Learn to scrap dynamic content using Docker and Selenium

```
library(rvest)
library(tidyverse)
library(knitr)
library(plyr)
library(dplyr)
library(jsonlite)
library(lubridate)
library(RSelenium)
```

1 Introduction

We see two methods to capture departures and arrivals data for airport on **FlightRadar** website, using a browser running on a server, and using an XHR request.

For each airports page, FlightRadar website offer the possibility to see general informations, departures and arrivals flights information. For this tutorial we try to scrape the Bordeaux Mérignac Airport BOD (https://www.flightradar24.com/data/airports/bod/) arrival flights page (https://www.flightradar24.com/data/airports/bod/arrivals)

As you could see if you go to departures pages, you have two interesting buttons, one at the top of the page, and one at the bottom of the page.

To display all data available (something like 24h of past and future departures/arrivals), we simulate multiples clic on this two buttons, and we stop this behavior only when these buttons disapears from the page.

At the end of this tutorial, we present how to use Docker (https://www.docker.com/) container technology to export our webscrapping script on some remote server for an autonomous execution 24/24h.

2 Using Selenium server browser

2.1 Install and use Selenium

2.1.1 The Selenium RLibrary

Due to some defense created by webmaster to protect their data (test javascript, user-agent, infinite loading, etc.), you need to simulate an human behavior, if possible using a real browser.

To be short, Selenium (https://docs.seleniumhq.org/) is a multi-tools project focusing on task automation to test web aplication. It works with lots of Internet browsers, and lot of operating systems.

Selenium Webdriver give to developper an API to interact/pilot an headless internet browser without opening it. So, you, developper, you could use this API with your favorite langage (Java, Python, R, etc.) to sent commands to browser in order to navigate, move your mouse, click on DOM element, sent keyboard output to input forms, inject javascript, capture image of the page, extract html, etc.

First, you need to install and load RSelenium package, the R bindings library for Selenium Webdriver API :

```
install.packages("devtools")
devtools::install_github("ropensci/RSelenium")
```

Depending of your existing configuration and OS you probably need to install some dependent software packages.

It's possible to use directly Selenium in connection with your browser, but we prefer to use directly a server version. Why ? Because using server version of Selenium, you have the possibility :

a. to sent command on local or remote server running Selenium
b. which run a different browsers and/or OS,
c. to distribute tests over multiple machines.

Selenium is a fast moving project, and some release are really buggy, so try to choose a stable version, and don't desperate.

2.1.2 Run a Selenium server

!! Before continuing, read the documentation on Docker at the bottom of this document, it explain what is really Docker/images/container, and it explain how to install images/containers on your system !!

When it's done, we `pull and run` one of (https://github.com/SeleniumHQ/docker-selenium) Docker Selenium-Server image using terminal. For this tutorial we use **Firefox** !

In classic context (good internet connection), we pull images directly from the Docker Hub server, a central repository like CRAN for R.

```
sudo docker pull selenium/standalone-firefox:3.14.0-arsenic
```

But, because the image is heavy in size (1 GO for the two images used in this tutorial), we prefer to directly load the image given by USB key by your teachers. Open a terminal on the folder where located the images.

```
sudo docker load --input=r-alpine.tar
sudo docker load --input=rSelenium.tar
```

Create the Selenium container which contain Firefox :

```
sudo docker run --shm-size=2g --name selenium -d -p 4445:4444 selenium/standalone-firefox:3.14.0-arsenic
```

Type `sudo docker ps` to see if server correctly run and listen to port **4445**

2.1.3 Connect to Selenium Server

Connect and open the browser on the server.

```
user_agent_list = c ("Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.0", "Mozilla/5.0 (Windows NT 6.1; r
v:27.3) Gecko/20130101 Firefox/27.3", "Mozilla/5.0 (X11; Linux x86_64; rv:28.0) Gecko/20100101 Firefox/28.0", "Mozilla/5.
0 (Windows NT 6.2; Win64; x64;) Gecko/20100101 Firefox/20.0", "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.
36 (KHTML, like Gecko) Chrome/42.0.2311.135 Safari/537.36 Edge/12.246", "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/41.0.2226.0 Safari/537.36", "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_0) AppleWebKi
t/537.36 (KHTML, like Gecko) Chrome/32.0.1664.3 Safari/537.36", "Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/27.0.1453.93 Safari/537.36","Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chro
me/27.0.1453.90 Safari/537.36", "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/535.11 (KHTML, like Gecko) Ubuntu/11.04 Chro
mium/17.0.963.56 Chrome/17.0.963.56 Safari/535.11")

fprof <- makeFirefoxProfile(list(general.useragent.override=sample(user_agent_list,1)))

remDr <- remoteDriver(remoteServerAddr = "localhost", port = 4445L, extraCapabilities = fprof )
remDr$open()
```

```
## [1] "Connecting to remote server"
## $acceptInsecureCerts
## [1] FALSE
##
## $browserName
## [1] "firefox"
##
## $browserVersion
## [1] "61.0.1"
##
## $`moz:accessibilityChecks`
## [1] FALSE
##
## $`moz:headless`
## [1] FALSE
##
## $`moz:processID`
## [1] 73
##
## $`moz:profile`
## [1] "/tmp/rust_mozprofile.hYVGLKxPdfkd"
##
## $`moz:useNonSpecCompliantPointerOrigin`
## [1] FALSE
##
## $`moz:webdriverClick`
## [1] TRUE
##
## $pageLoadStrategy
## [1] "normal"
##
## $platformName
## [1] "linux"
##
## $platformVersion
## [1] "4.15.0-34-generic"
##
## $rotatable
## [1] FALSE
##
## $timeouts
## $timeouts$implicit
## [1] 0
##
## $timeouts$pageLoad
## [1] 300000
##
## $timeouts$script
## [1] 30000
##
##
## $webdriver.remote.sessionid
## [1] "413c56f3-b518-4c34-8aeb-37f2fb3cd9c9"
##
## $id
## [1] "413c56f3-b518-4c34-8aeb-37f2fb3cd9c9"
```

```
remDr$maxWindowSize()

remDr$executeScript("return navigator.userAgent;", list(""))
```

```
## [[1]]
## [1] "Mozilla/5.0 (X11; Linux x86_64; rv:28.0) Gecko/20100101 Firefox/28.0"
```

2.1.4 Basic command for RSelenium

Johnd Harrison (https://github.com/johndharrison), the creator and first commiter of RSelenium binding library for Selenium, create a big tutorial with lots of commands covered : https://rpubs.com/johndharrison/RSelenium-Basics (https://rpubs.com/johndharrison/RSelenium-Basics)

Some of them :

- `remDr$maxWindowSize()` : maximize windows of the browser.
- `remDr$navigate("https://www.google.fr")` : navigate to url
- `remDr$screenshot(display = TRUE)` : take a screenshoot of the webpage and display it in RStudio Viewer
- `remDr$findElement(...)` : Find and element in the html structure, using different method : xpath, css, etc.
- `remDr$executeScript(...)` : Execute a js script in the remote browser
- `remDr$clickElement(...)` : Clic on element

2.2 From analysis to automation

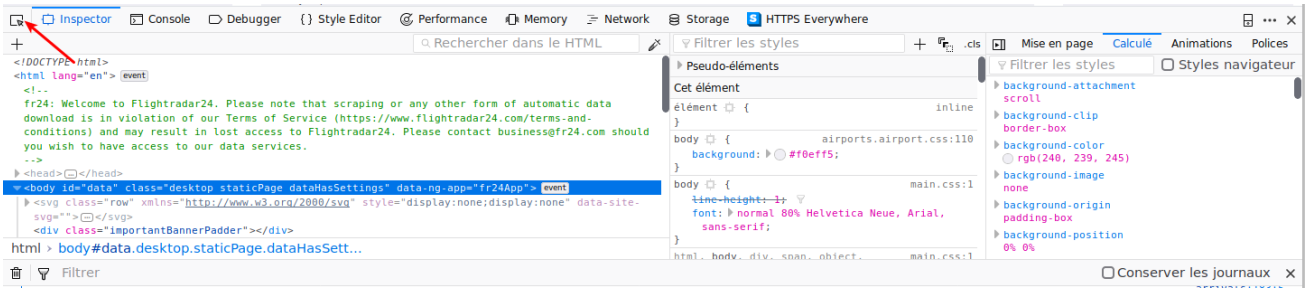
2.2.1 Analysis of website HTML structure

Open `Web Developer tools` in your favorite browser on the arrivals webpage of BOD :
`https://www.flightradar24.com/data/airports/bod/arrivals`

We investigate what happens in the html code when we clic the **load earlier** or **load later** button. Why we do that ? To understand how we could automate things later.

