

Programmation R sous Hadoop

Installation du framework Hadoop (distribution Cloudera)

Installation et configuration de R et RStudio Server

Programmation Mapreduce avec R

Programmation avec accès aux données sur HDFS



Objectif du tutoriel

L'objectif de ce tutoriel est de montrer, *in fine*, la programmation sous R de l'algorithme de comptage de mots – le fameux « wordcount » – à partir d'un ensemble de fichiers stockés sur HDFS.

L'exemple « wordcount » fait référence. Il est décrit partout sur le web. Mais, à bien y regarder, les tutoriels qui le reprennent sont (très) rarement reproductibles. Les fichiers de travail ne sont pas disponibles. On ne voit pas vraiment comment on y accède avec R lorsqu'ils sont stockés sur le système de fichier HDFS. Bref, on ne peut pas faire tourner les programmes et se rendre compte réellement de leur mode de fonctionnement.

Nous allons reprendre tout cela étape par étape. Nous décrivons avec force détails chaque stade de processus, en partant de l'installation d'un cluster hadoop mono-nœud sur une machine virtuelle jusqu'à la programmation sous R, en passant par l'installation de R et de l'environnement de programmation client – serveur RStudio Server.

Les étapes et, par conséquent les sources d'erreurs, sont nombreuses. Nous utiliserons moult copies d'écran pour appréhender concrètement chaque opération. D'où ce format de présentation inhabituel pour un tutoriel.



Etapes

1. Installation et configuration d'un cluster simple nœud
2. Installation et configuration de R et RStudio Server
3. Programmation R sous Hadoop – MapReduce
4. Programmation R sous Hadoop – Accès aux fichiers sur HDFS
5. Bibliographie



Installation et configuration d'un cluster simple noeud

Installation du framework hadoop

Il est possible d'installer le framework Hadoop directement sur une machine existante (ex. [installation sous Ubuntu](#)). Mais l'opération reste délicate, nécessitant un certain savoir faire informatique (système).

Heureusement, des éditeurs proposent des solutions clés en main avec la création d'une machine virtuelle – faisant office d'un cluster à un nœud – intégrant déjà Hadoop correctement configuré et fonctionnel. Nous utiliserons la distribution [Cloudera](#) basé sur le système d'exploitation CentOS dans ce tutoriel.



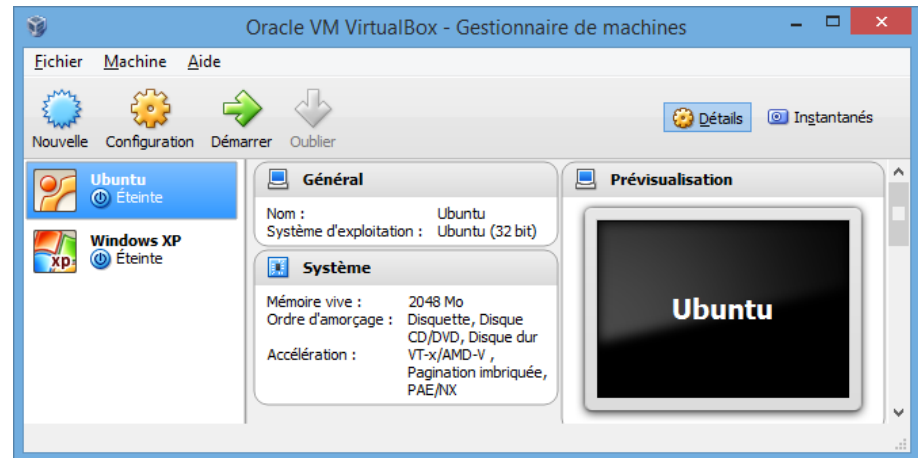
Chargement et installation de VirtualBox

VirtualBox est un logiciel libre.

VirtualBox est un logiciel de « virtualisation » c.-à-d. c'est une application hôte qui permet d'accueillir et faire tourner un système d'exploitation comme un logiciel quelconque. On dispose ainsi d'une machine virtuelle pleinement fonctionnelle.



Une fois installé, on peut y incorporé plusieurs machines virtuelles.



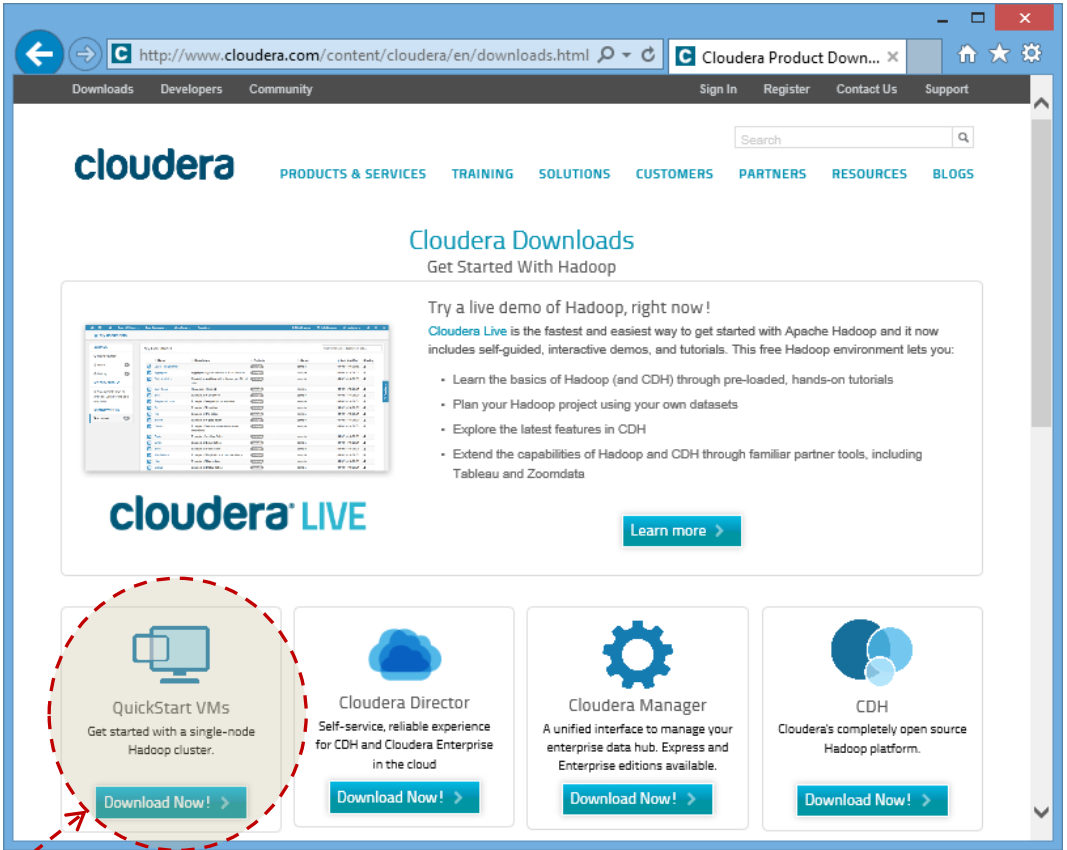
Dans cette copie d'écran, on constate que j'ai déjà installé deux machines invitées sous VirtualBox : l'une tournant sous Ubuntu et l'autre sous Windows XP.



Chargement de Cloudera (1/2)

Cloudera propose une solution clé en main – gratuite – permettant de bénéficier d'une configuration Hadoop fonctionnelle sous une nouvelle machine virtuelle invitée de VirtualBox (ou autre, ex. VMWare).

<http://www.cloudera.com/content/cloudera/en/downloads.html>



Il s'agit bien de créer un cluster hadoop simple nœud sous la forme d'une machine virtuelle (VM)



Chargement de Cloudera (2/2)

La machine virtuelle est basée sur CentOS 6.4

Support Developers Sign In Register Contact Us Downloads

Contact Sales: 866-843-7207

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.

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 >

Nous chargeons la machine destinée à VirtualBox dans ce tutorial.

Le fichier archive une fois téléchargé




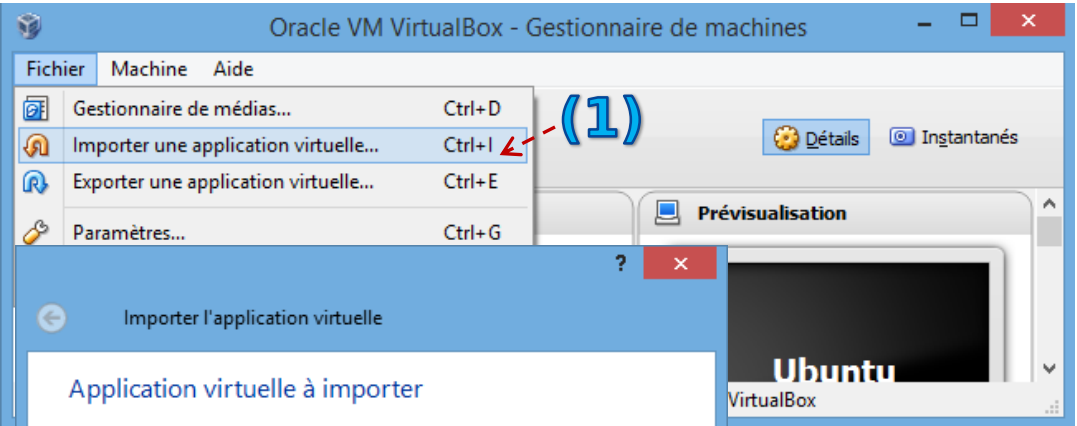
Nom	Modifié le	Type	Taille
cloudera-quickstart-vm-5.3.0-0-virtualbox.7z	18/03/2015 16:35	Fichier 7Z	3 780 637 Ko



