Deploying a scalable Data Science environment using Docker

Sergio Martín-Santana, Carlos J. Pérez-González , Marcos Colebrook, José L. Roda-García, Pedro González-Yanes

Abstract Within the Data Science stack, the infrastructure layer supporting the distributed computing engine is a key part that plays an important role in order to obtain timely and accurate insights in a digital business. However, sometimes the expense of using such Data Science facilities in a commercial cloud infrastructure is not affordable to everyone. In this sense, we present a computing environment based on free software tools over commodity computers. Thus, we show how to deploy an easily scalable Spark cluster using Docker including both Jupyter and RStudio that support Python and R programming languages. Moreover, we present a successful case study where this computing framework has been used to analyze statistical results using data collected from meteorological stations located in the Canary Islands (Spain).

1 Introduction

The NIST Big Data Working Group (NBD-WG) [1][2] provides a nice definition on the concept of Data Science:

> "Data Science is the extraction of actionable knowledge directly from data through a process of discovery, or hypothesis formulation and hypothesis testing. It can also be understood as the activities happening in the processing layer of the system architecture, against data stored in the data layer, in order to extract knowledge from the raw data."

S. Martín-Santana
Máster en Ingeniería Informática
Universidad de La Laguna, Tenerife, Spain
e-mail: Sergio.MS.91@gmail.com

C.J. Pérez-González
Depto. de Matemáticas, Investigación Operativa y Computación
Universidad de La Laguna, Tenerife, Spain
e-mail: cpgonzal@ull.edu.es

M. Colebrook (✉), J.L. Roda-García
Departamento de Ingeniería Informática y de Sistemas
Universidad de La Laguna, Tenerife, Spain
e-mails: mcolesan@ull.edu.es, jlroda@ull.edu.es

Pedro González-Yanes
Centro de Cálculo de la Escuela Superior de Ingeniería y Tecnología (Secc. Ing. Informática)
Universidad de La Laguna, Tenerife, Spain
e-mail: pgonyan@ull.edu.es

This definition implies a data life cycle, which is the set of processes that transform raw data into valuable and actionable knowledge, by means of principles, techniques and methods from many disciplines and domains (see Fig. 1) within the context of Big Data Engineering. For a brief introduction and a recent state-of-the-art on the concept of Big Data, the reader is referred to [3].

Furthermore, such data life cycle is developed inside a Data Science stack (see Fig. 2), in which the infrastructure layer supporting the distributed computing engine plays an important role in order to obtain timely and accurate insights in a digital business.

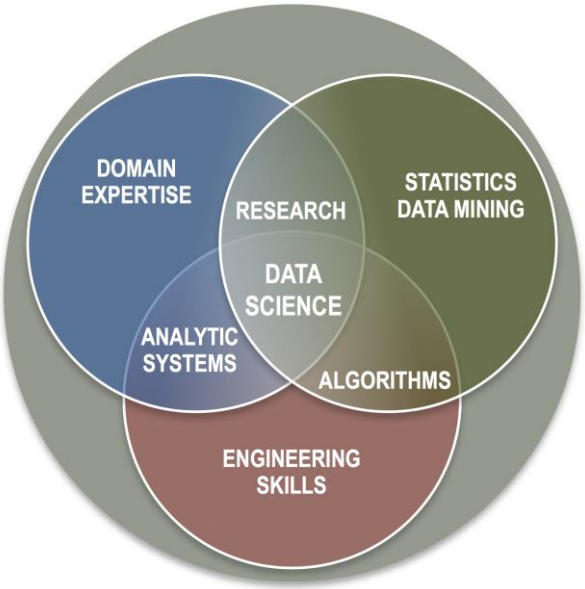

Fig. 1. Data Science definition from the point of view of the skills needed (adapted from [4])

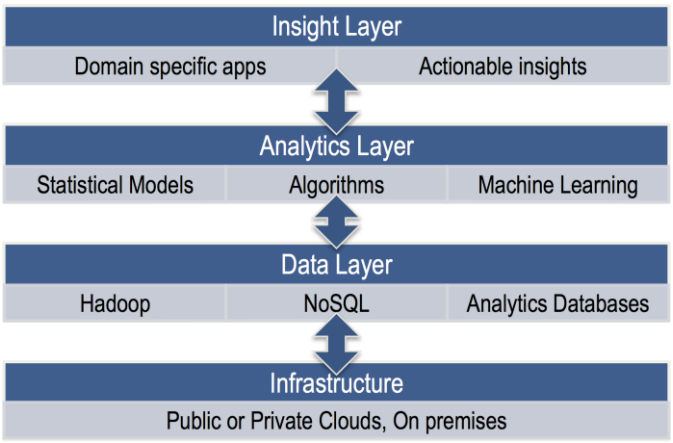

Fig. 2. Data Science stack (adapted from [6])

Indeed, the market research company Forrester [5] pointed out that a new type of company has arisen nowadays: the **insights-driven business**, which builds systems using Data Science platforms to create competitive advantage through data. Moreover, it is predicted that these companies will earn a revenue of $1.2 trillion in 2020.

In this sense, companies that adopt a Data Driven Decision Making (DDDM) achieve a 5-6% increase in productivity and production growth [7]. Besides, the relationship between DDDM and performance appears also in other performance indicators such as asset utilization, return on equity and market value.

According to [8] there is no alternative: 65% of the firms think that there is a high risk of becoming uncompetitive if they do not implement a data driven mindset, since data is becoming a key component of their market value.

Forrester [5] also suggests that Data Science platforms, which comprise data integration, data exploration, model development and deployment, could accelerate insights maturity if the firms follow some key recommendations:

- Unify the Data Science technology into a single platform.
- Treat Data Science platforms as a strategic and transformative investment.

Within this enterprise context, Linden et al. from the consulting firm Gartner [9] define a Data Science platform as:

> "A cohesive software application that offers a mixture of basic building blocks essential for creating all kinds of data science solutions, and for incorporating those solutions into business processes, surrounding infrastructure and products."

Additionally, their analysis of the 16 top vendors in Data Science platforms yields the following conclusions:

- The implementation of open source platforms is increasing the adoption of Data Science.
- Apache Spark is becoming a *de facto* Data Science foundation for the vendors.
- Open source languages like Python, R and Scala dominate this market. Even more, almost all Data Science platform vendors support Python and R.

Therefore, in order to facilitate the adoption of Data Science platforms, the Big Data Senior Steering Group (BD-SSG) [10] suggests to enhance infrastructures to support handling and analyzing large amounts of data, since state-of-the-art infrastructures are essential in a data-driven industry sector. They also noticed that there is a need to invest in infrastructure pilot programs, testbeds, and sandboxes for testing new techniques at scale, across a variety of application domains, and to engage in proofs of concept with both open source and proprietary solutions. Thus, future infrastructures may help moving the computation to the data.

Besides, the Big Data Value Association [11] also recommends building good infrastructures to develop a Data Economy, raising as a challenge a distributed trust infrastructure for data management, with flexible structures based on data microservices in a decentralized way. Regarding this matter, the European Union

[12] is currently working in the development of enabling technologies, infrastructures and skills for the benefit of the SME (Small & Medium-sized Enterprises).

Likewise, the Edison Data Science Framework [2] promotes infrastructures, including typical frameworks such as Hadoop [13] and Spark [14], to support data handling during the whole data lifecycle. On the other hand, the NBD-WG [15] suggests creating a vendor-neutral, technology- and infrastructure-independent framework that could enable stakeholders using the best analytics tools on the most suitable computing platform and cluster. Besides, in order to support Big Data stores and processing, the infrastructure should be scalable in terms of easy addition of new resources, with possible platforms including public and/or private clouds [1].

Nevertheless, digital businesses investing only in infrastructure projects are not guaranteed to succeed, as pointed out by the UK's Science and Technology Committee of the House of Commons [16]. Acquiring more digital skills, trusting on public data sharing, progressing in open data and data protection are essential factors to remain in the right pace for Big Data and Data Science. Furthermore, the UK's government has been committed to creating a coordinated infrastructure, and access to advanced software and hardware to the small businesses (SME).