Because we want to automate clic on this two buttons, we need to understand WHEN we need to stop clicking :) If we clic an infinite number of time, an error probably trigger when one of the two button disapear.

Select the **Selector tools** and click on the load earlier button.



If you clic the right thing, normaly you have highlighted some part of the html code which interest us :

Now, Iif you highlight and clic with the web tool selector on the load later flights button, you have something like that :

Things are not so very differences between this two buttons objects. It seems that only the timestamp, the data page number and the button text change …

Hightlight and clic one more time on the *load earlier flights* button. Clic another time to load a new page of data. You see that the html code change during the data load to desactivate clic on the button. Not so interesting. Now repeat the clic and stop only when the button disapear on your screen.

Great, a new css style attribute appear to indicate that now this button object is hidden : `style="display: none;"`

How could we re-use this important information during data harvesting to detect if the button is activated/desactivated ? The best solution was to use XPATH query !

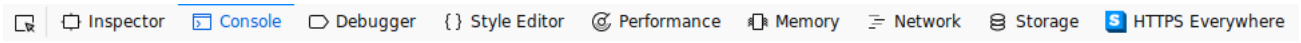
Load the page in the selenium server

```
remDr$navigate("https://www.flightradar24.com/data/airports/bod/arrivals")
Sys.sleep(5) # time to load !
remDr$screenshot(file = "screenshoot.png")
```

2.2.2 Resolve infinite loading using XPath

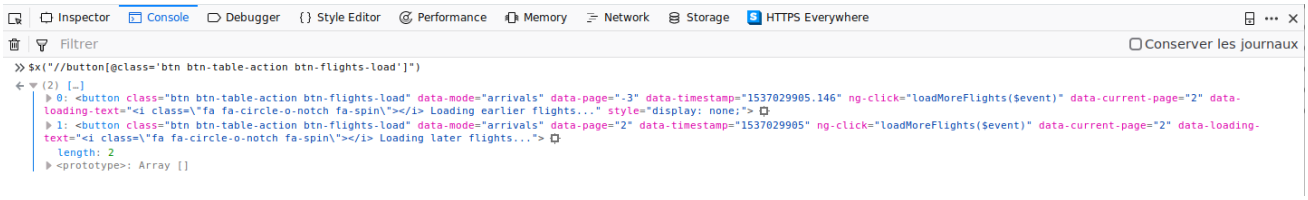
Building XPATH correct expression could be difficult. A good way to test validity of your XPATH expressions was to use an interactive way, using the web developper console. There are some good cheatsheet pages which resume all the possibilities of XPATH : 1 (https://devhints.io/xpath) 2 (https://gist.github.com/LeCoupa/8c305ec8c713aad07b14)

Clic on console tab :



Type this in the console : `$x("//button[@class='btn btn-table-action btn-flights-load']")`

The result is an interactive array you could develop as a tree if you want.



Clic Clic Clic to make disapear one of the loading button, and now we trying to select only the available button. XPATH understand boolean operator (or,and, etc.) so we filter by `@class` and `style` :

`$x("//button[@class='btn btn-table-action btn-flights-load' and contains(@style,'display: none;')]")`

Great, this query return only the valid button. We use later this query to stop our loop of infernal button clic.

Now we try to build this query using RSelenium with `findElement()` function :

```
loadmorebutton <- remDr$findElements(using = 'xpath', "//button[@class='btn btn-table-action btn-flights-load' and not(co
ntains(@style,'display: none;'))]")
```

Display the text of each element retrieved by function `findElements()` using the `getElementText()` function

```
unlist(lapply(loadmorebutton, function(x){x$getElementText()}))
```

```
## [1] "Load earlier flights" "Load later flights"
```

Now, how to simulate a clic on one of this button ?

An easy way was to call `clickElement()` function on the first loadmorebutton webelement :

```
tryCatch({
suppressMessages({
  loadmorebutton[[1]]$clickElement()})},
error = function(e) {
    loadmorebutton[[1]]$errorDetails()$message
  })
```

This command return an error message (if not, you're lucky !), not very explicit, so if you want more details, you could call the function `errorDetails()` like our `trycatch` block.

An element of the webpage overlapp our button, so browser say us that's not possible to clic on this webelement. Use snapshot function to see the page :

```
remDr$screenshot(file = 'screenshoot_overlap.png' )
```

If we hide these elements using XPath and javascript injection, everything goes to normal. First we accept cookies.

```
hideCookie <- function (x){
  cookiesButton <- x$findElement(using = 'xpath',"//div[@class='important-banner__close']")
  cookiesButton$clickElement()
}

hideCookie(remDr)
remDr$screenshot(file = 'screenshoot_hide.png')
```

The navbar element create problem, so we hide it using javascript injection :

```
hideNavBar <- function (x) {
  script <- "document.getElementById('navContainer').hidden = true;"
  x$executeScript(script)
}
hideNavBar(remDr)
```

```
## list()
```

Now you can `clickElement()` without problem :)

```
tryCatch({
suppressMessages({
  loadmorebutton[[1]]$clickElement()})},
error = function(e) {
    remDr$errorDetails()$message
  })
```

See changes before and after using `remDr$screenshot(display = TRUE)` command

2.3 Exercices

- Create a function which clic on button until they all disapears :)

Show/hide

- Extract data for only one day using XPATH and Rvest !

Show/hide

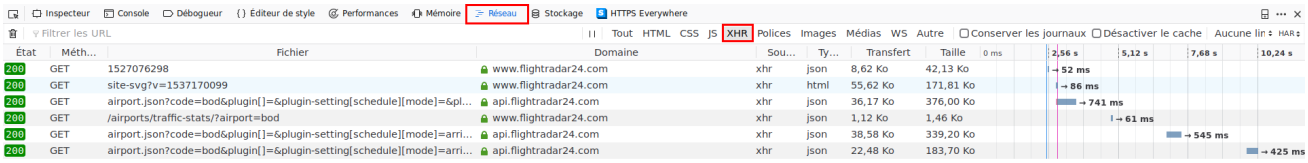
3 Scrap using XHR requests

3.1 Analyse Network data in browser

Sometimes, a defense is also a point of vulnerability. Many site use some sort of internal API to query and feed their website, it's an easy way for developpers to distribute the data. But for us, this is also a perfect data SOF (Single Point Of Failure) (https://en.wikipedia.org/wiki/Single_point_of_failure).

We try to see if this is the case with flight radar :)

Open the dev tools in the browser, clic on **Network** tab, then **XHR** tab.



