

# Big Data – Spark

## Hands-On 1

Arnaud Nauwynck - Oct 2022

# 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



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>



Download

Libraries ▾

Documentation ▾

Examples

Community ▾

Developers ▾

## Download Apache Spark™

1. Choose a Spark release:

2. Choose a package type:

3. Download Spark: [spark-3.3.0-bin-hadoop3.tgz](#)

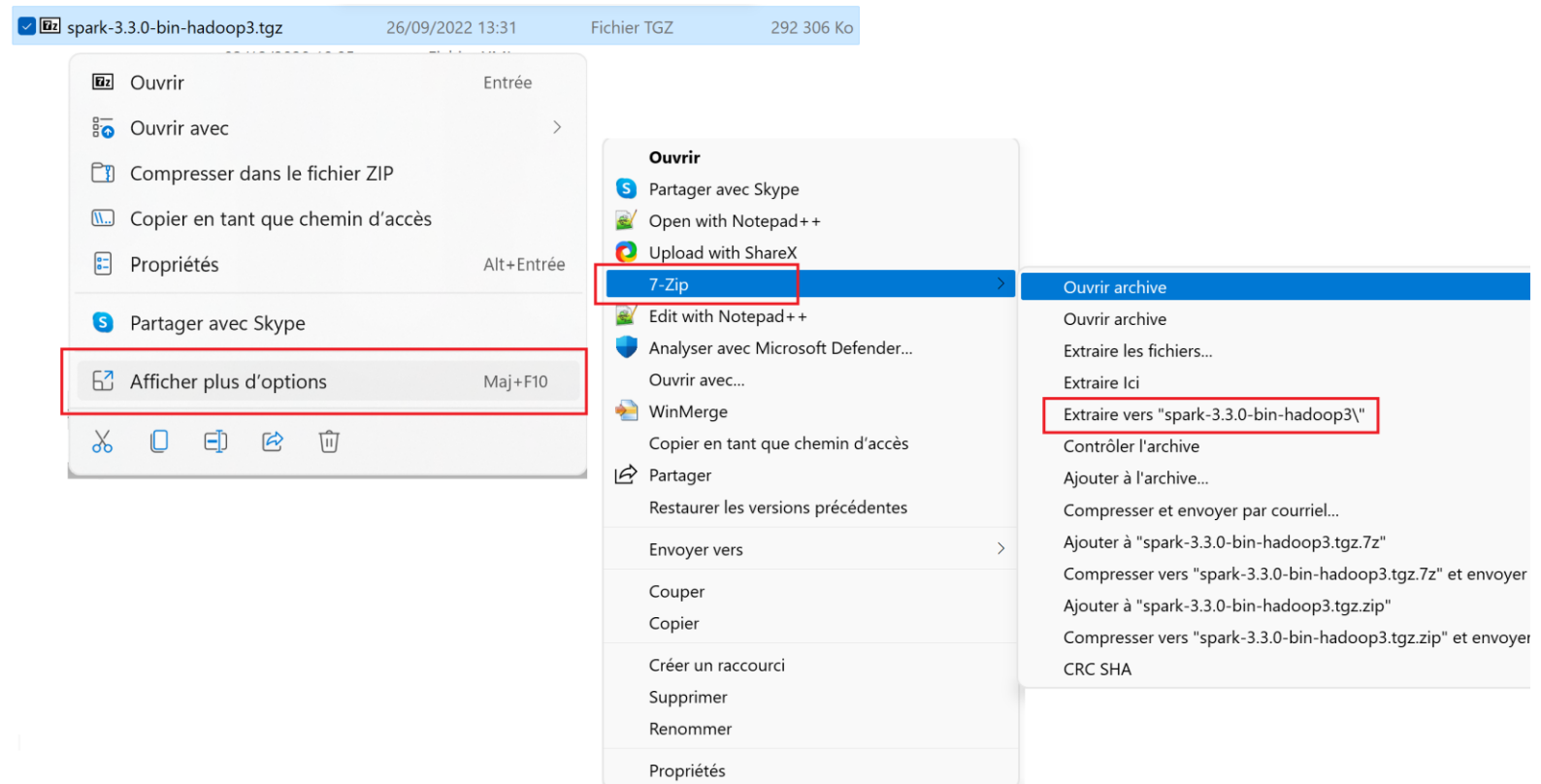
~ 285 Mo

# Step 2: Unzip


for example in C:\apps\hadoop\spark

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
















From Windows ...



# Step 2... check extract « .tar » file!

Ce PC > Windows-SSD (C:) > apps > hadoop > spark-3.3.0-bin-hadoop3				
<input type="checkbox"/> Nom	Modifié le	Type	Taille	
 spark-3.3.0-bin-hadoop3.tar	26/09/2022 13:31	Fichier TAR	329 940 Ko	

SD (C:) > apps > hadoop > spark-3.3.0

<input type="checkbox"/> 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

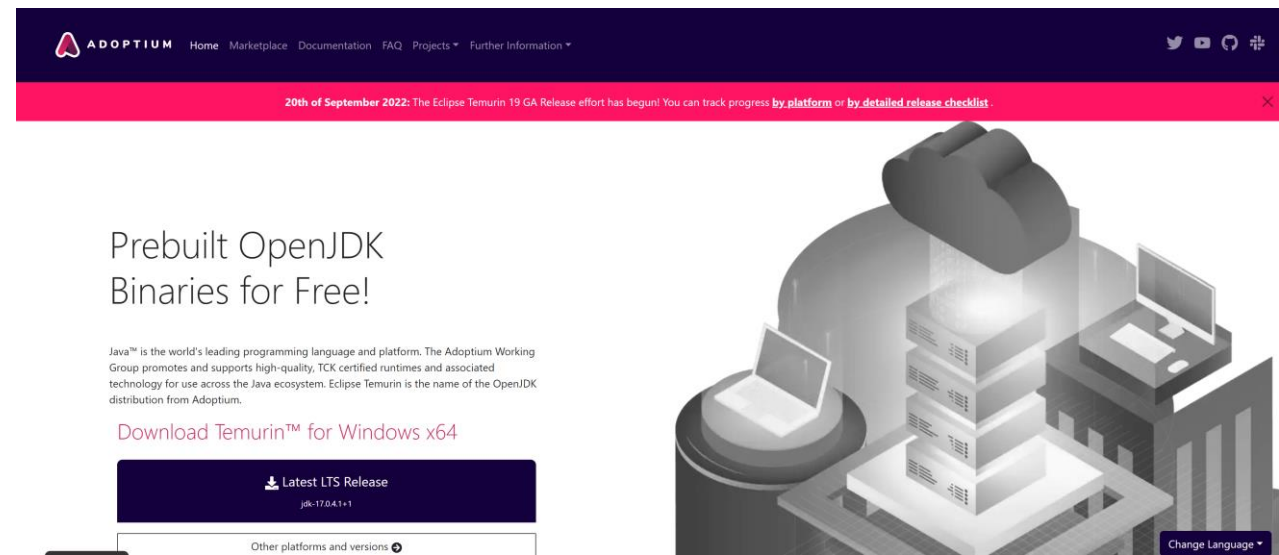
## Install JDK 17

Notice: most of Hadoop still use Java 8+linux.. But cmd error on windows?

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

Download from « adoptium » (previously « adoptOpenJdk »)

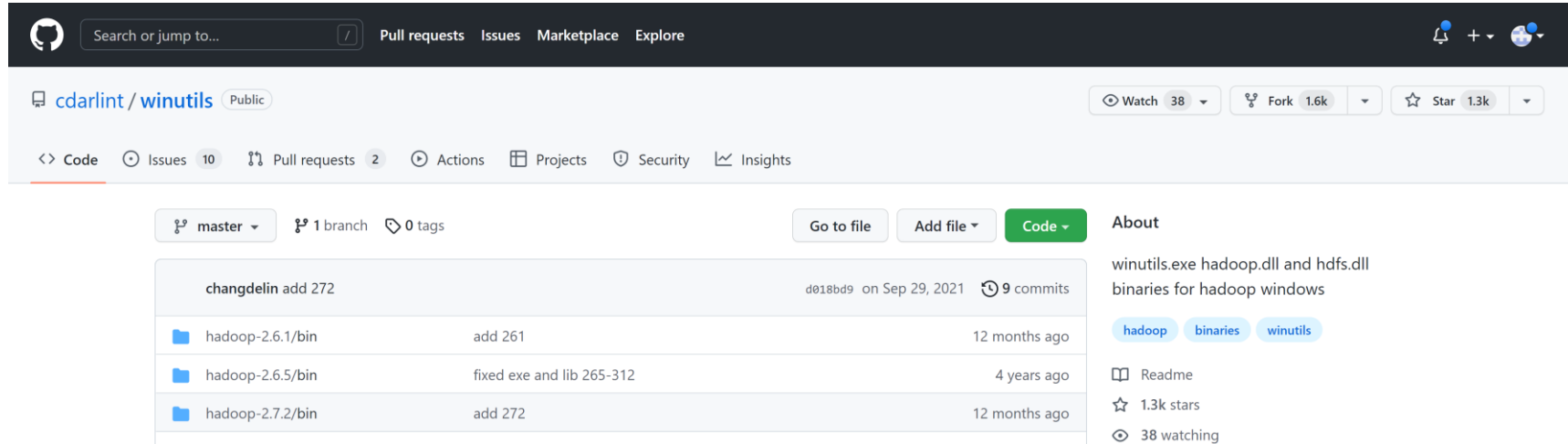
<https://adoptium.net/>



# Step 4: on Windows only

## Download « WinUtils »

<https://github.com/cdarlint/winutils>



The screenshot shows the GitHub repository page for `cdarlint/winutils`. The repository is public and has 38 watchers, 1.6k forks, and 1.3k stars. The main branch is `master`, with 1 branch and 0 tags. The repository contains a file `changdelin` with 272 additions, committed on Sep 29, 2021, with 9 commits. The file structure includes folders for `hadoop-2.6.1/bin`, `hadoop-2.6.5/bin`, and `hadoop-2.7.2/bin`. The `About` section describes the repository as containing `winutils.exe`, `hadoop.dll`, and `hdfs.dll` binaries for Hadoop on Windows. The repository also has a README, 1.3k stars, and 38 watchers.