From the above paragraphs, it is clearly stated that the infrastructure layer plays an outstanding role within the Data Science stack. However, sometimes the expense of using such Data Science facilities in a private and commercial cloud infrastructure is not affordable to a small business. Accordingly, in the next sections we present a Data Science computing environment based on open source software tools that can be easily deployed over commodity (personal) computers.

Finally, the remainder of the chapter is organized as follows. In Sect. 2, we show the most important tools and environments for Data Science nowadays. Sect. 3 presents the full project and simple guides on how to deploy our Data Science stack in Windows, Linux or Mac. This stack has been used to analyze data from meteorological stations located in the Canary Islands (Spain), and the results are presented in Sect. 4. Finally, the conclusions are provided in Sect. 5.

2 Tools and frameworks for Data Science

In Data Science there are many tasks that must be carried out frequently. For instance, loading and processing datasets, obtaining summarized statistics, visualizing the information in tables and charts, etc. The amount of tools and applications that are available to accomplish these jobs has increased in the last years, which implies installing programs and libraries in desktop or server computers with all the problems derived of this process.

Among the main difficulties that usually arise are those concerning to errors due to not complying with the dependencies between the required software versions or the lack of experience of the users in dealing with these computer system aspects. In this sense, virtualization is the solution to afford these issues since it

provides the possibility to create and deploy software-based systems (so called virtual machines and containers) that emulate the physical ones.

A virtual machine consists in a guest system that packages both the computer architecture and the software applications along with the operating systems (plus all the code and dependencies required) to be executed in the host system. A container represents another level of virtualization where the host operating system kernel and its resources are shared to allow the execution of multiple light-weight and isolated processes. Consequently, each container takes up less space than virtual machines (container images are typically moderate in size), and run almost instantly.

The containers technology helps setting up the collection of useful tools for different stages in a Data Science project. Thus, each container represents a recipe for each application that can be shared and versioned. In the following sections, we describe and discuss the most usual programming languages and developing frameworks in order to create the stack of containers for Data Science.

2.1 Containers in Data Science

Since its first appearance in 2013 [17], Docker containers have implied a big impact in simplifying the process to create Data Science stacks. Basically, containers are lightweight versions of traditional virtual machines but without the need of large amounts of storage space on servers (see Fig. 3). Besides, they can be easily created and deleted, and they boot up quickly. Restoring a normal virtual machine usually can imply excessive time to get going, but Docker containers start up almost immediately.

The containers run from images that are essentially snapshots of a running container at a particular time point. These images can be used as templates to create and run other containers. This is the main reason why they are important in Data Science, since images are created containing the required tools for doing data analysis, either for a general use or for specific analyses. Lots of base images of containers can be downloaded for free from registries like Docker Hub [18]. The key idea is that many containers can be launched as required and, consequently, it turns into an easy task creating reproducible Data Science environments.

Running a container with the libraries and tools for a particular analysis reduces the effort to debug packages across different environments because they run identically on systems as Mac OS X, Windows or Linux. Due to this feature, Docker containers are very convenient to allow the users launching a variety of isolated applications in a platform as, for example, Jupyter [19] and RStudio [20] sessions configured with a set of basic packages, but also lending the users the possibility to install other libraries.

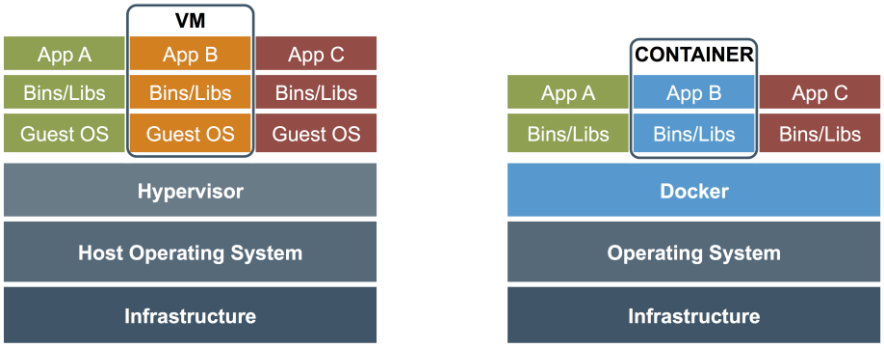

Fig. 3. Difference between Virtual Machines (*left*) and Docker containers (*right*) (adapted from [17])

2.2 The R language

The R language [21] represents the most well-known free, open source programming language and environment for statistical computing and graphics. Indeed, it is powerful and highly extensible with more than 10,000 add-on packages.

There are many large and active communities (for instance, the LinkedIn's R group has more than 100,000 members), and there are currently hundreds of R Meetup groups. This proves the increasing interest in the R statistics language, especially for data analysis. The programming environment allows for command-line scripting and, therefore, the data analysis steps can be serialized in such a way that can be reused with other data in contrast with interfaces guided with option menus.

The variety of tasks that can be accomplished in R are, among others, the following (we describe in parenthesis the aspects of data analysis that could be accomplished with these simple tasks):

- Exploring and manipulating data (ETL processing)
- Fitting and validation of predictive or classification models (machine learning)
- Creating visually attractive graphs (data visualization)
- Connecting with different data sources (systems integration)
- Making illustrative reports or dashboards (business intelligence)

The reader may find many R language tutorials in the Internet, some of them designed even for novice users without any programming background. These tutorials help users to understand the basics and fundamentals of R about importing and exporting data, exploring and manipulating data and, for advanced users, how to use loops and create functions.

R is one of the key tools in Data Science because it covers several data mining, machine learning and statistical techniques. There are also complete tutorials which explain how to perform descriptive statistics and make inferences on data, apply linear and logistic regression models as well as classification and clustering techniques, fit time series, apply variable selection and dimensionality reduction, etc.

2.3 RStudio

RStudio is an integrated development environment (IDE) that enhances the standard R and eases the work of R programmers [20]. It is available as open source for free, but there are also enterprise versions with additional features (administrative tools, enhanced security and authentication for multiple users, metrics and monitoring functionality, etc.).

RStudio is a very interesting application because it supports several premium characteristics such as intelligent code completion, syntax highlighting, integration of R help and the management of structured R documentation, and a tool for interactive debugging (see Fig. 4). The product can be used in a personal desktop installation or in a server version to centralize access and computation.

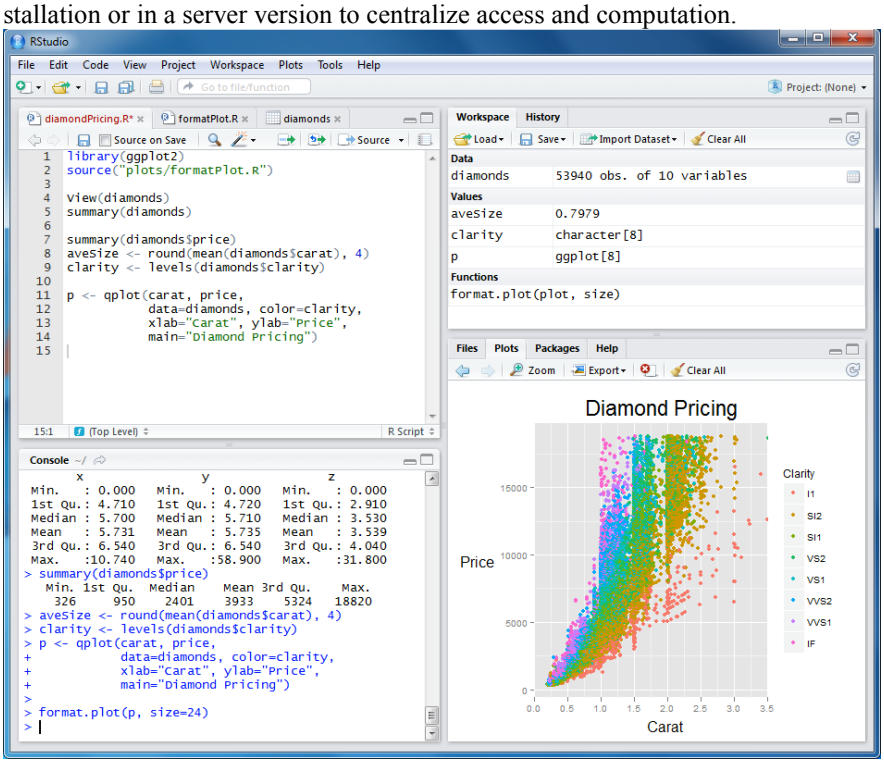

Fig. 4. The RStudio IDE (source: [20]).

