

R Programming under Hadoop

Installation of the Hadoop framework (Cloudera)
Installation and configuration of R and RStudio Server
Mapreduce programming in R
Accessing files in HDFS using R



The aim of this tutorial is to show the programming of the famous “word count” algorithm from a set of files stored in HDFS file system.

The "word count" is a state-of-the-art example for the programming under Hadoop. It is described everywhere on the web. But, unfortunately, the tutorials which describe the task are often not reproducible. The dataset are not available. The whole process, including the installation of the Hadoop framework, are not described. We do not know how to access to the files stored in the HDFS file system. In short, we cannot run programs and understand in details how they work.

In this tutorial, we describe the whole process. We detail first the installation of a virtual machine which contains a single-node Hadoop cluster. Then we show how to install R and RStudio Server which allows us to write and run a program. Last, we write some programs based on the mapreduce scheme.

The steps, and therefore the source of errors, are numerous. We will use many screenshots to actually understand each operation. This is the reason of this unusual presentation format for a tutorial.



Steps

1. Installation of the single-node Hadoop cluster
2. Installation of R and RStudio Server
3. MapReduce programming in R
4. Accessing files in HDFS using R
5. References



Installation of a virtual machine containing a single-node Apache Hadoop cluster

We can install the Hadoop framework directly on an existing machine. But the operation remains delicate, requiring a certain knowledge about operating system (e.g. [installation on Ubuntu Linux](#)).

Fortunately, some editors offer turnkey solutions with the creation of a virtual machine - as a single-node cluster - including already correctly configured and functional Hadoop framework. We will use the QuickStart VM of Cloudera based on the CentOS operating system in this tutorial.



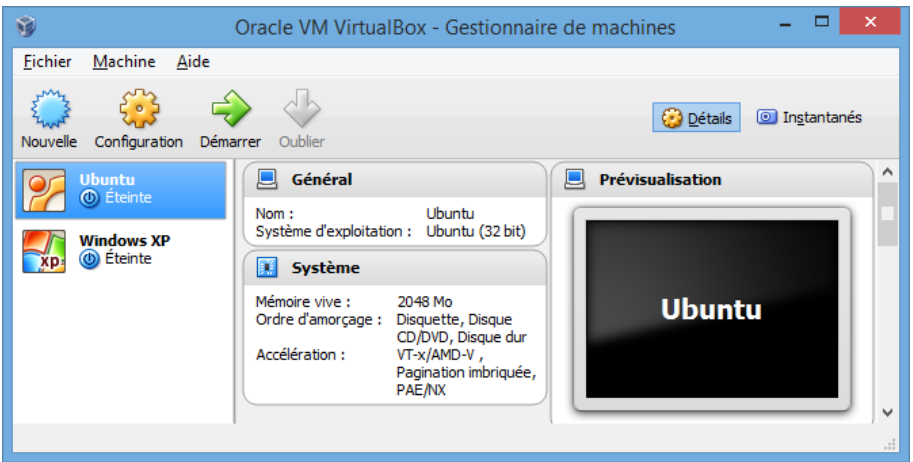
Installation of VirtualBox

VirtualBox is an open source tool.

VirtualBox is a "virtualization" tool i.e. it is a host application that allows to host and run an operating system as a software (guest operating system). Thus, we have a fully functional virtual machine.



We can incorporate multiple virtual machines.



In this screenshot, we see that I have already installed two guests machines into VirtualBox: one running under Ubuntu and the other under Windows XP.

Downloading Cloudera (1/2)

The QuickStart VMs contain a single-node Apache Hadoop cluster, complete with example data, queries, scripts, and Cloudera Manager to manage your cluster.

http://www.cloudera.com/content/cloudera/en/downloads/quickstart_vms/cdh-5-3-x.html



The aim is to create a single-node Hadoop cluster running under a **virtual machine (VM)**.



Downloading Cloudera (2/2)

The virtual machine runs CentOS 6.4

Support Developers Sign In Register Contact Us Downloads

Contact Sales: 866-843-7207

Search

cloudera

PRODUCTS & SERVICES TRAINING SOLUTIONS CUSTOMERS PARTNERS RESOURCES BLOGS

Downloads > QuickStart VMs | Cloudera Manager | CDH | Connectors | CDH4 Components

QuickStart VMs for CDH 5.3.x

A Single-Node Hadoop Cluster and Examples for Easy Learning!

Start testing Hadoop with Cloudera's QuickStart VMs. The QuickStart VMs contain a single-node Apache Hadoop cluster, complete with example data, queries, scripts, and Cloudera Manager to manage your cluster.

The VMs run CentOS 6.4 and are available for VMware, VirtualBox, and KVM.

All require a 64-bit host OS.

Version: QuickStart VM with CDH 5.3.x

Please Note: Cloudera QuickStart VMs are for demo purposes only and are not to be used as a starting point for clusters.

Create New Virtual Machine

Welcome to the New Virtual Machine Wizard!

This wizard will guide you through the steps that are necessary to create a new virtual machine for VirtualBox.

Use the Continue button to go to the next page of the wizard and the Go Back button to return to the previous page. You can also press Esc if you want to cancel the execution of this wizard.

Go Back Continue

Download System Requirements Installed Products Getting Started

Platform	Release Date	Hash	Bits
VMware	December 23, 2014	SHA1: d452977b6f6caa5f9ce9c9066fc866c2fe95f818	Download for VMware >
KVM	December 23, 2014	SHA1: f9e35b72c25aab95b2d1939e75c588031e56cdfd	Download for KVM >
VirtualBox	December 23, 2014	SHA1: 7f19f078b7bd33f842a895cb0a44143a6ad6776c	Download for VirtualBox >

We download the machine designed for VirtualBox in this tutorial.

Here is the archive file when it was downloaded.




Nom	Modifié le	Type	Taille
cloudera-quickstart-vm-5.3.0-0-virtualbox.7z	18/03/2015 16:35	Fichier 7Z	3 780 637 Ko

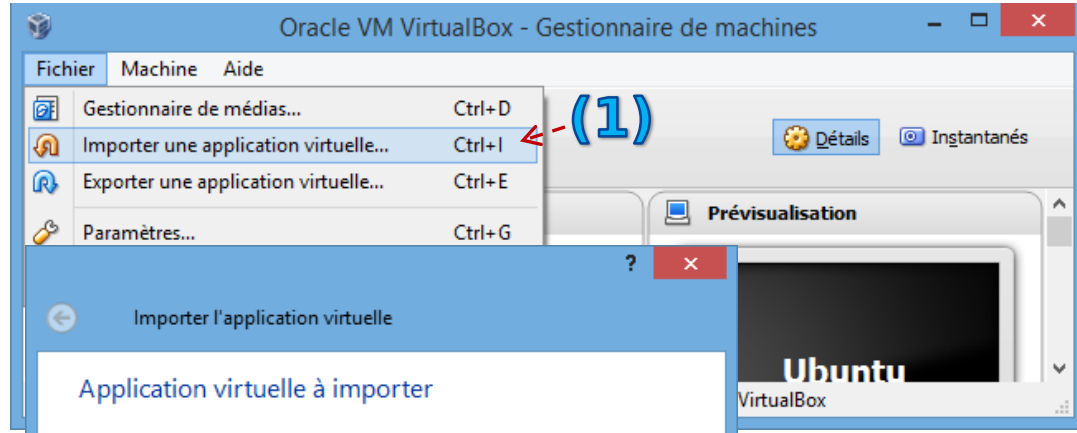


Importing the virtual machine into VirtualBox (1/3)

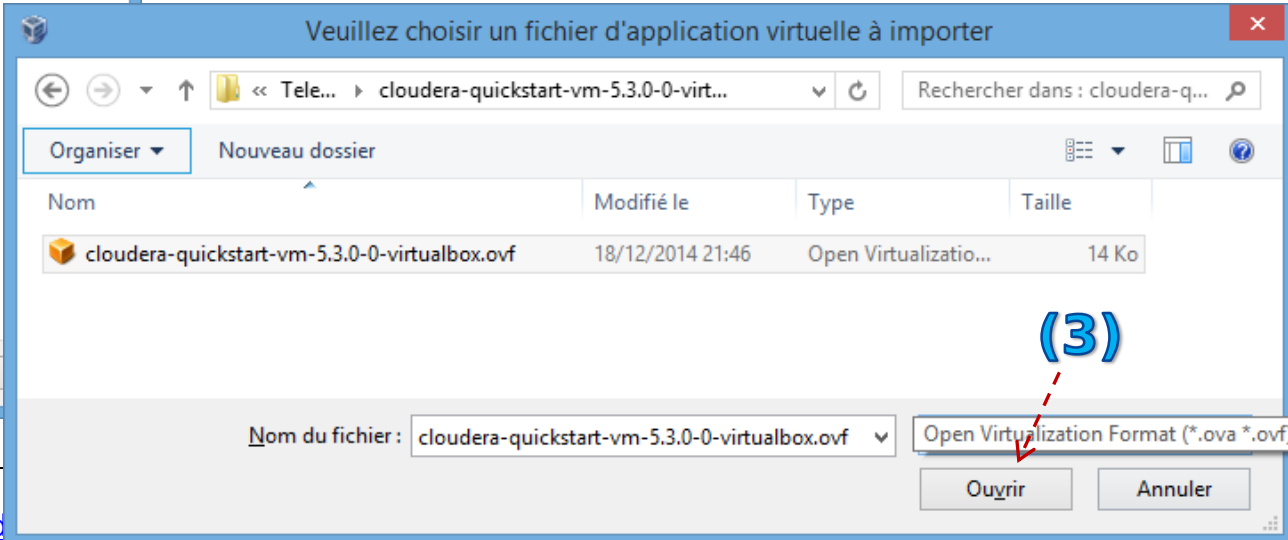
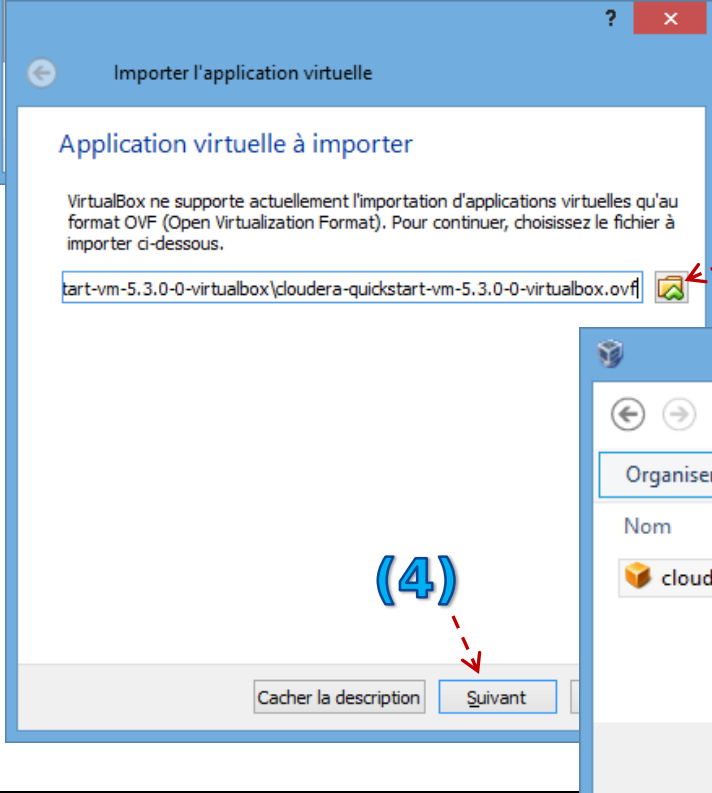
The archive contains 2 files