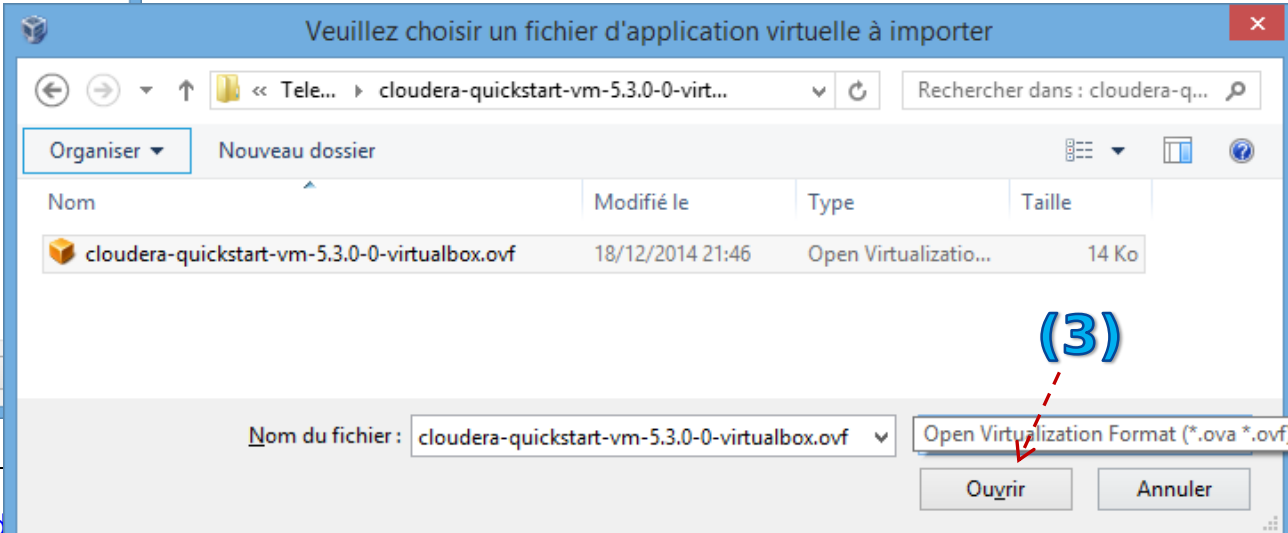
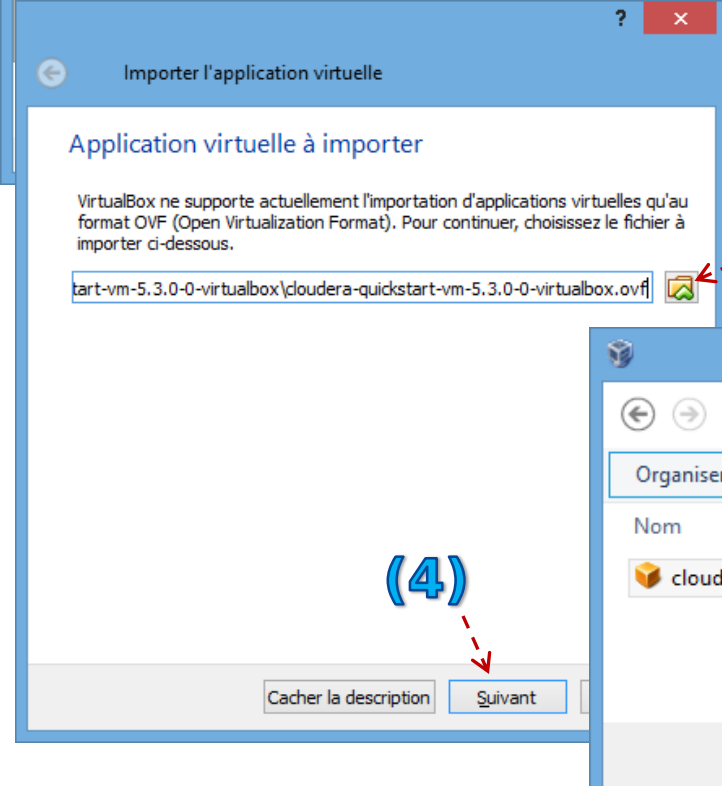
Importation de la machine virtuelle sous VirtualBox (1/3)

L'archive comporte 2 fichiers →

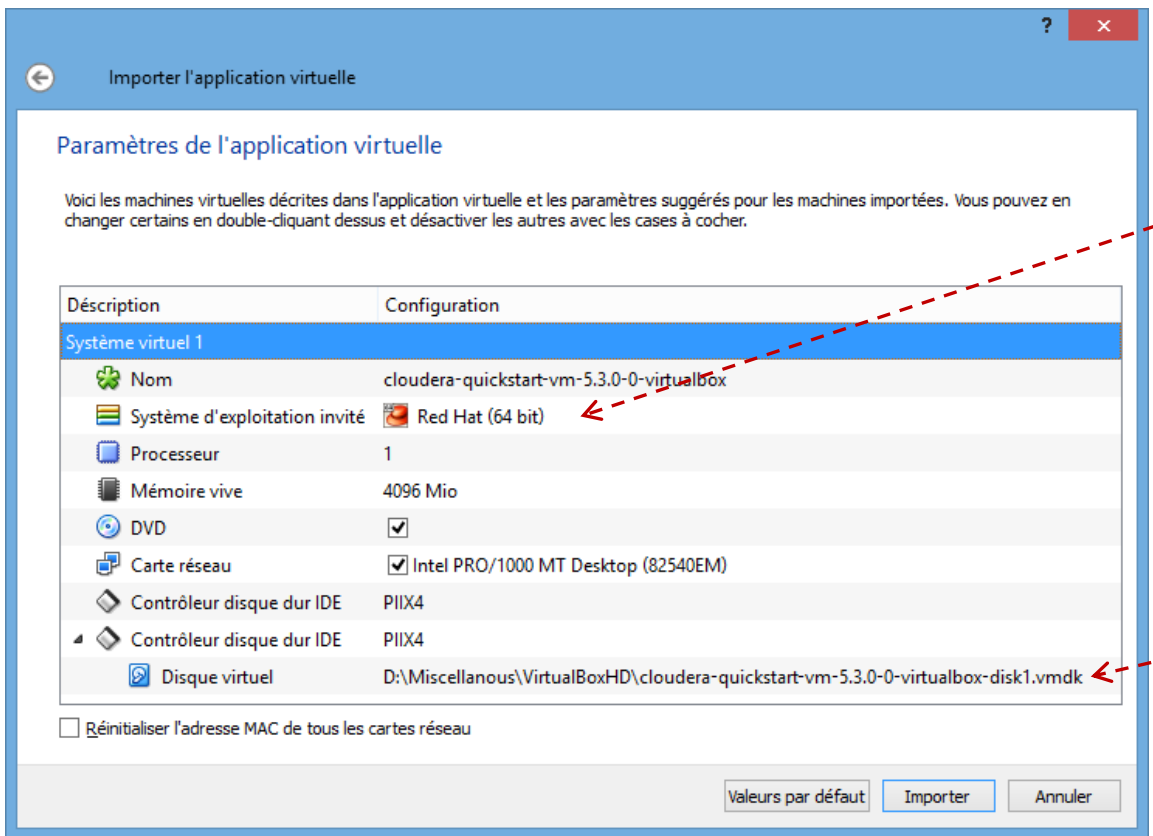
 cloudera-quickstart-vm-5.3.0-0-virtualbox.ovf	18/12/2014 21:46	Open Virtualizatio...	14 Ko
 cloudera-quickstart-vm-5.3.0-0-virtualbox-disk1.vmdk	18/12/2014 21:51	Virtual Machine Di...	3 837 156 Ko



Il y a plusieurs étapes dans le processus d'importation que l'on initie en cliquant sur le menu adéquat dans VirtualBox



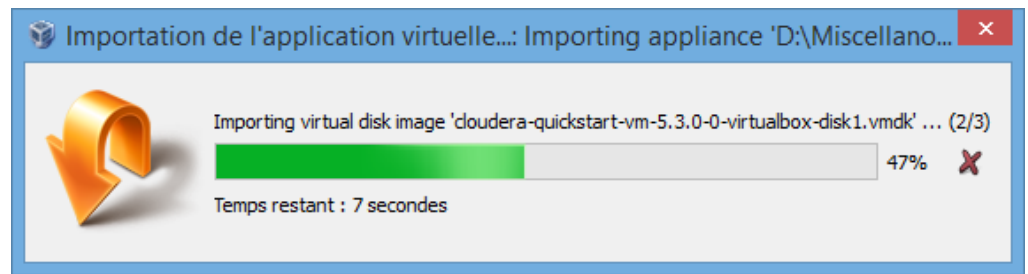
Importation de la machine virtuelle sous VirtualBox (2/3)



On a l'impression de pouvoir choisir, mais en réalité c'est toujours CentOS qui est installé.

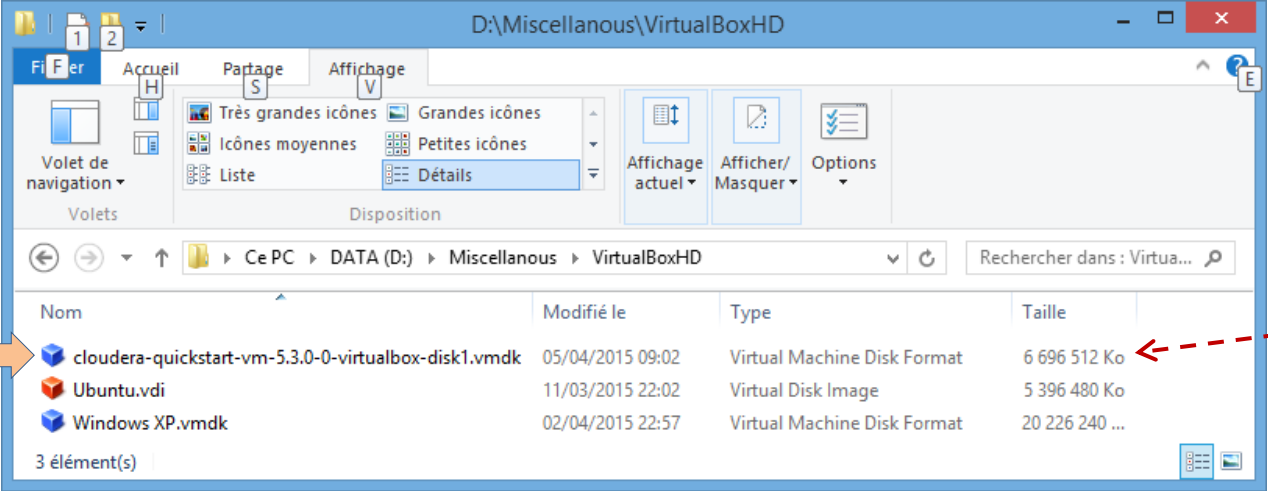
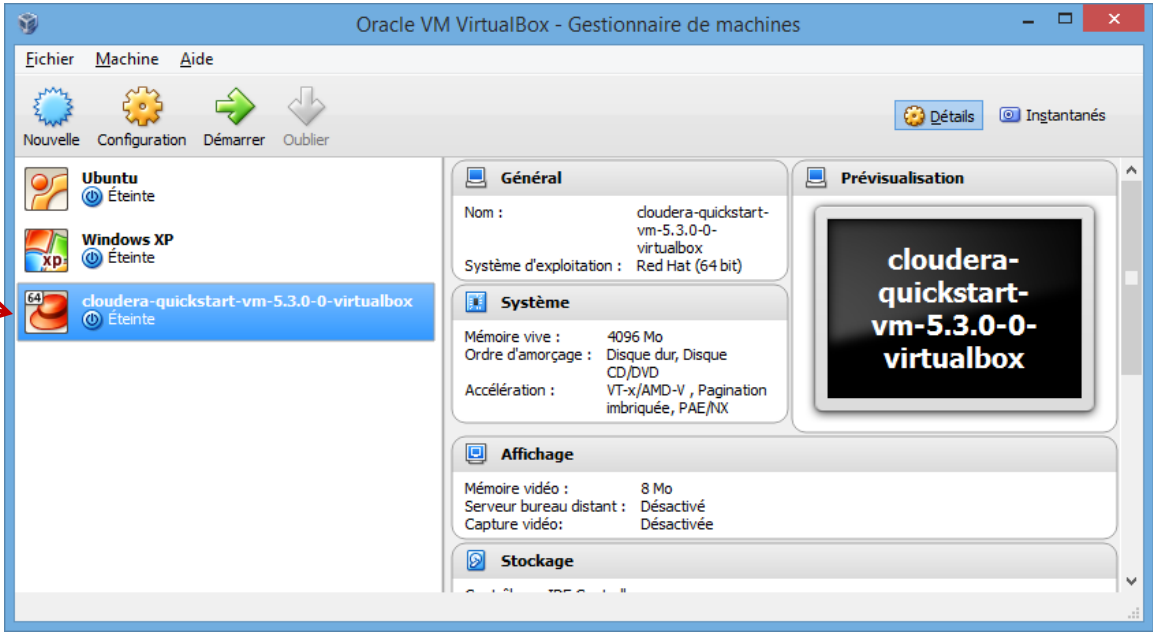
Choix du dossier de destination du fichier contenant la machine virtuelle. **Attention, il faut prévoir de la place parce que sa taille va enfler considérablement au fil des opérations.**

Le processus d'importation est démarré lorsque l'on clique sur IMPORTER.