2.4 The Python language

Python [21] is a general purpose programming language and one of the most popular tools for data analysis. It is very frequent to use it when analyzing huge amounts of data due to several strengths. In a similar way to R, Python provides many powerful libraries appropriate to process very large and growing data sets, and there is a wide support from open source community users.

It is relatively easy to write code in Python and to make this code understandable by other users. Python also integrates very well with other open source platforms commonly used in Data Science, as Spark [14] and Hadoop [13]. These are the reasons that have contributed to the enthusiastic adoption of Python by the programmers.

A Python environment can be easily set up. There are free distributions like Anaconda [23] or Canopy [24] containing the core Python language, as well as other essential libraries for data analysis including the following:

- Numpy and Scipy: fundamental scientific computing
- Pandas: data manipulation and analysis
- Matplotlib: plotting and visualization
- Scikit-learn: machine learning and data mining
- StatsModels: statistical modeling, testing, and analysis

Again as in the R case, there are many excellent internet resources (among others, DataCamp [25] and Codecademy [26]) to learn how to code in Python. They are an excellent option to gain knowledge in programming concepts that will be useful and valuable in working with data.

2.5 Jupyter notebooks

One of the most important Python extensions is the Jupyter notebook (also known as IPython notebook) [19]. The notebooks are executable documents that, when launched from the Jupyter web interface, a browser is opened to show an environment to place not only code and executing data analysis, but even to introduce rich text, formatted expressions and embedded images and videos.

With Jupyter is possible to include several kernels that are computational engines for executing code of many other languages apart from Python (as for example R). The notebooks also provide options to export the content in several formats including PDF, HTML and Markdown. Consequently, notebook documents can be used as reports containing both the analysis description and the final results (figures, tables and graphics).

Other interesting Python IDE for data analysis is Rodeo [27], from the Yhat company. This program is similar to RStudio for R, and can be seen as a simple, lightweight alternative front-end to the Python notebooks.

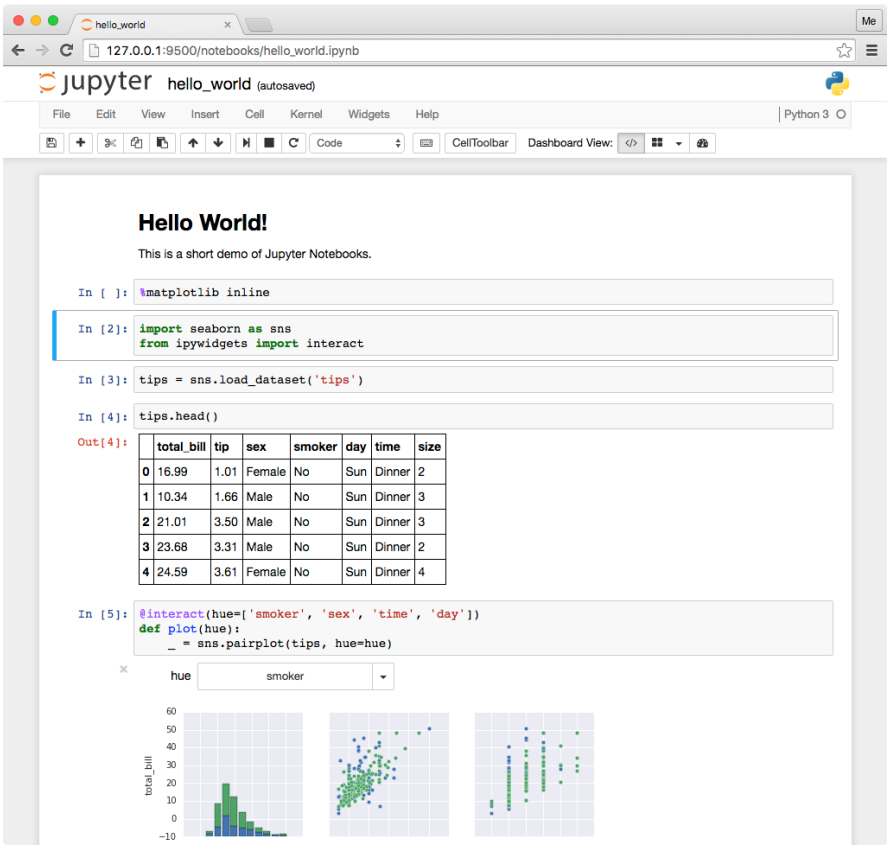

Fig. 5. An example of the Jupyter notebook (source: [19])

The R and Python languages described above, as well as the RStudio and Jupyter development environments (IDE), are included within the applications layer, whereas the Docker containers are used in the infrastructure layer to ease the deployment process (see Fig. 2). The reader might have noticed that the data layer is missing in the previous schema. Such layer can be connected from inside the user container, as we explain in the following section where we describe the development of the Data Science Stack project.

3 The Data Science Stack Project

The integration of different languages, libraries and platforms for use in a real environment is a complex task. It is really crucial to model the ecosystem in which you are going to work. In this section, we describe the starting situation of this

project, as well as the reasons to justify the change of the model towards a more efficient one.

We rely on the TOGAF architecture framework [28] that allows us to model, through the points of view, the existing system at the beginning of the project called AS-IS and after a set of change recommendations, model the target system called TO-BE. The Layered Viewpoint diagrams designed with the Archimate tool [29] will be used to describe both situations, and they are composed of three main layers: business, application and technology.

The key idea is to reflect in a single diagram the complete system. Besides, the Layered Viewpoint diagrams offer a joint vision of the actors, the main existing business processes, the software components and the technological infrastructure.

3.1 Initial situation

Currently, there exists an infrastructure where the researcher users request the creation of different systems for data storage, data analysis and information visualization. One way to accomplish these tasks is through virtual machines that are generated from templates by system administrators. In this case, it is necessary to reserve the appropriate computing resources. This step involves several problems for the end user like waiting until the resources are available or the administration staff can complete the work.

To understand the starting point of our project, we will use a TOGAF-based diagram, which visually represents the AS-IS model [28]. This model is divided in layers where actors, processes, components and infrastructure are presented (the aforementioned business, application and technology layers).

In the current model, the execution for a research user of our system is presented, who may ask for a new infrastructure or simply use it in the system. To do so, this request should be solved by a system administrator. The application will be registered and, once accepted, it will be created. Then, the system can be really deployed. The business processes are executed in an application for the management of the forms and will have a mechanism to create virtual machines for later deployment. Finally, the complete system is executed according to the infrastructure layer presented, in which a web and virtualization servers are available (see Fig. 6).