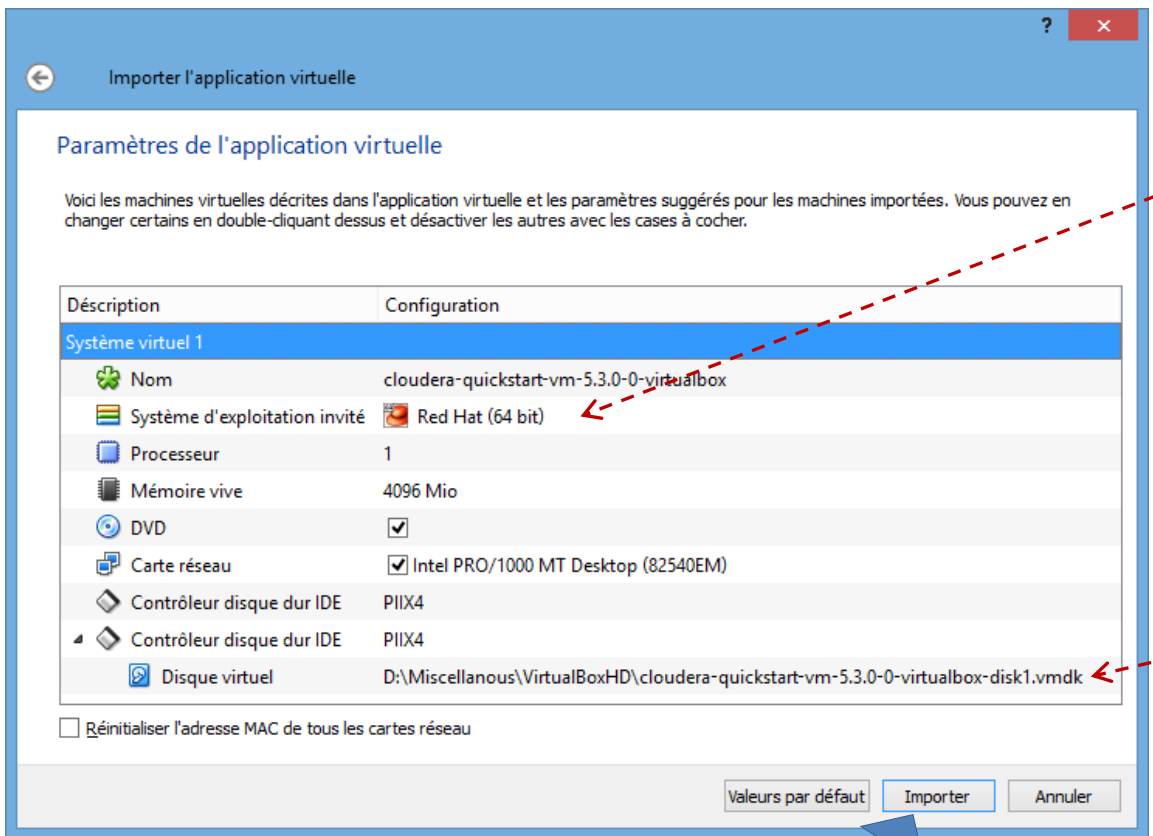
 cloudera-quickstart-vm-5.3.0-0-virtualbox.ovf	18/12/2014 21:46	Open Virtualizatio...	14 Ko
 cloudera-quickstart-vm-5.3.0-0-virtualbox-disk1.vmdk	18/12/2014 21:51	Virtual Machine Di...	3 837 156 Ko



There are several steps in the import process.



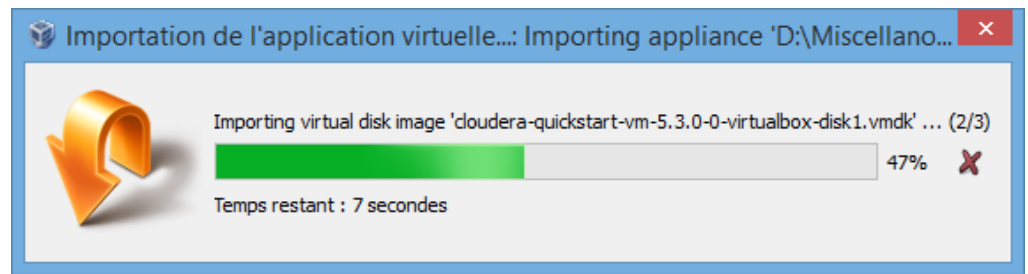
Importing the virtual machine into VirtualBox (2/3)



It seems we can choose, but in reality it is always **CentOS** which is installed.

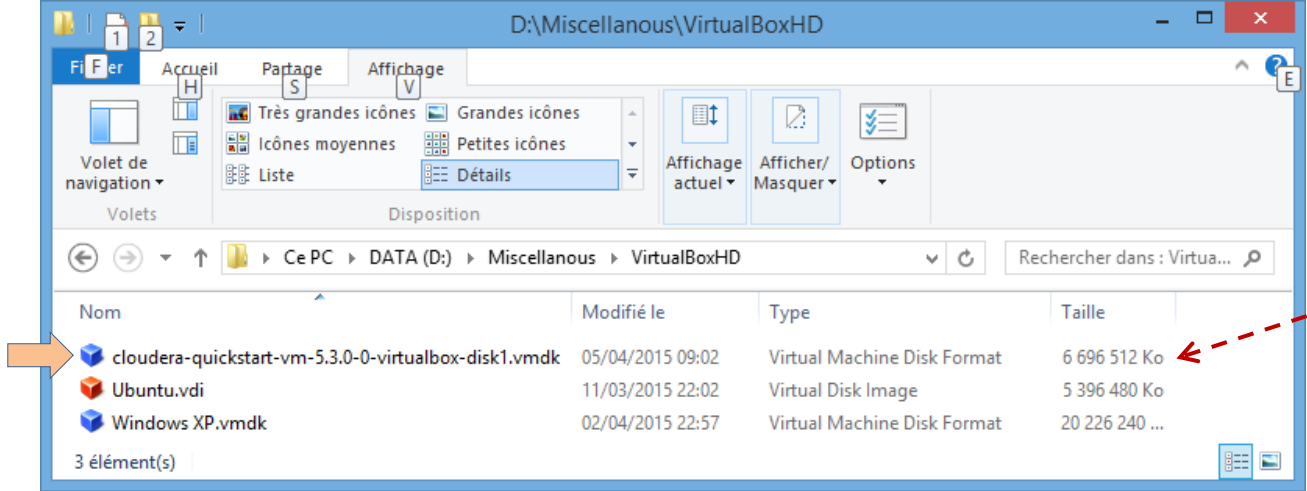
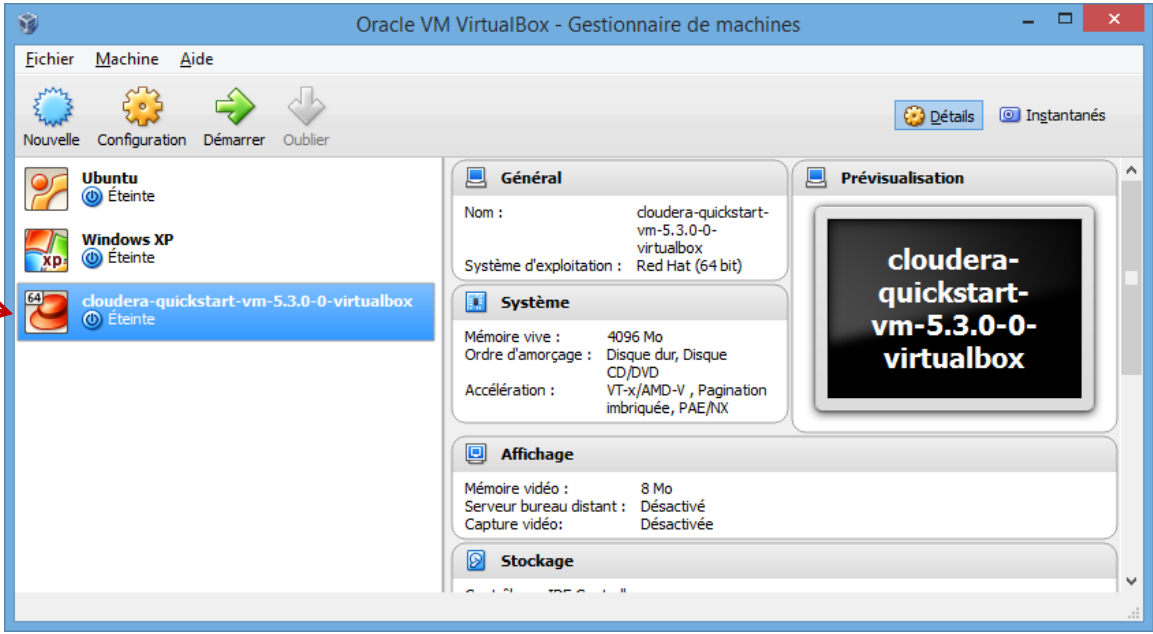
We can choose the destination folder of the file containing the virtual machine. Its size can increase considerably during the operations.

The import process is started when you click on the IMPORT button.