Lucky guy/girl, do you see it ? Each `GET` query call an `aiport.json` file on the server :

`https://api.flightradar24.com/common/v1/airport.json?code=bod&plugin[]=&plugin-setting[schedule][mode]=&plugin-setting[schedule][timesta`

If we decompose the query, we have :

- an airport code : **bod**
- a timestamp : **1537297562**
- a page number : **1**
- a limit by page : **100**

Copy paste this url in your browser to see how the result json is structured. Insteresting data is located into schedule `result > response > airport > arrivals` :

- **item :** number of total items
- **page :** actual page and number of page
- **timestamp :** date of capture
- **data :** a list of 100 flights corresponding to actual page

3.2 Generate a custom GET Query

We have the capacity to generate custom query to download data at custom timestamp. This query return data structured in json , so we try to convert this data to data.frame using the jsonlite wonderfull package :) Why wonderfull ? Because jsonlite had an option to flatten the structure of json which normally contain data.frame into data.fram into data.frame …

```
timestamp <- as.numeric(as.POSIXct(now()))

url <- paste("https://api.flightradar24.com/common/v1/airport.json?code=bod&plugin[]=&plugin-setting[schedule][mode]=&plu
gin-setting[schedule][timestamp]=",timestamp,"&page=1&limit=100&token=",sep="")

# https://cran.r-project.org/web/packages/jsonlite/vignettes/json-aaquickstart.html
json <- jsonlite::fromJSON(url,flatten = T)
```

We extract information for the first page of Arrivals data collected by airport from json.

```
pageOfData <- json$result$response$airport$pluginData$schedule$arrivals$data
filteredData <- pageOfData %>% select(flight.airline.code.icao, flight.airline.name, flight.airport.origin.name, flight.a
irport.origin.code.icao, flight.airport.origin.position.latitude, flight.airport.origin.position.longitude)

filteredData <- rename(filteredData, c(flight.airline.code.icao = "ICAO", flight.airline.name= "Name", flight.airport.ori
gin.name = "Origin", flight.airport.origin.code.icao="Origin ICAO", flight.airport.origin.position.latitude = "Latitude",
flight.airport.origin.position.longitude = "Longitude" ))

knitr::kable(filteredData, caption = "page 1 of arrival for BOD")
```

page 1 of arrival for BOD

ICAO	Name	Origin	Origin ICAO	Latitude	Longitude
CLG	Chalair Aviation	Nantes Atlantique Airport	LFRS	47.15694	-1.607770
AFR	Air France	Lyon Saint Exupery Airport	LFLL	45.71964	5.089108
AFR	Air France	Paris Orly Airport	LFPO	48.72333	2.379444

ICAO	Name	Origin	Origin ICAO	Latitude	Longitude
CLG	Chalair Aviation	Brest Bretagne Airport	LFRB	48.44722	-4.421660
TUI	TUI	Casablanca Mohammed V International Airport	GMMN	33.36746	-7.589960
EZY	easyJet	Marseille Provence Airport	LFML	43.43666	5.215000
AFR	Air France	Paris Charles de Gaulle Airport	LFPG	49.01252	2.555752
KLM	KLM	Amsterdam Schiphol Airport	EHAM	52.30861	4.763889
HOP	HOP!	Lille Airport	LFQQ	50.56333	3.086944
VOE	Volotea	Bastia Poretta Airport	LFKB	42.55000	9.484722
VOE	Volotea	Palermo Falcone-Borsellino Airport	LICJ	38.17595	13.091010
VOE	Volotea	Luqa Malta International Airport	LMML	35.85749	14.477500
FPO	ASL Airlines France	Paris Charles de Gaulle Airport	LFPG	49.01252	2.555752
HOP	HOP!	Marseille Provence Airport	LFML	43.43666	5.215000
AFR	Air France	Paris Orly Airport	LFPO	48.72333	2.379444
CLG	Chalair Aviation	Rennes Saint-Jacques Airport	LFRN	48.07194	-1.732220
HOP	HOP!	Lyon Saint Exupery Airport	LFLL	45.71964	5.089108
AFR	Air France	Paris Orly Airport	LFPO	48.72333	2.379444
VOE	Volotea	Strasbourg Airport	LFST	48.54361	7.637222
CLG	Chalair Aviation	Nantes Atlantique Airport	LFRS	47.15694	-1.607770
EZY	EasyJet	Nice Cote d'Azur Airport	LFMN	43.66527	7.215000
IBE	Iberia	Madrid Barajas Airport	LEMD	40.49355	-3.566760
SWR	Swiss	Zurich Airport	LSZH	47.46472	8.549167
BAW	British Airways	London Gatwick Airport	EGKK	51.14805	-0.190270
AFR	Air France	Paris Charles de Gaulle Airport	LFPG	49.01252	2.555752
HOP	HOP!	Lille Airport	LFQQ	50.56333	3.086944
HOP	HOP!	Lyon Saint Exupery Airport	LFLL	45.71964	5.089108
EZY	EasyJet	Lyon Saint Exupery Airport	LFLL	45.71964	5.089108
AFR	Air France	Paris Orly Airport	LFPO	48.72333	2.379444
CLG	Chalair Aviation	Brest Bretagne Airport	LFRB	48.44722	-4.421660
CLG	Chalair Aviation	Montpellier Mediterranee Airport	LFMT	43.58333	3.961389
AFR	Air France	Paris Charles de Gaulle Airport	LFPG	49.01252	2.555752
EZY	EasyJet	London Gatwick Airport	EGKK	51.14805	-0.190270
KLM	KLM	Amsterdam Schiphol Airport	EHAM	52.30861	4.763889
AFR	Air France	Paris Orly Airport	LFPO	48.72333	2.379444
PBD	Pobeda	Moscow Vnukovo International Airport	UUWW	55.59153	37.261478
VOE	Volotea	Split Airport	LDSP	43.53894	16.297960
TSC	Air Transat	Montreal Pierre Elliott Trudeau Airport	CYUL	45.47055	-73.740799
CLG	Chalair Aviation	Nantes Atlantique Airport	LFRS	47.15694	-1.607770
RYR	Ryanair	Brussels South Charleroi Airport	EBCI	50.46000	4.452778
VLG	Vueling	Palma de Mallorca Airport	LEPA	39.55167	2.738808
EZY	EasyJet	Bristol Airport	EGGD	51.38266	-2.719080
THY	Turkish Airlines	Istanbul Ataturk International Airport	LTBA	40.97692	28.814600
EZY	EasyJet	Belfast International Airport	EGAA	54.65750	-6.215830
VLG	Vueling	Malaga Costa Del Sol Airport	LEMG	36.67490	-4.499100
BEE	Flybe	Birmingham Airport	EGBB	52.45385	-1.748020
AFR	Air France	Paris Charles de Gaulle Airport	LFPG	49.01252	2.555752
EZY	EasyJet	Milan Malpensa Airport	LIMC	45.63060	8.728111
HOP	HOP!	Nice Cote d'Azur Airport	LFMN	43.66527	7.215000
VOE	Volotea	Santorini Thira National Airport	LGSR	36.39916	25.479330
JAF	TUI fly Belgium	Corfu International Airport	LGKR	39.60194	19.911659
EZY	EasyJet	Lisbon Humberto Delgado Airport	LPPT	38.78131	-9.135910