Search or jump to... Pull requests Issues Marketplace Explore

cdarlint / winutils Public

Watch 38 Fork 1.6k Star 1.3k

<> Code Issues 10 Pull requests 2 Actions Projects Security Insights

master 1 branch 0 tags

Go to file Add file Code

changdelin add 272 d018bd9 on Sep 29, 2021 9 commits

hadoop-2.6.1/bin	add 261	12 months ago
hadoop-2.6.5/bin	fixed exe and lib 265-312	4 years ago
hadoop-2.7.2/bin	add 272	12 months ago

About

winutils.exe hadoop.dll and hdfs.dll binaries for hadoop windows

hadoop binaries winutils

Readme

1.3k stars

38 watching

# 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/ ... optionnal for this TD1, but needed for after

Re-download spark, packaging = « with user-provided Apache Hadoop »



## Download Apache Spark™

1. Choose a Spark release: 3.3.0 (Jun 16 2022) ▾

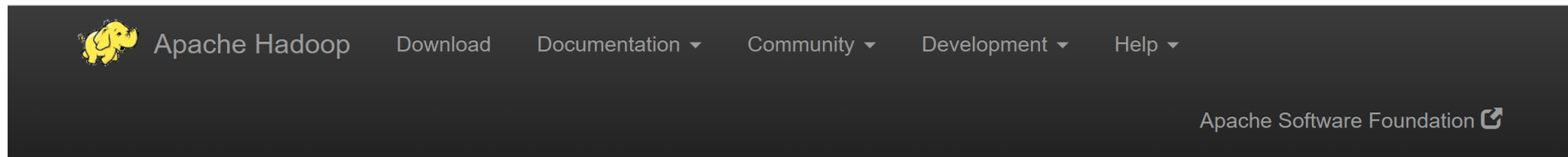
2. Choose a package type:

- Pre-built for Apache Hadoop 3.3 and later ▾
- 3. Pre-built for Apache Hadoop 3.3 and later
- Pre-built for Apache Hadoop 3.3 and later (Scala 2.13)
- 4. Pre-built for Apache Hadoop 2.7 and
- Pre-built with user-provided Apache Hadoop
- Source Code

Note that Spark 3 is pre-built with Scala 2.12 in general and Spark 3.2+ provides additional pre-built distribution with Scala 2.13.

# Step 5/ Download user-defined Hadoop

<https://hadoop.apache.org/releases.html>



## Download

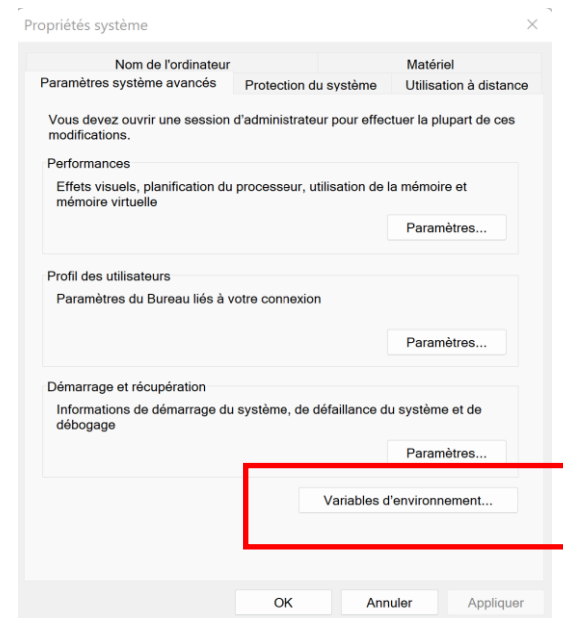
Hadoop is released as source code tarballs with corresponding binary tarballs for convenience. The downloads are distributed via mirror sites and should be checked for tampering using GPG or SHA-512.

Version	Release date	Source download	Binary download	Release notes
3.3.4	2022 Aug 8	<a href="#">source</a> ( <a href="#">checksum signature</a> )	<a href="#">binary</a> ( <a href="#">checksum signature</a> ) <a href="#">binary-aarch64</a> ( <a href="#">checksum signature</a> )	<a href="#">Announcement</a>
3.2.4	2022 Jul 22	<a href="#">source</a> ( <a href="#">checksum signature</a> )	<a href="#">binary</a> ( <a href="#">checksum signature</a> )	<a href="#">Announcement</a>
2.10.2	2022 May 31	<a href="#">source</a> ( <a href="#">checksum signature</a> )	<a href="#">binary</a> ( <a href="#">checksum signature</a> )	<a href="#">Announcement</a>

# Step 6 : Configure Environment Variables

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

Rather then Edit Windows System



# Step 6: set JAVA\_HOME , PATH, SPARK\_HOME, SPARK\_SCALA\_VERSION HADOOP\_HOME

```
setenv-spark-td1.cmd
1  @echo off
2  @echo ... executing setenv-spark-td1.cmd
3
4  REM set JAVA_HOME=C:\apps\jdk\jdk-8 ... does not work on windows
5  set JAVA_HOME=C:\apps\jdk\jdk-17.0.1.12
6  set PATH=%JAVA_HOME%\bin;%PATH%
7
8  REM currently use hadoop jars provided in spark, but could be separated
9  set HADOOP_HOME=C:\apps\hadoop\spark-3.3.0-hadoop-3.3
10
11 set SPARK_HOME=C:\apps\hadoop\spark-3.3.0-hadoop-3.3
12 set PATH=%SPARK_HOME%\bin;%PATH%
13
14 set SPARK_SCALA_VERSION=2.12
15
16 @echo .. using JAVA_HOME: %JAVA_HOME%
17 @echo .. using HADOOP_HOME: %HADOOP_HOME%
18 @echo .. using SPARK_HOME: %SPARK_HOME%
19 @echo .. using SPARK_SCALA_VERSION: %SPARK_SCALA_VERSION%
```

Step 6... If using explicit user-defined Hadoop  
... need spark-env.{sh | cmd}  
SPARK\_DIST\_CLASSPATH=\$( hadoop classpath )

```
C:\apps\hadoop>hadoop classpath
C:\apps\hadoop\conf-localfs;C:\apps\hadoop\hadoop-3.3.4\share\hadoop\common;C:\apps\hadoop\hadoop-3.3.4\share\hadoop\common\lib\*;C:\apps\hadoop\hadoop-3.3.4\share\hadoop\common\*;C:\apps\hadoop\hadoop-3.3.4\share\hadoop\hdfs;C:\apps\hadoop\hadoop-3.3.4\share\hadoop\hdfs\lib\*;C:\apps\hadoop\hadoop-3.3.4\share\hadoop\hdfs\*;C:\apps\hadoop\hadoop-3.3.4\share\hadoop\yarn;C:\apps\hadoop\hadoop-3.3.4\share\hadoop\yarn\lib\*;C:\apps\hadoop\hadoop-3.3.4\share\hadoop\yarn\*;C:\apps\hadoop\hadoop-3.3.4\share\hadoop\mapreduce\*
```