Importing the virtual machine into VirtualBox (3/3)

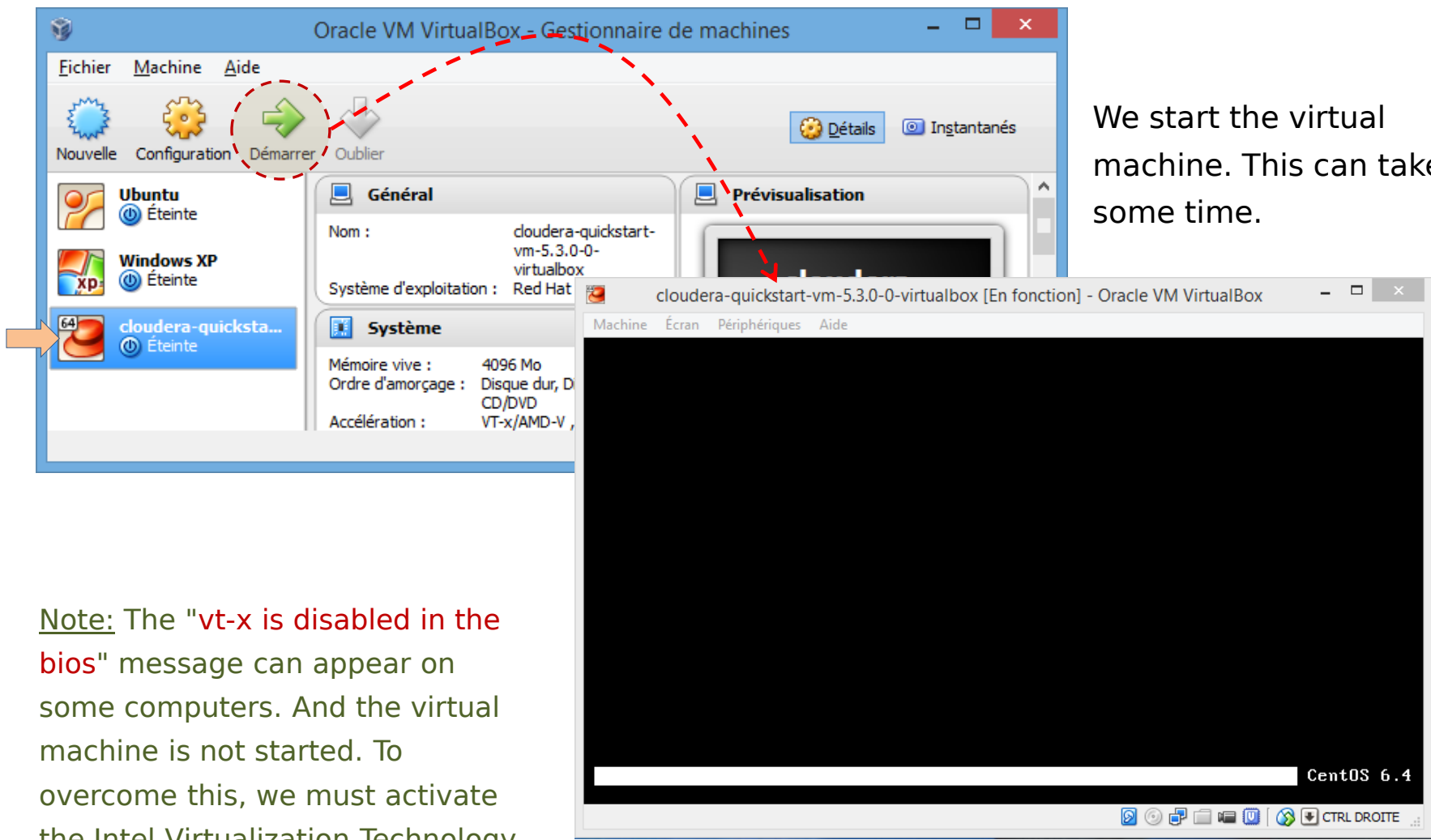
The virtual machine is now available into VirtualBox.



The size of the corresponding file is approximately 6 GB! And this is only the beginning.



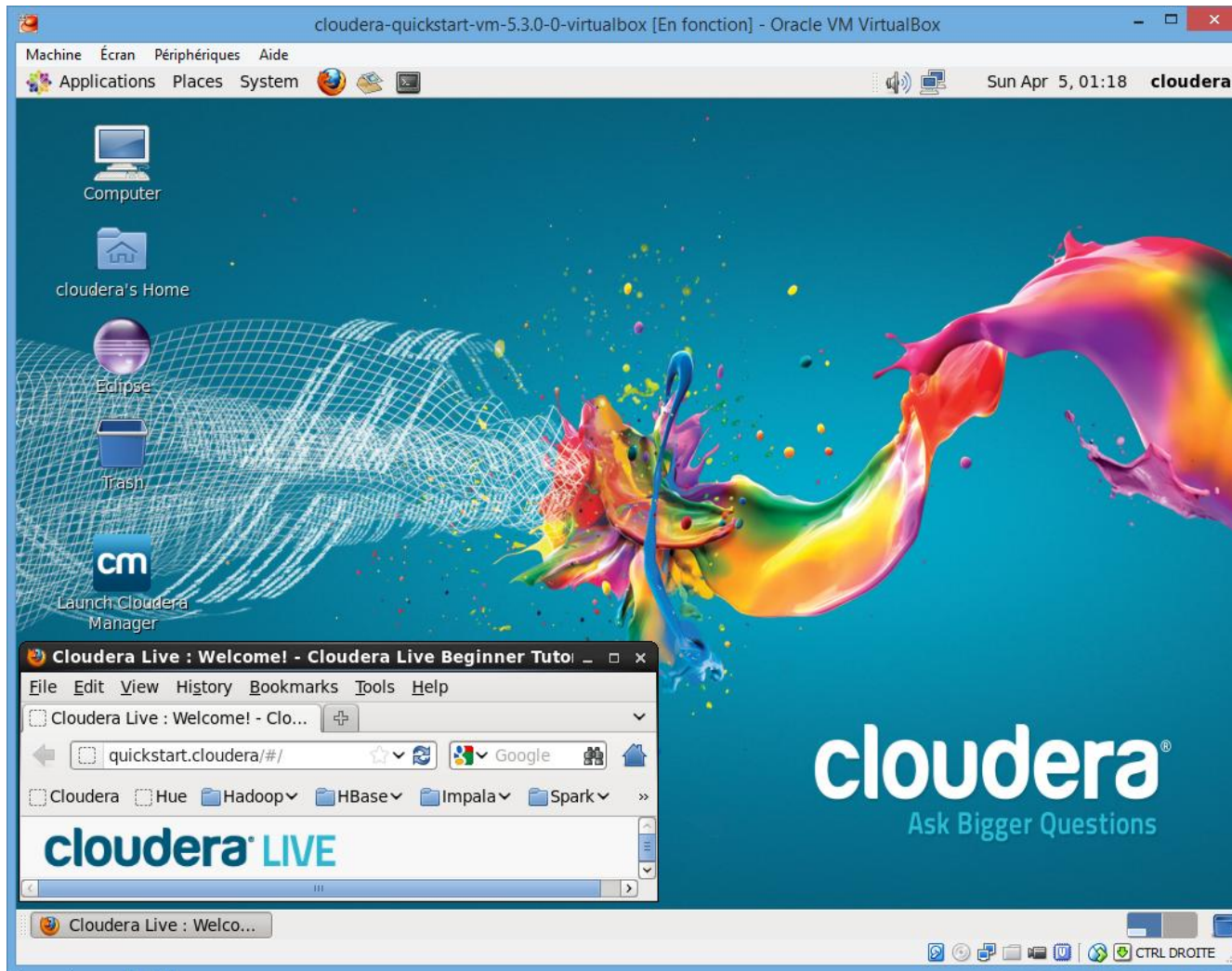
Starting the virtual machine (1/3)



We start the virtual machine. This can take some time.

Note: The "vt-x is disabled in the bios" message can appear on some computers. And the virtual machine is not started. To overcome this, we must activate the Intel Virtualization Technology in the BIOS.

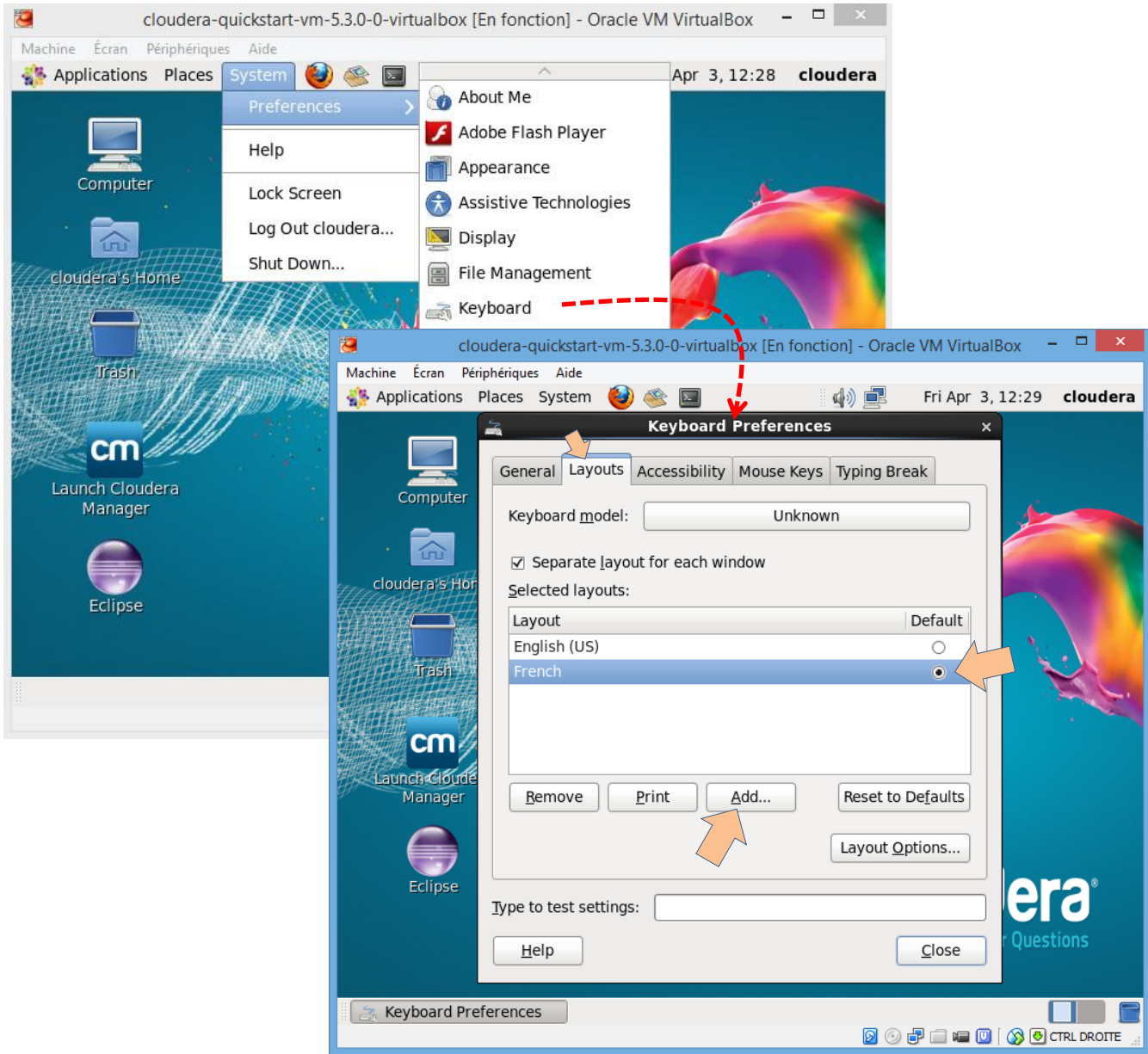
Starting the virtual machine (2/3)



