hadoop>
```

Spark and custom user-defined Hadoop?
.. More difficult

Missing hadoop jars
In spark classpath ?!

Invite de commandes

```
Microsoft Windows [version 10.0.22000.978]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\arnaud>cd c:\apps\hadoop

c:\apps\hadoop>call setenv-spark-td1.cmd
... executing setenv-spark-td1.cmd
.. using JAVA_HOME: C:\apps\jdk\jdk-8
.. using HADOOP_HOME: C:\apps\hadoop\hadoop-3.3.4
.. using SPARK_HOME: C:\apps\hadoop\spark-3.3.0
c:\apps\hadoop>
c:\apps\hadoop>
c:\apps\hadoop>spark-shell -version
Error: A JNI error has occurred, please check your installation and try again
Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/hadoop/fs/FSDataInputStream
        at java.lang.Class.getDeclaredMethods0(Native Method)
        at java.lang.Class.privateGetDeclaredMethods(Class.java:2701)
        at java.lang.Class.privateGetMethodRecursive(Class.java:3048)
        at java.lang.Class.getMethod0(Class.java:3018)
        at java.lang.Class.getMethod(Class.java:1784)
        at sun.launcher.LauncherHelper.validateMainClass(LauncherHelper.java:650)
        at sun.launcher.LauncherHelper.checkAndLoadMain(LauncherHelper.java:632)
Caused by: java.lang.ClassNotFoundException: org.apache.hadoop.fs.FSDataInputStream
        at java.net.URLClassLoader.findClass(URLClassLoader.java:387)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:418)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:352)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:351)
        ... 7 more

c:\apps\hadoop>
```

Spark-shell … print(« Hello World »)

Invite de commandes - spark-shell

```
c:\apps\hadoop>spark-shell
22/09/26 14:44:53 WARN Shell: Did not find winutils.exe: java.io.FileNotFoundException: java.io.FileNotFoundException: H
ADOOP_HOME and hadoop.home.dir are unset. -see https://wiki.apache.org/hadoop/WindowsProblems
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/09/26 14:45:00 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
22/09/26 14:45:03 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
Spark context Web UI available at http://DESKTOP-2EGCC8R:4041
Spark context available as 'sc' (master = local[*], app id = local-1664196304279).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.3.0
      /_/

Using Scala version 2.12.15 (OpenJDK 64-Bit Server VM, Java 1.8.0_345)
Type in expressions to have them evaluated.
Type :help for more information.

scala> 22/09/26 14:45:14 WARN ProcfsMetricsGetter: Exception when trying to compute pagesize, as a result reporting of P
rocessTree metrics is stopped


scala>

scala> print("Hello Spark world")
Hello Spark world
scala>
```

Objectives

1/ Install on your (Windows) PC a minimalist local SPARK

2/ Configure it, launch spark-shell

3/ Discover spark-shell scala> REPL

4/ Execute basic spark commands on DataSets, Files

10 mn Pause

Discover Spark-shell
type <enter> <enter> <enter> …

Invite de commandes - spark-shell

```
scala>
scala>
scala>
scala>
scala>
scala>
scala>
scala>
scala>
scala>
scala>
scala>
scala>
scala>
scala>
scala>
scala>
```

REPL = Read – Eval – Print - Loop

For each line,

Spark evaluate (compile in scala code)

Result is printed, with type information and assigned to variable « res${i} »
where « i » is incremented

Variables can be re-used by name

```
scala> 1
res10: Int = 1

scala> 2
res11: Int = 2

scala> res10 + res11
res12: Int = 3

scala> "Whaouh!"
res13: String = Whaouh!

scala>
```

Not only spark-shell…
pyspark => need python exe
sparkR => need ‘R’ exe
spark-sql => need connect to Hive MetaStore Server

Invite de commandes

```
.. using SPARK_HOME: C:\apps\hadoop\spark-3.3.0-hadoop-3.3
c:\apps\hadoop>
c:\apps\hadoop>
c:\apps\hadoop>pyspark

c:\apps\hadoop>sparkR
'R' n’est pas reconnu en tant que commande interne
ou externe, un programme exécutable ou un fichier de commandes.