# if using explicit HADOOP\_HOME (recommended for fine-grained configuration / versions)

```
setenv-spark-td1-hadoop3.cmd
1 @echo off
2 @echo ... executing setenv-spark-td1.cmd
3
4 set JAVA_HOME=C:\apps\jdk\jdk-17.0.1.12
5 set PATH=%JAVA_HOME%\bin;%PATH%
6
7 set HADOOP_HOME=C:\apps\hadoop\hadoop-3.3.4
8 set PATH=%HADOOP_HOME%\bin;%PATH%
9
10 set SPARK_HOME=C:\apps\hadoop\spark-3.3.0
11 set PATH=%SPARK_HOME%\bin;%PATH%
12
13 set SPARK_SCALA_VERSION=2.12
14
15 REM also need to add result of "hadoop classpath" to spark
16 set SPARK_DIST_CLASSPATH=C:\apps\hadoop\conf-localfs
17 set SPARK_DIST_CLASSPATH=%SPARK_DIST_CLASSPATH%;%HADOOP_HOME%\share\hadoop\common;%HADOOP_HOME%\share\hadoop\common\lib\*;%H
18 set SPARK_DIST_CLASSPATH=%SPARK_DIST_CLASSPATH%;%HADOOP_HOME%\share\hadoop\hdfs;%HADOOP_HOME%\share\hadoop\hdfs\lib\*;%HADOOP
19 REM set SPARK_DIST_CLASSPATH=%SPARK_DIST_CLASSPATH%;%HADOOP_HOME%\share\hadoop\yarn;%HADOOP_HOME%\share\hadoop\yarn\lib\*;%H
20
21 @echo .. using JAVA_HOME: %JAVA_HOME%
22 @echo .. using HADOOP_HOME: %HADOOP_HOME%
23 @echo .. using SPARK_HOME: %SPARK_HOME%
24 @echo .. using SPARK_SCALA_VERSION: %SPARK_SCALA_VERSION%
25 @echo .. using SPARK_DIST_CLASSPATH: %SPARK_DIST_CLASSPATH%
26
```

# Step 6 ... Error if forgetting SPARK\_DIST\_CLASSPATH

```
Accès refusé.  
Erreur : impossible d'initialiser la classe principale org.apache.spark.deploy.SparkSubmit  
Caused par : java.lang.NoClassDefFoundError: org/apache/hadoop/fs/FSDataInputStream
```

NO Detailed!!! ... Internally spark-shell expand to command =>

```
C:\apps\jdk\jdk-17.0.1.12\bin\java
```

```
-cp "C:\apps\hadoop\spark-3.3.0\bin\..\conf\;C:\apps\hadoop\spark-3.3.0\jars\*;C:\apps\hadoop\conf-localfs"
```

```
"-Dscala.usejavacp=true" -Xmx1g -XX:+IgnoreUnrecognizedVMOptions
```

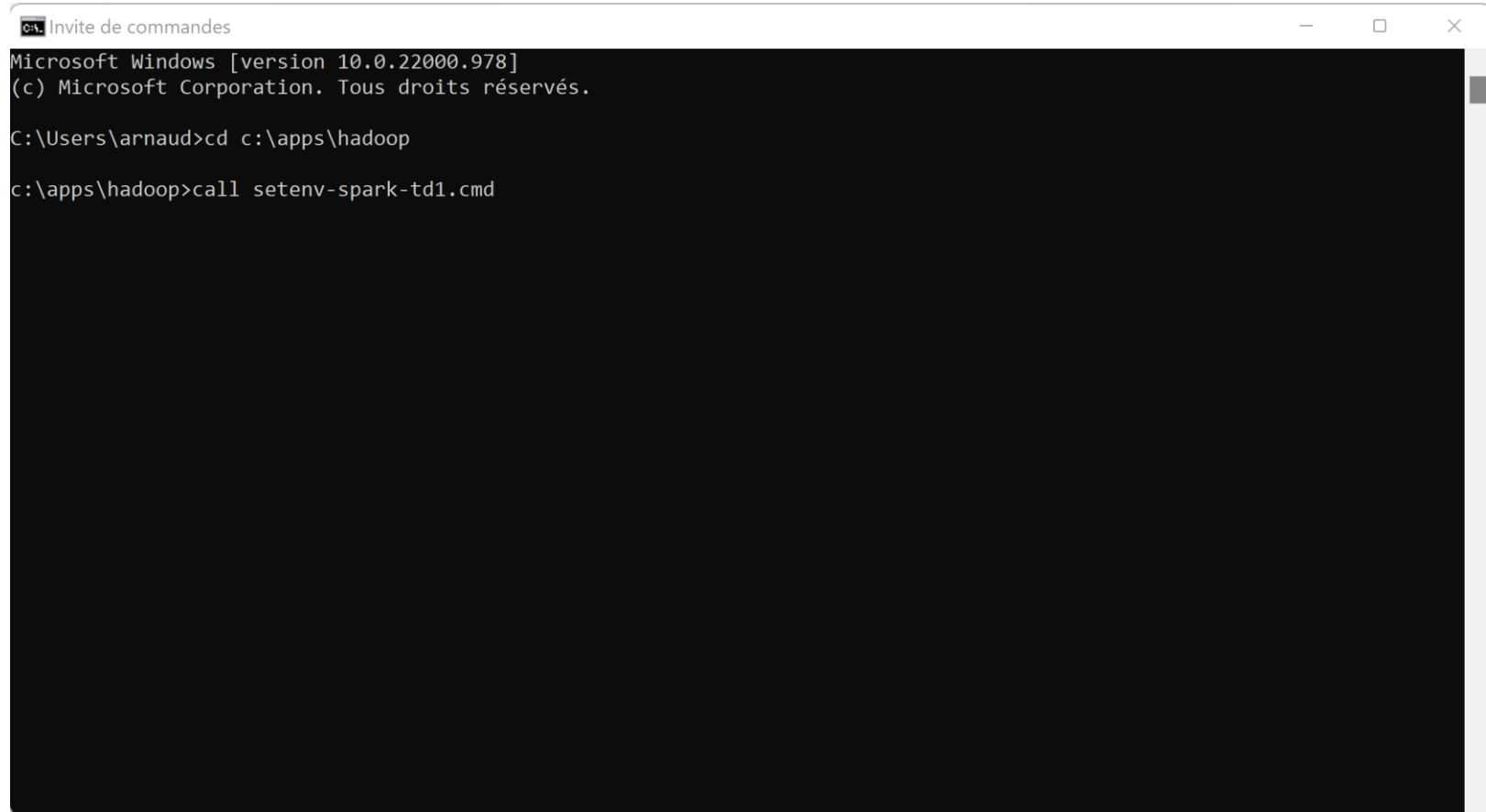
```
"--add-opens=java.base/java.lang=ALL-UNNAMED" "--add-opens=java.base/java.lang.invoke=ALL-UNNAMED" "--add-opens=
```

```
org.apache.spark.deploy.SparkSubmit
```

```
--class org.apache.spark.repl.Main --name "Spark shell" spark-shell
```

# Step 6: Sanity Checks ...

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



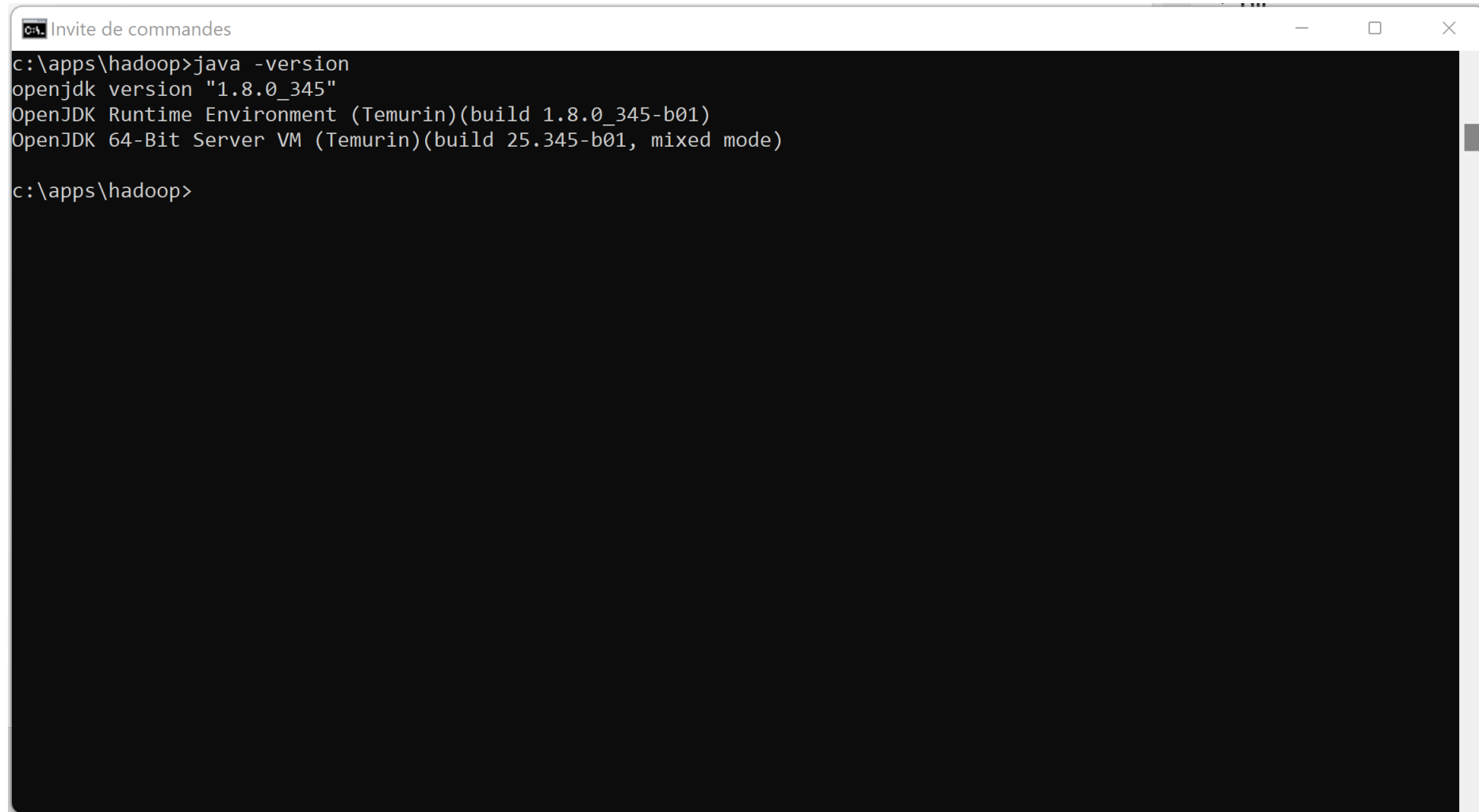
```
Invite de commandes
Microsoft Windows [version 10.0.22000.978]
(c) Microsoft Corporation. Tous droits réservés.

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

c:\apps\hadoop>call setenv-spark-td1.cmd
```