Hadoop is already functional when the virtual machine is started. There are no specific operations to do at this step.



Starting the virtual machine (3/3)



If necessary, we must install the keyboard layout for our country.

Here, I install the French keyboard for me. I set it as the default keyboard.



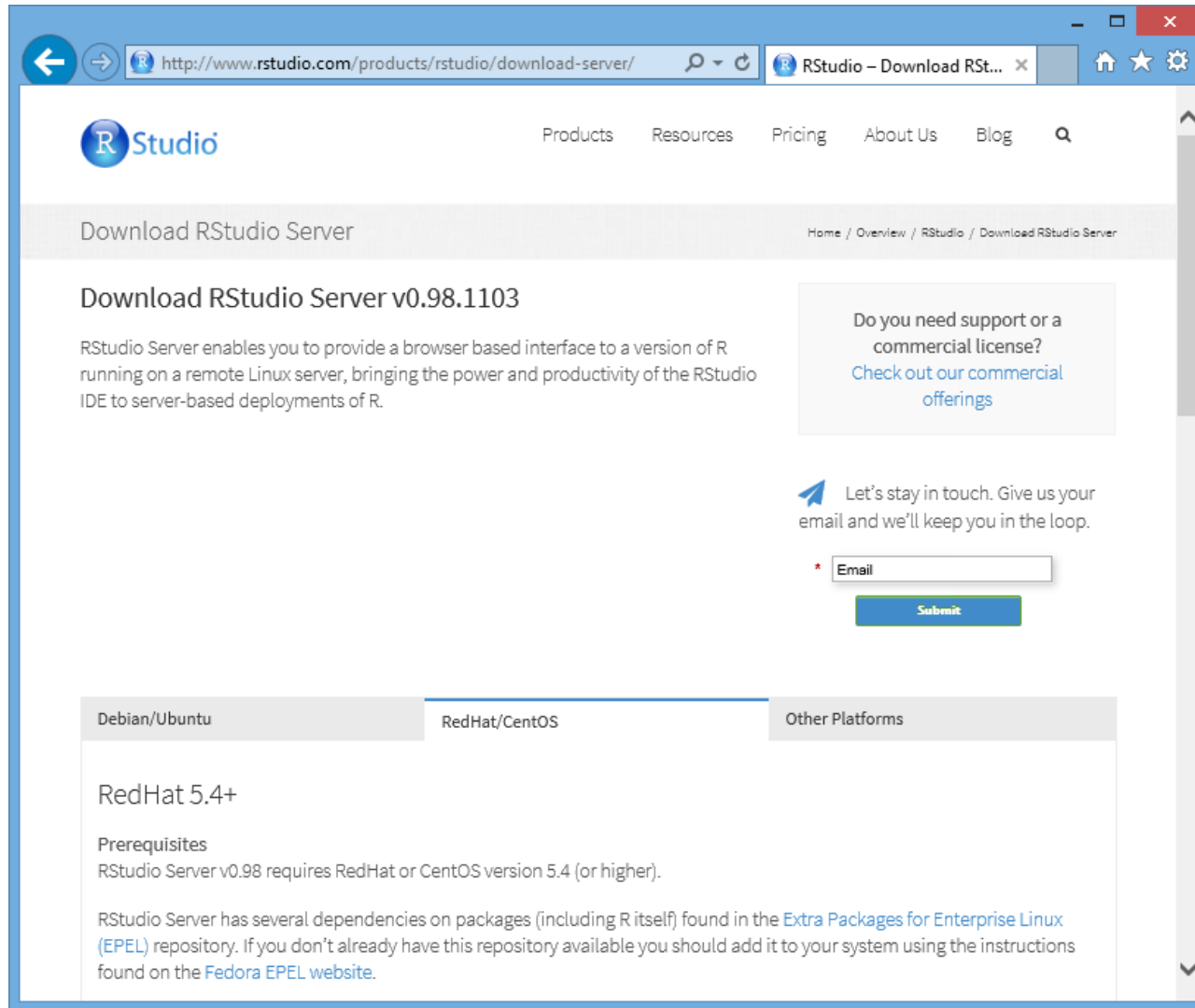
Installation of R and RStudio Server

The R tool allows to execute our programs.

RStudio Server enables to provide a browser based interface to a version of R running on a remote Linux server. In our case, the same computer is the client and the server, we use 127.0.0.1 as IP address (local host). But the approach can be applied generally to remote access using a properly configured server.



Installation of R and RStudio Server (1/2)



Download RStudio Server

Download RStudio Server v0.98.1103

RStudio Server enables you to provide a browser based interface to a version of R running on a remote Linux server, bringing the power and productivity of the RStudio IDE to server-based deployments of R.

Do you need support or a commercial license?
[Check out our commercial offerings](#)

Let's stay in touch. Give us your email and we'll keep you in the loop.

* Email

Debian/Ubuntu | **RedHat/CentOS** | Other Platforms

RedHat 5.4+

Prerequisites
RStudio Server v0.98 requires RedHat or CentOS version 5.4 (or higher).

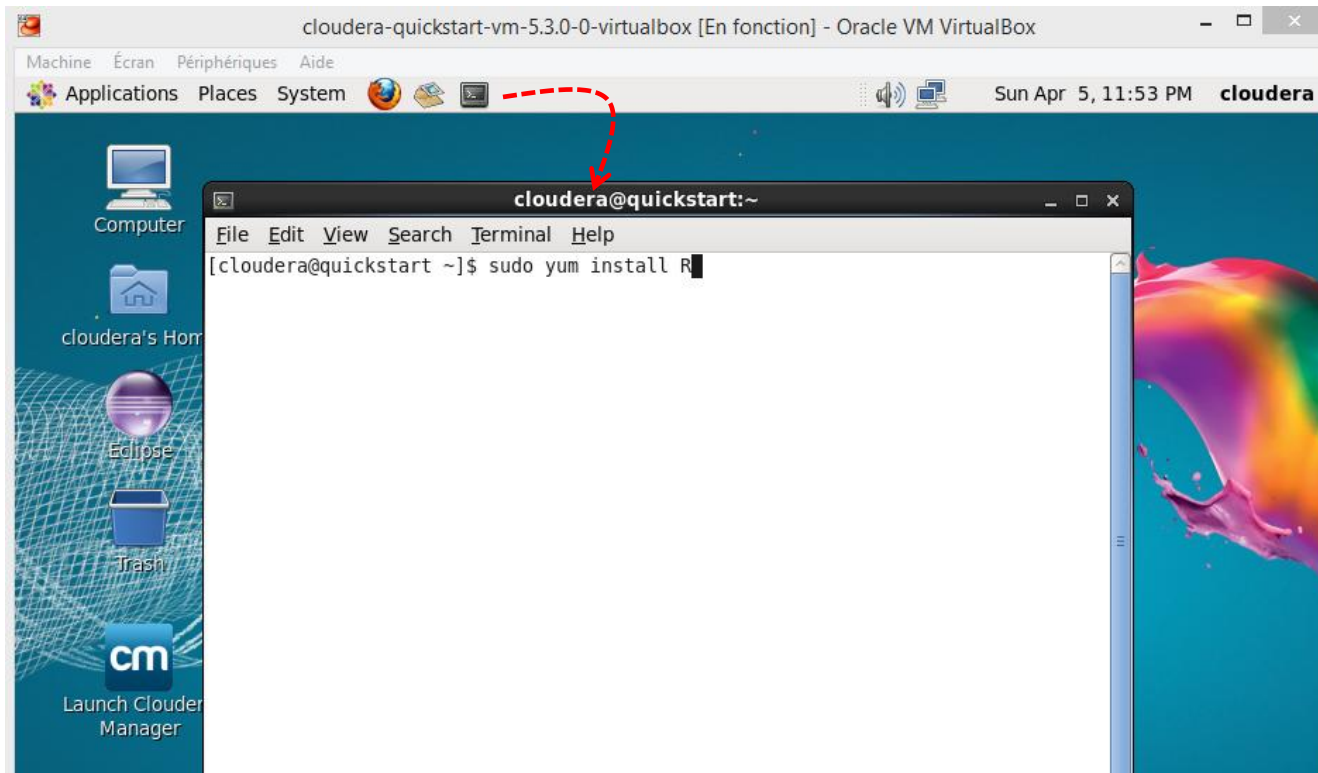
RStudio Server has several dependencies on packages (including R itself) found in the [Extra Packages for Enterprise Linux \(EPEL\)](#) repository. If you don't already have this repository available you should add it to your system using the instructions found on the [Fedora EPEL website](#).

All appropriate instructions are described on the website of RSTUDIO. We choose CentOS because this is the operating system that has been installed with our virtual machine.

<http://www.rstudio.com/products/rstudio/download-server/>



Installation of R and RStudio Server (2/2)



We open a terminal and we set the commands below.