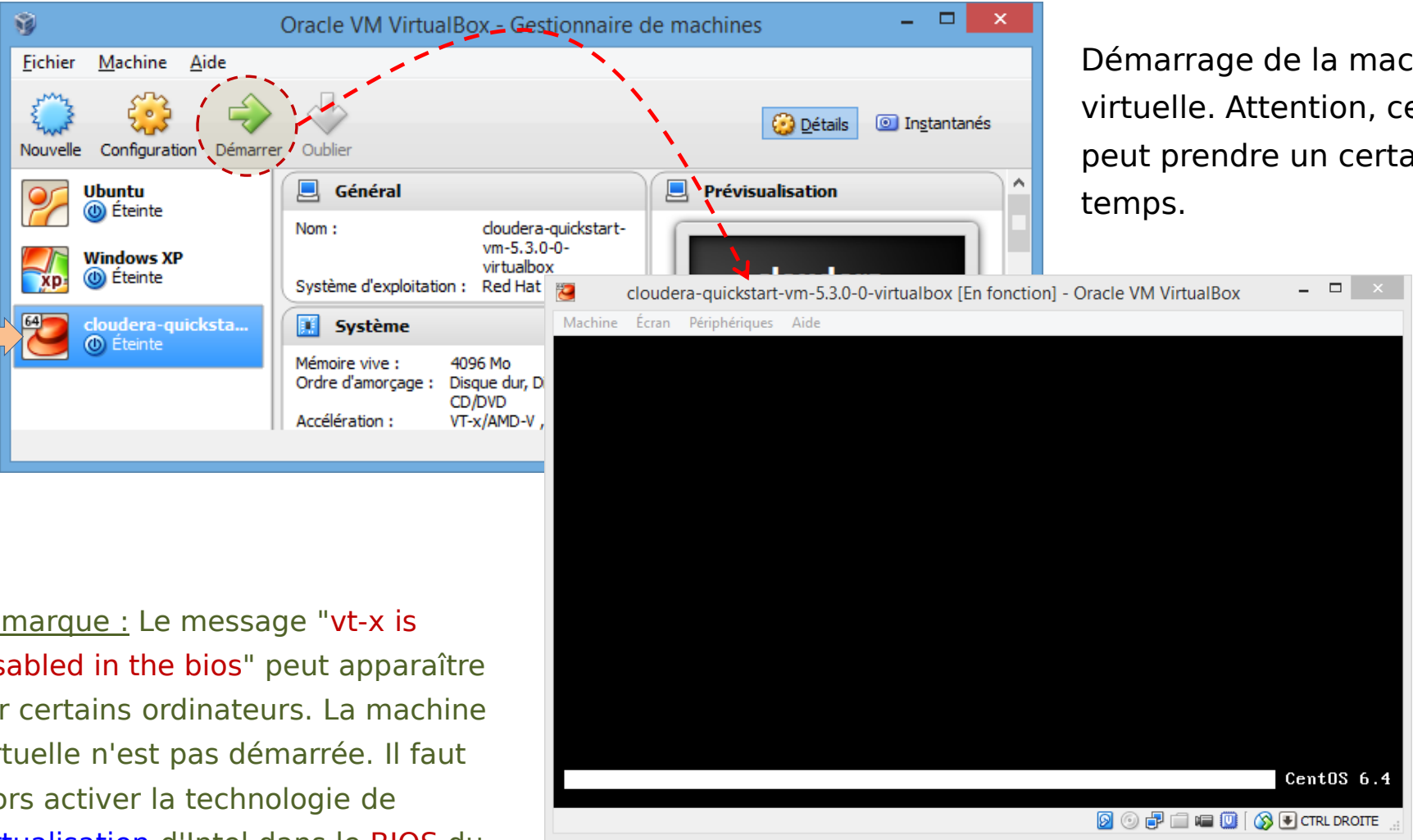
Importation de la machine virtuelle sous VirtualBox (3/3)

La machine virtuelle est maintenant installée dans VirtualBox



Le fichier qui lui est associé fait plus de 6 Go ! Et ce n'est que le début.

Démarrage de la machine virtuelle (1/3)

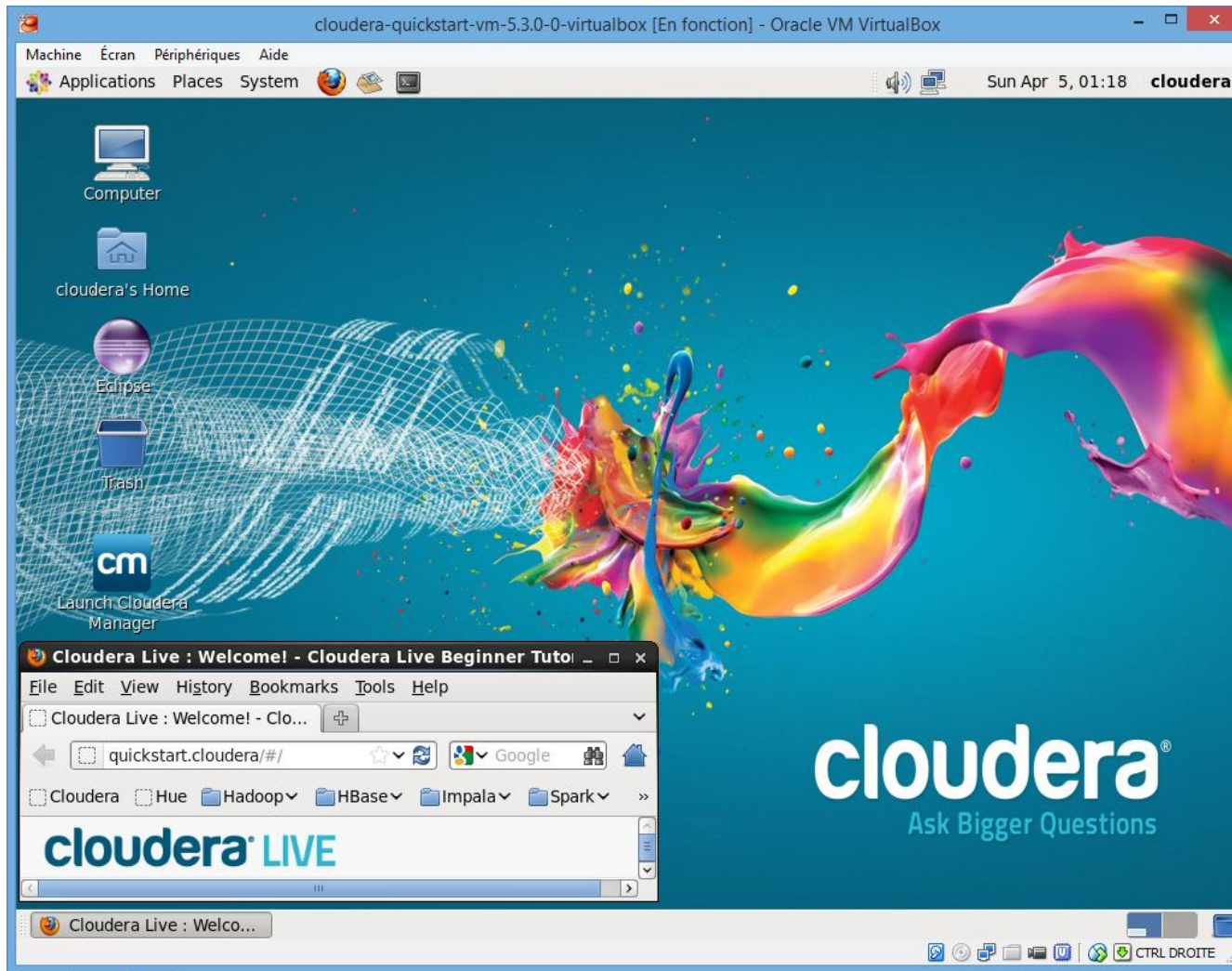


Démarrage de la machine virtuelle. Attention, cela peut prendre un certain temps.

Remarque : Le message "vt-x is disabled in the bios" peut apparaître sur certains ordinateurs. La machine virtuelle n'est pas démarrée. Il faut alors activer la technologie de virtualisation d'Intel dans le BIOS du PC c.-à-d. le mettre à ENABLED.