# Sanity Check: Java

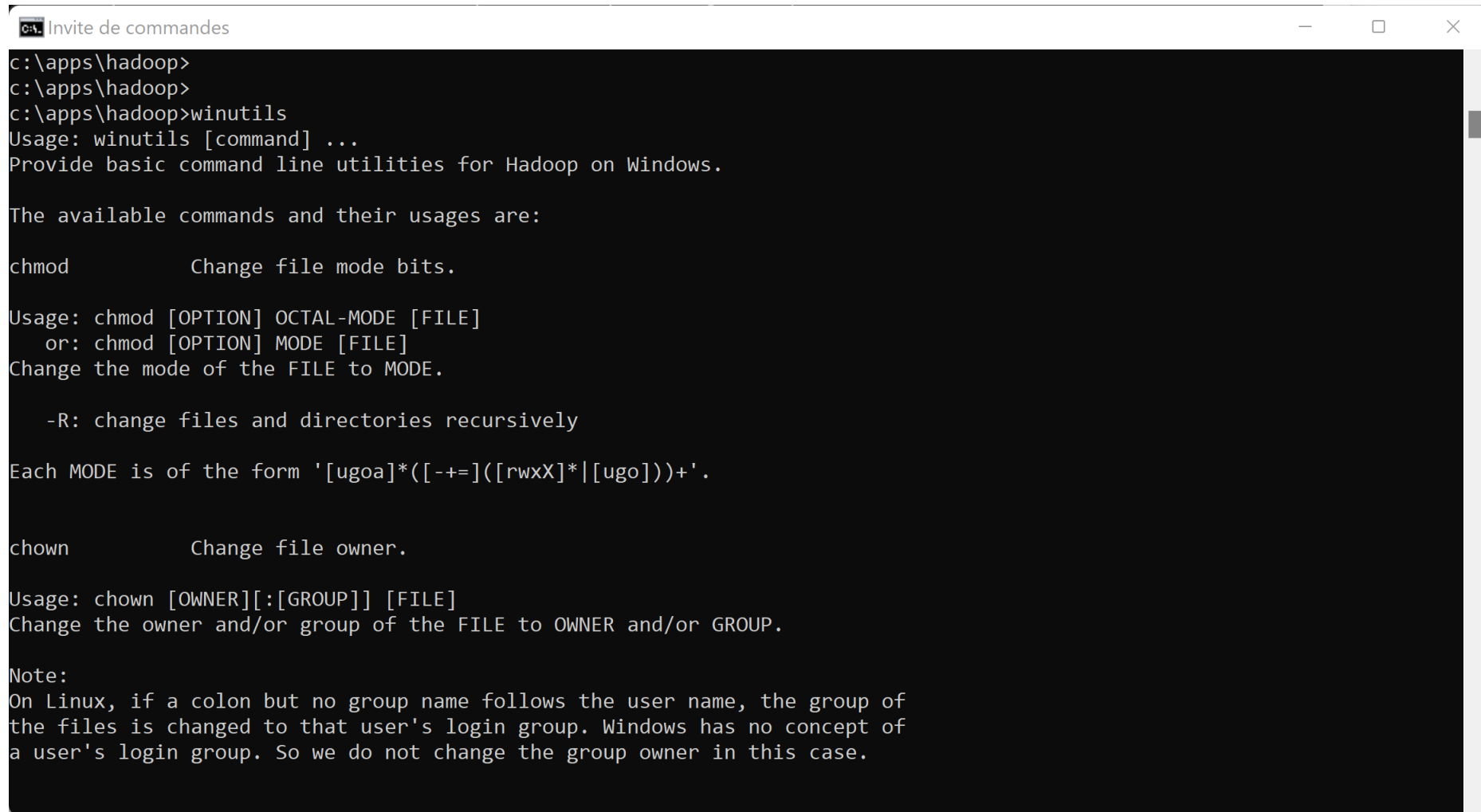


```
C:\apps\hadoop>java -version
openjdk version "1.8.0_345"
OpenJDK Runtime Environment (Temurin)(build 1.8.0_345-b01)
OpenJDK 64-Bit Server VM (Temurin)(build 25.345-b01, mixed mode)

C:\apps\hadoop>
```

The image shows a Windows Command Prompt window titled "Invite de commandes". The command prompt is at the directory "c:\apps\hadoop". The command "java -version" has been executed, resulting in the following output: "openjdk version \"1.8.0\_345\"", "OpenJDK Runtime Environment (Temurin)(build 1.8.0\_345-b01)", and "OpenJDK 64-Bit Server VM (Temurin)(build 25.345-b01, mixed mode)". The prompt is now waiting for the next command.

# Sanity check: WinUtils



```
C:\> 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 '[ugo]*([-+]=([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 »

```
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

### error when using JDK8 ... but ok when JDK17 ?!

Naively think  
missing hadoop jars  
in spark classpath ?!  
... NOT Only !!

Root cause  
was jdk version +  
Missing env var  
SPARK\_SCALA\_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 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/FSDaataInputStream
    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.FSDaataInputStream
    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>
```

# Optionnal when using custom Hadoop

## Sanity check HADOOP\_HOME + PATH

C:> hdfs dfs -ls file:///c:/data

```
C:\apps\hadoop>hdfs dfs -ls file:///c:/data
Found 6 items
d-----      - arnaud  Aucun           4096 2022-09-27 16:35 file:///c:/data/OpenData-gouv.fr
-----      1 arnaud  Aucun           450 2021-12-31 19:41 file:///c:/data/README.txt
-----      1 arnaud  Aucun           450 2021-12-31 19:41 file:///c:/data/loremIpsum.txt
-----      1 arnaud  Aucun           371 2022-09-27 14:50 file:///c:/data/sample-data1-no-header.csv
-----      1 arnaud  Aucun           405 2022-09-27 13:57 file:///c:/data/sample-data1.csv
-r--r--r--    1 arnaud  Aucun       2777803 2022-09-27 15:12 file:///c:/data/sample2.csv

C:\apps\hadoop>dir c:\data
Le volume dans le lecteur C s'appelle Windows-SSD
Le numéro de série du volume est 3EFF-0F82

Répertoire de c:\data

30/09/2022  20:26    <DIR>          .
31/12/2021  20:41             450 loremIpsum.txt
27/09/2022  16:35    <DIR>          OpenData-gouv.fr
31/12/2021  20:41             450 README.txt
27/09/2022  14:50             371 sample-data1-no-header.csv
27/09/2022  13:57             405 sample-data1.csv
27/09/2022  15:12             2 777 803 sample2.csv
              5 fichier(s)             2 779 479 octets
              2 Rép(s)  633 680 625 664 octets libres
```

# hdfs dfs -ls ...

« hdfs dfs » is unix like commands for accessing Hadoop FileSystem

example

hdfs dfs -ls /some-hdfs-dir

# => implicitly converted to «hdfs dfs -ls hdfs://<hdfsServer>/some-hdfs-dir »

hdfs dfs -ls s3:// ... for accessing Amazon S3

hdfs dfs -ls abfss:// ... for accessing Azure Storage

hdfs dfs -ls [file:///c:/local-dir](#) ... for accessing local dir ! Notice 3 slashes

# Spark-shell ... print(« Hello World »)

```
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
ADOOO_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> ...

A screenshot of a terminal window titled "Invite de commandes - spark-shell". The terminal background is black, and the text is white. It shows a series of 14 "scala>" prompts, one on each line, indicating an interactive Scala session. The window has a standard macOS title bar with a red, yellow, and green button on the left, and minimize, maximize, and close buttons on the right. A vertical scrollbar is visible on the right side of the terminal area.