Installation of R

```
$ sudo yum install R
```

Installation of Rstudio Server (64 bit)

```
$ sudo yum install openssl1098e # Required only for RedHat/CentOS 6 and 7
```

```
$ wget http://download2.rstudio.org/rstudio-server-0.98.1103-x86\_64.rpm
```

```
$ sudo yum install --nogpgcheck rstudio-server-0.98.1103-x86_64.rpm
```



Installation of packages available on the CRAN server

```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
[cloudera@quickstart ~]$ sudo R  
  
R version 3.1.3 (2015-03-09) -- "Smooth Sidewalk"  
Copyright (C) 2015 The R Foundation for Statistical Computing  
Platform: x86_64-redhat-linux-gnu (64-bit)  
  
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.  
  
Natural language support but running in an English locale  
  
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.  
  
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.  
  
> █
```

To use the R Hadoop packages for programming with Hadoop, we need first to install some packages available on the CRAN server.

We launch R in the administrator mode from a terminal i.e. using the command **\$sudo R**

Then, inside R terminal, we use the **install.packages()** command by specifying a repository (repos).

```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
[cloudera@quickstart ~]$ sudo R  
  
R version 3.1.3 (2015-03-09) -- "Smooth Sidewalk"  
Copyright (C) 2015 The R Foundation for Statistical Computing  
Platform: x86_64-redhat-linux-gnu (64-bit)  
  
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.  
  
Natural language support but running in an English locale  
  
> install.packages(c("Rcpp", "RJSONIO", "bitops", "digest", "functional", "reshape2",  
"stringr", "plyr", "caTools", "rJava"), repos="http://cran.us.r-project.org") █
```

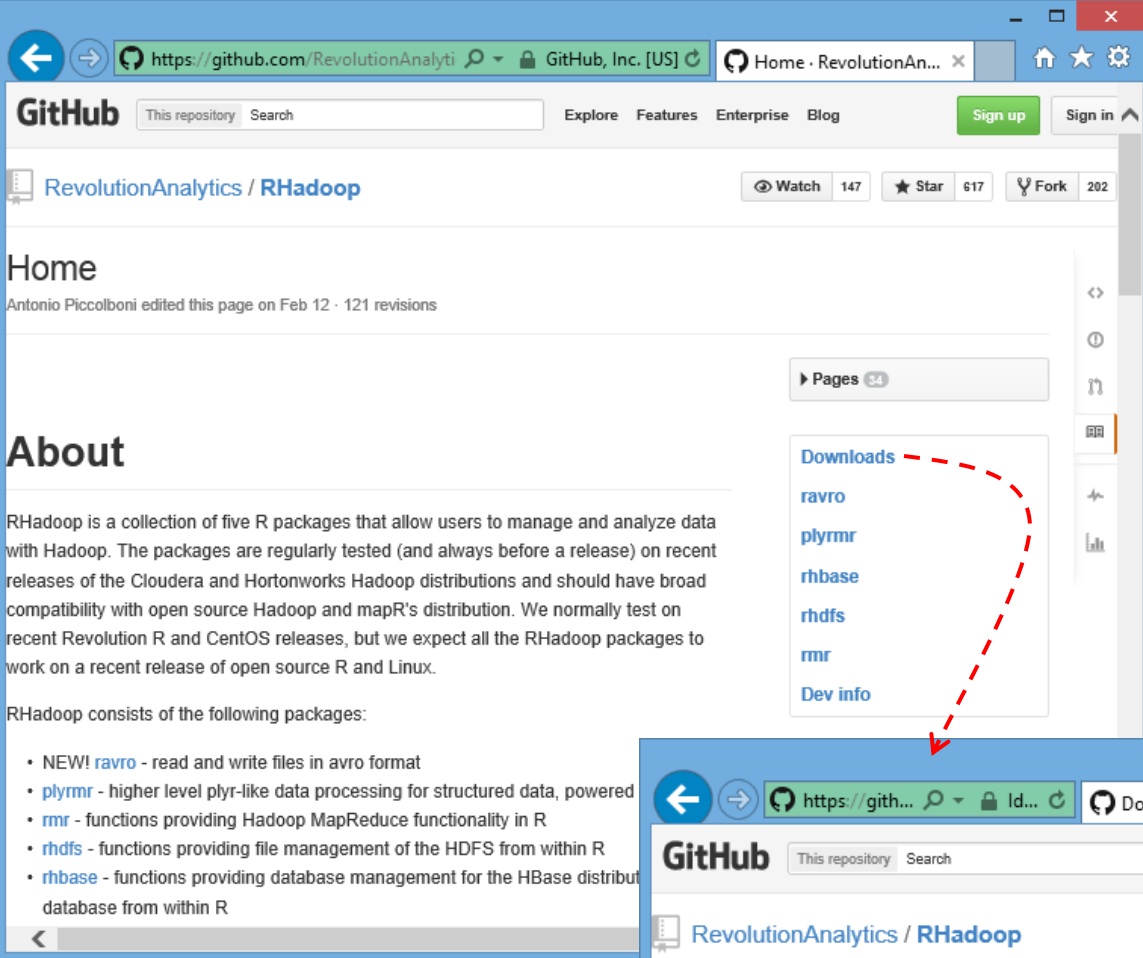


Installation of the RHadoop packages (1/3)

We want to use two packages from the RHadoop collection.

rmr which provides **MAPREDUCE** functionality in R.

rhdfs which provides functions for file management of the **HDFS** from within R.



We download these two files into the "Downloads" directory



Installation of the RHadoop packages (2/3)

```
cloudera@quickstart:~/Downloads
File Edit View Search Terminal Help
[cloudera@quickstart ~]$ cd Downloads/
[cloudera@quickstart Downloads]$ ls -l
total 92
-rw-rw-r-- 1 cloudera cloudera 25105 Apr  6 00:52 rhdfs_1.0.8.tar.gz
-rw-rw-r-- 1 cloudera cloudera 63087 Apr  6 00:52 rmr2_3.3.1.tar.gz
[cloudera@quickstart Downloads]$ sudo R
R version 3.1.3 (2015-03-09) -- "Smooth Sidewalk"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> install.packages("rmr2_3.3.1.tar.gz")
```

We open a terminal.
We change the current directory and we launch R in the administrative mode.

`install.packages()` allows also to install packages from local files.



Installation of the RHadoop packages (3/3)

```
cloudera@quickstart:~/Downloads
File Edit View Search Terminal Help
** help
*** installing help indices
    converting help for package 'rmr2'
      finding HTML links ... done
bigdataobject           html
dfs.empty               html
equijoin                html
fromdfstodfs           html
hadoop-setting          html
keyval                  html
make.io.format          html
mapreduce               html
rmr-package             html
rmr.options             html
rmr.sample              html
rmr.str                 html
scatter                 html
status                  html
tomaptoreduce           html
vsum                    html
** building package indices
** testing if installed package can be loaded
Warning: S3 methods 'gorder.default', 'gorder.factor', 'gorder.data.frame', 'gorder.
matrix', 'gorder.raw' were declared in NAMESPACE but not found
* DONE (rmr2)
Making 'packages.html' ... done
> Sys.setenv(HADOOP_HOME="/usr/lib/hadoop")
> Sys.setenv(HADOOP_CMD="/usr/lib/hadoop/bin/hadoop")
> install.packages("rhdfs_1.0.8.tar.gz")
```

For the "rmr" packages, we must define first some environment variables for the HADOOP system.

We use the `Sys.setenv()` command under R.



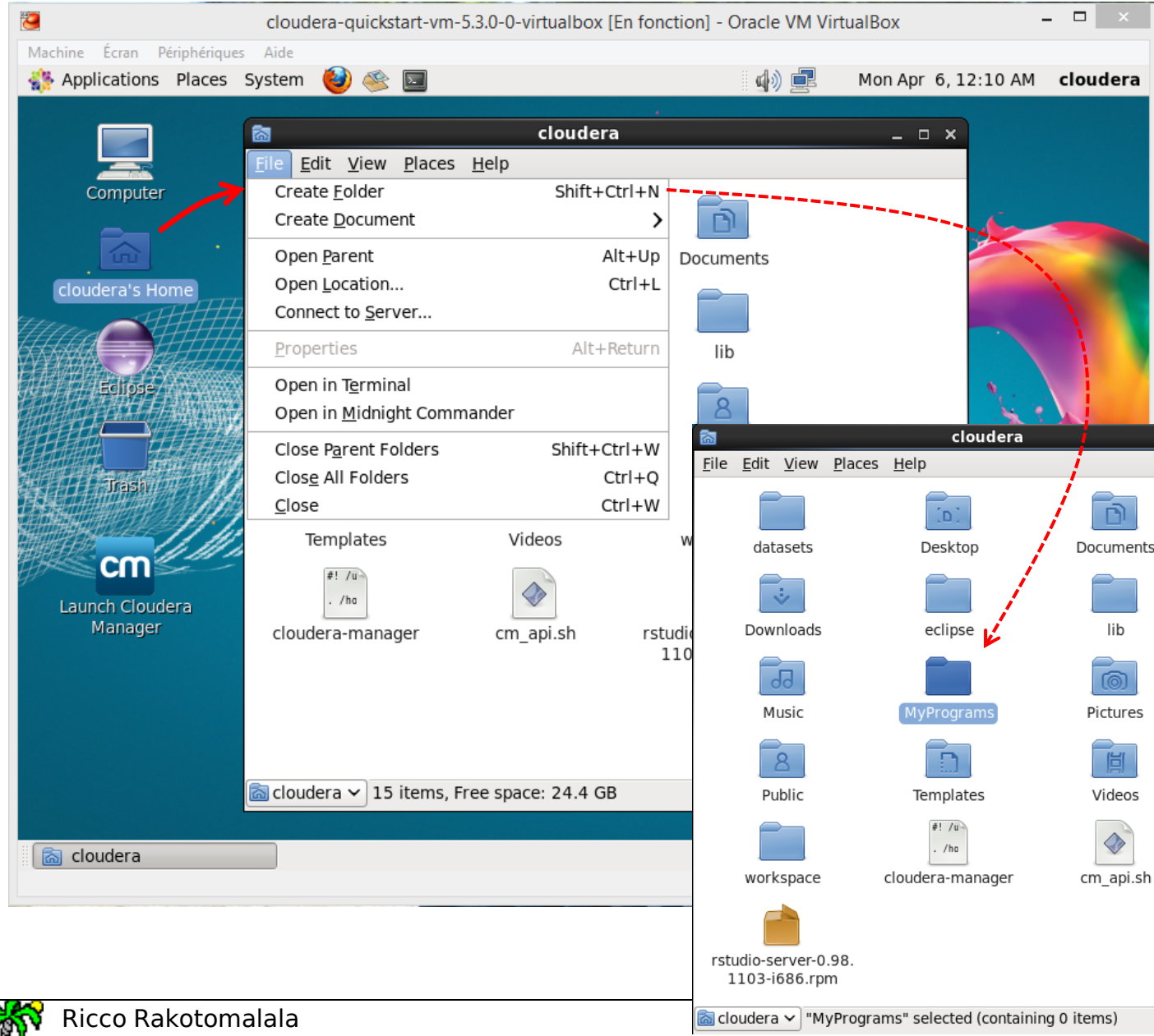
R programming under Hadoop (1)

MapReduce functionalities in R

In this first example, a vector of words is generated in memory. It is stored in a temporary file and then the `mapreduce()` function from the “`rmr2`” package is called. This function calls internally the `map()` and `reduce()` functions that we have coded before.