c:\apps\hadoop>spark-sql
22/09/26 15:13:24 WARN Shell: Did not find winutils.exe: java.io.FileNotFoundException: java.io.FileNotFoundException: H
ADOOP_HOME and hadoop.home.dir are unset. -see https://wiki.apache.org/hadoop/WindowsProblems
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/09/26 15:13:24 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
22/09/26 15:13:26 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
22/09/26 15:13:26 WARN Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 4042.
22/09/26 15:13:29 WARN metastore: Failed to connect to the MetaStore Server...
22/09/26 15:13:32 WARN metastore: Failed to connect to the MetaStore Server...
Terminer le programme de commandes (O/N) ? O
Terminer le programme de commandes (O/N) ? O

c:\apps\hadoop>
```

Discover built-ins commands, :help

Invite de commandes - spark-shell

```
scala> :help
All commands can be abbreviated, e.g., :he instead of :help.
:completions <string>    output completions for the given string
:edit <id>|<line>        edit history
:help [command]          print this summary or command-specific help
:history [num]           show the history (optional num is commands to show)
:h? <string>             search the history
:imports [name name ...] show import history, identifying sources of names
:implicits [-v]          show the implicits in scope
:javap <path|class>      disassemble a file or class name
:line <id>|<line>        place line(s) at the end of history
:load <path>             interpret lines in a file
:paste [-raw] [path]     enter paste mode or paste a file
:power                   enable power user mode
:quit                    exit the interpreter
:replay [options]        reset the repl and replay all previous commands
:require <path>          add a jar to the classpath
:reset [options]         reset the repl to its initial state, forgetting all session entries
:save <path>             save replayable session to a file
:sh <command line>       run a shell command (result is implicitly => List[String])
:settings <options>      update compiler options, if possible; see reset
:silent                  disable/enable automatic printing of results
:type [-v] <expr>        display the type of an expression without evaluating it
:kind [-v] <type>        display the kind of a type. see also :help kind
:warnings                show the suppressed warnings from the most recent line which had any

scala>
```

Discover History

Exercise :

a/ Type 5 basic commands:
print(« line1 ») <ENTER>
print(« line2 ») <ENTER>
….
print(« line5 ») <ENTER>

b/ navigate scroll UP and Down to retype command N

c/ replay all commands, using « :replay »

c/ save session in file « my-supper-commands.txt »

Discover History (stop + relaunch spark-shell)

Exercise :

a/ Stop you spark-shell

b/ restart a spark-shell

c/ do you still see your previous commands (Scroll Up)

d/ do you still see your history ?

e/ Find in which file are written your commands ?
… search for hidden « .scala_history » file, in your home directory C:\users\<login>\.scala_history

Exercise using « :save », « :load »

a/ save all commands session in file « my-super-commands.txt »

b/ load and re-execute commands from file

Exercise using « :replay », « :reset »

a/ replay all in once your previous commands, using « :replay »

b/ reset commands… see History after

Discover Multi-Line edit support

How to type 1 command containing several lines ??

For example a for loop, with if –then-else
Or more realistic … SQL query on select line: SELECT … FROM … WHERE …

Answer:

Type **« :paste »**

then copy& paste multiple lines… or simply type several lines with <enter>

When finished, press **« Control + D »**

Using :paste … Control+D

new 1

```
for(i <- 0 to 5) {
    if (i % 2 == 0) {
        println("Foo " + i);
    } else {
        println("Bar " + i);
    }
}
```

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

for(i <- 0 to 5) {
if (i % 2 == 0) {
println("Foo " + i);
} else {
println("Bar " + i);
}
}


// Exiting paste mode, now interpreting.

Foo 0
Bar 1
Foo 2
Bar 3
Foo 4
Bar 5

scala>
```

Using Multiple line String (example: SQL) triple quotes

new 1

```
val sqlQuery = """
  SELECT a,
    b,
    c
  FROM tableXyz
  WHERE 1=1
    AND a LIKE '%text%'
    AND b > 10
"""
println(sqlQuery)
// will not work here..  spark.sql(sqlQuery).show(10)
```

Invite de commandes - spark-shell

```
scala> :paste
// Entering paste mode (ctrl-D to finish)


val sqlQuery = """
  SELECT a,
    b,
c
  FROM tableXyz
  WHERE 1=1
AND a LIKE '%text%'
AND b > 10
"""
println(sqlQuery)
// will not work here..  spark.sql(sqlQuery).show(10)



// Exiting paste mode, now interpreting.


  SELECT a,
    b,
c
  FROM tableXyz
  WHERE 1=1
AND a LIKE '%text%'
AND b > 10

sqlQuery: String =
"
  SELECT a
```

Spark-shell is a « REPL » for Spark in « scala »