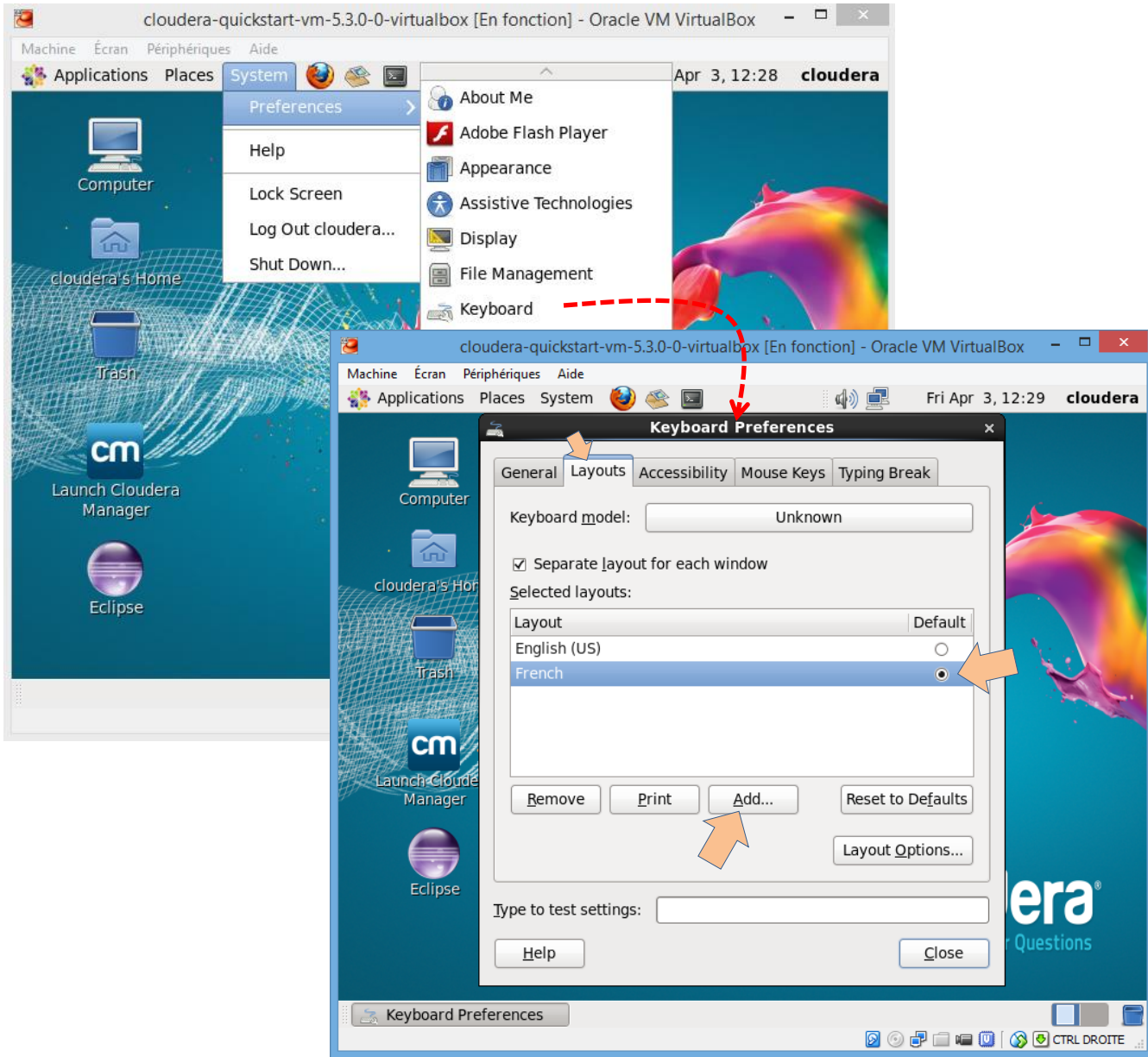
Démarrage de la machine virtuelle (2/3)



Au démarrage de la machine, **Hadoop est fonctionnel**. Il n'y a pas de manipulations particulières à faire de ce côté-là.



Démarrage de la machine virtuelle (3/3)



Installation du clavier français pour les différentes manipulations ultérieures. Nous aurons à saisir des commandes dans un terminal notamment.

Il faut cliquer sur ADD, puis sélectionner FRENCH et le choisir comme clavier par défaut.

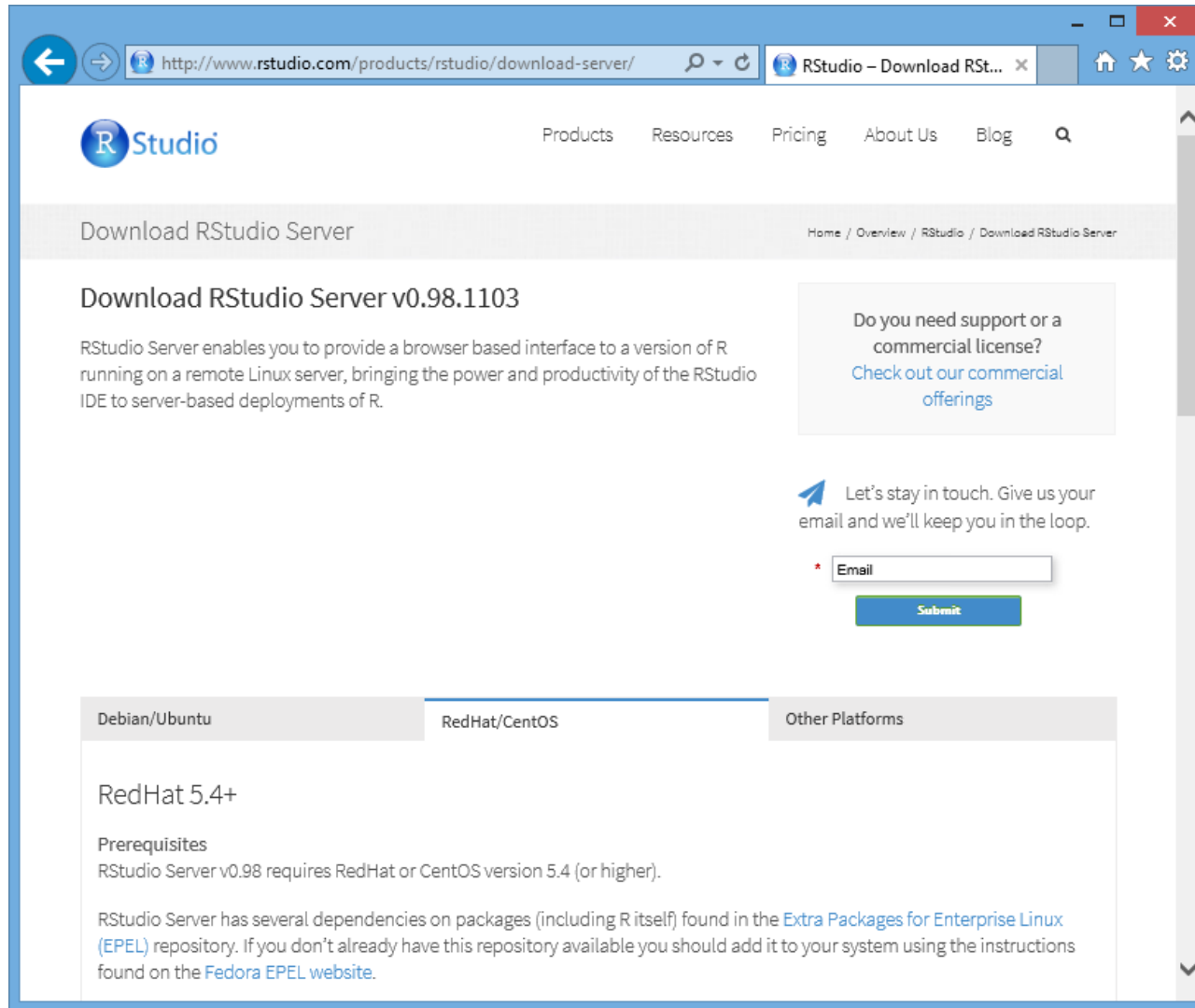
Installation et configuration de R et RStudio Server

Le logiciel R sert à interpréter nos programmes.

RStudio Server nous permet d'accéder à l'éditeur de code RStudio via un navigateur. On peut accéder au serveur en utilisant son numéro IP. Dans notre cas, la même machine fait office de client et de serveur, nous utilisons 127.0.0.1 (machine locale). Mais la démarche est très facilement généralisable à un accès distant en utilisant le numéro IP d'un serveur correctement configuré.



Installation de R et RStudio Server (1/2)



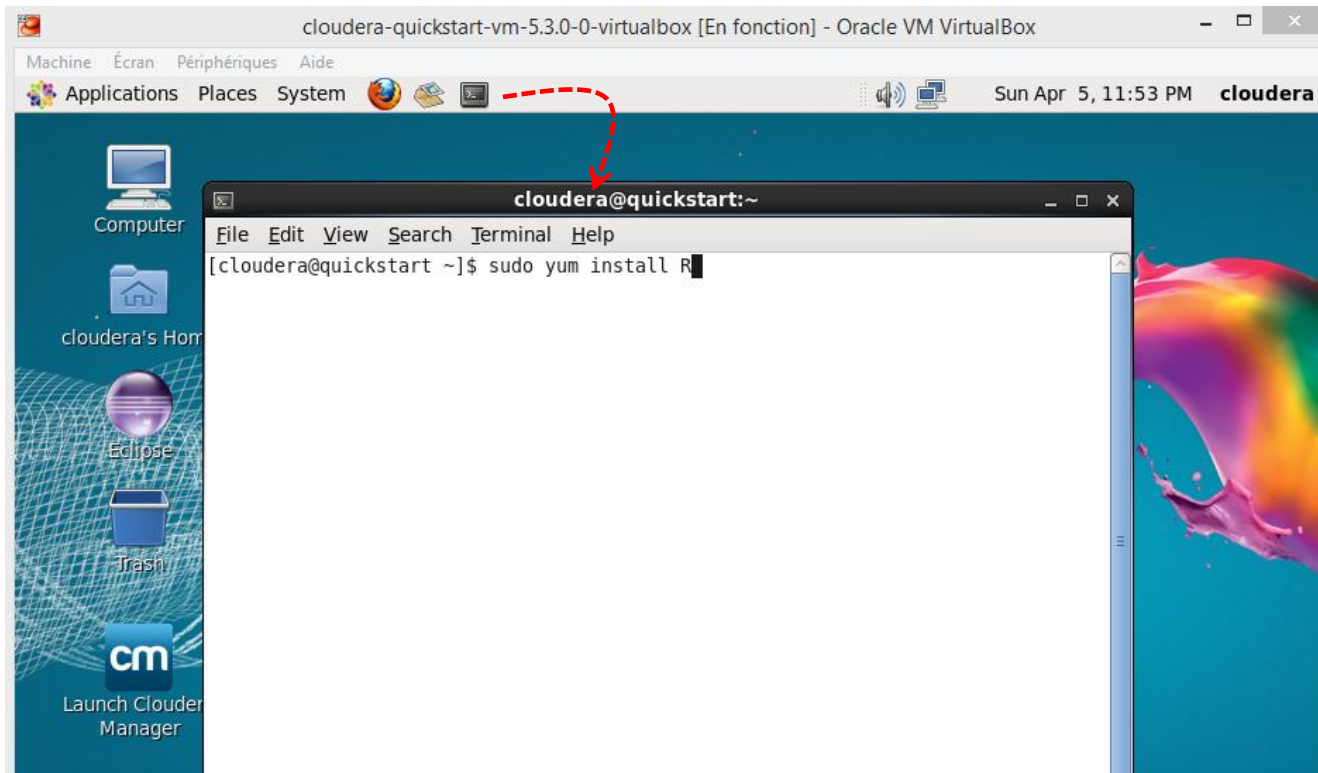
The screenshot shows the RStudio website's download page for RStudio Server. The browser address bar displays <http://www.rstudio.com/products/rstudio/download-server/>. The page features a navigation menu with links for Products, Resources, Pricing, About Us, and Blog. The main heading is "Download RStudio Server", with a breadcrumb trail: Home / Overview / RStudio / Download RStudio Server. The version "Download RStudio Server v0.98.1103" is prominently displayed. Below this, a paragraph explains that RStudio Server provides a browser-based interface to R running on a remote Linux server. A callout box asks if the user needs support or a commercial license, with a link to "Check out our commercial offerings". A newsletter sign-up form prompts users to "Let's stay in touch" and includes an "Email" input field and a "Submit" button. At the bottom, there are tabs for "Debian/Ubuntu", "RedHat/CentOS", and "Other Platforms". The "RedHat/CentOS" tab is active, showing "RedHat 5.4+" and "Prerequisites" which state that RStudio Server v0.98 requires RedHat or CentOS version 5.4 or higher. It also mentions dependencies on packages from the Extra Packages for Enterprise Linux (EPEL) repository.