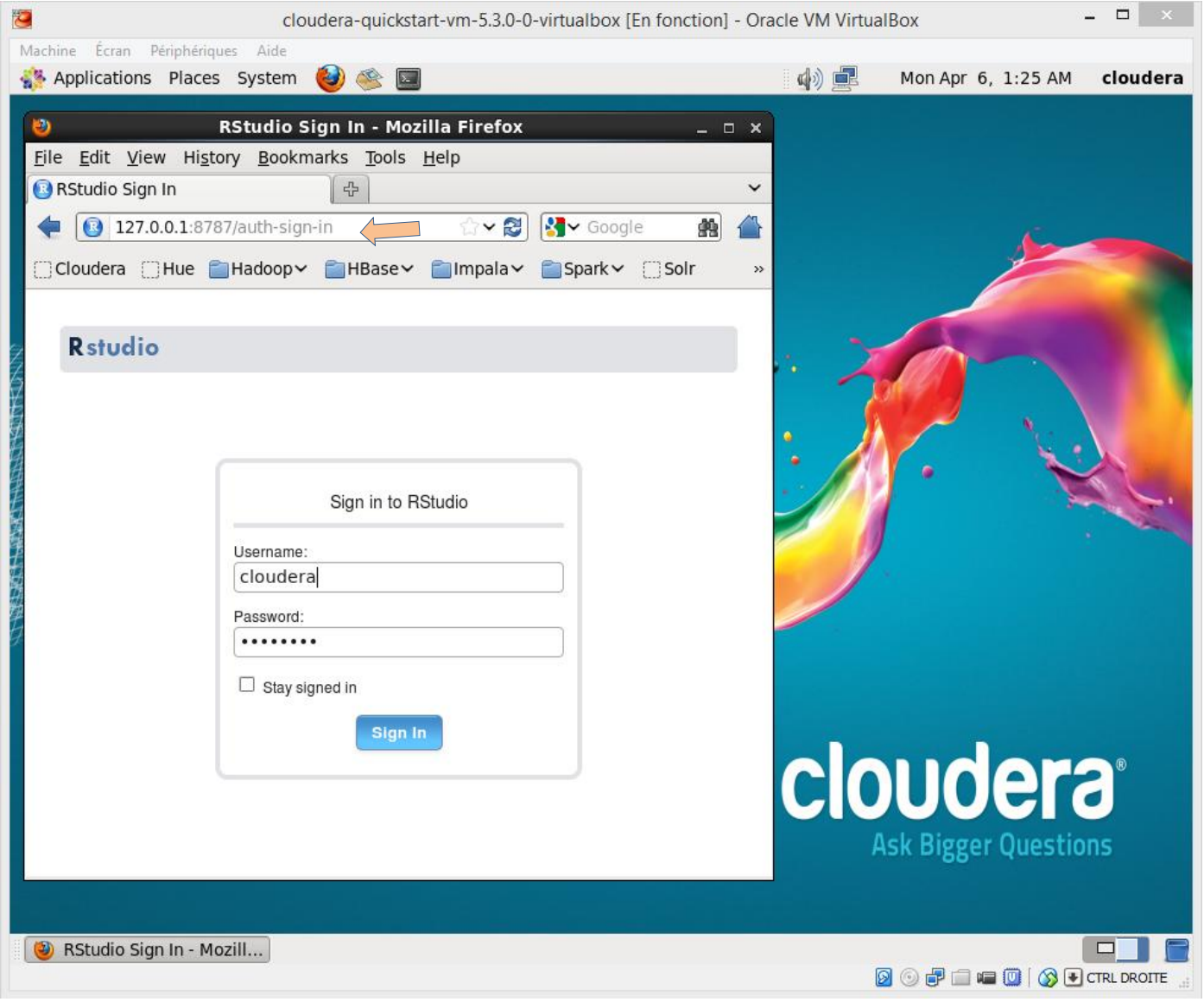
Creating a directory for our programs



The programs that we are going to write will be stored in the "MyPrograms" folder.



Accessing to RStudio using a web browser (1/2)



The same machine is client and server. We use the local host IP address: 127.0.0.1