ICAO	Name	Origin	Origin ICAO	Latitude	Longitude
EZY	EasyJet	Palma de Mallorca Airport	LEPA	39.55167	2.738808
VOE	Volotea	Mahon Menorca Airport	LEMH	39.86259	4.218647
HOP	HOP!	Dusseldorf International Airport	EDDL	51.28945	6.766775
VLG	Vueling	Barcelona El Prat Airport	LEBL	41.29707	2.078463
RYR	Ryanair	London Stansted Airport	EGSS	51.88500	0.235000
AFR	Air France	Paris Orly Airport	LFPO	48.72333	2.379444
VOE	Volotea	Dubrovnik Airport	LDDU	42.56135	18.268240
TAP	TAP Portugal	Lisbon Humberto Delgado Airport	LPPT	38.78131	-9.135910
EIN	Aer Lingus	Dublin Airport	EIDW	53.42138	-6.270000
AFR	Air France	Paris Orly Airport	LFPO	48.72333	2.379444
EZY	EasyJet	Barcelona El Prat Airport	LEBL	41.29707	2.078463
HOP	HOP!	Marseille Provence Airport	LFML	43.43666	5.215000
KLM	KLM	Amsterdam Schiphol Airport	EHAM	52.30861	4.763889
EZY	EasyJet	Rhodes International Airport	LGRP	36.40541	28.086189
CLG	Chalair Aviation	Rennes Saint-Jacques Airport	LFRN	48.07194	-1.732220
EZY	EasyJet	Basel Mulhouse-Freiburg EuroAirport	LFSB	47.59890	7.528300
EZY	EasyJet	London Luton Airport	EGGW	51.87472	-0.368330
VOE	Volotea	Strasbourg Airport	LFST	48.54361	7.637222
HOP	HOP!	Lyon Saint Exupery Airport	LFLL	45.71964	5.089108
EZY	EasyJet	Lille Airport	LFQQ	50.56333	3.086944
AFR	Air France	Paris Charles de Gaulle Airport	LFPG	49.01252	2.555752
HOP	HOP!	Lyon Saint Exupery Airport	LFLL	45.71964	5.089108
EZY	EasyJet	Berlin Schonefeld Airport	EDDB	52.38000	13.522500
IBE	Iberia	Madrid Barajas Airport	LEMD	40.49355	-3.566760
AFR	Air France	Paris Orly Airport	LFPO	48.72333	2.379444
DLH	Lufthansa	Frankfurt Airport	EDDF	50.02642	8.543125
BEL	Brussels Airlines	Brussels Airport	EBBR	50.90138	4.484444
EZY	EasyJet	Geneva International Airport	LSGG	46.23806	6.108950
CLG	Chalair Aviation	Brest Bretagne Airport	LFRB	48.44722	-4.421660
VOE	Volotea	Malaga Costa Del Sol Airport	LEMG	36.67490	-4.499100
HOP	HOP!	Lille Airport	LFQQ	50.56333	3.086944
FPO	ASL Airlines France	Faro Airport	LPFR	37.01442	-7.965910
SWR	Swiss	Zurich Airport	LSZH	47.46472	8.549167
CLG	Chalair Aviation	Brest Bretagne Airport	LFRB	48.44722	-4.421660
EZY	EasyJet	Lyon Saint Exupery Airport	LFLL	45.71964	5.089108
AFR	Air France	Paris Orly Airport	LFPO	48.72333	2.379444
CLG	Chalair Aviation	Montpellier Mediterranee Airport	LFMT	43.58333	3.961389
HOP	HOP!	Marseille Provence Airport	LFML	43.43666	5.215000
EZY	EasyJet	Nice Cote d'Azur Airport	LFMN	43.66527	7.215000
AFR	Air France	Paris Charles de Gaulle Airport	LFPG	49.01252	2.555752
EZY	easyJet (Europcar Livery)	Amsterdam Schiphol Airport	EHAM	52.30861	4.763889
VOE	Volotea	Toulon-Hyeres Airport	LFTH	43.09734	6.146031
AFR	Air France	Paris Orly Airport	LFPO	48.72333	2.379444
HOP	HOP!	Lyon Saint Exupery Airport	LFLL	45.71964	5.089108
EZY	EasyJet	Lyon Saint Exupery Airport	LFLL	45.71964	5.089108
FPO	ASL Airlines France	Paris Charles de Gaulle Airport	LFPG	49.01252	2.555752
CLG	Chalair Aviation	Nantes Atlantique Airport	LFRS	47.15694	-1.607770
EZY	easyJet	Brussels Airport	EBBR	50.90138	4.484444

3.3 Exercices

- Get all pages of arrivals data by generating the correct query to API :)

4 Automation of scraping using Docker containers !

4.1 Why ?

This is the ultimate and probably the most complex part of this big tutorial.

In real webscraping project, there are two possible use case :

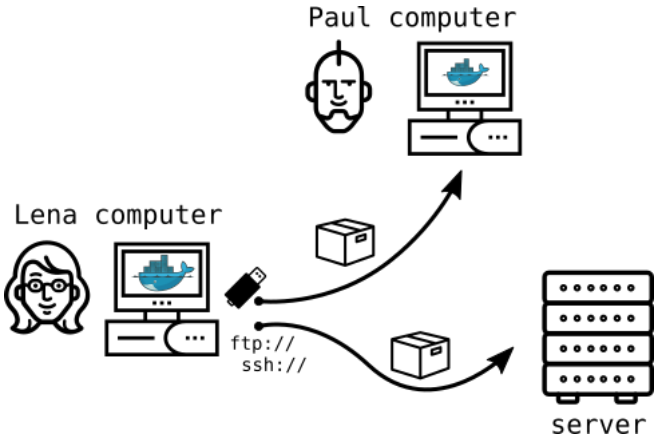
- a one shoot data harvesting or
- a daily/monthly/etc. data harvesting.

Take a very practical example, if you need to collect **one year of data on a daily basis** you cannot use your personnal computer. You need to connect and run your program from a distant server (somewhere on internet).

4.2 What is Docker ?

To be really really short on subject, Docker (https://en.wikipedia.org/wiki/Docker_(software)) is a technology which encapsulate one or multiple software with all theirs dependencies into an isolated (and if possible immutable) container which run on top of any system interoperable with Docker tools (Window/Linux/Mac). You could run multiple isolated containers (A, B , ...), with capacity to exchange informations using a common dedicated local network, on the same machine.

It was a very interesting technology because **if you develop a program in a container on your local machine, it works on any server compatible with Docker.** If you know the concept of Virtual Machine (VM) (https://en.wikipedia.org/wiki/Virtual_machine), the idea is the same, but Docker is a **more efficient** technology (see this comparison (https://www.docker.com/resources/what-container) on Docker official documentation).



Lena create an independent container which contain :

- the software
- the dependencies, libraries, etc.
- the data collected or needed by software

Two use case are possible :

- If Lena want to share container `MyContainer` to paul, she (command `docker export --output myContainer.tar myContainer`) the container on a USB key. Later, Paul copy and load the container `MyContainer` on his machine (command `docker load`).
- If Lena want to run `MyContainer` container on some server on the web which contain docker program, she export the container(command `docker --output myContainer.tar myContainer`), copy this container on the server (using FTP protocol for example) and load this container (command `docker load --input myContainer.tar`) on the server.

Be careful 1 if your container contain a Volume you need a special procedure to migrate it.

Be careful 2 If your container need another container to work, you need to export both.

Here we are, we use this **Docker (https://www.docker.com/) technology** to encapsulate our webscrapping script into one **portable** container. After that you could save yours and launch it on any webserver which run Docker.

There are three big step to understand in containers lifecycle:

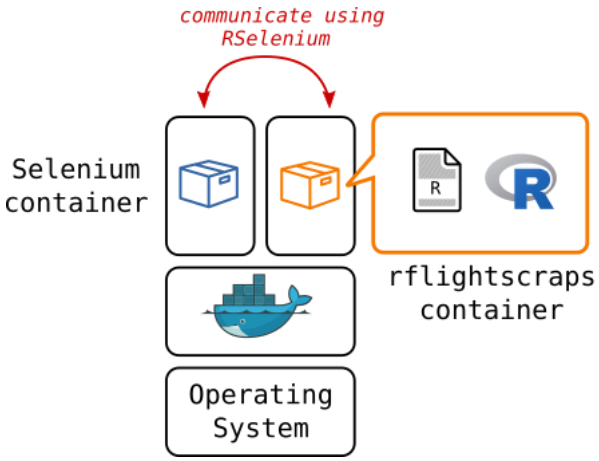
- First, we describe the composition of an **image** into a **Dockerfile** file using a special Docker syntax. **It's like a recipe into cookbook.** For example, you could find lot of recipes for general software on this site : DockerHub (https://hub.docker.com/explore/).

- Next, like a recipe in the real life, you need to concretize this recipe into some delicious cake. Image need to be built before usage. From an **Image/Recipe of a cake** we create a **Container/Cake**
- Finally, you **run** the builted image.