# 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!

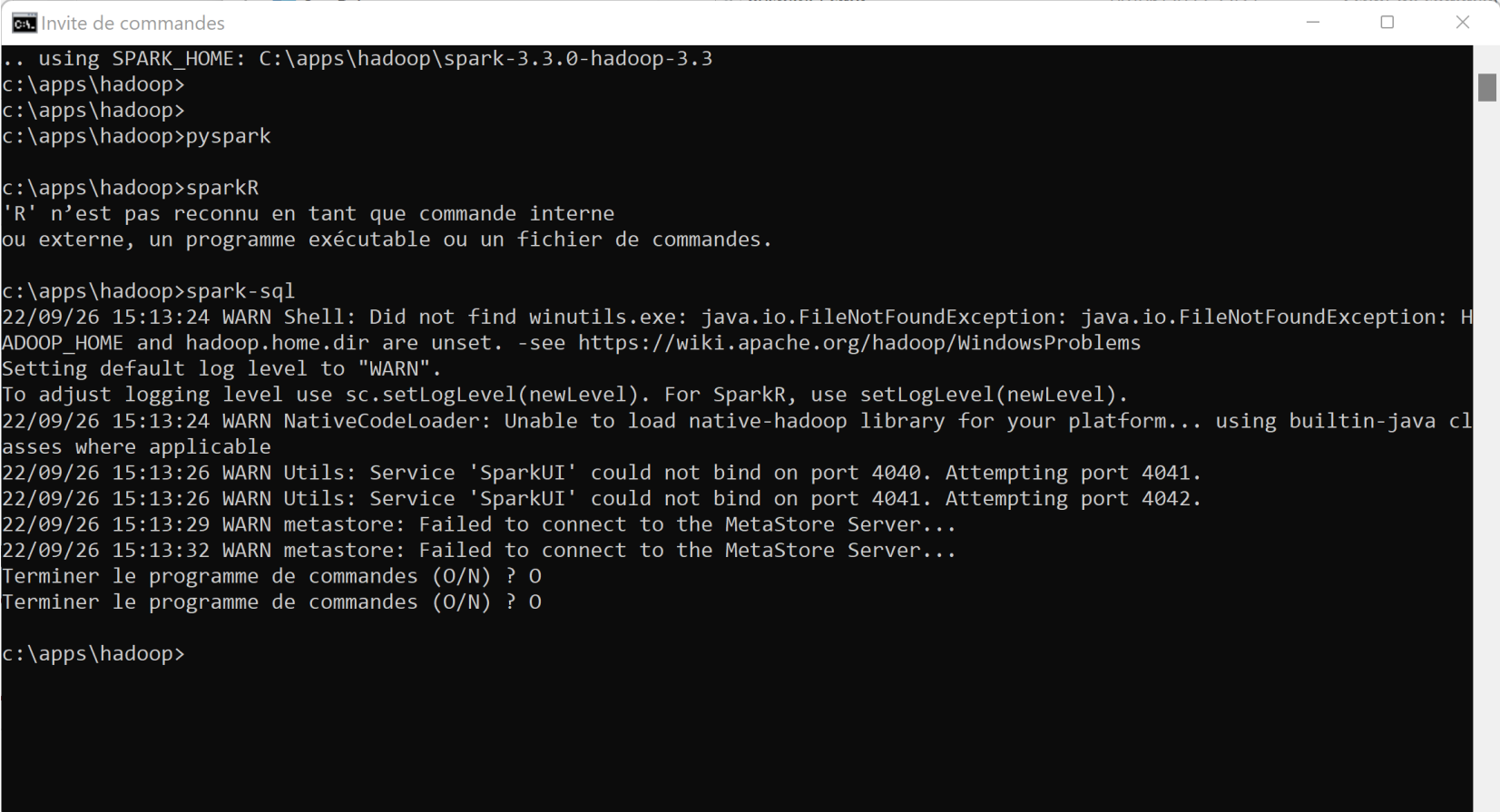
scala>
```

Not only spark-shell...

pyspark => need python exe

sparkR => need 'R' exe

spark-sql => need connect to Hive MetaStore Server



```
.. 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) ? 0
Terminer le programme de commandes (O/N) ? 0

c:\apps\hadoop>
```

# Discover built-ins commands, :help

```
C:\> 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 »

# Interactive Line Edit Jline ( ~GNU readline)

Remember the following standard shortcuts

CTRL+L : clear screen

CTRL+a / CTRL+e : jump to begin/end of line

CTRL+d : delete char

CTRL+k : kill rest of line, put in buffer

CTRL+y : paste line from buffer

Scroll Up / Down : previous/next line from history

# 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 ...

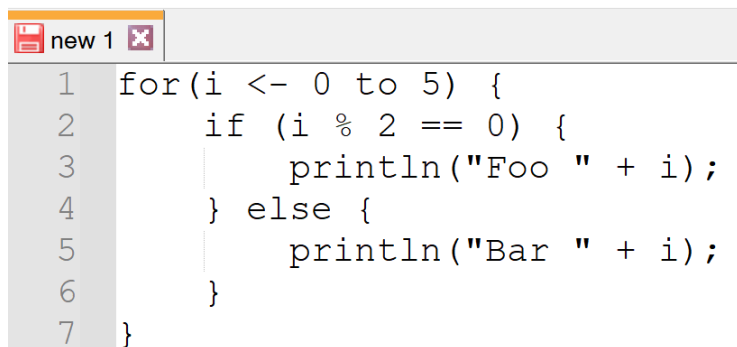
Answer:

Type « **:paste** »

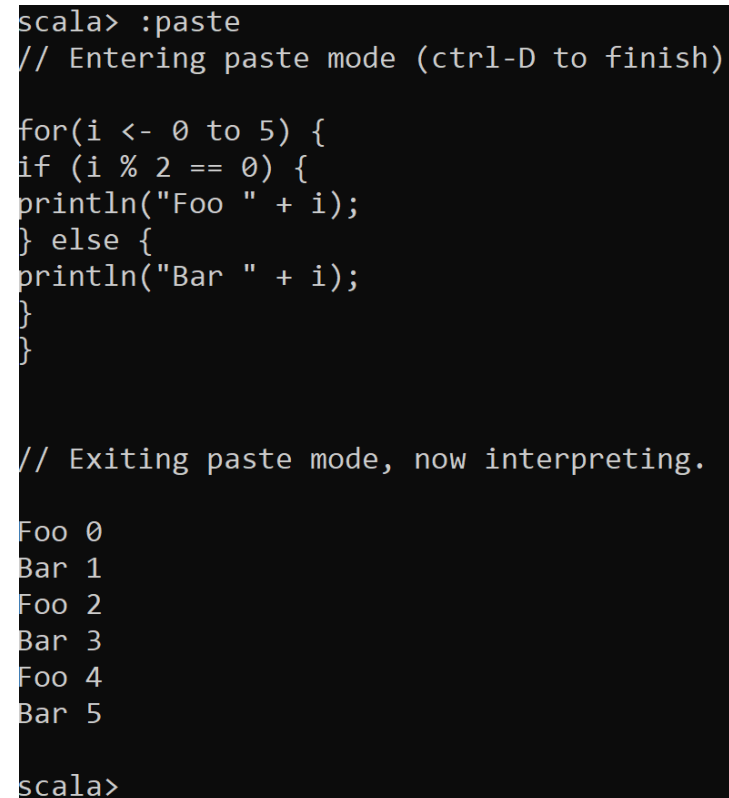
then copy& paste multiple lines... or simply type several lines with <enter>