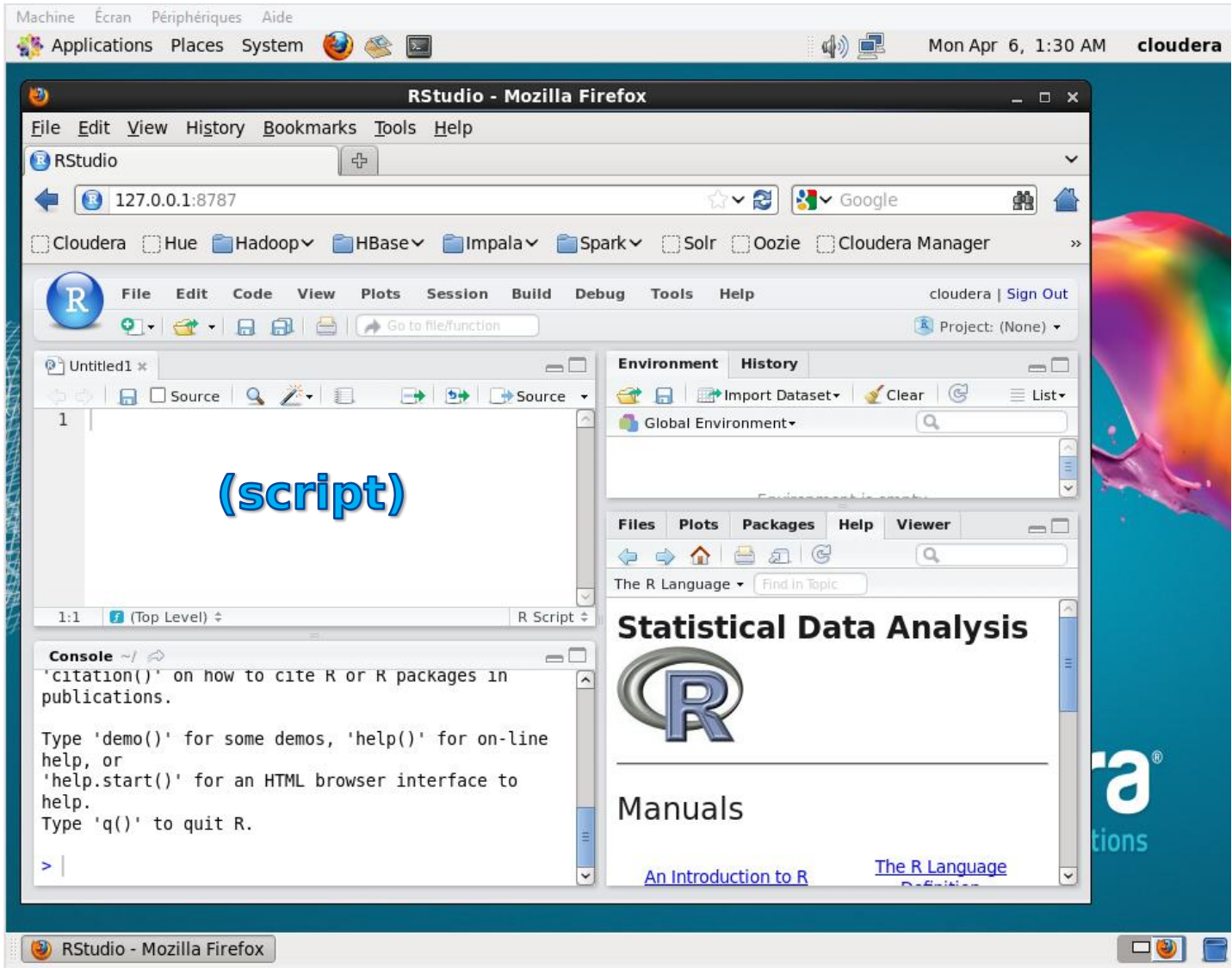
The port is 8787

The standard login is username : cloudera password : cloudera

See <https://support.rstudio.com/hc/en-us/articles/200552306-Getting-Started>



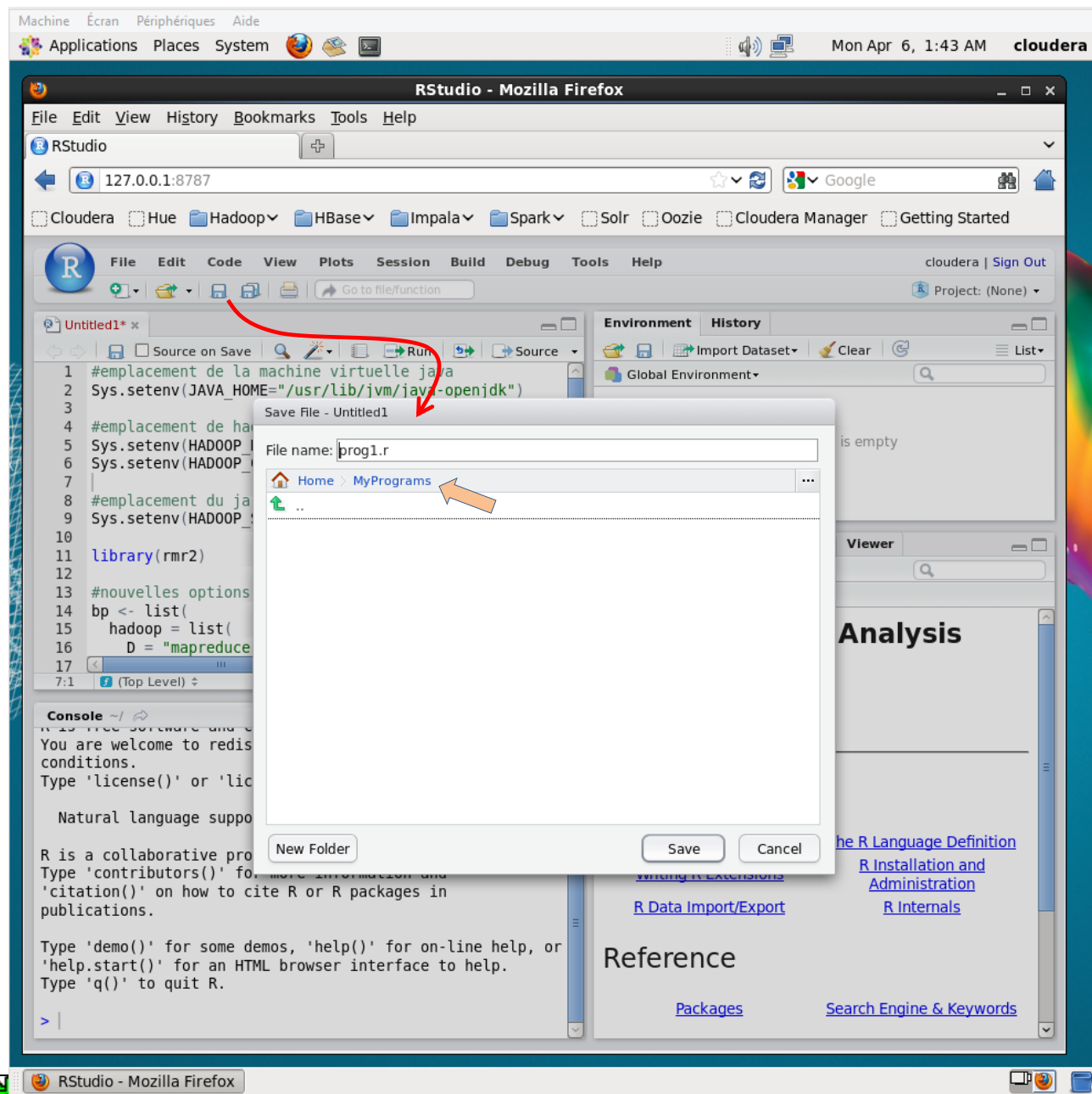
Accessing to RStudio using a web browser (2/2)



We have the usual RStudio development environment.

To write a new R script, we click on the FILE / NEW FILE / R SCRIPT menu.

Mapreduce in R (1/4)



The program is stored in the "MyPrograms" directory.

```
#directory of the JAVA virtual machine
Sys.setenv(JAVA_HOME="/usr/lib/jvm/java-openjdk")

#directory of the hadoop system
Sys.setenv(HADOOP_HOME="/usr/lib/hadoop")
Sys.setenv(HADOOP_CMD="/usr/lib/hadoop/bin/hadoop")

#directory of the utility which allows to create and run mapreduce jobs
Sys.setenv(HADOOP_STREAMING="/usr/lib/hadoop-mapreduce/hadoop-streaming-2.5.0-cdh5.3.0.jar")

#loading the "rmr2" package
library(rmr2)

#options for the memory management
#otherwise, the error message "Java heap space" interrupts the execution of the program
bp <- list(
  hadoop = list(
    D = "mapreduce.map.java.opts=-Xmx1024M",
    D = "mapreduce.reduce.java.opts=-Xmx2048M",
    D = "mapreduce.map.memory.mb=1280",
    D = "mapreduce.reduce.memory.mb=2560"
  )
)

#modification of the settings
rmr.options(backend.parameters = bp)

#hadoop mode
rmr.options(backend="hadoop")
```



MapReduce in R (3/4) – Writing the map() and reduce() functions. Handling a vector of words.

```
#map function
mymap <- function(k,v){
  keyval(v,1)
}

#reduce function
myreduce <- function(k,v){
  n <- length(v)
  keyval(k,n)
}

#vector of words
b <- c("one","two","one","one","two")

#transformed and copied in a temporary file on HDFS
a <- to.dfs(b)

#launching the mapreduce function
sortie <- mapreduce(input=a, map=mymap, reduce=myreduce)

#retrieving the output from a temporary file on HDFS
# printing the results
print(from.dfs(sortie))
```

After the calling of the map() function

key	value
one	1
two	1
one	1
one	1
two	1

Splitting based on the key

Key = one	1	1	1
Key = two	1	1	

Calling of REDUCE (Wikipedia [EN] :
The framework calls the application's
Reduce function once for each unique
key in the sorted order)

After the calling of REDUCE

Key = one	3
Key = two	2



MapReduce in R (4/4) – Reading the R terminal output

```
Console ~ | ↵
> #jeu de données créé en mémoire
> b <- c("one","two","one","one","two")
>
> #copié temporairement sur le système HDFS
> a <- to.dfs(b)
15/04/06 05:06:02 INFO zlib.ZlibFactory: Successfully loaded & initialized
native-zlib library
15/04/06 05:06:02 INFO compress.CodecPool: Got brand-new compressor [.deflate]
>
> #lancement
> sortie <- mapreduce(input=a, map=mymap, reduce=myreduce)
packageJobJar: [] [/usr/lib/hadoop-mapreduce/hadoop-streaming-2.5.0-
cdh5.3.0.jar] /tmp/streamjob5719859679609417903.jar tmpDir=null
15/04/06 05:06:04 INFO client.RMProxy: Connecting to ResourceManager at
/0.0.0.0:8032
15/04/06 05:06:05 INFO client.RMProxy: Connecting to ResourceManager at
/0.0.0.0:8032
15/04/06 05:06:05 INFO mapred.FileInputFormat: Total input paths to process : 1
15/04/06 05:06:05 INFO mapreduce.JobSubmitter: number of splits:2
15/04/06 05:06:05 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1428320419241_0002
15/04/06 05:06:06 INFO impl.YarnClientImpl: Submitted application
application_1428320419241_0002
15/04/06 05:06:06 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1428320419241_0002/
15/04/06 05:06:06 INFO mapreduce.Job: Running job: job_1428320419241_0002
15/04/06 05:06:13 INFO mapreduce.Job: Job job_1428320419241_0002 running in
uber mode : false
15/04/06 05:06:13 INFO mapreduce.Job: map 0% reduce 0%
```

The vector of words is created into the R memory

This vector is copied into a temporary file in HDFS file system. This file will be the **input** of the **mapreduce()** function of "rmr2".

Initialization of the execution. Then the **map()** and **reduce()** functions are called internally.

```
Console ~ | ↵
Total committed heap usage (bytes)=391979008
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=776
File Output Format Counters
Bytes Written=744
rmr
reduce calls=2
15/04/06 05:06:30 INFO streaming.StreamJob: Output directory:
/tmp/file178076b50cbe

> print(from.dfs(sortie))
$key
[1] "one" "two"

$val
[1] 3 2

> |
```

At the end of the execution, the output is copied into another temporary file on HDFS

We can retrieve the object in memory and print it.

R programming under Hadoop (2)

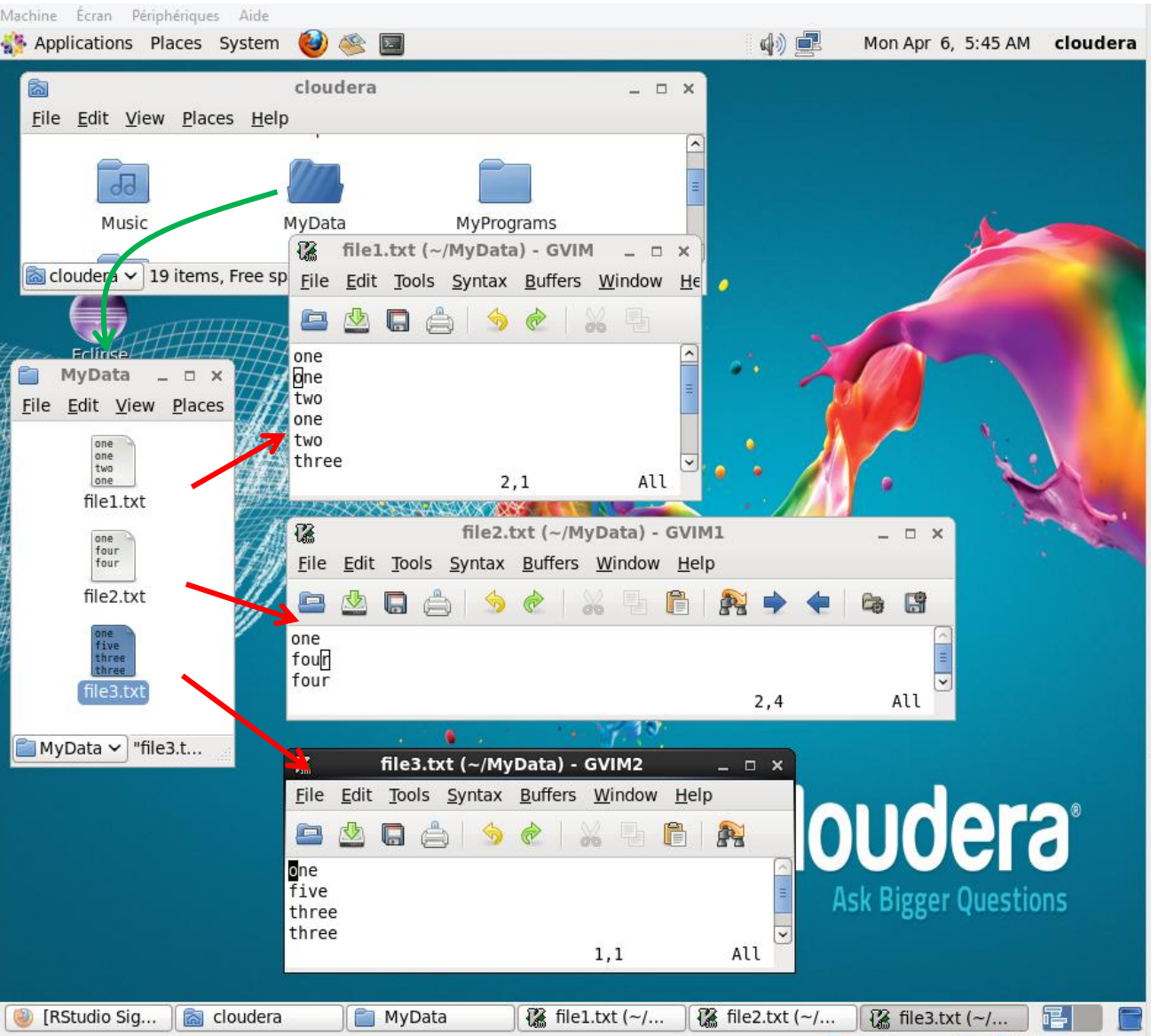
Accessing files in HDFS

In more realistic perspective, we will access to a set of files stored into a folder in the HDFS file system.