You could find lot of other informations on the web, but also on this online tutorial (https://hackmd.io/s/SylAFG_Kf#) which resume lot of commands.

In this tutorial, we give you the corresponding `Dockerfile` which contain the recipe to build a `Container` ready to scrape the flightradar website. All script to do that are in `docker-scripts` folder.

At the end of the tutorial you have this architecture, with two container (RSelenium and Flightscrapradar) which communicate to scrap data on flightradar website.



4.3 Installing Docker

4.3.1 Linux way

On linux Ubuntu, you found documentation here (https://docs.docker.com/install/linux/docker-ce/ubuntu/). First step, install the key and repository.

```
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common
```

Add key and repository :

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"

sudo apt-get update
```

Install docker-ce :

```
sudo apt-get install docker-ce
```

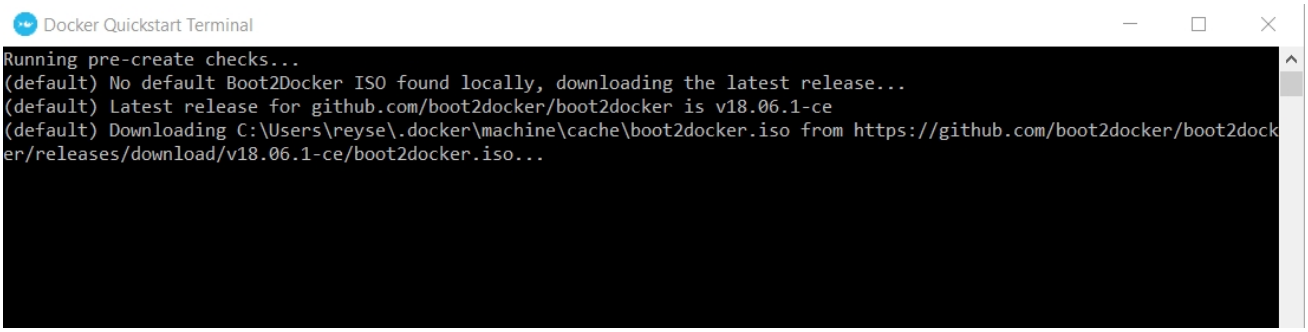
You are ready to jump to the tutorial which correspond to your OS.

4.3.2 Windows and Mac Way

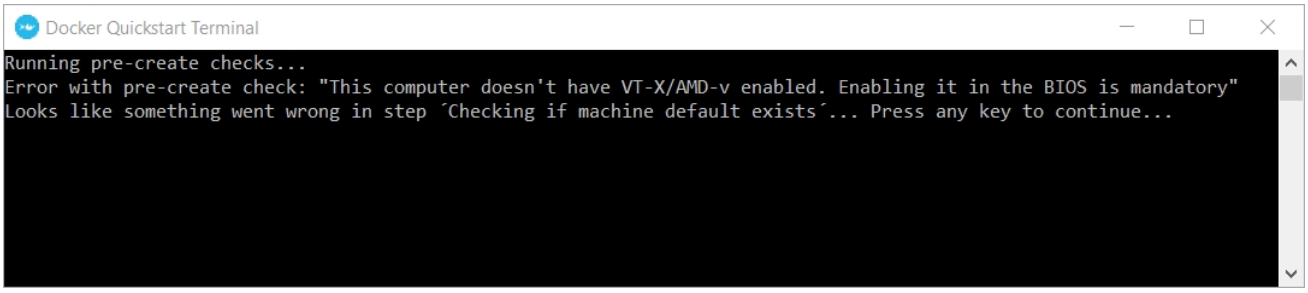
There are two way to install Docker for Windows and Mac, a new way Windows (https://docs.docker.com/docker-for-windows/) / Mac (https://docs.docker.com/docker-for-mac/) and an old way. For this tutorial we use the the old way due to better compatibility.

Install Docker Tools for windows using the `DockerToolbox.exe` (or `.dmg` for mac) file. You could find the official documentation is available here Windows (https://docs.docker.com/toolbox/toolbox_install_windows/) and Mac (https://docs.docker.com/toolbox/toolbox_install_mac/)

After that you could launch Docker quickstart terminal directly after installation or using the icon in start menu.

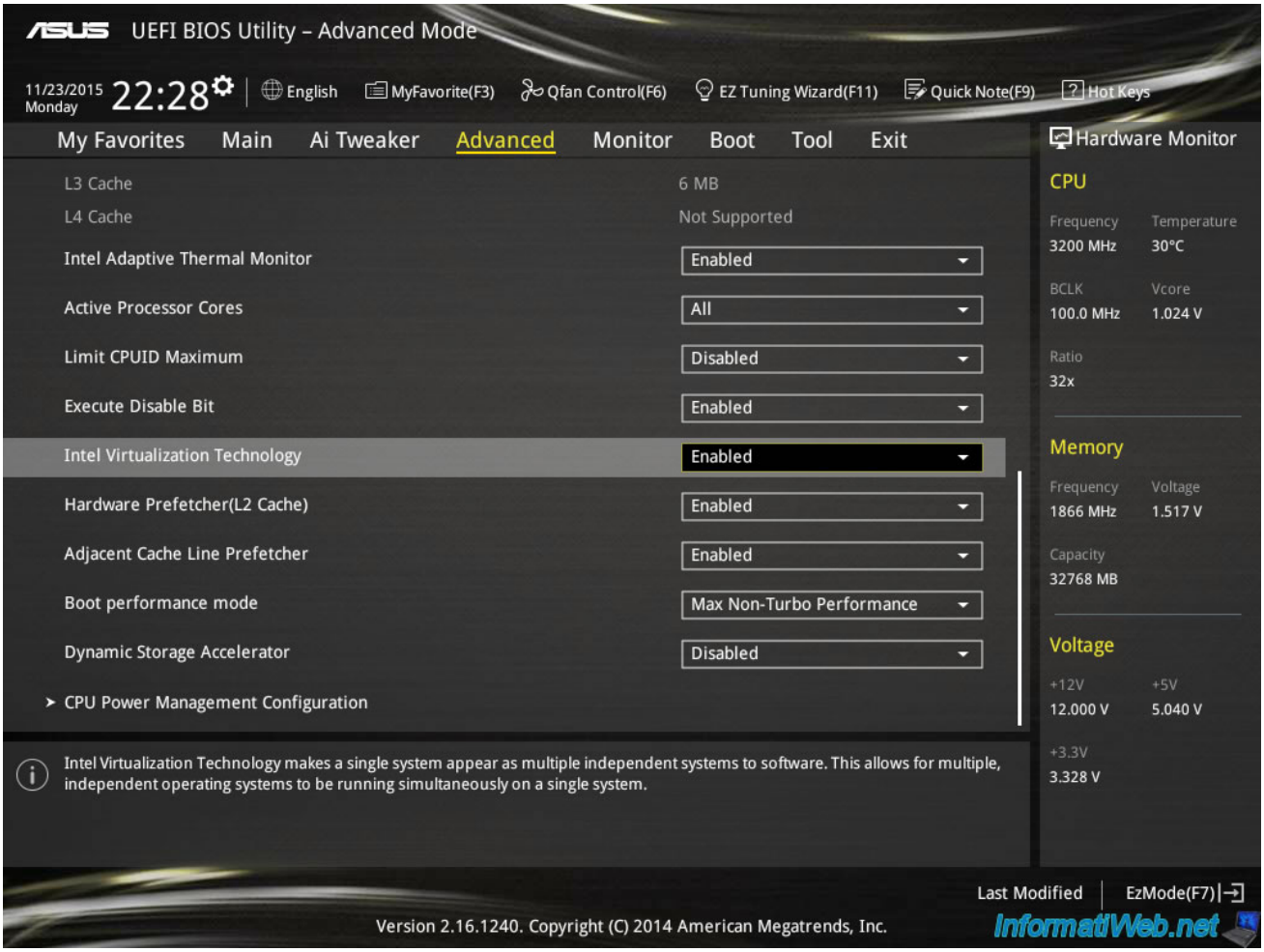


Docker first download an iso, and after that test if your system is ready to run containers. If you see an error like this, you need to run another step.

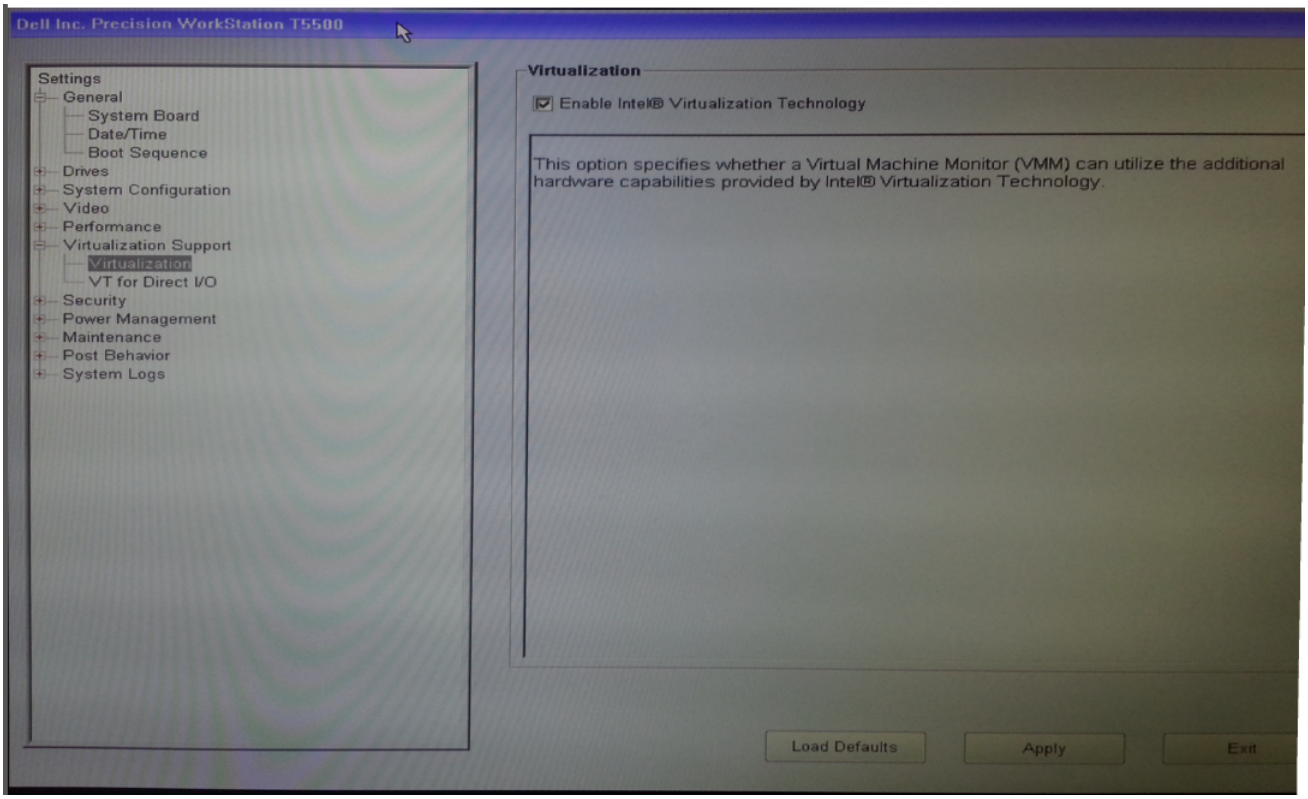