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

# Using :paste ... Control+D



```
1 for(i <- 0 to 5) {  
2     if (i % 2 == 0) {  
3         println("Foo " + i);  
4     } else {  
5         println("Bar " + i);  
6     }  
7 }
```



```
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 x
1 val sqlQuery = """
2   SELECT a,
3     b,
4     c
5   FROM tableXYZ
6   WHERE 1=1
7     AND a LIKE '%text%'
8     AND b > 10
9   """
10 println(sqlQuery)
11 // will not work here..  spark.sql(sqlQuery).show(10)
```

```
C:\> 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

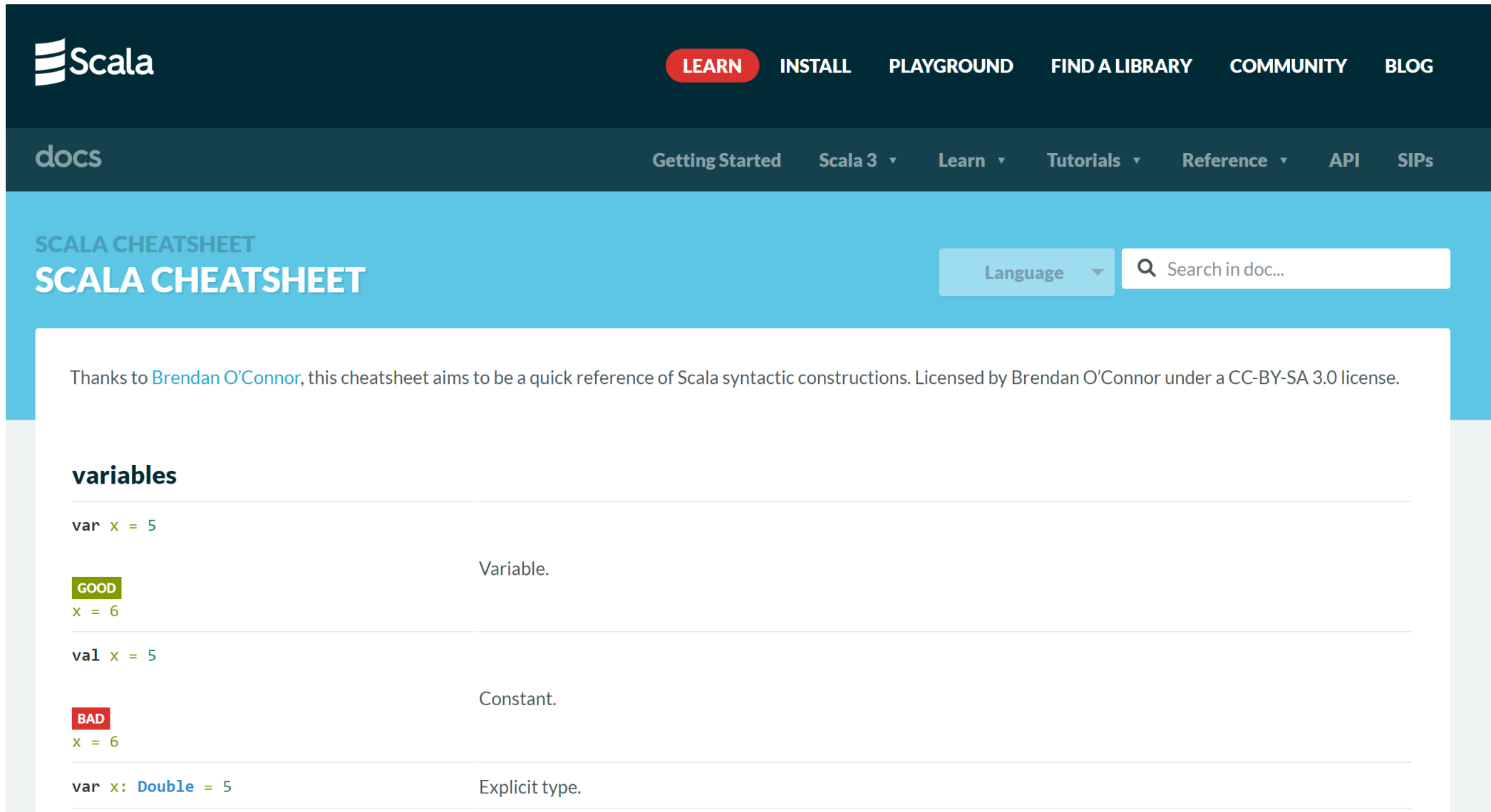
```
scala> java.lang.System.out.println("Hello plain old java from Spark-scala")  
Hello plain old java from Spark-scala
```

... 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/



The screenshot shows the Scala Cheatsheet page. At the top is a dark blue header with the Scala logo on the left and navigation links (LEARN, INSTALL, PLAYGROUND, FIND A LIBRARY, COMMUNITY, BLOG) on the right. Below this is a lighter blue bar with the word 'docs' on the left and a list of document sections (Getting Started, Scala 3, Learn, Tutorials, Reference, API, SIPs) on the right. The main content area has a light blue background. It features the title 'SCALA CHEATSHEET' in large white letters, a 'Language' dropdown menu, and a search bar. Below the title is a paragraph of text: '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.' The main content is organized into a table with two columns: code snippets and their descriptions.

SCALA CHEATSHEET	
Thanks to <a href="#">Brendan O'Connor</a> , 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.	
<b>variables</b>	
<code>var x = 5</code>	Variable.
<b>GOOD</b> <code>x = 6</code>	
<code>val x = 5</code>	Constant.
<b>BAD</b> <code>x = 6</code>	
<code>var x: Double = 5</code>	Explicit type.

5-10 minutes  
Reading



# 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

# Answer Exercise 1

```
scala> var ls = Seq( (1, false, "hello1"), (2, true, "hello 2"), (3, false, "Hello 3"))  
ls: Seq[(Int, Boolean, String)] = List((1,false,hello1), (2,true,hello 2), (3,false,Hello 3))
```

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

List has « :: » operator, not Seq !

```
var ls = List[Object]()
```

```
var ls2 = a :: ls
```

« a » and « ls » must have same generic type!

Use « List[Type] » instead of « List() »

Type is ... (Int, Boolean, String)

# Answer Exercise 2 ... the (non-idiomatic) Java Way..

```
scala> var ls = List[(Int,Boolean,String)]()  
ls: List[(Int, Boolean, String)] = List()  
  
scala> ls = (1, false, "hello " +1) :: ls  
ls: List[(Int, Boolean, String)] = List((1,false,hello 1))  
  
scala> ls = (2, true, "hello " +2) :: ls  
ls: List[(Int, Boolean, String)] = List((2,true,hello 2), (1,false,hello 1))  
  
scala> ls = (3, false, "hello " +3) :: ls  
ls: List[(Int, Boolean, String)] = List((3,false,hello 3), (2,true,hello 2), (1,false,hello 1))
```

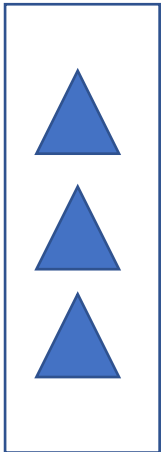


Exercise 3 .. Same using « functional » code style:  
transform the sequence (1, 2 ... 10000) into Tuples  
using « map » lambda function

# 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;} »

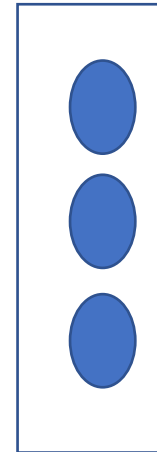
Source : List[X]



transformationFunction : X -> Y



target : List[Y]



# Answer Exercise 3

```
scala> var ls = (1 to 100000).map(i => (i, i%2 == 0, s"Hello ${i}"))  
ls: scala.collection.immutable.IndexedSeq[(Int, Boolean, String)] = Vector((1,false,Hello 1), (2,true,Hello 2), (3,false,Hello 3), (4,true,Hello 4), (5,false,Hello 5), (6,true,Hello 6), (7,false,Hello 7), (8,true,Hello 8), (9,false,Hello 9), (10,true,Hello 10), (11,false,Hello 11), (12,true,Hello 12), (13,false,Hello 13), (14,true,Hello 14), (15,false,Hello 15), (16,true,Hello 16), (17,false,Hello 17), (18,true,Hello 18), (19,false,Hello 19), (20,true,Hello 20), (21,false,Hello 21), (22,true,Hello 22), (23,false,Hello 23), (24,true,Hello 24), (25,false,Hello 25), (26,true,Hello 26), (27,false,Hello 27), (28,true,Hello 28), (29,false,Hello 29), (30,true,Hello 30), (31,false,Hello 31), (32,true,Hello 32), (33,false,Hello 33), (34,true,Hello 34), (35,false,Hello 35), (36,true,Hello 36), (3...
```

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

# Answer Exercise 4

```
scala> ls.size
res44: Int = 100000

scala> ls(0)
res45: (Int, Boolean, String) = (1,false,Hello 1)

scala> ls(0)._1
res46: Int = 1

scala> ls(0)._2
res47: Boolean = false

scala> ls(0)._3
res48: String = Hello 1
```

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

# Answer Exercise 5

```
scala> var ls = (1 to 100000).map(i => (i, i%2 == 0, s"Hello ${i}"))
ls: scala.collection.immutable.IndexedSeq[(Int, Boolean, String)] = Vector((1,false,Hello 1), (2,true,Hello 2), (3,false,Hello 3), (4,true,Hello 4), (5,false,Hello 5), (6,true,Hello 6), (7,false,Hello 7), (8,true,Hello 8), (9,false,Hello 9), (10,true,Hello 10), (11,false,Hello 11), (12,true,Hello 12), (13,false,Hello 13), (14,true,Hello 14), (15,false,Hello 15), (16,true,Hello 16), (17,false,Hello 17), (18,true,Hello 18), (19,false,Hello 19), (20,true,Hello 20), (21,false,Hello 21), (22,true,Hello 22), (23,false,Hello 23), (24,true,Hello 24), (25,false,Hello 25), (26,true,Hello 26), (27,false,Hello 27), (28,true,Hello 28), (29,false,Hello 29), (30,true,Hello 30), (31,false,Hello 31), (32,true,Hello 32), (33,false,Hello 33), (34,true,Hello 34), (35,false,Hello 35), (36,true,Hello 36), (3...