Toutes les instructions adéquates sont décrites sur le site de RSTUDIO. Nous choisissons CentOS puisque c'est le système qui a été installé avec la distribution CLOUDERA.

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



Installation de R et RStudio Server (2/2)



Nous ouvrons un terminal pour pouvoir introduire les commandes d'installation.

Voici les instructions à saisir dans le terminal

Installation de R

```
$ sudo yum install R
```

Installation de Rstudio Server (64 bit)

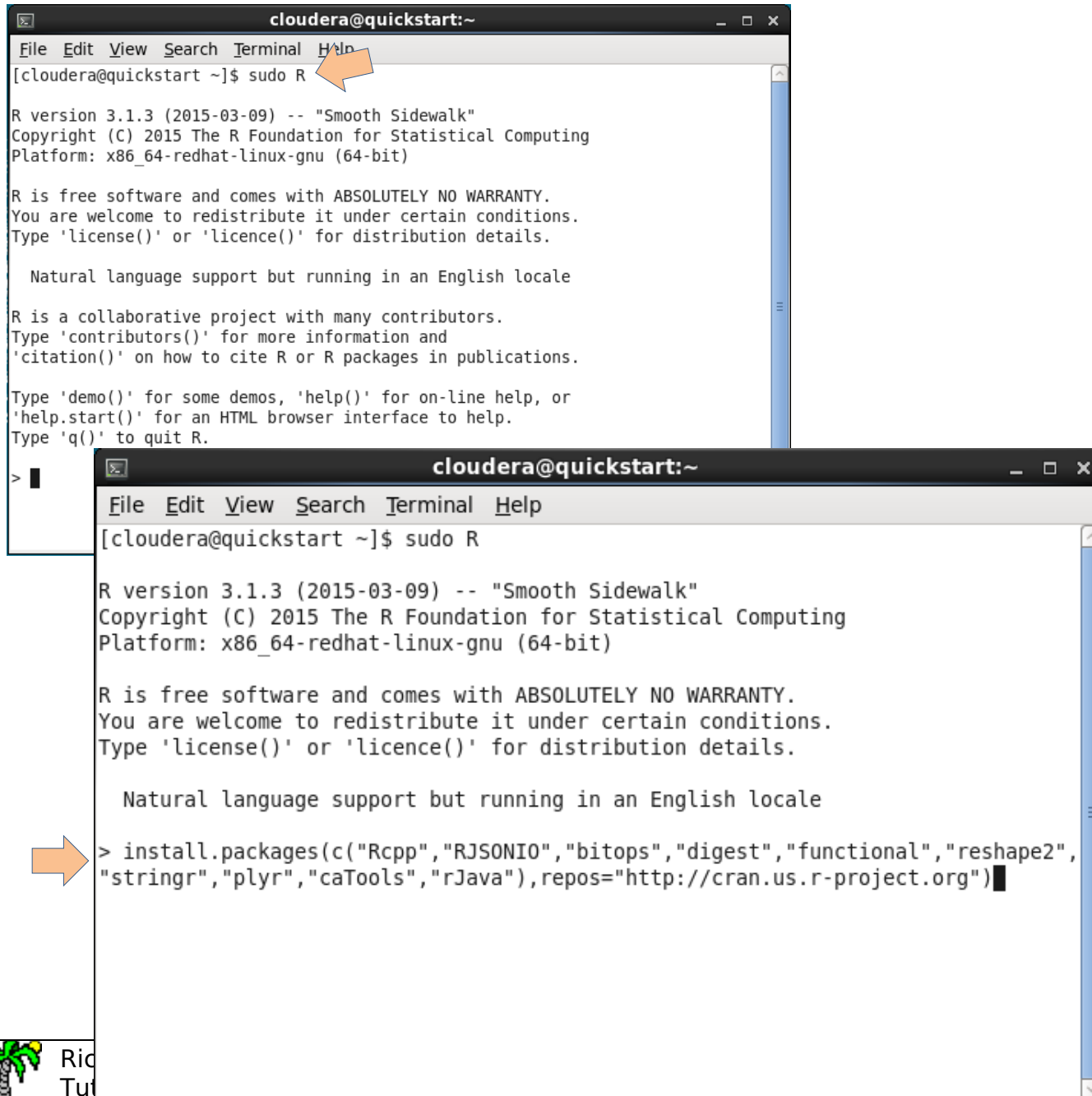
```
$ sudo yum install openssl1098e # Required only for RedHat/CentOS 6 and 7
```

```
$ 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 des packages référencés sous R



The image shows two terminal windows from a Cloudera Quickstart environment. The top window shows the R version 3.1.3 (2015-03-09) being installed. An orange arrow points to the command `sudo R` in the terminal prompt. The bottom window shows the same R version being installed, followed by the command `install.packages(c("Rcpp", "RJSONIO", "bitops", "digest", "functional", "reshape2", "stringr", "plyr", "caTools", "rJava"), repos="http://cran.us.r-project.org")`. An orange arrow points to this command.

```
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.  
  
> █  
  
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") █
```

Pour pouvoir programmer sous Hadoop, nous avons besoin d'une série de packages référencés sur le CRAN.

Nous les installons en démarrant R en mode administrateur via le terminal c.-à-d. avec la commande `$ sudo R`

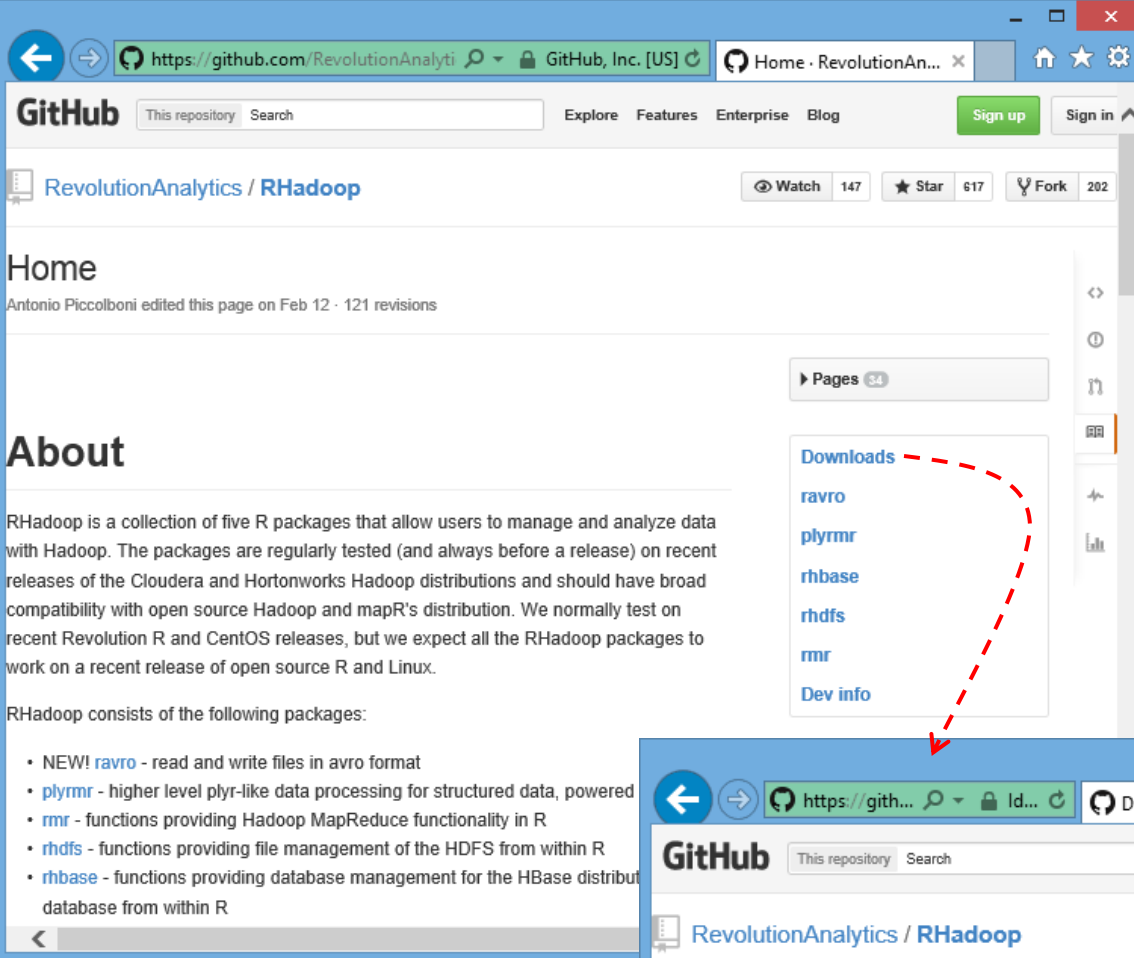
Vient ensuite la commande `install.packages` indiquant les librairies et le dépôt source (repos).

Installation des packages spécifiques sous R (1/3)

Nous souhaitons utiliser 2 packages de la collection RHadoop de la société RevolutionAnalytics.

rmr pour pouvoir programmer selon le paradigme MAPREDUCE

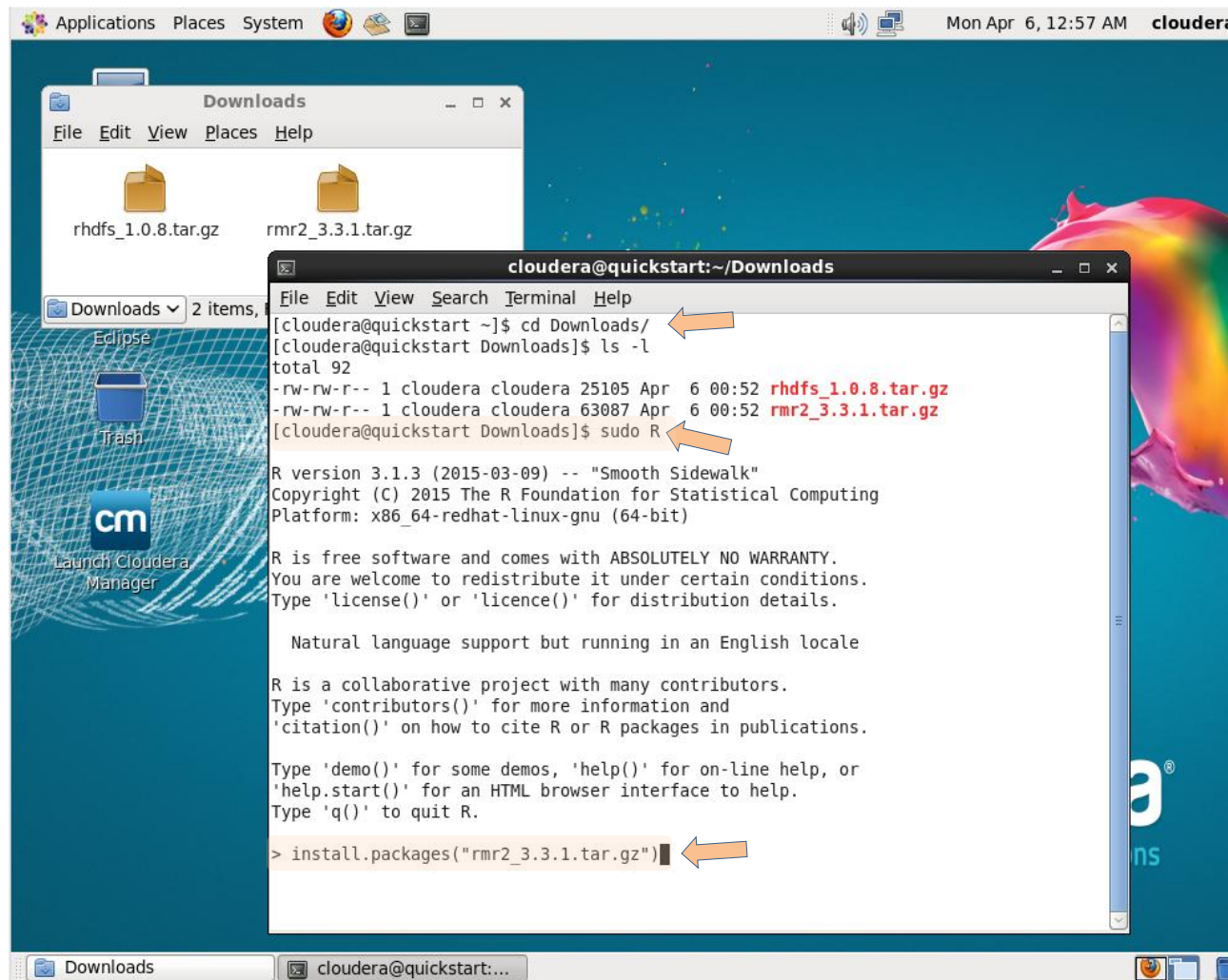
rhdfs pour disposer des commandes permettant de manipuler des fichiers sur le système HDFS



Nous chargeons les deux fichiers dans le dossier « Downloads »



Installation des packages spécifiques sous R (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")
```