Restart your computer, and try to activate an option in the BIOS (Del key during initialization of your computer) probably named “Vanderpool technology” or “VT-X technology” or “Virtualization technology”. Save and restart. Some pictore for UEFI Bios on HP, DELL, ASUS motherboard/systems.

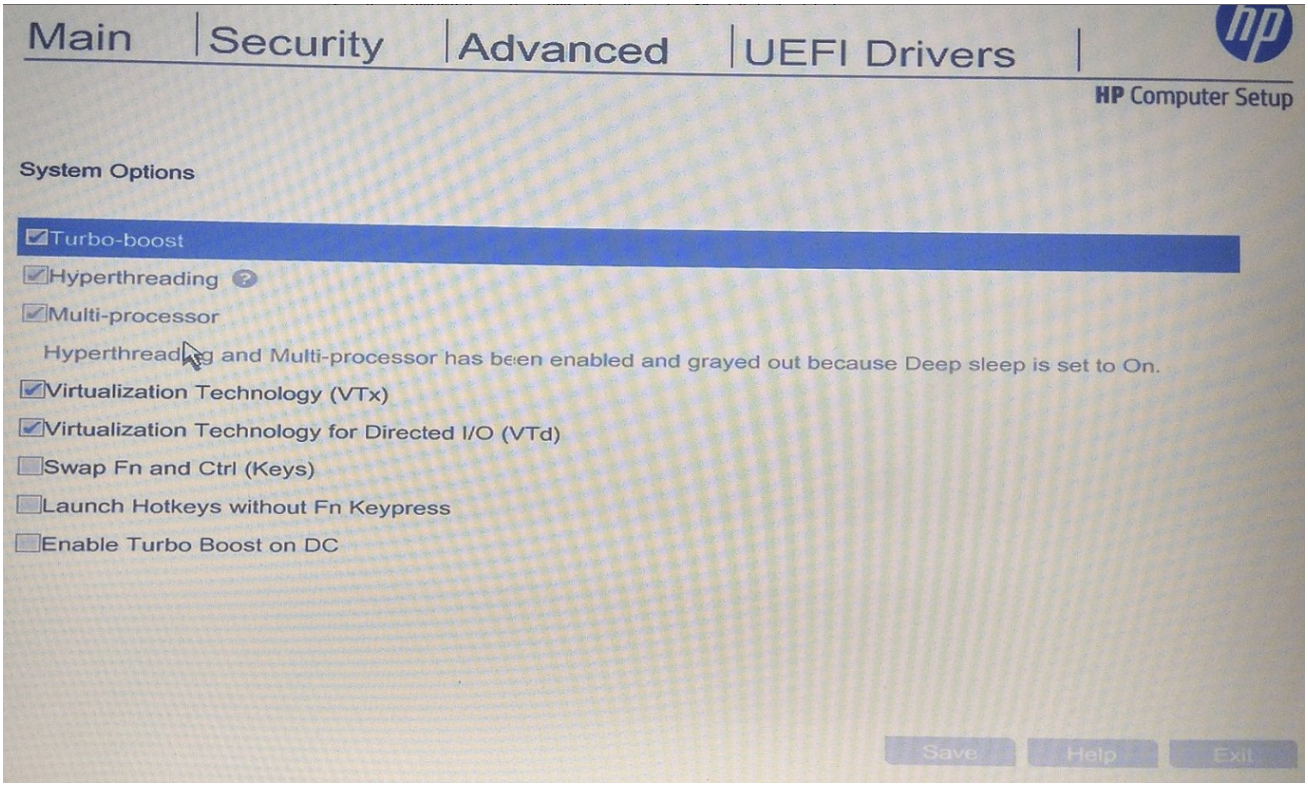
Asus


UEFI BIOS Utility – Advanced Mode
11/23/2015 Monday 22:28
English
MyFavorite(F3)
Qfan Control(F6)
EZ Tuning Wizard(F11)
Quick Note(F9)
Hot Keys
My Favorites
Main
Ai Tweaker
Advanced
Monitor
Boot
Tool
Exit
L3 Cache 6 MB
L4 Cache Not Supported
Intel Adaptive Thermal Monitor Enabled
Active Processor Cores All
Limit CPUID Maximum Disabled
Execute Disable Bit Enabled
Intel Virtualization Technology Enabled
Hardware Prefetcher(L2 Cache) Enabled
Adjacent Cache Line Prefetcher Enabled
Boot performance mode Max Non-Turbo Performance
Dynamic Storage Accelerator Disabled
CPU Power Management Configuration
Intel Virtualization Technology makes a single system appear as multiple independent systems to software. This allows for multiple, independent operating systems to be running simultaneously on a single system.
Hardware Monitor
CPU
Frequency 3200 MHz
Temperature 30°C
BCLK 100.0 MHz
Vcore 1.024 V
Ratio 32x
Memory
Frequency 1866 MHz
Voltage 1.517 V
Capacity 32768 MB
Voltage
+12V 12.000 V
+5V 5.040 V
+3.3V 3.328 V
Last Modified
EzMode(F7)
Version 2.16.1240. Copyright (C) 2014 American Megatrends, Inc.
InformatiWeb.net

Dell


Dell Inc. Precision WorkStation T5500
Settings
General
System Board
Date/Time
Boot Sequence
Drives
System Configuration
Video
Performance
Virtualization Support
Virtualization
VT for Direct I/O
Security
Power Management
Maintenance
Post Behavior
System Logs
Virtualization
Enable Intel® Virtualization Technology
This option specifies whether a Virtual Machine Monitor (VMM) can utilize the additional hardware capabilities provided by Intel® Virtualization Technology.
Load Defaults
Apply
Exit

HP



You are ready to jump to the tutorial which correspond to your OS.

4.4 Tutorial Docker / Linux

Copy the folder `docker-images` on the USB Key (ask teachers) into the `scrap-flightradar` folder of this tutorial.

Now, go to this folder using terminal command (`cd pathofthefolder`), and load the two images on your system.