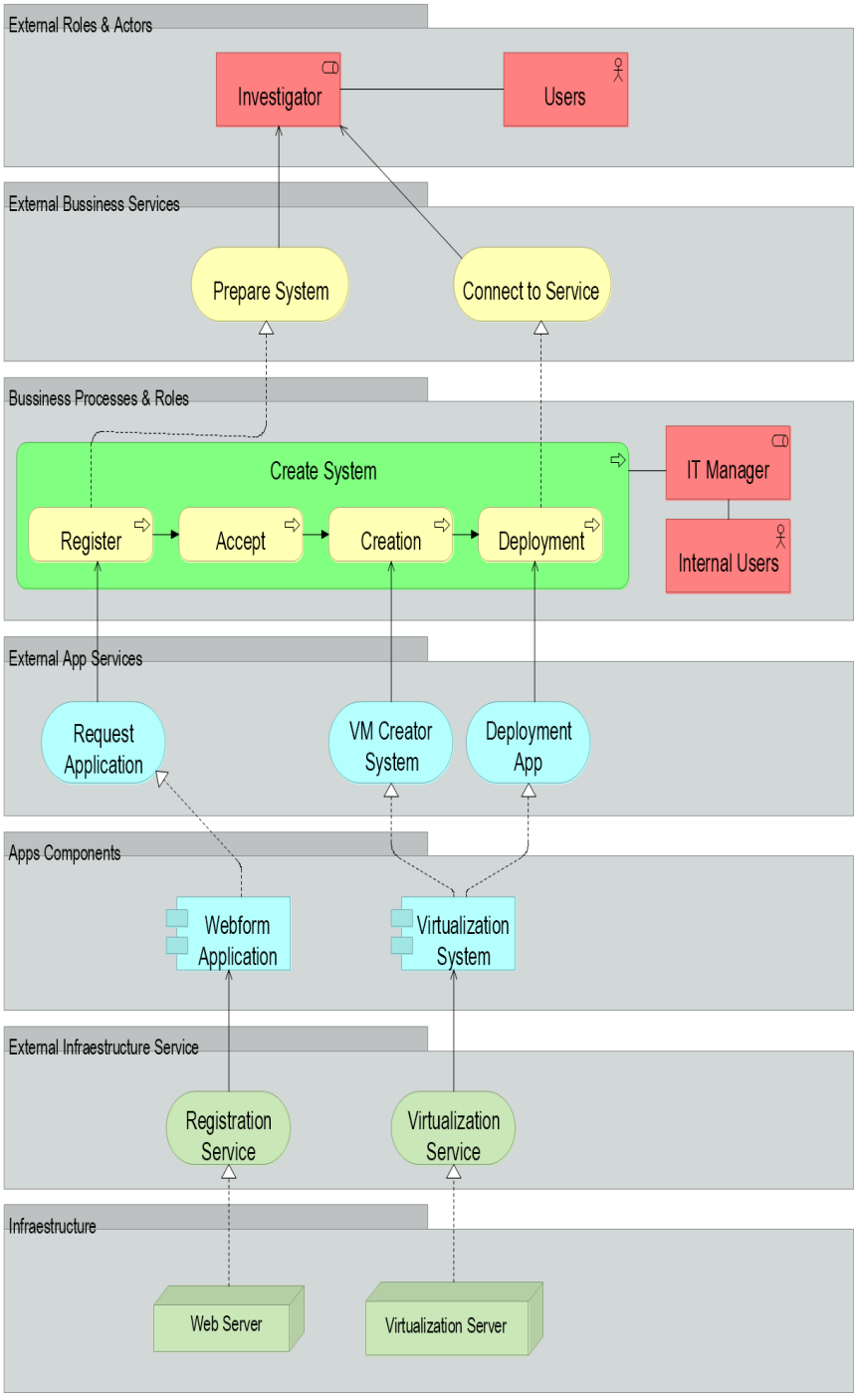

Fig. 6. The situation of the current system described as a TOGAF AS-IS diagram

3.2 Evolution of the model to a more efficient solution

In order to speed up the deployment and implementation of Data Science systems, we propose a different strategy based in the dockerization of the system. Under this strategy, computer resources will be optimized, since no system resources reservation will be needed.

The proposed model aims to minimize and simplify the Administrator's task. Using the system described below, we could meet the requirements of the researchers without adding complex tasks. Initially, this solution will have two development environments, such as Jupyter and RStudio, which allow the applications development through languages such as Python and R. Besides, Spark is considered for the massive processing of data, and it also provides real-time data processing. All of these components run on a single Docker container, making a lot easier its maintenance and future evolution.

Hence, from the system administrator’s point of view, his/her work could focus on creating a single, equal based recipe for all researchers that would include all the required tools. From the point of view of the data scientist, he/she would have a complete set of tools for researching, with low cost extensibility to add new libraries, languages or tools.

This new solution avoids, or at least diminishes, the high costs of the equipment to allocate test systems. Now the researcher will be able to carry out small tests on his personal equipment, and to later transfer them to a production system in a large infrastructure.

The final solution is presented in Fig. 7 through a TOGAF-based model where the architecture of the system is visually displayed.

3.3 Solution development: A container based in Docker

We have developed an environment with different applications installed as modules by executing a Dockerfile, which consists of several scripts doing specific tasks. Our proposal of this system uses the latest version of Ubuntu Xenial OS, but other operating systems can be used as Debian, CentOS, Alpine, etc., although this might imply some modifications of the steps described in the following paragraphs.

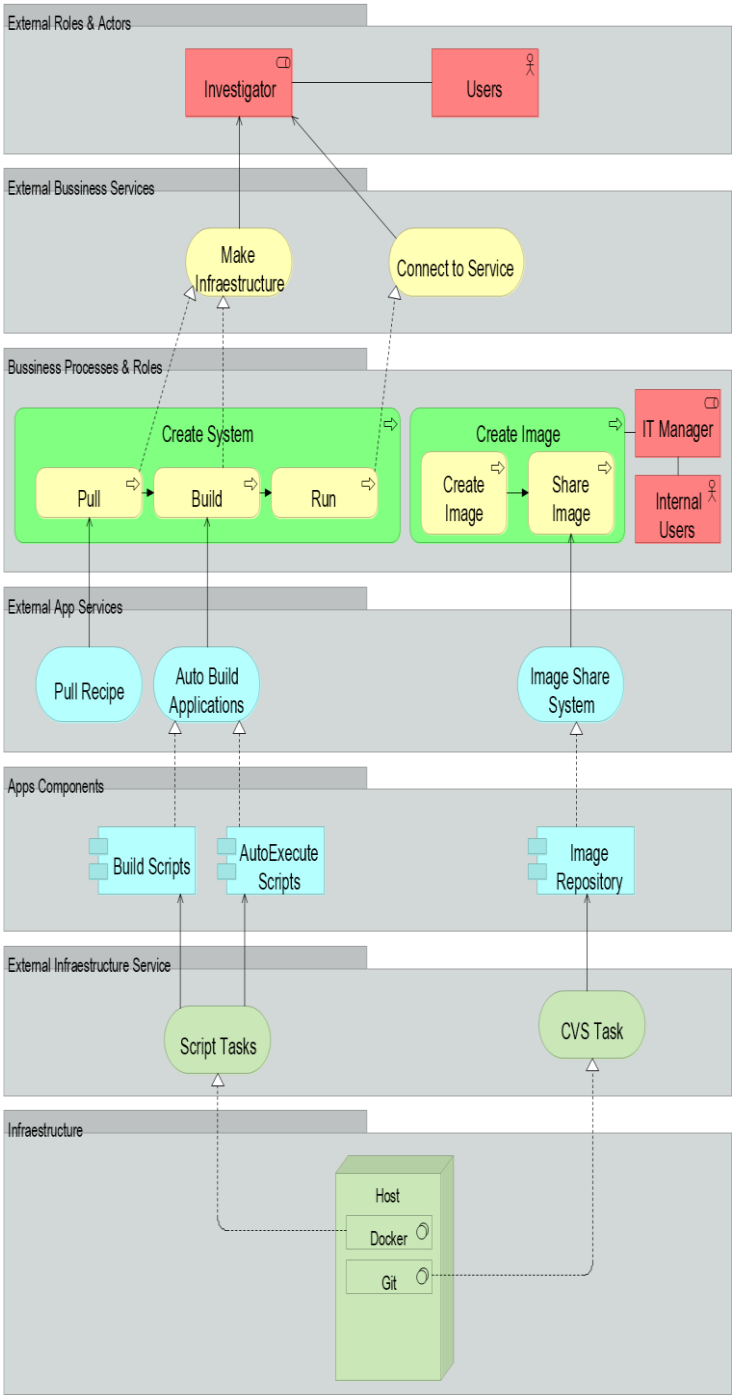

Fig. 7. The TOGAF TO-BE diagram as an evolution of the TOGAF AS-IS diagram

When the Dockerfile is executed, the first script (`base.sh`) installs some additional base system libraries and applications (e.g. `wget`, `default-jre`, `tar`, `openssl`, `libssl`, etc.) that are required for our Data Science environment.