De nouveau, nous lançons le terminal, nous changeons de répertoire (absolument nécessaire pour éviter d'avoir à spécifier des chemins) et nous démarrons R en mode administrateur.

`install.packages()` permet aussi d'installer des librairies à partir des fichiers chargés localement.



Installation des packages spécifiques sous R (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")
```

Attention, pour le package « rhdfs », il faut d'abord spécifier les valeurs des **variables d'environnement** - indiquant la localisation du système hadoop sur notre machine - avant de pouvoir l'installer.

Nous utilisons la commande `Sys.setenv()` sous R.



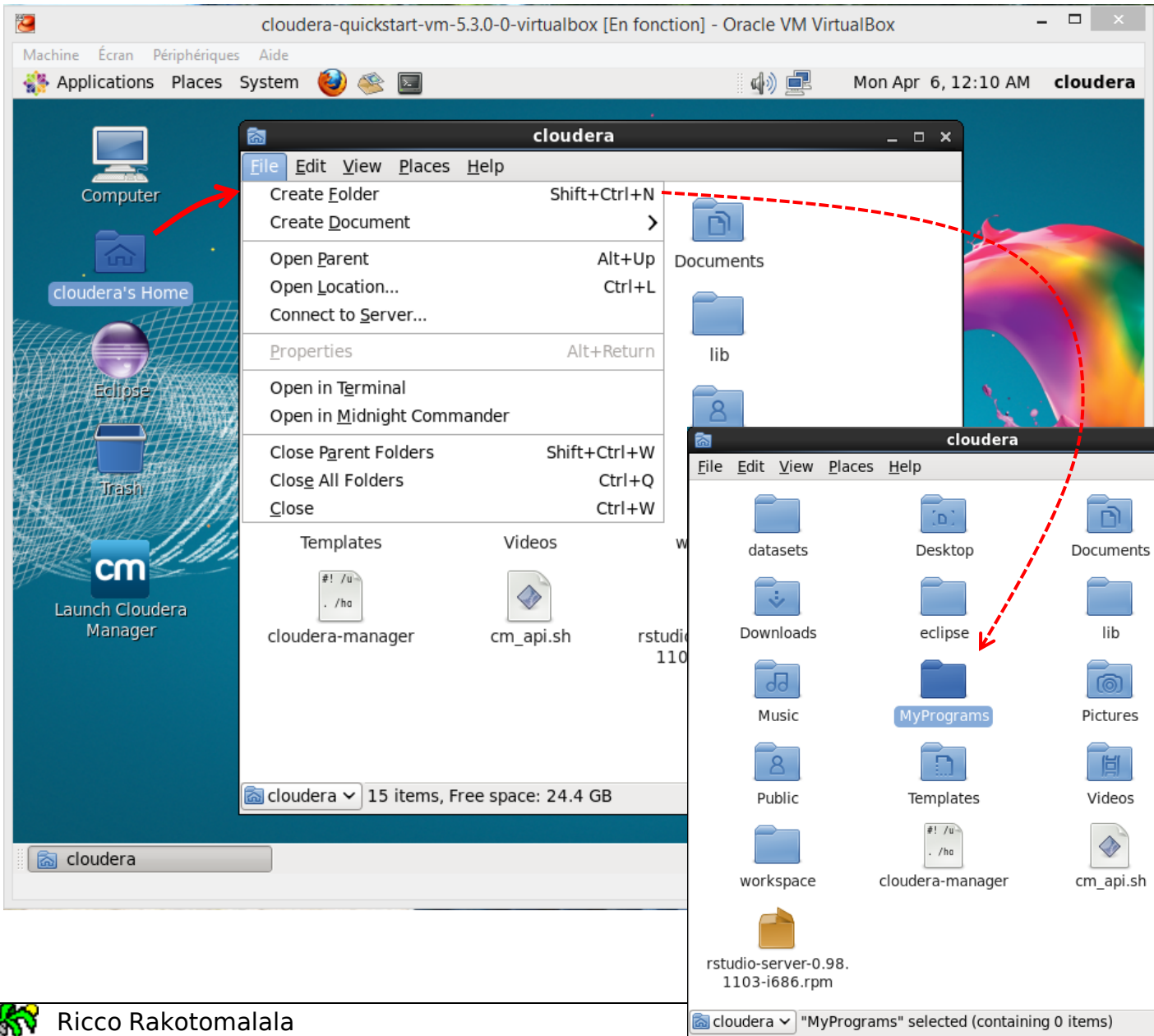
Programmation R sous Hadoop (1)

Le principe MapReduce

Première tentative. Des données sont générées en mémoire, elles sont stockées dans un fichier temporaire puis la fonction `mapreduce()` de `rmr2` est appelée. Elle se charge d'appeler successivement en interne les fonction `map()` et `reduce()` que nous avons préalablement codées.



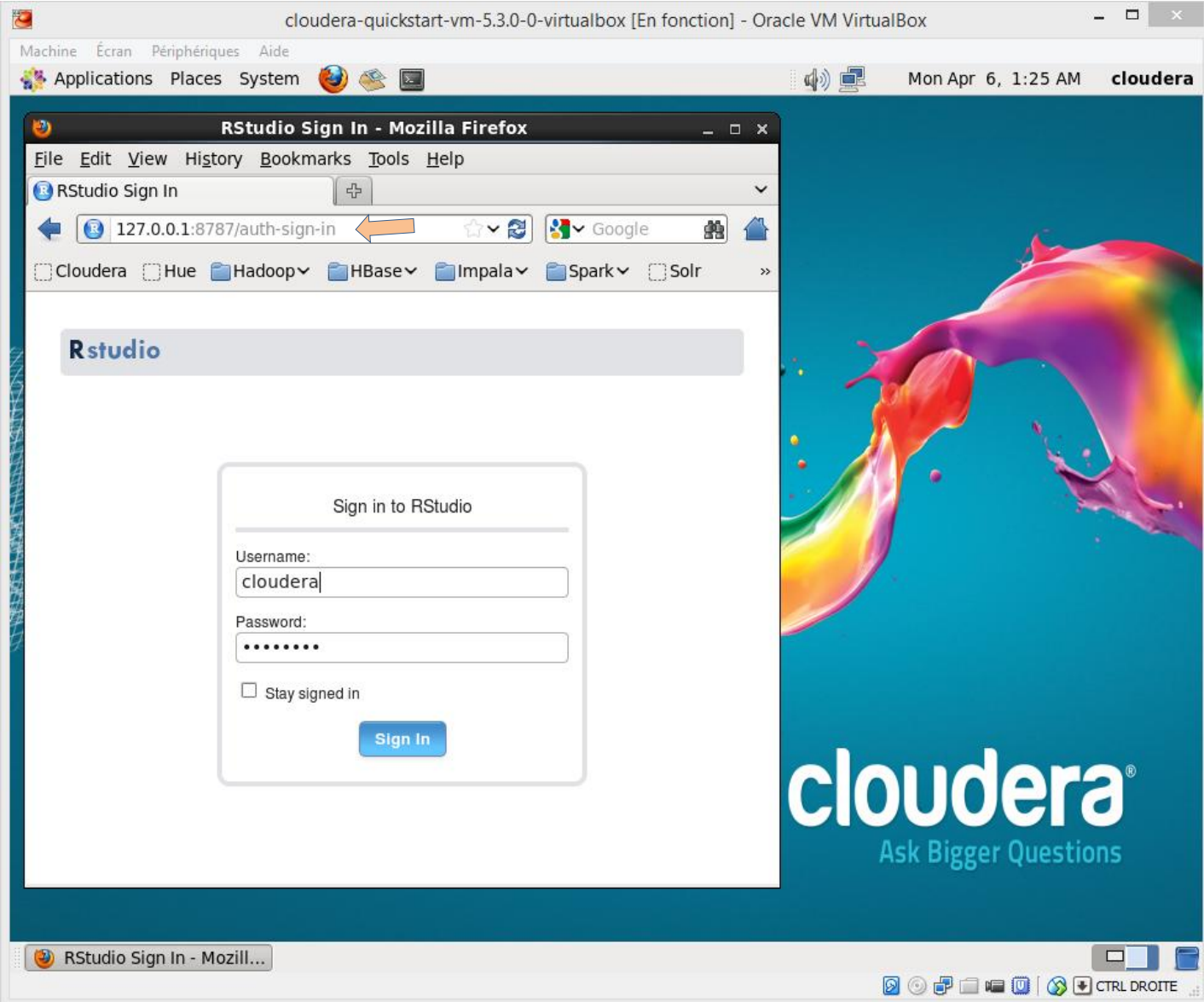
Création d'un dossier pour nos programmes



Les programmes que nous allons écrire seront stockés dans ce dossier



Accès à RStudio via un navigateur web (1/2)

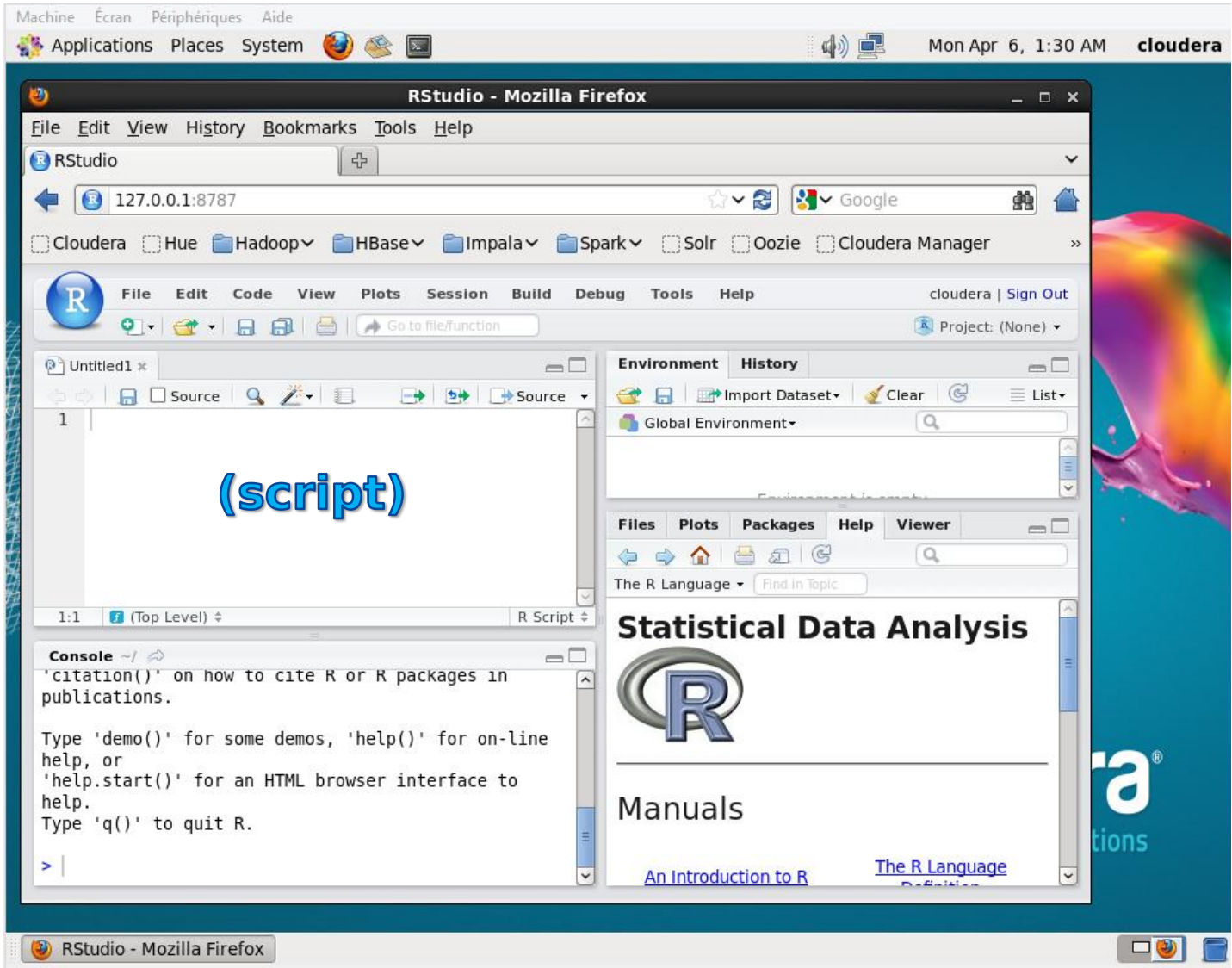


Le serveur est sur la machine locale. Nous utilisons l'IP : **127.0.0.1**

Le port est spécifique, nous utilisons **8787**