```
sudo docker load --input=r-alpine.tar
sudo docker load --input=rSelenium.tar
```

BUILD image

Go to `docker-scripts` folder into the folder which contain this tutorial on your disk.

The building of this image take lot of times (ten minutes), this is due to the huge dplyr library. Run the `docker build` command in the folder which contain the `Dockerfile` description of the image.

```
docker build . --tag=rflightscraps
```

LAUNCH Container

a. Using a **binded volume**, this is the easiest way actually. Open your terminal, go to the folder of your script. Create a new folder named `localbackup` and run the container `rflightscraps` with correct path.

```
mkdir localbackup
docker run --name rflightscraps -d -e UID=1000 -e GID=1000 --mount type=bind,source=$(pwd)/localbackup,destination=/usr/l
ocal/src/flight-scrap/docker-scripts/data rflightscraps
```

To see if your container is running and consult the logs of execution :

```
sudo docker ps
sudo docker logs rflightscraps
```

To consult the result of automatic harvesting, consult the `docker-scripts/localbackup` folder using `ls` unix command. You see a list of csv which correspond to harvest made every minute. If you want to change this, you need to modify the `crontab` file following the cron syntax (https://crontab.guru/#*_*_*_*_*), and rebuild/relaunch the image (it take less time, because you only modify one file, no need to recompile).

b. Same thing, but using a *named volume*, a more portable way to share data between docker container, but it lacks some features on permissions to correctly export data.

Create a named volume, independent from filesystem

```
docker volume create --name myDataVolume
docker volume ls
```

Mount the volume :

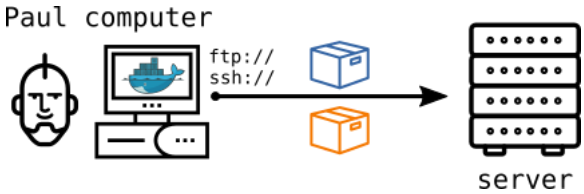
```
docker run --mount type=volume,source=myDataVolume,destination=/usr/local/src/flight-scrap/docker-scripts/data rflightscr
aps
```

Export data :

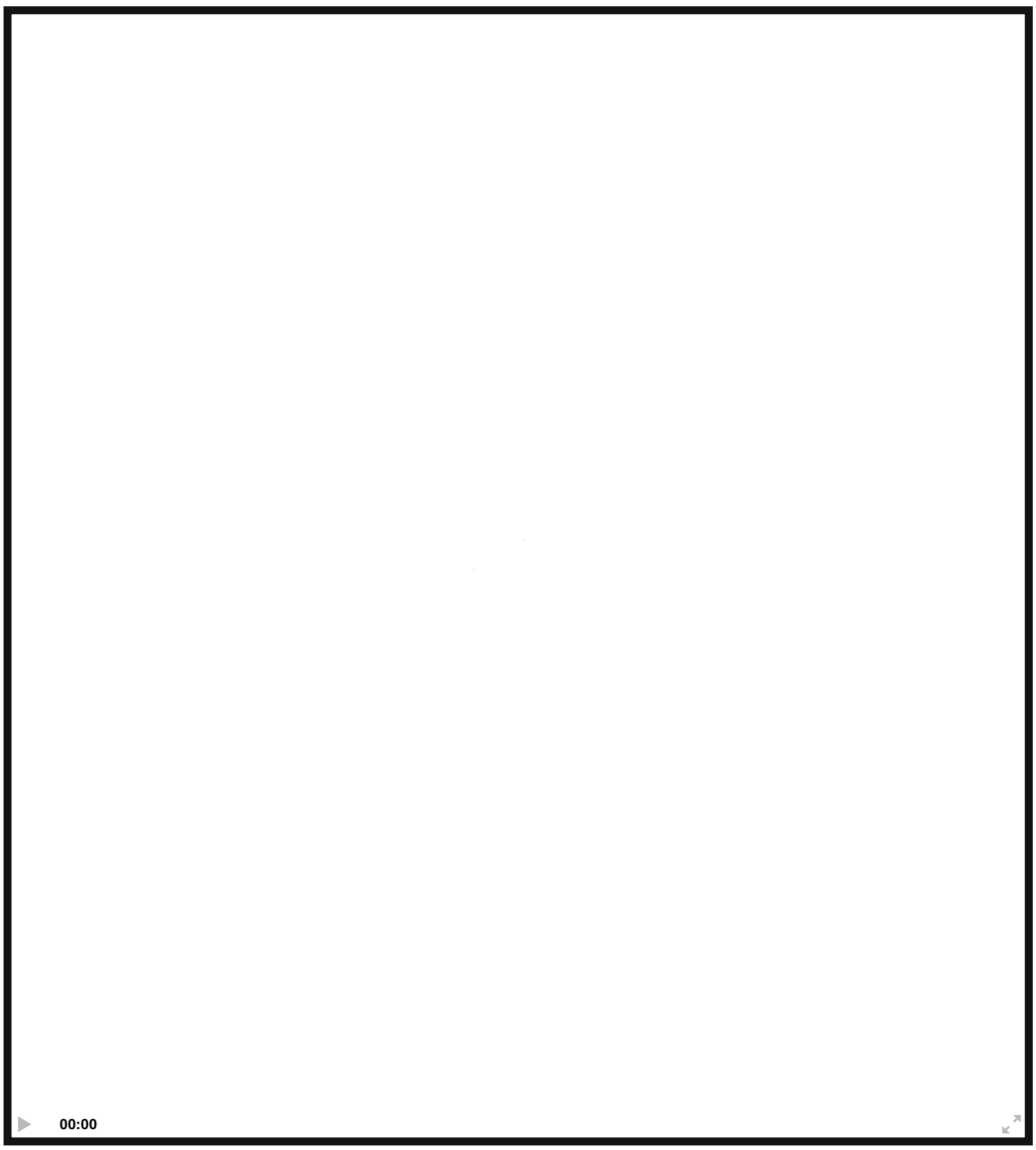
- Using a `alpine` image, we mount the named volume (`myDataVolume`) to a `/alpine_data` folder inside the `alpine` container.
- Then, we create a new folder inside the `alpine` container named `/alpine_backup` .
- We then create an archive containing the contents of the `/alpine_data` folder and we store it inside the `/alpine_backup` folder (inside the container).
- We also mount the `/alpine_backup` folder from the container to the docker host (your local machine) in a folder named `/local_backup` inside the current directory.