scala> val ds = spark.createDataset(ls)
ds: org.apache.spark.sql.Dataset[(Int, Boolean, String)] = [_1: int, _2: boolean ... 1 more field]
```

### Equivalent 1-liner solution :

```
scala> val dsDirect = spark.createDataset( (1 to 100000).map(i => (i, i%2 == 0, s"Hello ${i}")) )
dsDirect: org.apache.spark.sql.Dataset[(Int, Boolean, String)] = [_1: int, _2: boolean ... 1 more field]
```



## 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.

# Answer Exercise 6

```
scala> ds.count  
res49: Long = 100000
```

```
scala> ds.show(2)  
+---+-----+-----+  
| _1|    _2|    _3|  
+---+-----+-----+  
|  1|false|Hello 1|  
|  2| true|Hello 2|  
+---+-----+-----+  
only showing top 2 rows
```

```
scala> ds.take(1)(0)  
res57: (Int, Boolean, String) = (1,false,Hello 1)
```

```
scala> ds.take(1)(0)._1  
res58: Int = 1
```

```
scala> ds.take(1)(0)._2  
res59: Boolean = false
```

```
scala> ds.take(1)(0)._3  
res60: String = Hello 1
```

By default... « .show() » => same as « .show(20) »  
.. => same as « .show(20, true) »

```
scala> ds.show()  
+---+-----+-----+  
| _1|    _2|    _3|  
+---+-----+-----+  
|  1|false|Hello 1|  
|  2| true|Hello 2|  
|  3|false|Hello 3|  
|  4| true|Hello 4|  
|  5|false|Hello 5|  
|  6| true|Hello 6|  
|  7|false|Hello 7|  
|  8| true|Hello 8|  
|  9|false|Hello 9|  
| 10| true|Hello 10|  
| 11|false|Hello 11|  
| 12| true|Hello 12|  
| 13|false|Hello 13|  
| 14| true|Hello 14|  
| 15|false|Hello 15|  
| 16| true|Hello 16|  
| 17|false|Hello 17|  
| 18| true|Hello 18|  
| 19|false|Hello 19|  
| 20| true|Hello 20|  
+---+-----+-----+  
only showing top 20 rows
```

# 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

# Answer Exercise 7

```
scala> val ds2 = ds.withColumnRenamed("_1", "id")
ds2: org.apache.spark.sql.DataFrame = [id: int, _2: boolean ... 1 more field]

scala> val ds3 = ds2.withColumnRenamed("_2", "even")
ds3: org.apache.spark.sql.DataFrame = [id: int, even: boolean ... 1 more field]

scala> val ds4 = ds3.withColumnRenamed("_3", "text")
ds4: org.apache.spark.sql.DataFrame = [id: int, even: boolean ... 1 more field]
```

Ds ... is unmodified ... still has \_1, \_2, \_3 columns

```
scala> ds
res62: org.apache.spark.sql.Dataset[(Int, Boolean, String)] = [_1: int, _2: boolean ... 1 more field]
```

More idiomatic: cascade methods

```
scala> val dsRenamed = ds.withColumnRenamed("_1", "id").withColumnRenamed("_2", "even").withColumnRenamed("_3", "text")
dsRenamed: org.apache.spark.sql.DataFrame = [id: int, even: boolean ... 1 more field]
```

# 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.Dataset[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



# Answer Exercise 8

```
scala> val dsFilter = ds4.filter(ds4("id") >= 100)
dsFilter: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [id: int, even: boolean ... 1 more field]

scala> dsFilter.count
res63: Long = 99901
```

# Exercise 8

Filter rows where

« id » is « >= 100 »

AND

«text » contains substring « 12 »

# Hint Exercise 8

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

You can manipulate columns objects almost as values in scala

# Answer Exercise 8

```
scala> val dsFilter = ds4.filter( ds4("id") >= 100 && ds4("text").contains("12") )
dsFilter: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [id: int, even: boolean ... 1 more field]

scala> dsFilter.show(10)
+---+-----+-----+
| id| even|    text|
+---+-----+-----+
|112| true|Hello 112|
|120| true|Hello 120|
|121|false|Hello 121|
|122| true|Hello 122|
|123|false|Hello 123|
|124| true|Hello 124|
|125|false|Hello 125|
|126| true|Hello 126|
|127|false|Hello 127|
|128| true|Hello 128|
+---+-----+-----+
only showing top 10 rows
```

# Alternative Answer Exercise 8

```
scala> val dsFilter = ds4.filter( $"id" >= 100 && $"text".contains("12") )
dsFilter: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [id: int, even: boolean ... 1 more field]

scala> dsFilter.show(10)
+---+-----+-----+
| id| even|    text|
+---+-----+-----+
|112| true|Hello 112|
|120| true|Hello 120|
|121|false|Hello 121|
|122| true|Hello 122|
|123|false|Hello 123|
|124| true|Hello 124|
|125|false|Hello 125|
|126| true|Hello 126|
|127|false|Hello 127|
|128| true|Hello 128|
+---+-----+-----+
only showing top 10 rows
```

# Exercise 9

Redo exercise 7,8 using filter by SQL text expression

# Answer Exercise 9

```
scala> val dsFilter = ds4.filter( "id >= 100 AND text LIKE '%12%' " )
dsFilter: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [id: int, even: boolean ... 1 more field]

scala> dsFilter.count
res71: Long = 3969

scala> dsFilter.show(10)
+---+-----+-----+
| id| even|      text|
+---+-----+-----+
|112| true|Hello 112|
|120| true|Hello 120|
|121|false|Hello 121|
|122| true|Hello 122|
|123|false|Hello 123|
|124| true|Hello 124|
|125|false|Hello 125|
|126| true|Hello 126|
|127|false|Hello 127|
|128| true|Hello 128|
+---+-----+-----+
only showing top 10 rows
```

# 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 »)` »

# Answer Exercise 10

```
scala> val dsFilter = ds4.filter(row => ( row.getAs[Int]("id") >= 100 && row.getAs[String]("text").contains("12") ) )
dsFilter: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [id: int, even: boolean ... 1 more field]
```

```
scala> dsFilter.count
res74: Long = 3969
```

```
scala> dsFilter.show(10)
```

```
+---+-----+-----+
| id| even|    text|
+---+-----+-----+
|112| true|Hello 112|
|120| true|Hello 120|
|121|false|Hello 121|
|122| true|Hello 122|
|123|false|Hello 123|
|124| true|Hello 124|
|125|false|Hello 125|
|126| true|Hello 126|
|127|false|Hello 127|
|128| true|Hello 128|
+---+-----+-----+
```

```
only showing top 10 rows
```

# 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;  
    }  
}
```

# Answer Exercise 11

```
scala> case class CustomBean(id: Int, even: Boolean, text: String) {}  
defined class CustomBean  
  
scala> val b = CustomBean(1, false, "Hello 1")  
b: CustomBean = CustomBean(1,false,Hello 1)  
  
scala> val ls=Seq(b, YourCustomBean(2, true, "Hello 2"), YourCustomBean(3, false, "Hello 3"))  
ls: Seq[Product with Serializable] = List(CustomBean(1,false,Hello 1), YourCustomBean(2,true,Hello 2), YourCustomBean(3,  
false,Hello 3))
```

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

<https://spark.apache.org/docs/latest/sql-getting-started.html#creating-datasets>

- 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' »

# Answer Exercise 12

```
scala> val beansDs = spark.createDataset( (1 to 100000).map( i => CustomBean(i, i%2==0, s"Hello ${i}") ) )
beansDs: org.apache.spark.sql.Dataset[CustomBean] = [id: int, even: boolean ... 1 more field]
```

```
scala> beansDs.show(2)
```

id	even	text
1	false	Hello 1
2	true	Hello 2

```
only showing top 2 rows
```

```
scala> beansDs.filter( $"id" > 100 && $"text".contains("12") ).show(4)
```

id	even	text
112	true	Hello 112
120	true	Hello 120
121	false	Hello 121
122	true	Hello 122

```
only showing top 4 rows
```

# 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

# Answer Exercise 13

Type, then conversion:

```
scala> ds
res81: org.apache.spark.sql.Dataset[(Int, Boolean, String)] = [_1: int, _2: boolean ... 1 more field]

scala> val df = ds.toDF
df: org.apache.spark.sql.DataFrame = [_1: int, _2: boolean ... 1 more field]
```

After a filter() operation, Spark has already entered the magic DataFrame world !

```
scala> dsFilter
res82: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [id: int, even: boolean ... 1 more field]
```

Converting CustomBean->DataFrame... same as Tuple -> DataFrame

```
scala> beansDs
res86: org.apache.spark.sql.Dataset[CustomBean] = [id: int, even: boolean ... 1 more field]

scala> val beansDF = beansDs.toDF
beansDF: org.apache.spark.sql.DataFrame = [id: int, even: boolean ... 1 more field]
```



# 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 ]`

c/ DataFrame .... `Dataset[Row]` ... means YourCustomBean is the preferred spark one: « Row »

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

# Answer Exercise 14

Scala collection are only on the Spark **driver** side  
... you may not be able to instantiate 500 Giga objects of these

Dataset (and DataFrame) are distributed over spark **executors**

To send data over the network, they are serialized then un-serialized.  
So it is still real « Tuple » or « CustomBean » object pointers, but on spark executors.