Le login est standardisé, username : **cloudera**
password : **cloudera**

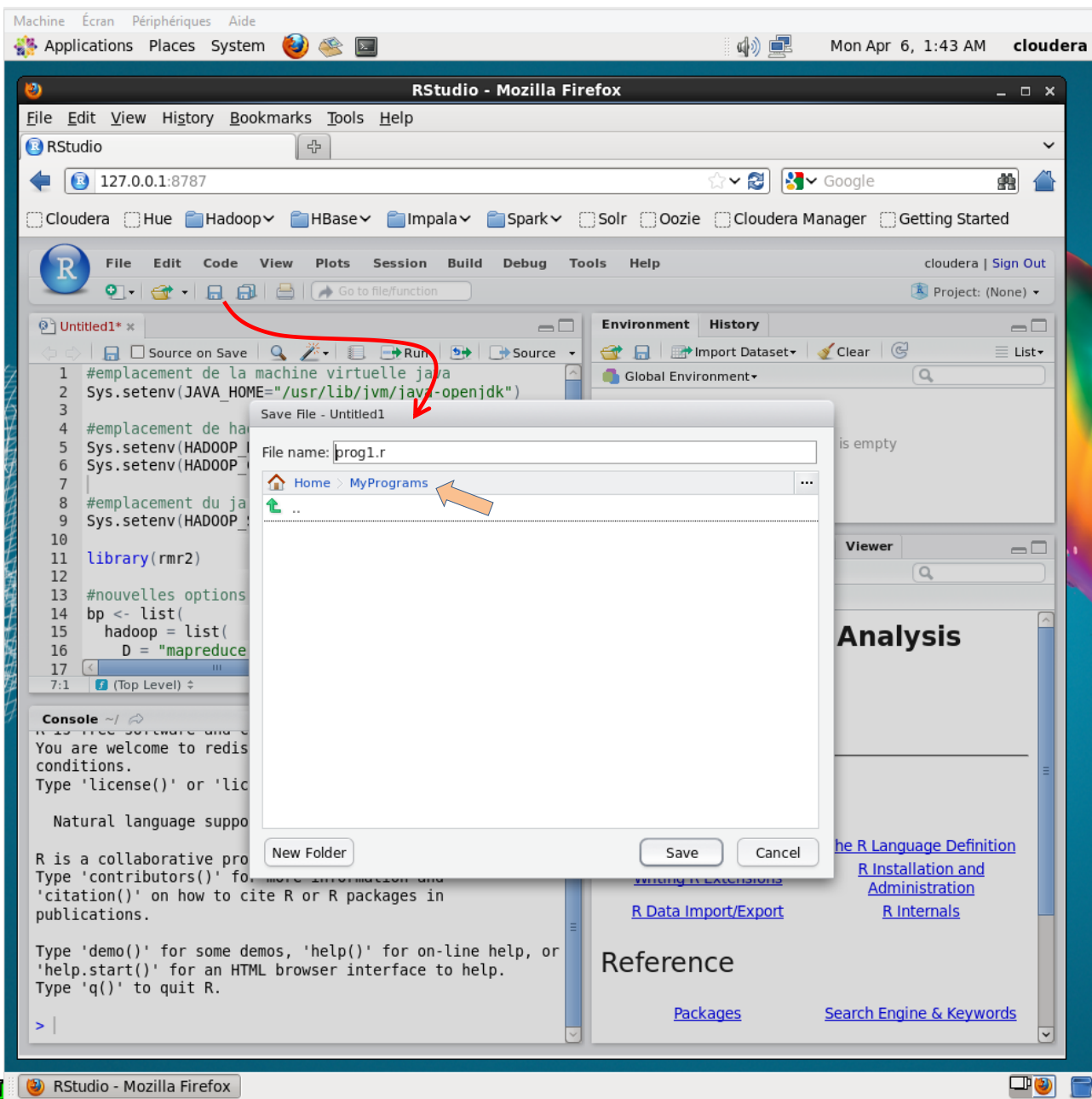
Accès à RStudio via un navigateur web (2/2)



Nous retrouvons l'environnement de développement habituel de RStudio.

Remarque : Si l'éditeur de script n'est pas disponible, nous pouvons créer un fichier vierge en cliquant sur le menu FILE / NEW FILE / R SCRIPT

Programmation MapReduce sur des données créées en mémoire (1/4)



Le programme une fois rédigé est sauvegardé dans le dossier « MyPrograms » créé préalablement.

```
#emplacement de la machine virtuelle java
Sys.setenv(JAVA_HOME="/usr/lib/jvm/java-openjdk")

#emplacement de hadoop
Sys.setenv(HADOOP_HOME="/usr/lib/hadoop")
Sys.setenv(HADOOP_CMD="/usr/lib/hadoop/bin/hadoop")

#emplacement du jar se chargeant du streaming
Sys.setenv(HADOOP_STREAMING="/usr/lib/hadoop-mapreduce/hadoop-streaming-2.5.0-cdh5.3.0.jar")

#chargement du package « rmr2 »
library(rmr2)

#nouvelles options - très importantes pour la gestion de la mémoire
#sinon le message d'erreur « Java heap space » sera renvoyé durant l'exécution
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 des paramètres
rmr.options(backend.parameters = bp)

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



Programmation MapReduce (3/4) – Ecriture des fonctions map() et reduce(). Application sur un vecteur de mots.