```
docker run --rm -v myDataVolume:/alpine_data -v $(pwd)/local_backup:/alpine_backup alpine:latest tar cvf /alpine_backup/scrap_data_"$(da
```

EXPORT Containers



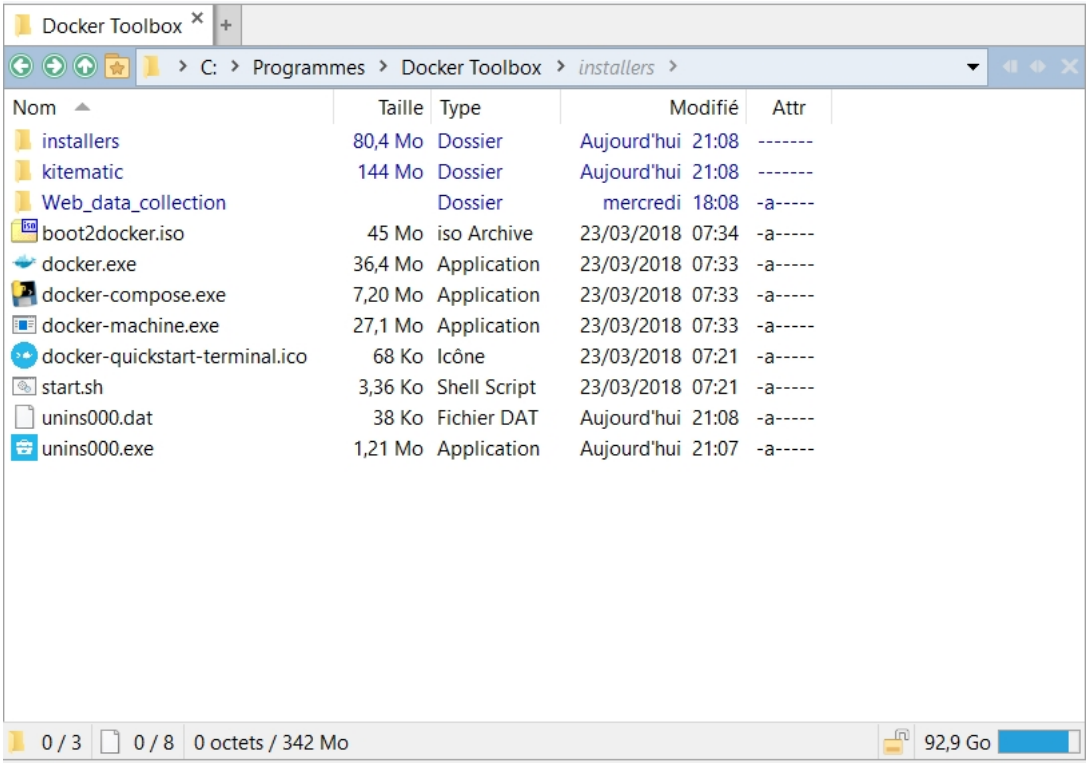
If you don't have a server yet, you could buy one with docker already installed for cheap / month : - https://www.digitalocean.com/products/one-click-apps/docker/ (https://www.digitalocean.com/products/one-click-apps/docker/) - https://www.ovh.com/fr/vps/vps-cloud.xml (https://www.ovh.com/fr/vps/vps-cloud.xml)



4.5 Tutorial Docker / Windows & Mac

PREPARE image

Copy `docker-scripts` and `docker-images` folders into `c:\Program Files\Docker Toolbox`



After that, into Terminal of Docker Toolbox you see this folders.

```
reyse@INSMOUTH MINGW64 /c/Program Files/Docker Toolbox/Web_data_collection/scrap_flightradar
$ ls -l
total 1217
drwxr-xr-x 1 reyse 197609       0 sept. 21 17:43 docker-images/
drwxr-xr-x 1 reyse 197609       0 sept. 22 11:54 docker-scripts/
drwxr-xr-x 1 reyse 197609       0 sept. 21 12:20 docker-test/
drwxr-xr-x 1 reyse 197609       0 sept. 18 18:29 images/
-rw-r--r-- 1 reyse 197609 1212081 sept. 21 17:30 scrap_flightradar.html
-rw-r--r-- 1 reyse 197609   19370 sept. 21 17:57 scrap_flightradar.Rmd
-rw-r--r-- 1 reyse 197609     205 sept. 20 15:34 scrap_flightradar.Rproj
```

Go to `docker-images` folder using `cd` command, and load the two images :

```
docker load --input=r-alpine.tar
docker load --input=rSelenium.tar
```

```
reyse@INSMOUTH MINGW64 /c/Program Files/Docker Toolbox
$ docker load -i r-alpine.tar
cd7100a72410: Loading layer [==================================================>]  4.403MB/4.403MB
385866c7f983: Loading layer [==================================================>]  290.1MB/290.1MB
Loaded image: artemklevtsov/r-alpine:latest

reyse@INSMOUTH MINGW64 /c/Program Files/Docker Toolbox
$ docker load -i rSelenium.tar
0a42ee6ceccb: Loading layer [==================================================>]  118.8MB/118.8MB
c2af38e6b250: Loading layer [==================================================>]  15.87kB/15.87kB
5e95929b2798: Loading layer [==================================================>]  14.85kB/14.85kB
2166dba7c95b: Loading layer [==================================================>]  5.632kB/5.632kB
bcff331e13e3: Loading layer [==================================================>]  3.072kB/3.072kB
6eba6110ccc4: Loading layer [==================================================>]  3.072kB/3.072kB
70f675152cda: Loading layer [==================================================>]  152.4MB/152.4MB
0e3b1af8adf7: Loading layer [==================================================>]  1.304MB/1.304MB
8c63cb20b8cd: Loading layer [==================================================>]  353.8kB/353.8kB
385f1fee572c: Loading layer [==================================================>]  23.25MB/23.25MB
48b1b5ef5604: Loading layer [==================================================>]    103MB/103MB
1ab56b003e8b: Loading layer [==================================================>]  22.58MB/22.58MB
402ffa13597a: Loading layer [==================================================>]  38.79MB/38.79MB
195cbc461a59: Loading layer [==================================================>]  5.632kB/5.632kB
738eacc64f56: Loading layer [==================================================>]  280.3MB/280.3MB
cfbe9d798bbe: Loading layer [==================================================>]  10.37MB/10.37MB
5bfc0c2f26ee: Loading layer [==================================================>]  3.584kB/3.584kB
cf2e1eac6e59: Loading layer [==================================================>]  8.704kB/8.704kB
9e56646bb1c7: Loading layer [==================================================>]  3.584kB/3.584kB
Loaded image: selenium/standalone-firefox:3.14.0-arsenic
```

BUILD image

Go to `docker-scripts` folder into the folder which contain this tutorial on your disk.

The building of this image take lot of times (ten minutes), this is due to the huge dplyr library. Run the `docker build` command in the folder which contain the `Dockerfile` description of the image.

```
docker build . --tag=rflightscraps
```

LAUNCH container

We use a **binded volume**, this is the easiest way actually.

First, create a new folder named `localbackup` into your users folder on windows : `C:\Users\yourname` After that, change the path by yours in this command and run it.

```
docker run --name rflightscraps -d -e UID=1000 -e GID=1000 --mount type=bind,source=/c/Users/reyse/localbackup,destinatio
n=/usr/local/src/flight-scrap/docker-scripts/data rflightscraps
```

The end, close the session !

```
remDr$close()
```