In the next step of the Dockerfile execution, the scripts to install and configure the different applications of the Data Science stack are placed inside the container. This aspect provides modularity to our system because each tool is aggregated in an easy and independent way.

Thus, the first applications to be installed are versions 2.7 and 3.6 of the Python programming language using the instructions contained in the script named `python.sh`.

In addition to Python, the next scripts (`R.sh` and `Rstudio.sh`) install the last version of the R language, along with the integrated development environment (IDE) RStudio server. Both applications are configured to place the packages libraries into our system project folder and define the web server ports to access the programs by using a simple web browser (Chrome, Firefox, Safari, etc.).

The installation of the Spark framework for distributed computing is carried out by a script called `spark.sh`. This application can be used only with one master node (standalone mode), but the execution in cluster mode (with additional worker nodes) is possible by configuring some environment variables defined in the script `SparkConf.sh`. This script allows the users to modify several characteristics of the worker nodes as the number of CPUs assigned to them or the RAM available for each one.

Another important development tool that we aggregated to the Data Science stack is Jupyter, which provides the functionality to use both Python and R in the notebooks. The script `jupyter.sh` installs the main program and the kernel libraries to allow the execution of those languages in a notebook, as well as the configuration of the port to access the application through the web browser.

Table 1 shows the list of the applications configured in the Data Science stack with their corresponding ports and addresses to be run using a web browser. Besides, Table 2 summarizes the list of scripts and files used in the deployment.

Table 1. Web applications with default and configured ports in the Data Science stack, and links for accessing them

Application	Default port	Configured ports in deployment	Address link of application
RStudio	8787	10087	http://localhost:10087 (user: rstudio, password: rstudio)
Spark	8080, 8081, 7077	10080	http://localhost:10080
Jupyter	8888	10088	http://localhost:10088

Table 2. A summary of all the files used in the deployment

Located in	File name	Function
/	build_image.bat	Builds the container image in Windows
	build_image.sh	Builds the container image in Ubuntu or Mac
	Dockerfile	Docker instructions to build the images
install/	base.sh	Installs base system libraries and applications
	custom_python.sh	Allows installing certain versions of additional python libraries
	jupyter.sh	Installs Jupyter notebook and languages' kernels
	jupyter_notebook_config.py	Jupyter notebook configuration file
	PyLibraries.sh	Basic Python libraries
	python.sh	Installs versions 2.7 and 3.6 of Python
	R.sh	Installs the R programming language
	Rconfig.R	Installs the R kernel in Jupyter
	RStudio.sh	Installs the RStudio IDE
	spark.sh	Installs Spark
	SparkConf.sh	Spark configuration file
	start.sh	Entry point in Dockerfile: runs the container
Start/Ubuntu_Mac/	Docker_install.sh	Installs Docker in Ubuntu or Mac
	execute_worker.sh	Deploys one Spark worker node
	start_master.sh	Starts the Spark master
	start_workers.sh	Deploys additional worker nodes for Spark
Start/Windows/	runworker.bat	Deploys one Spark worker node
	start_master.bat	Starts the Spark master
	start_workers.bat	Deploys additional worker nodes for Spark

After the Dockerfile execution, the user can access the different applications by typing the links of Table 1 in the address bar of any browser, so now the system is ready to be used for the required Data Science tasks.

3.4 Deployment of the system

In the procedure to launch the environment, two scripts called `start_master` and `start_workers` are provided. The first one creates the network to run the container or cluster, and creates the user folder to place the notebooks in the host machine of the system. This folder is mounted as a data volume of the container and allows the users to access the files of the data stack environment from the host computer where this system is running. Besides, several R and Python libraries will be installed only the first time this script is run.

The second script is optional and enables the user to configure a cluster of N additional system instances to be run as workers. This cluster is valid for using the Spark framework, where the number of instances can be introduced in the `NWorkers` argument of the script.

In the following paragraphs, we describe the steps to deploy the data stack environment in a host computer with Windows and Linux Ubuntu operating systems.

Install and run in Windows (64 bits)

1. Open a command prompt (CMD) or Powershell using the Windows menu options.
2. Download the Github project typing:

```
git clone https://github.com/taroull/DockerForScience.git --config
core.autocrlf=input
```

3. Change to working directory typing: `cd DockerForScience`
4. If Docker is not installed yet, the user can download and install it from the official web page [17]. When Docker is installed, a process window is started and pinned into a quick access of the desktop. Before proceeding, users must enable the "Shared Drives" option in the settings.

In some systems, the user might need to configure appropriately the firewall or antivirus program to avoid problems when executing the following steps.

5. Optionally, the script `install\custom_python.sh` might be edited to include in variable `Libraries` some additional python libraries (separated by spaces) that the user wants to install in the stack. For example:

```
#!/bin/bash
#Specify as "<library>[==version] ... [<libraryN>[==version]]"
Libraries="pandas==0.21.0 scipy bokeh plotly"
```

6. Execute the script `.\build_image.bat`, placed in the root folder of the project, to generate the Docker image in the host system. This process may take some minutes depending on file sizes and the internet connection speed.

7. After creating the image, the system can be started by executing `Start\Windows\start_master.bat`.
8. Consequently, all applications in Table 1 are accessible by typing the corresponding address link within a web browser.
9. Optionally, we can use the script `Start\Windows\start_workers.bat <NWorkers>` to deploy additional worker nodes for Spark. The number of slaves is specified in `<NWorkers>`, being 2 the default value of this parameter.

Install and run in Ubuntu/Debian or Mac OSX

1. Open a linux terminal.
2. Download the Github project using:

```
git clone https://github.com/taroull/DockerForScience.git --config
core.autocrlf=input
```

3. Change to the working directory by typing: `cd DockerForScience`
4. If Docker is not installed already, the user can execute the install script `Start/Ubuntu_Mac/Docker_install.sh`
5. Optionally, the script `install/custom_python.sh` might be edited to include in variable `Libraries` some additional python libraries (separated by spaces) that the user wants to install in the Data Science stack. See the example shown in point 5 of the previous section.
6. Execute the script `./build_image.sh` (it may require preceding a `sudo` command if the user has no admin privileges), placed in the root folder of the project, to generate the Docker image in the host system. The building process may take some minutes depending on file sizes and the internet connection speed.
7. After creating the built image, the system can be started by executing `Start/Ubuntu_Mac/start_master.sh`.
8. Now the applications in Table 1 are accessible by typing the corresponding address link in a web browser.
9. Furthermore, and only for Spark, we can use the script `Start/Ubuntu_Mac/start_workers.sh <NWorkers>` to start additional worker nodes, where the number of instances is set through the parameter `NWorkers`. The default value of this parameter is two instances.

Finally, and regardless of the system considered (Windows, Ubuntu or Mac), the user might access the container through the following command:

```
docker exec -ti master bash
```

Bear in mind that the deployment process might take a while depending on the speed of both your computer and the internet connection. Once we have set up our Data Science stack, the three main applications shown in Table 1 are now running

in their corresponding ports at `localhost`, and ready to be used in a web browser (see Fig. 8, Fig. 9 and Fig. 10). Therefore, we can now proceed to show a case study in the next section.