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

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

#données créées en mémoire
b <- c("one","two","one","one","two")

#transformées et copiées dans un fichier temporaire sur HDFS
a <- to.dfs(b)

#lancement de la procédure mapreduce
sortie <- mapreduce(input=a, map=mymap, reduce=myreduce)

#récupération de la sortie temporaire sur HDFS
#affichage dans le terminal R
print(from.dfs(sortie))
```

A la sortie de MAP

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

Partition basée sur la clé

Key = one	1	1	1
Key = two	1	1	

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

A la sortie de REDUCE

Key = one	3
Key = two	2



```
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%
```

Un vecteur de mots est généré en mémoire

Ces données en mémoire sont copiées dans un fichier temporaire sur HDFS, qui sert d'entrée (**input**) pour la fonction `mapreduce()` de `rmr2`

Initialisation de l'exécution. Puis appel interne des fonctions `map()` et `reduce()`

```
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

> |
```

A l'issue de l'exécution, la sortie est copiée dans un fichier temporaire sur HDFS

Que l'on peut charger en mémoire dans un format reconnaissable par R et afficher



Programmation R sous Hadoop (2)

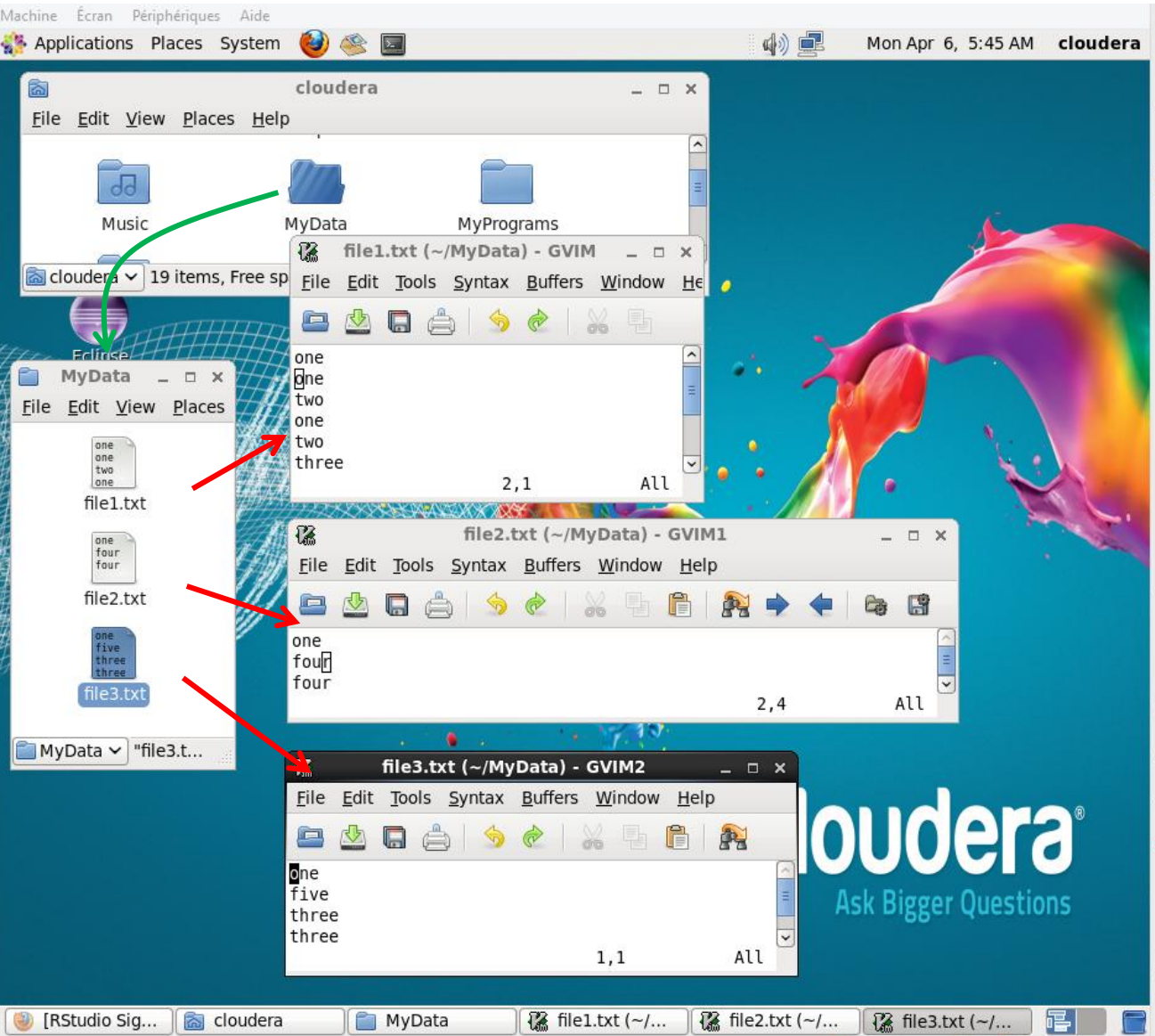
Accès à des fichiers stockés sur HDFS

De manière plus réaliste maintenant, nous accédons à un ensemble de fichiers stockés dans un dossier du système de fichiers HDFS.

Il nous faut donc tout d'abord expliciter le format et le contenu de ces fichiers, puis montrer comment nous pouvons les copier dans un dossier spécialement créé à cet effet sur HDFS, enfin modifier notre programme pour qu'il traite ces fichiers.



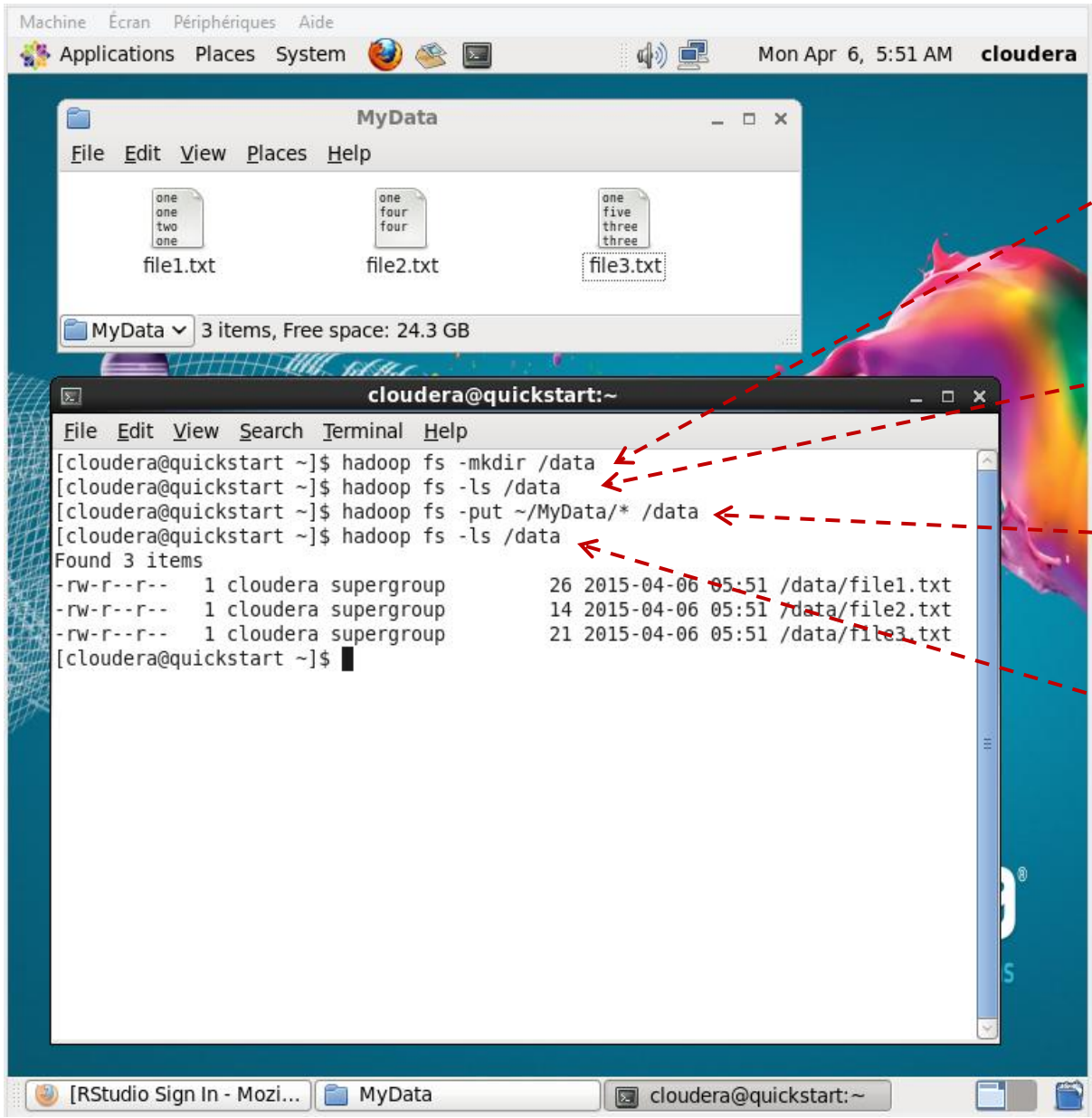
Création de 3 fichiers de données dans un répertoire quelconque



3 fichiers sont créés dans notre dossier personnel « MyData » : file1.txt, file2.txt, file3. txt

Chacun contient une liste de mots que l'on visualiser ici.

Copies des fichiers dans un dossier « Data » sur HDFS



Création d'un dossier **/data** sur HDFS. Qu'il y ait un nœud ou plus sur le cluster n'est pas un souci, pour nous c'est transparent.

Le dossier est vide pour l'instant

Nous copions nos fichiers de notre système local vers HDFS

Les fichiers sont bien visibles dans le dossier **/data** sur HDFS.



Programmation à partir de fichiers sur HDFS

La partie spécifications des paramètres est identique à précédemment. Idem en ce qui concerne les fonction `map()` et `reduce()`

La procédure se charge de scanner l'ensemble des fichiers contenus dans le dossier `/data`

```
#on a besoin de la librairie hdfs maintenant
library(rhdfs)

#initialiser l'accès à hdfs
hdfs.init()

#afficher le contenu du dossier data
hdfs.ls("/data")

#programmation en accédant au contenu du dossier /data sur HDFS
sortie.bis <- mapreduce(input="/data", input.format="text", map=mymap, reduce=myreduce)
print(from.dfs(sortie.bis))
```

Ces fichiers sont au format « texte » et non au format « natif ». On ne passe pas par l'étape intermédiaire `to.dfs()` ici.



Sorties du programme (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%
```

Les 3 fichiers du dossier **/data** sont bien visibles.

Et ils vont être traités.

On peut suivre à la console l'évolution du processus.

```
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
```



Sorties du programme (2/2)

```
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
```

```
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() a été appelé 5 fois parce qu'il y a 5 valeurs différentes de clés.

Les résultats sont stockés dans un fichier temporaire parce que nous n'avons pas renseigné le paramètre « output ».

Et nous avons bien les comptages consolidés des 3 fichiers : file1.txt, file2.txt et file3.txt !

Bibliographie



Tutoriel Tanagra, « [MapReduce avec R](#) », février 2015.

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

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

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

RevolutionAnalytics, « [RHadoop](#) ».