Scala basic commands …

```
scala> var x = 5
x: Int = 5

scala> print("x:" + x)
x:5
scala> print(s"x: ${x}")
x: 5
```

More Scala …

```
scala> (1 to 3).foreach(x => println(x))
1
2
3
```

More Scala with … Sequence, implicit Lambda, « _ » var

```
scala> var ls=Seq(1, 3, "Hello")
ls: Seq[Any] = List(1, 3, Hello)

scala> ls.foreach(println(_))
1
3
Hello
```

« Scala » .. ~ a super set of « Java »
Scala runs on the JVM

```
scala> java.lang.System.out.println("Hello plain old java from Spark-scala")
Hello plain old java from Spark-scala
```

… but not exactly

```
scala> for(int i=0; i < 3; i++) System.out.println("hello " + i)
<console>:1: error: illegal start of simple pattern
       for(int i=0; i < 3; i++) System.out.println("hello " + i)
                 ^
<console>:1: error: '<-' expected but ';' found.
```

```
scala> for(i <- 0 to 2) System.out.println("hello " + i)
hello 0
hello 1
hello 2
```

https://docs.scala-lang.org/cheatsheets/

5-10 minutes
Reading

Scala

LEARN INSTALL PLAYGROUND FIND A LIBRARY COMMUNITY BLOG

docs

Getting Started Scala 3 Learn Tutorials Reference API SIPs

SCALA CHEATSHEET

SCALA CHEATSHEET

Language

Search in doc...

Thanks to Brendan O'Connor, this cheatsheet aims to be a quick reference of Scala syntactic constructions. Licensed by Brendan O'Connor under a CC-BY-SA 3.0 license.

variables

`var x = 5`
GOOD
`x = 6`
	Variable.

`val x = 5`
BAD
`x = 6`
	Constant.

`var x: Double = 5`	Explicit type.

Objectives

1/ Install on your (Windows) PC a minimalist local SPARK

2/ Configure it, launch spark-shell

3/ Discover spark-shell scala> REPL

4/ Execute basic spark commands on DataSets

Introduction to Exercises 1,2,3,4,5 …
The goal is to create a spark DataSet..

Expected DataSet = List of 100_000 rows
each row containing a Tuple of
(int : i,
boolean : true if i is even,
string: « hello ${i} »)

Exercise 1 : create a List of 3 Tuples

var ls = ???

With expected content:

(1, false, "hello 1")
(2, true, "hello 2")
(3, false, "Hello 3")

Hint Exercise 1

In scala,

« (a, b, c) » creates a Tuple with 3 fields _1,_2,_3 with _1=a, _2=b, _3=c
« Seq(a, b, c) » create a Sequence (synonym: List) of 3 elements: a, b, c

Exercise 2: same, but create an empty List, and successively add 3 rows
… the (non idiomatic) Java way

Hint exercise 2

List are immutable !
You can create a new List, as the concatenation of 1 element and an existing list

```
var ls = Seq()
var ls2 =  a :: ls
```

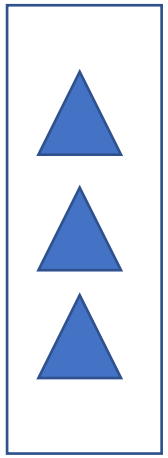
« a » and « ls » must have same generic type!
Use « Seq[Type] » instead of « Seq() »

Exercise 3 .. Same using « functional » code style: transform the sequence (1, 2 … 10000) into Tuples using « map » lambda function

Hint Exercise 3

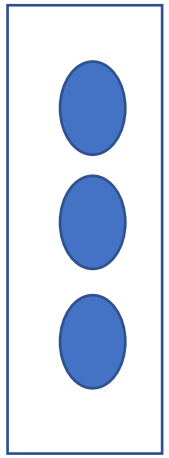
« .map(f) » is a method of List,
to map all elements from source type X to target type Y
taking a function « f » as parameter : « f(X) -> Y »
… f can be written as a lambda: « x => { ..return y;} »

Source : List[X]



transformationFunction : X -> Y

target : List[Y]



Exercise 4: display result from previous exercise

4a / display length of result
4b/ display row 0 tuple
4c/ display row 0, individual fields from tuple

Exercise 5: create a spark DataSet of 100_000 rows
each row containing a Tuple of
int : i
boolean : true if i is even
string: « hello ${i} »

Hint Exercise 5

Just use

spark.createDataset() method

« spark » is a built-in variable
You can type « spark.create » and press TAB TAB to autocomplete

Exercise 6: Display result DataSet of ex. 5

6a / display number of rows
6b/ show first rows (default: 20)
6c/ show only first 2 rows
6d/ show individual columns of row0

Hint Exercise 6

Like on Seq or List … Dataset accept many methods,
like « .show() », « .count() », « .take() » etc.