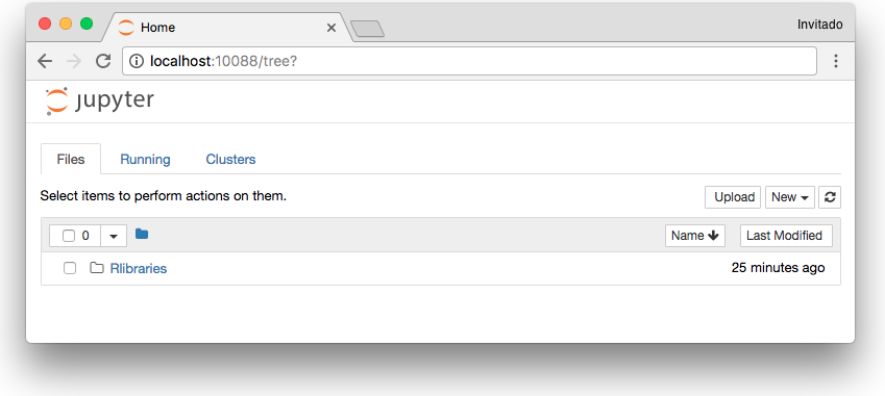

Fig. 8. The Jupyter notebook running in a web browser at localhost:10088

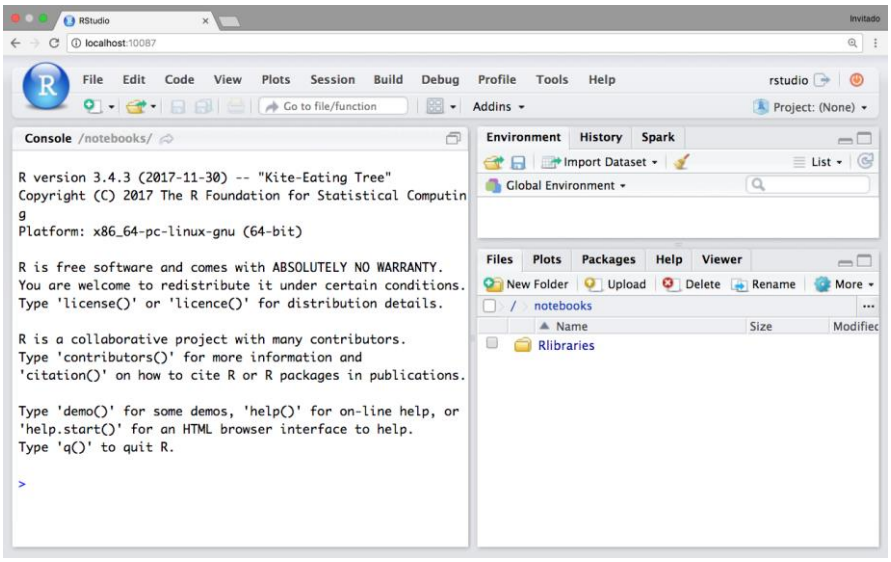

Fig. 9. RStudio running in a web browser

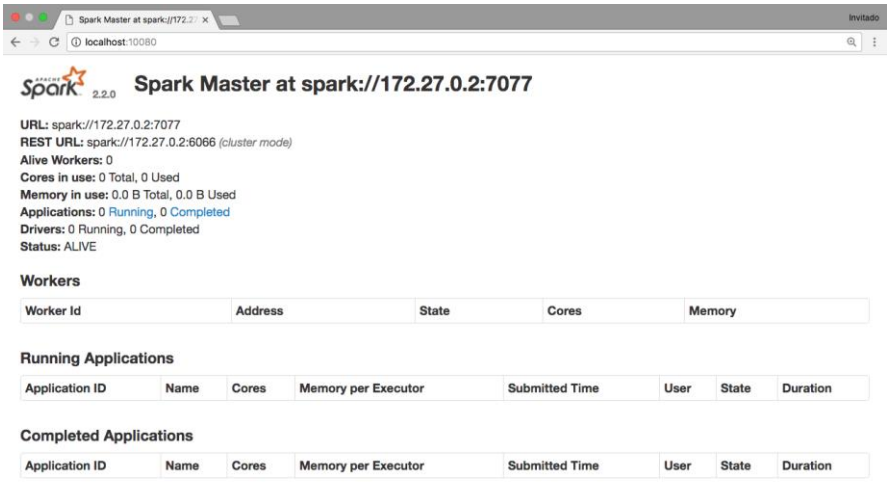

Fig. 10. The Spark Master deployed and running at localhost:10080

4 A case study

This section presents an illustrative example of data analysis by using the Data Science stack developed in this chapter. In particular, the problem consists in studying the hourly observations collected in meteorological stations located in the Canary Islands (Spain) in order to obtain different statistical results using a Jupyter notebook and several R libraries. These stations are included in the United States Air Force (USAF) Master Station Catalog and registered in the National Climatic Data Center's (NCDC) archive of weather and climate data.

The notebook can be downloaded by running the following command in the notebooks folder:

```
git clone https://github.com/taroull/Notebook-DataSciencewithR
```

For a given station, we can collect information by providing the beginning and ending years. Particularly, using the appropriate function in R, we get a data frame with information on hourly meteorological observations specifying a geographical bounding box and/or time bounds.

In this sense, the geographical position of the stations can be represented by using the `ggmap` library in R. The subsequent map (see Fig. 12) is easily obtained (see Fig. 11), and it can give us a good reference of the main locations where the data processed in the study were collected.

```
stations_canary$offset<--0.04
stations_canary$offset[stations_canary$name %in% c("IZANA","TENERIFE NORTE","TENERIFE SUR")]<-0.05

mapPoints <- ggmap(map) +
    geom_point(aes(x = lon, y = lat), data = stations_canary, col="red", alpha = .5) +
    geom_text(aes(x = lon, y = lat+offset, label=name), data = stations_canary, size = 2.0, hjust=
"left")+
    xlab("Longitude (degrees)") + ylab("Latitude (degrees)")
print(mapPoints)
```

Fig. 11. R script to draw the stations in a map

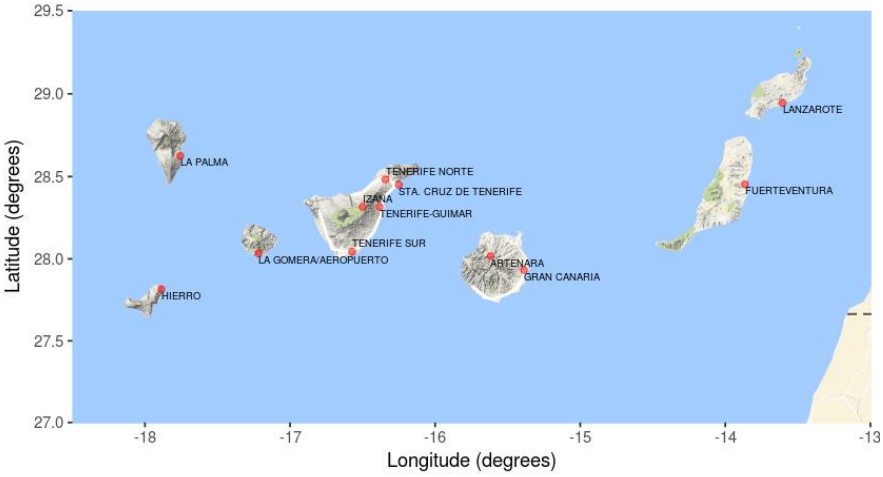

Fig. 12. The meteorological stations drawn on a map of the Canary Islands

In particular, we are interested in studying the weather conditions in 2017 in the Teide National Park, located in the island of Tenerife. Consequently, we get hourly information of the nearest station about time, wind speed and direction, and temperatures through 2017 by using the script described in Fig. 13, and the result is shown just underneath as a formatted table.

```
temps_at_izana$time<-as.POSIXct(strptime(temps_at_izana$time,format="%Y-%m-%d %H:%M:%S"),format="%
Y-%m-%d %H:%M:%S")
head(temps_at_izana)
```

time	wind_direction	wind_speed	temp
2017-01-01 22:00:00	260	11.3	5.0
2017-01-01 23:00:00	250	10.8	4.5
2017-01-02 00:00:00	260	11.3	5.5
2017-01-02 01:00:00	260	10.8	5.7
2017-01-02 02:00:00	250	12.9	5.1
2017-01-02 03:00:00	250	13.4	4.2

Fig. 13. The meteorological data of the Teide National Park's station

In order to analyze these data, first we can plot all these values (see Fig. 14) to represent the evolution of daily temperatures in the National Park in 2017 using `ggplot`. The plot (see Fig. 15) uses a color gradient to appreciate the transition from the cold temperatures (blue) to the warm ones (red), and includes a smooth regression model fitting to the data.

```
ggplot(temps_at_izana,aes(x = time,y = temp)) +
  geom_point(aes(colour = temp)) +
  scale_colour_gradient2(name = "Temperature level",low = "blue", mid = "green" , high = "red", mi
dpoint = 12) +
  geom_smooth(color = "red",size = 1) +
  scale_y_continuous(limits = c(-5,30), breaks = seq(-5,30,5)) +
  ggtitle ("Daily temperatures and local smooth fitting in 2017 \n Izaña (Canary Islands)") +
  xlab("Month") +  ylab ("Temperature ( ºC )") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
`geom_smooth()` using method = 'gam'
```