Dataset[Row] is special ... Spark internally is able to handle native memory / swap storage.

# 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 ») }` ?

# Answer Exercise 15

On scala Tuple => you use `_1, _2, ...` field names

On your custom bean => you use your class fieldName (or getter)

```
Example: case class YourCustomBean(id: Int, text: String, foo: String) {}  
val bean = YourCustomBean(123, « text », « foo »);  
bean.foo
```

In SQL expression, you use the column name, example `« _1 »`, `« id »`, `« text »`

On spark.sql.Row => you use `org.apache.spark.sql.Row` methods !!

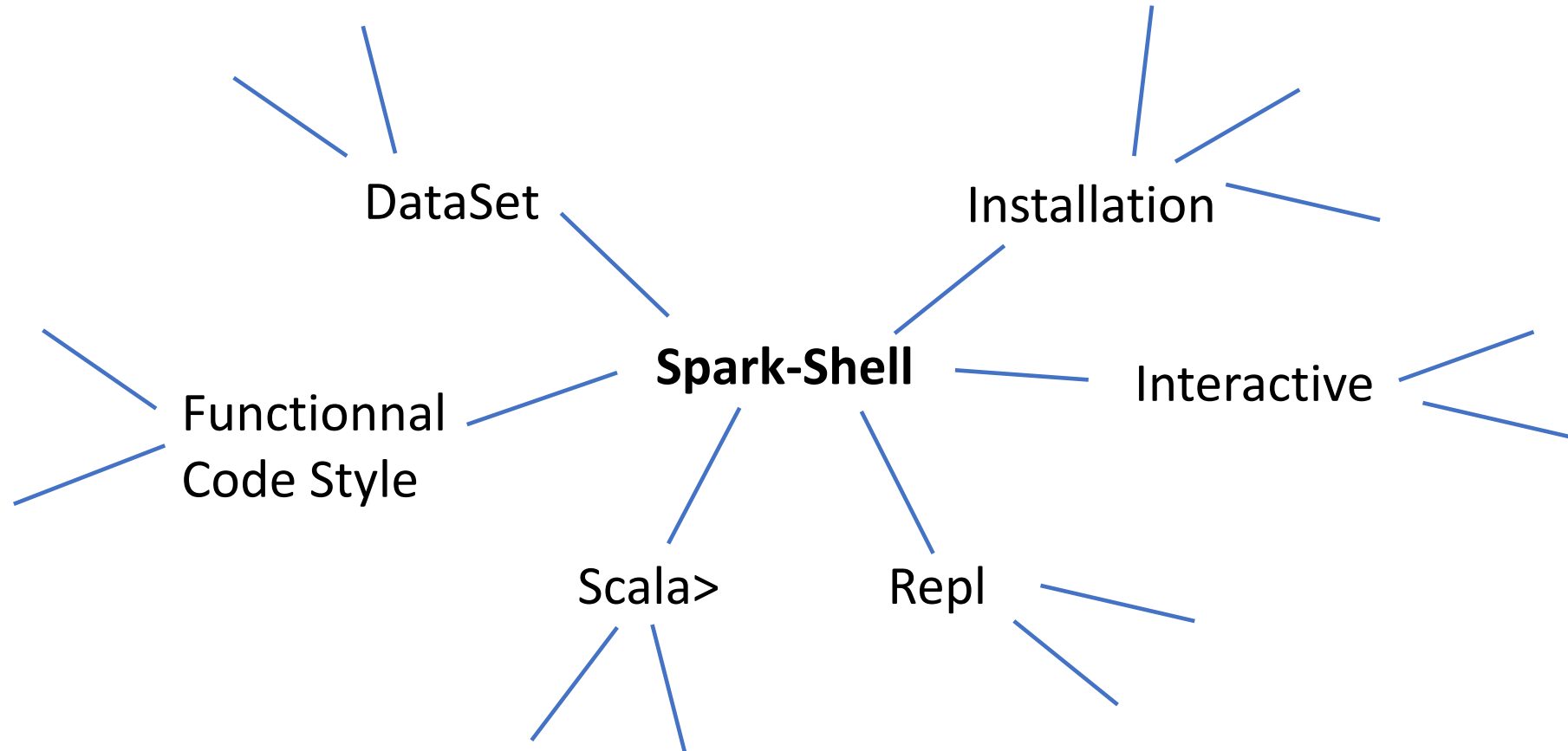
```
Example: row.getAs[Int](« foo »)
```

# 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`



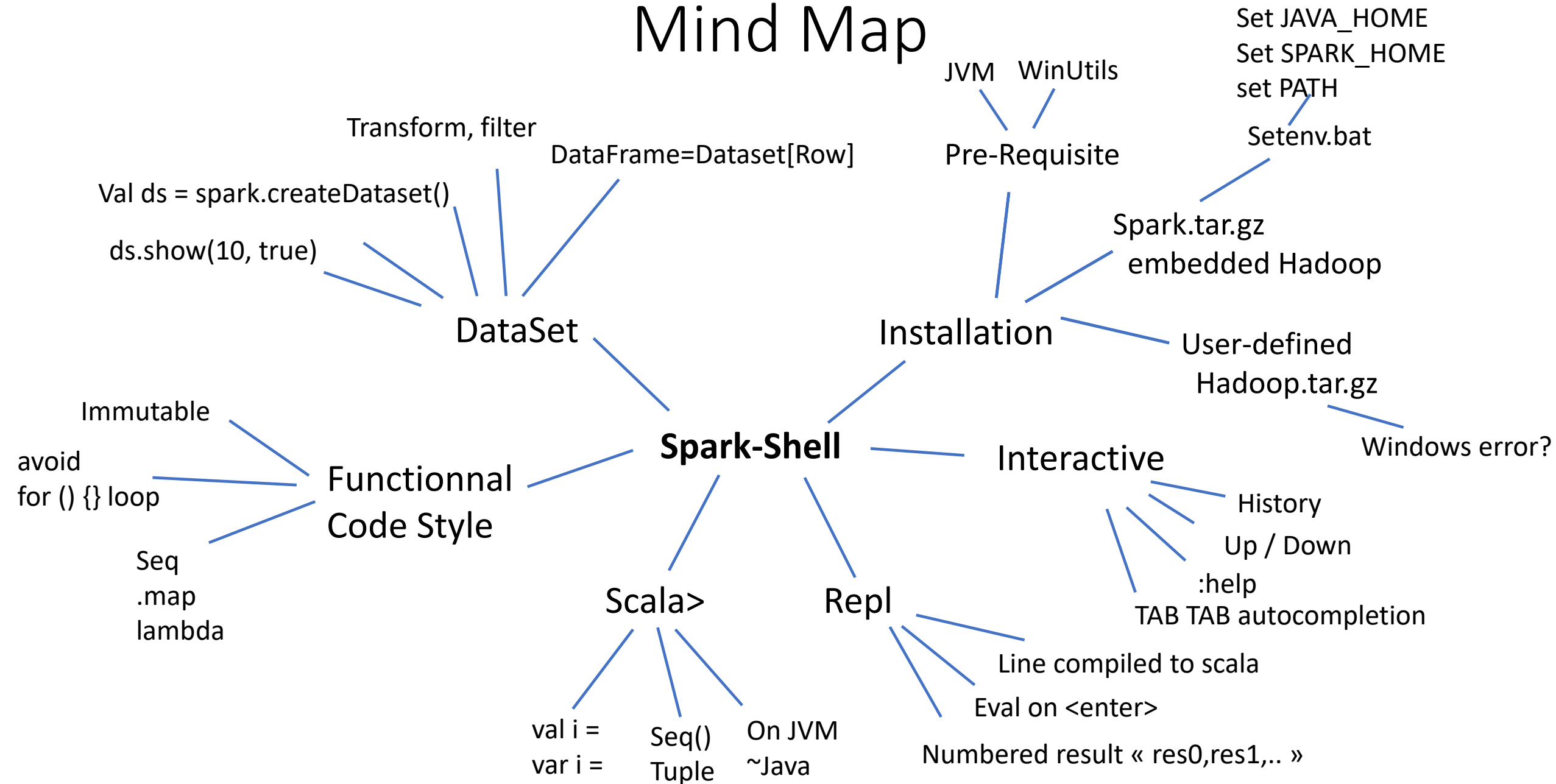
4/ Execute basic spark commands on DataSets

Take-Away

What You learned ?



# Mind Map



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
- ...