Exercise 7: Rename columns

Rename columns _1,_2,_3
to
« id », « even », « text »

Hint Exercise 7

Use dataset.withColumnRenamed()

… remember Dataset are immutable !
Most Dataset method returns a new Dataset

so cascade calls (or assign intermediate results)
repeat the operation 3 times

Exercise 8

Filter rows where « id » is « >= 100 »
Using « filter(Column) » operators

To create an expression on column « id » of dataset « ds »,
You can use « ds(« id ») »
OR implicit equivalent $«id »

Hint Exercise 8

Type « .filter» and pres TAB TAB to autocomplete
You can see there are 3 possible overloads for method

```
scala> val dsFilter = ds4.filter

def filter(func: org.apache.spark.api.java.function.FilterFunction[org.apache.spark.sql.Row]): org.apache.spark.sql.Data
set[org.apache.spark.sql.Row]
def filter(func: org.apache.spark.sql.Row => Boolean): org.apache.spark.sql.Dataset[org.apache.spark.sql.Row]
def filter(conditionExpr: String): org.apache.spark.sql.Dataset[org.apache.spark.sql.Row]
def filter(condition: org.apache.spark.sql.Column): org.apache.spark.sql.Dataset[org.apache.spark.sql.Row]
```

use « .filter(ds(« columnName») operator…) »

It is more complex than using SQL conditionExpr… cf exercise 9
But It is simpler than exercise 10 … using lambda predicateFunc : row -> boolean

Exercise 8

Filter rows where

« id » is « >= 100 »

AND

«text » contains substring « 12 »

Hint Exercise 8

Spark as overriden most of the natural operators on class « Column »
Including >=, &&, ||, !, contains, …

You can manipulate columns objects almost as values in scala

Exercise 9

Redo exercise 7,8 using filter by SQL text expression

Exercise 10

Same as exercise 7,8,9… but using « .filter(row => …) »
i.e. Filter with lambda function on row

Use getAs[Type](« columnName ») to extract the value of a column, and cast

Hint Exercise 10

Method signature is
.filter((or.apache.spark.sql.Row row) => boolean)

You need to extract column value for object « Row », and interpret as Int or String

for column « id » as Int … use « row.getAs[Int]("id") »
For column « text » as String … use « row.getAs[String](« text") »

Exercise 11 : same as exercise 1 (on Tuple), but using your own custom « case-class »

Declare a scala case-class (also named Bean, POJO or Record in java)
To replace the Tuple « (Int,boolean,String) » of exercise 1

Should be equivalent to java class:

```
public class CustomBean {
    public int id;
    public boolean even;
    public String text;

    public CustomBean(int id, boolean even, String text) {
       this.id = id;  this.even = even;   this.text = text;
   }
}
```

Exercise 12 : Same as Exercise 5,6,7,8,9 .. Using CaseClass

a/ create a List of 100 000 CustomBean objects, using map + lambda function

b/ create a Dataset[CustomBean] with 100 000 rows

c/ display results : show 10 rows, show row 0, show individual fields of row 0

d/ filter Dataset with « id > 100 and text contains ’12’ »

Exercise 13 : convert to DataFrame (Dataframe = Dataset<Row>)

a/ What was the type of « ds » (from exercise 5) ?
b/ What is the type of « dsFilter » (from exercise 8) ?
c/ What is the type of « beanDs » (from exercise 12) ?

d/ Convert a Dataset of tuple « ds » (from exercise 5) to a spark DataFrame
e/ Convert a Dataset of custom Bean « beanDs » (from exercise 12) to a spark DataFrame

Exercise 14 : Question

Do you understand the in-memory difference between ?

a/ in memory scala collection of Tuple: Set(Tuple (…))
b/ Dataset on scala Tuple: Dataset[(..Tuple)]
c/ Dataset on custom class Dataset[YourCustomBean]
d/ DataFrame = Dataset[Row]

What do you think is the most efficient for developping type-safe code, and internally for Spark ?

Exercise 15: question

When do you use { ._1, ._2 } ?

When do you use { .id, .text } ?