We first detail the format and the contents of these files, then we show how we copy them into a folder specially created for this purpose in HDFS. Finally, we modify our program in order to process these files.



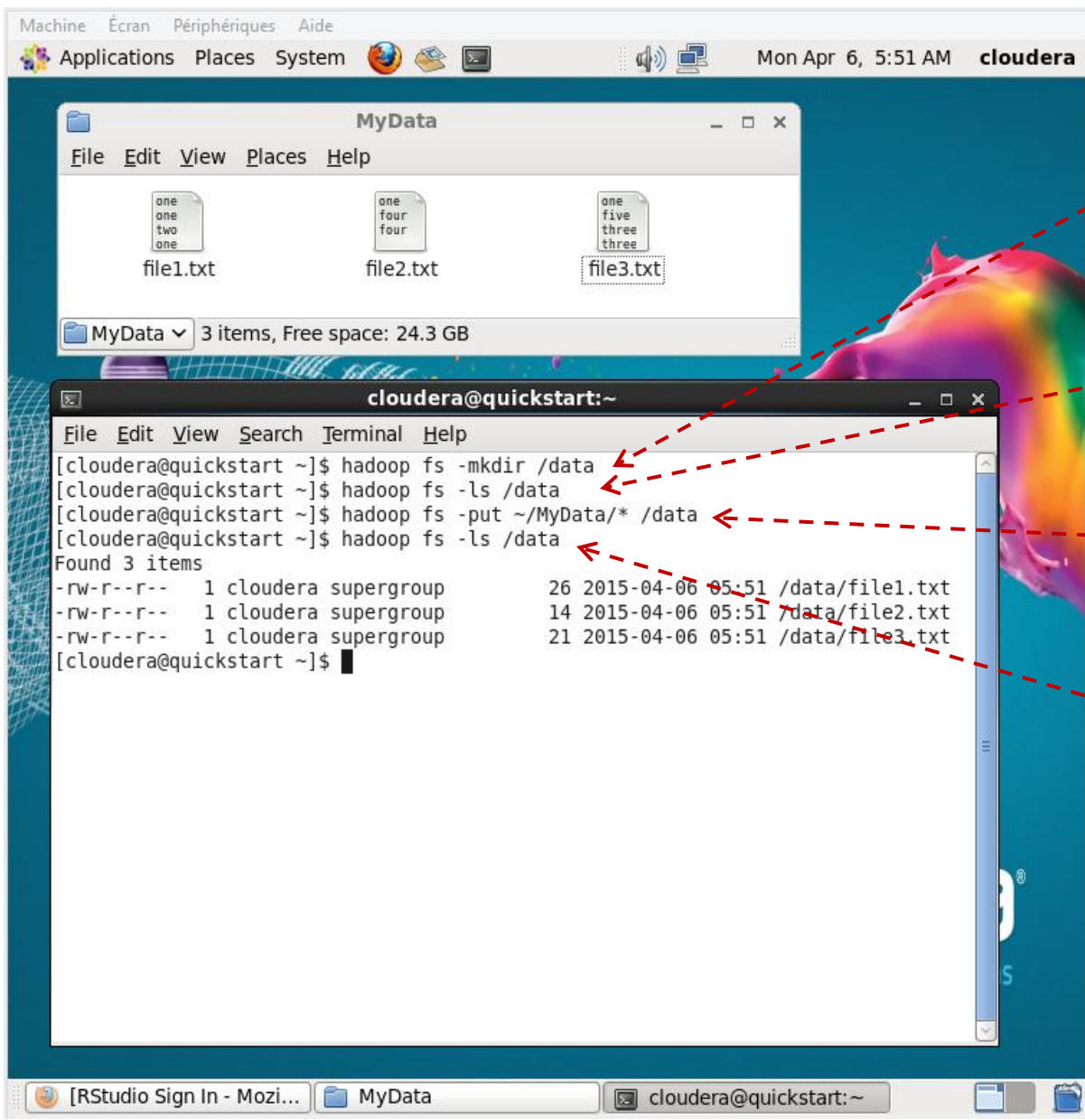
Creating 3 files (text format)



3 files are created in the "MyData" directory:
`file1.txt`, `file2.txt`, `file3.txt`

Each file contains a list of words that we can see in the screen shot.

Copying the files from the local directory to HDFS



We create the directory **"/data"** on HDFS. The command is valid regardless the number of nodes on the cluster.

The directory is empty for the moment.

We copy the files from the local directory to HDFS.

The files are visible into the directory **"/data"** now.

R program to handle files in HDFS

The global settings, the functions `map()` and `reduce()` are the same as before.

The input parameter now allows to scan the files in the `/data` directory in HDFS

```
#we need the rhdfs package here
library(rhdfs)

#initializing the access to hdfs
hdfs.init()

#displaying the content of the /data directory
hdfs.ls("/data")

#calling the mapreduce() function by reading the content of /data in HDFS
sortie.bis <- mapreduce(input="/data", input.format="text", map=mymap, reduce=myreduce)
print(from.dfs(sortie.bis))
```

The files are in the text file format. We do not need to create a temporary file using the **to.dfs()** procedure here.



Output of the program (1/2)

```
Console ~/
> #afficher le contenu du dossier data
> hdfs.ls("/data")
permission owner group size modtime file
1 -rw-r--r-- cloudera supergroup 26 2015-04-06 05:51 /data/file1.txt
2 -rw-r--r-- cloudera supergroup 14 2015-04-06 05:51 /data/file2.txt
3 -rw-r--r-- cloudera supergroup 21 2015-04-06 05:51 /data/file3.txt
>
> #programmation en accédant aux fichiers contenu dans le dossier /data
> sortie.bis <- mapreduce(input="/data", input.format="text", map=mymap, reduce=myreduce)
packageJobJar: [] [/usr/lib/hadoop-mapreduce/hadoop-streaming-2.5.0-cdh5.3.0.jar]
/tmp/streamjob7113339506246022660.jar tmpDir=null
15/04/06 06:09:13 INFO client.RMPProxy: Connecting to ResourceManager at /0.0.0.0:8032
15/04/06 06:09:13 INFO client.RMPProxy: Connecting to ResourceManager at /0.0.0.0:8032
15/04/06 06:09:14 INFO mapred.FileInputFormat: Total input paths to process : 3
15/04/06 06:09:14 INFO mapreduce.JobSubmitter: number of splits:3
15/04/06 06:09:14 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1428320419241_0003
15/04/06 06:09:14 INFO impl.YarnClientImpl: Submitted application
application_1428320419241_0003
15/04/06 06:09:14 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1428320419241_0003/
15/04/06 06:09:14 INFO mapreduce.Job: Running job: job_1428320419241_0003
15/04/06 06:09:23 INFO mapreduce.Job: Job job_1428320419241_0003 running in uber mode : false
15/04/06 06:09:23 INFO mapreduce.Job: map 0% reduce 0%
```

The 3 files in the **/data** directory are visible.



And they will be processed.



We can observe the steps of the processing.



```
Console ~/
15/04/06 06:09:14 INFO impl.YarnClientImpl: Submitted application
application_1428320419241_0003
15/04/06 06:09:14 INFO mapreduce.Job: The url to track the job:
http://quickstart.cloudera:8088/proxy/application_1428320419241_0003/
15/04/06 06:09:14 INFO mapreduce.Job: Running job: job_1428320419241_0003
15/04/06 06:09:23 INFO mapreduce.Job: Job job_1428320419241_0003 running in uber mode : false
15/04/06 06:09:23 INFO mapreduce.Job: map 0% reduce 0%
15/04/06 06:09:41 INFO mapreduce.Job: map 22% reduce 0%
15/04/06 06:09:48 INFO mapreduce.Job: map 33% reduce 0%
15/04/06 06:09:51 INFO mapreduce.Job: map 78% reduce 0%
15/04/06 06:09:53 INFO mapreduce.Job: map 100% reduce 0%
15/04/06 06:10:01 INFO mapreduce.Job: map 100% reduce 100%
15/04/06 06:10:02 INFO mapreduce.Job: Job job_1428320419241_0003 completed successfully
15/04/06 06:10:02 INFO mapreduce.Job: Counters: 50
File System Counters
FILE: Number of bytes read=1797
FILE: Number of bytes written=447123
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=355
HDFS: Number of bytes written=1285
HDFS: Number of read operations=12
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
Job Counters
```



Output of the program (2/2)

The screenshot shows the RStudio interface with the following content:

```
42 #lecture du fichier sur le hdfs maintenant
43 library(rhdfs)
44
45 #initialisation
46 hdfs.init()
47
48 #afficher le contenu du dossier data
49 hdfs.ls("/data")
50
51 #programmation en accédant aux fichiers contenu dans le dossier /data
52 sortie.bis <- mapreduce(input="/data", input.format="text", map=mymap, reduce=my
53 print(from.dfs(sortie.bis))
54
```

Console output:

```
IO ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=61
File Output Format Counters
  Bytes Written=1285
rmr
  reduce calls=5
15/04/06 06:10:02 INFO streaming.StreamJob: Output directory: /tmp/file1fbc4e04826a
> print(from.dfs(sortie.bis))
$key
[1] "one" "two" "five" "four" "three"

$val
[1] 5 2 1 2 3
```

reduce() is called 5 times because there are 5 distinct values of key.

The results are stored in a temporary file because we have not specified the output parameter.

And we obtain the counting of the words for the three files: file1.txt, file2.txt et file3.txt !

References



Tutoriel Tanagra, « [Mapreduce with R](#) », February 2015.

Hugh Devlin, « [Mapreduce in R](#) », January 2014.

Cloudera, « [Cloudera Product Downloads](#) ».

RStudio, « [Download RStudio Server – RedHat/CentOS](#) ».

RevolutionAnalytics, « [RHadoop](#) ».