Fig. 14. R script that plots the data of the Teide National Park

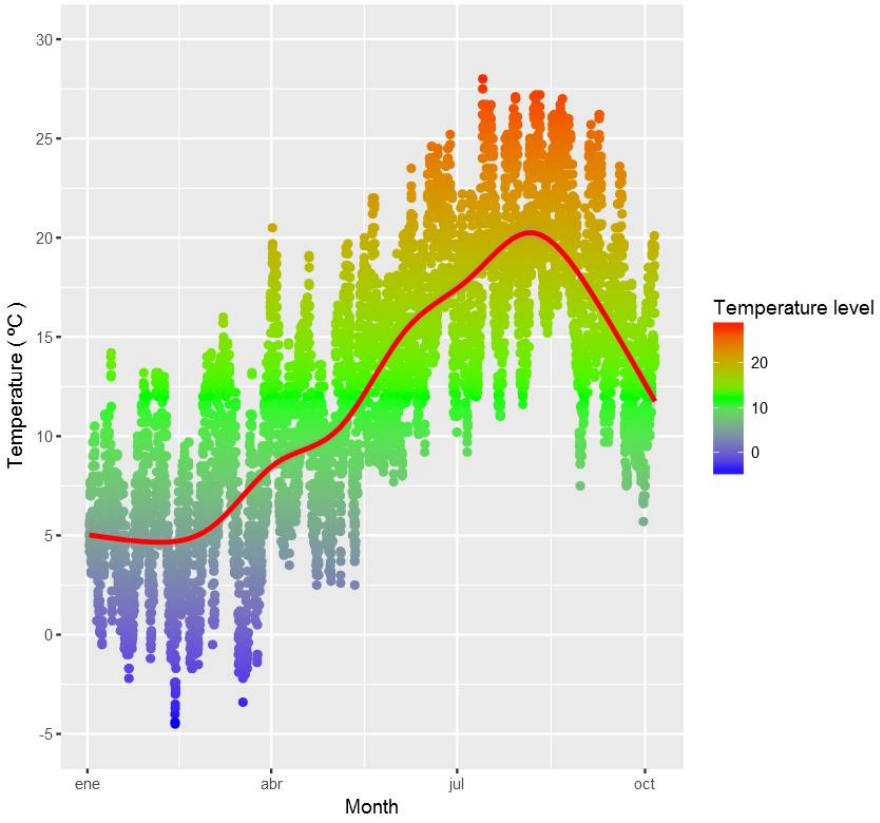

Fig. 15. Daily temperatures and local smooth fitting in Teide National Park in 2017

Another interesting result consists in representing the distribution of low and high temperatures along the different daily hours in 2017. The chart (see Fig. 17)

allows us to make a comparison study of which parts of the day are colder or warmer. Again, we can observe that this result is relatively simple to obtain with the `ggplot` library (see Fig. 16).

```
ggplot(temps_at_izana_temperatures) +
  geom_bar(aes(x = hour, fill = min.max.temp)) +
  scale_fill_discrete(name= "", labels = c("Daily high temperature","Daily low temperature")) +
  scale_y_continuous(limits = c(0,100)) +
  ggtitle ("Daily low and high temperatures distribution by hour \n Izaña (Canary Islands)") +
  xlab("Hour of the day") +  ylab ("Frequency") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

Fig. 16. R script to plot the histogram of temperatures by hour

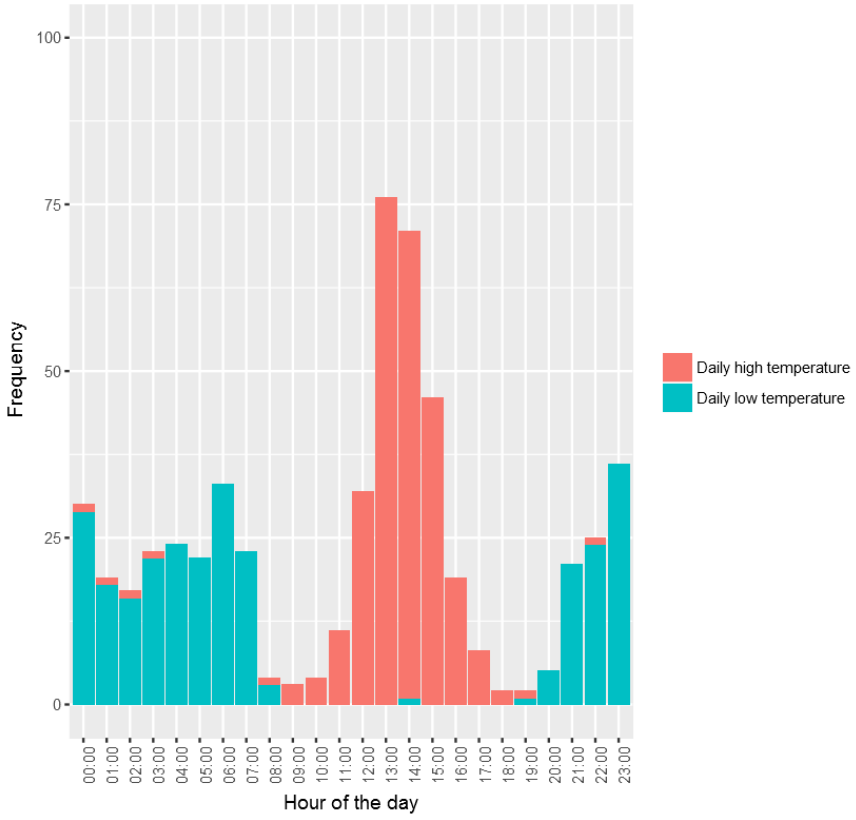

Fig. 17. Daily low and high temperature distribution by hour

Lastly, if the volume or complexity of the data is too large to be processed in a normal way using R, the user could easily connect to a Spark cluster in order to run the required analysis and get the final insights.

In this sense, and following the same example used in this section, we could now connect our notebook to a Spark master node to run some basic analysis (see Fig 18).

```
usePackage("sparklyr")
```

```
Loading required package: sparklyr
```

```
conf <- spark_config()   # Load variable with spark_config()
conf[["sparklyr.defaultPackages"]] <- NULL
conf$spark.executor.memory <- "16G"
sc <- spark_connect(master = "local", spark_home="/usr/local/spark",
                    config = conf)
```

Fig. 18. R script to connect to a Spark master node

Then, we copy our data to a Spark data frame to run some typical lazy Spark operations. In this case, we calculate the average temperatures grouped by month, as shown in Fig. 19.

```
temps_at_izana_avg<-temps_at_izana %>%
  group_by(month,month_name) %>%
  summarize(avg_wind=mean(wind_speed),avg_temp=mean(temp))
temps_at_izana_avg
```

```
# Source:   lazy query [?? x 4]
# Database: spark_connection
# Groups:   month
  month month_name avg_wind  avg_temp
  <chr>      <chr>    <dbl>     <dbl>
1    01      enero 9.073950  4.790476
2    02    febrero 7.803506  4.460518
3    03      marzo 7.712921  6.617900
4    04      abril 8.182260  9.586409
5    05       mayo 7.343103 12.443966
6    06      junio 5.969795 16.686657
7    07      julio 5.105517 18.588828
8    08     agosto 6.231077 19.353729
9    09 septiembre 7.713445 15.890588
```

Fig. 19. Example of some Spark lazy operations

Finally, we use the usual `collect()` function in Spark to run query and return the results back to R (see Fig. 20).