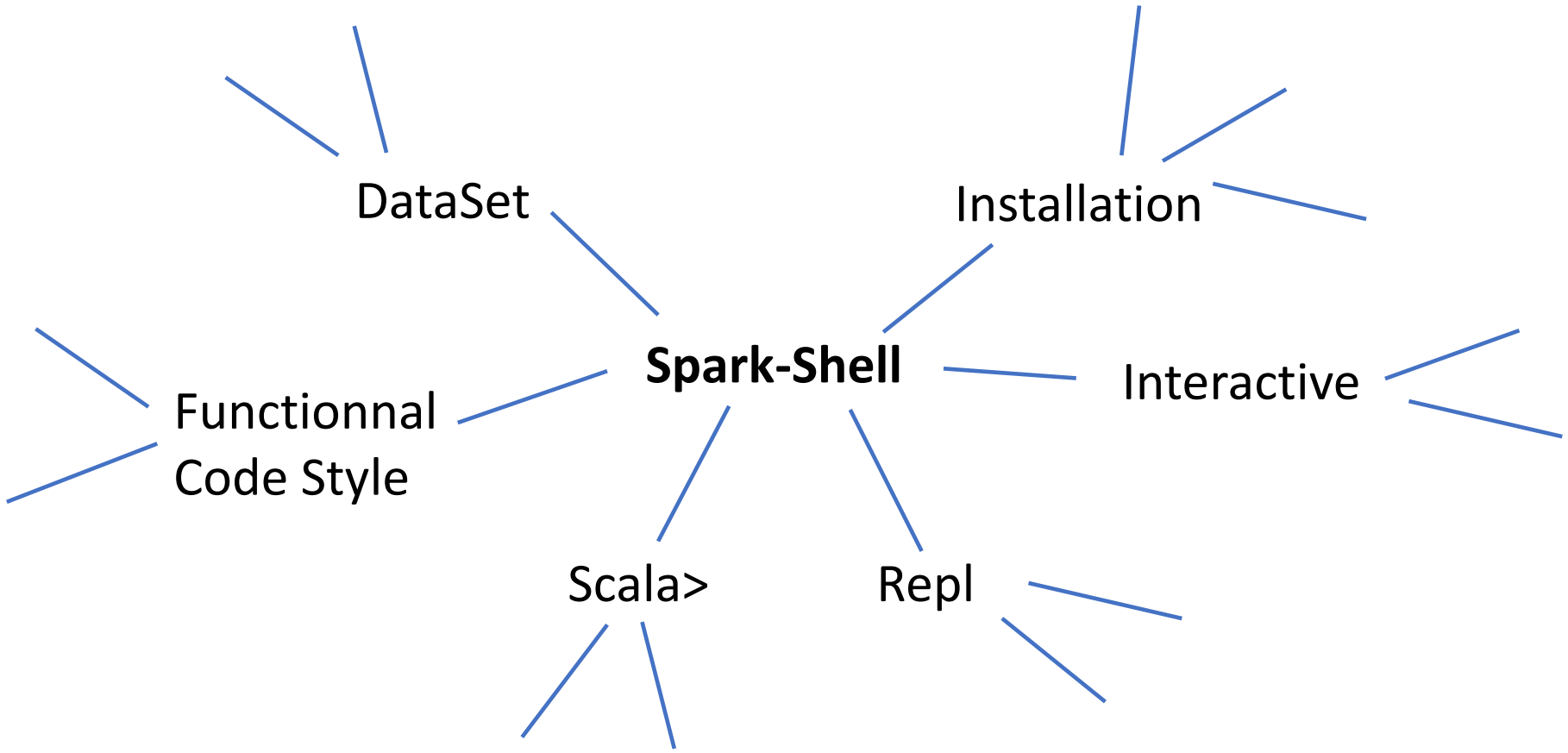
When do you use { .getAs[Int](« id »), .getAs[Int](« text ») } ?

Exercise 16 : MindMap

Draw a MindMap to summarize
what you did and learn from this TD session

Your MindMap should
start with word « spark-shell » in the middle
Then draw star edges to other word chapters and sub-chapters

Hint Exercise 16



Objectives

1/ Install on your (Windows) PC a minimalist local SPARK

2/ Configure it, launch spark-shell

3/ Discover spark-shell scala> REPL

4/ Execute basic spark commands on DataSets

Take-Away

What You learned ?

Questions ?

Next Steps

More CMs

More TDs

Spark concepts:
- File Input / Output
- PARQUET columnar files
- SQL
- Spark Clustering
- DAG, Distribution, Optimizations
- Java binding, UDF, map
- Spark Streaming
- …