```
temps_at_izana_avg <- collect(temps_at_izana_avg)
temps_at_izana_avg
```

month	month_name	avg_wind	avg_temp
01	enero	9.073950	4.790476
02	febrero	7.803506	4.460518
03	marzo	7.712921	6.617900
04	abril	8.182260	9.586409
05	mayo	7.343103	12.443966
06	junio	5.969795	16.686657
07	julio	5.105517	18.588828
08	agosto	6.231077	19.353729
09	septiembre	7.713445	15.890588

Fig. 20. Returning the average result from Spark back to R

The notebook is completed with further interesting results that provide a brief overview of the collected information. As a result, the notebook allows publishing not only the analysis but even the R script along with the data. This turns to be extremely useful in case other researchers want to reproduce and validate the study without the complexity of installing all the required tools individually. Therefore, containers help to develop and deploy these complex setups in a simple way and run them in any environment.

5 Conclusions

By virtue of what has been stated in previous sections of this chapter, it has been clearly established that the infrastructure layer plays a key role within the Data Science stack. Indeed, there is a great concern on building good infrastructures to allow the stakeholders to run testbeds, sandboxes and proofs of concept (see [10], [11], [12] and [15]).

On the other hand, a new type of company defined as the insights-driven business [5], that uses Data Science to create competitive advantage has arisen in the last years. Furthermore, companies that make data driven decisions can raise up to a 5-6% their productivity [7], or become completely uncompetitive otherwise [8].

However, some of these businesses cannot afford such Data Science services in a commercial cloud. Accordingly, and following the recommendations indicated in [9], we have developed a Data Science platform that can be easily deployed over commodity computers using open source software that includes Spark as the current *de facto* standard, as well as the R and Python languages.

To summarize, the main advantages of our Data Science stack approach are the following:

- The R libraries and the Python packages that are installed within the notebooks folder after the Docker image is created, can be reused when the container or the image are run again.
- The scripts can update in the future the applications already installed by just changing the variables defined inside them.
- The shared volume inside the Docker container allows the user saving the data and related files in a folder inside the host computer.
- As a result, the notebook allows publishing not only the analysis but even the R script along with the data. This turns to be extremely useful in case other researchers want to reproduce and validate the study.

Likewise, from the point of view of the main actors the advantages are:

- System administrators take less time to configure.
- Data analysts or researchers can really focus to accomplish their main duties rather than wasting time in administration tasks.

- The academia can benefit from the easiness to develop executable examples in a very simple way.

Besides, the project presented in this chapter can be very helpful for teaching and researching activities, and has been already used in subjects regarding Laws and Regulations, Computer Vision, and Bioinformatics, and in a project on Earth and Atmosphere Observation Research. In this sense, the Data Science stack provides the students with the necessary tools for accomplishing different tasks in order to solve problems concerning Data Science (preprocessing data, statistical analysis, etc.), and preventing all the problems that usually arise when installing and configuring software.

In conclusion, our key objective has been essentially twofold. On the one hand, we have developed a simple, easy and fast platform to deploy a scalable Data Science stack that includes the main foundation tools. On the other hand, it has been designed for use in insights-driven businesses, as well as for obtaining reproducible results in research, and for teaching academic subjects easier.

Finally, the lines of future research are mainly focused in improving the current stack by including new Data Science environments like Zeppelin [30], and installing Deep Learning applications such as TensorFlow [31].

Acknowledgments This work is partially supported by the Spanish Ministry of Education and Science, Research Projects MTM2016-74877-P and CGL2015-67508-R, National Plan of Scientific Research, Technological Development and Innovation. The authors wish to thank Adrián Muñoz-Barrera and Luis A. Rubio-Rodríguez for their support and assistance both in the configuration and deployment of the cluster and in the development of the solution.

References

[1] NIST (2015a) Big Data Interoperability Framework: Volume 5, Architectures White Paper Survey. http://dx.doi.org/10.6028/NIST.SP.1500-5. Accessed October 2017.

[2] EDSF (2017) The EDISON Data Science Framework, Release 2: http://edison-project.eu/edison/edison-data-science-framework-edsf. Accessed October 2017.

[3] Plaza-Martín V, Pérez-González CJ, Colebrook M, Roda-García JL, González-Dos-Santos T, González-González JC (2016) Analyzing Network Log Files Using Big Data Techniques. In: García-Márquez FP, Lev B (ed) Big Data Management. Springer International Publishing, p 227–256

[4] NIST (2015b) Big Data Interoperability Framework: Volume 1, Definitions, http://dx.doi.org/10.6028/NIST.SP.1500-1. Accessed October 2017.

[5] Forrester (2016) Data Science Platforms Help Companies Turn Data Into Business Value. https://www.datascience.com/resources/white-papers/forrester-data-science-platforms. Accessed October 2017.

[6] Hazard C (2014) Stacking the Deck: The Next Wave of Opportunity in Big Data https://www.kdnuggets.com/2014/05/stacking-deck-next-wave-opportunity-big-data.html. Accessed October 2017.

[7] Brynjolfsson E, Hitt LM, Kim HH (2011) Strength in Numbers: How Does Data-Driven Decisionmaking Affect Firm Performance?. SSRN Electronic Journal. doi:10.2139/ssrn.1819486

[8] Capgemini Consulting (2015) Big & Fast Data: The Rise of Insight-Driven Business, http://ww.capgemini.com/wp-content/uploads/2017/07/big_fast_data_the_rise_of_insight-driven_business-report.pdf. Accessed October 2017.
[9] Linden A, Krensky P, Hare J, Idoine CJ, Sicular S, Vashisth S (2017) Magic Quadrant for Data Science Platforms. https://www.gartner.com/doc/reprints?id=1-3TK9NW2&ct=170215&st=sb. Accessed October 2017.
[10] NITRD (2016) The Federal Big Data Research and Development Strategic Plan, http://ww.nitrd.gov/PUBS/bigdatardstrategicplan.pdf. Accessed October 2017.
[11] BDV (2017) Big Data Value Strategic Research and Innovation Agenda. http://ww.bdva.eu/sites/default/files/EuropeanBigDataValuePartnership_SRIA__v3_0.pdf. Accessed October 2017.
[12] COTEC (2017) Generación de talento Big Data en España (in Spanish). http://cotec.es/media/BIG-DATA-FINAL-web.pdf. Accessed October 2017.
[13] Apache Hadoop. http://hadoop.apache.org. Accessed October 2017.
[14] Apache Spark. https://spark.apache.org. Accessed October 2017.
[15] NIST (2015c) Big Data Interoperability Framework: Volume 3, Use Cases and General Requirements. http://dx.doi.org/10.6028/NIST.SP.1500-3. Accessed October 2017.
[16] HC-STC (2016) The big bata dilemma. http://www.publications.parliament.uk/pa/cm201516/cmselect/cmsctech/468/468.pdf. Accessed October 2017.
[17] Docker: the container platform provider. http://www.docker.com. Accessed October 2017.
[18] Docker hub. https://hub.docker.com. Accessed October 2017.
[19] Project Jupyter. http://jupyter.org. Accessed October 2017.
[20] RStudio: the open source and enterprise-ready profesional software for R. https://www.rstudio.com. Accessed October 2017.
[21] Python. https://www.python.org. Accessed October 2017.
[22] The R Project for Statistical Computing. https://www.r-project.org. Accessed October 2017.
[23] Anaconda Python distribution., https://www.anaconda.com/download/. Accessed October 2017.
[24] Enthought Canopy Python distribution., https://www.enthought.com/product/canopy/. Accessed October 2017.
[25] Datacamp: Learn Data Science Online. https://www.datacamp.com. Accessed October 2017.
[26] Codecademy: Learn to code interactively for free. https://www.codecademy.com. Accessed October 2017.
[27] Rodeo: a Python IDE built for analyzing data. https://www.datascience.com/blog/docker-containers-for-data-science. Accessed October 2017.
[28] The Open Group Architecture Framework (TOGAF) Version 9.1. The Open Group. http://www.opengroup.org/togaf. Accessed October 2017.
[29] Lankhorst MM (2004) Enterprise architecture modelling—the issue of integration. Advanced Engineering Informatics 18(4):205–216
[30] Zeppelin. https://zeppelin.apache.org. Accessed October 2017.
[31] Tensorflow. https://www.tensorflow.org. Accessed October 2017.