Infrastructure-as-Code for Scientific Computing Environments

Daniel Adorno Gomes

University of Trás-os-Montes and Alto Douro
Vila Real, Portugal
www.utad.pt
Email: adornogomes@gmail.com

Pedro Mestre
and Carlos Serôdio

Centro Algoritmi
University of Minho, Guimarães, Portugal
CITAB - Centre for the Research and Technology of
Agro-Environmental and Biological Sciences
University of Trás-os-Montes e Alto Douro
www.utad.pt
Email: pmestre@utad.pt, cserodio@utad.pt

***Abstract*—Infrastructure as Code (IaC) is receiving a lot of attention because of the positive results demonstrated in environment provisioning that supports both corporate and scientific applications. One of the biggest challenges related to the software development that supports scientific research projects is to obtain similar results to those that were published when that research is reproduced. Obtaining the same results involves recreating the same computational environment as the one used by the original researchers, and IaC is a paradigm that can help on this issue. In this paper, the authors provide a state-of-the-art and a literature review on IaC. Some work done on infrastructure provisioning based on IaC is presented and analyzed. A section shows how IaC can solve some typical issues on scientific computational environments. In other words, it is described how IaC practices were applied in a real case. Based on what is presented in this paper, and considering the scientific context, the use of IaC can help in many aspects like the quality of the developed software and the improvement of the results obtained in reproducible research.**

***Keywords*–IaC; devops; infrastructure-as-code; configuration script; continuous deployment; continuous delivery; continuous integration; reproducible research.**

I. Introduction

The provision of the computational environment that supports scientific research has always been one of the main problems faced by researchers. In 2009, Jon Claerbout, a researcher at Stanford University who worked with data analysis and development of algorithms for geophysical exploration, reported that he had great difficulty reproducing algorithms and results reported by other researchers [1]. He reported also, that the same situation was occurring in his laboratory. Quite frequently his students and researchers faced problems to reproduce the results generated by their own research [2].

Even when we have similar setups, but on different servers, if for example the operating system is not the same, the software that will run on those servers can have different behaviors. It is therefore essential to have the same dependencies between the software components of the environment, to get similar results to those that have been disclosed. The computational environment used in the context of a particular scientific research has to be exactly the same in all aspects related to software, especially with respect to operating systems, compilers, libraries and their respective versions [3].

Enterprise and web applications differ in many aspects from scientific software, but when the subject is the infrastructure, they have faced the same issues. To guarantee the behavior of the applications will be the same on development, test and production environments, many companies like Github, Mozilla, Facebook, Google and Netflix have adopted an approach called Infrastructure-as-Code (IaC). The idea behind IaC is to provide the entire computer and network infrastructure through scripts. That is, no task that is related to the provisioning of the computational environment must be done manually, but in a programmatic way, both to avoid mistakes and to make the environment reproducible as many times as needed [4][5].

As we can see in Figure 1, there was a growing interest on IaC, mainly, in the last 6 years. The chart, obtained from Google Trends, shows the numbers of search results for the term ”Infrastructure as Code” over the last 10 years.

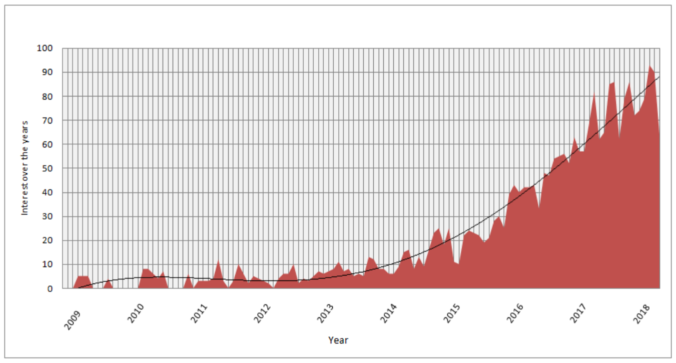


Figure 1. Search results for the term ”Infrastructure as Code”. Based on [4].

This paper presents a state-of-the-art and a literature review on IaC, that supports the section that highlights which IaC methods and techniques fit best for scientific computing environments.

The rest of the article is organized as follows: Section 2 presents the state-of-the-art on IaC. Section 3 makes a review of the literature on IaC. Section 4 presents a series of suggestions on how IaC can be applied on scientific computing environments. Section 5 describes a real case of applying IaC practices. Finally, in Section 6 it is presented the discussion and the conclusion.

II. INFRASTRUCTURE-AS-CODE

In the last decade, due the growing of cloud computing demands, a new practice has been adopted to provide computing and network infrastructure through the use of source code. This approach, called Infrastructure-as-Code (IaC), permits to manage infrastructure the same way as if it was a software systems [6].

With IaC we have a dynamic infrastructure where the servers are created with all the configuration and software needed, just by using commands in executable files such as shell-scripts. All the operations like to create a new server, change its configuration, install or uninstall software depends on just running a script.

Using scripts to define our infrastructure means that our environments will have more consistency and it will be more reliable. Manual provisioning can have different interpretations of the same instructions, resulting in different configurations and faults that are not easy to identify, mainly, when these environments are not monitored. Scriptable infrastructure can guarantee a consistent monitoring, as well [7].

This approach is based on some practices as follows [8]:

- Definition Files: all configurations are defined in shell-script files called definition files. This means that all the updates that we need to apply in the infrastructure are executed from these files for example install a database, increase the memory of a Virtual Machine and create a new server. There is no manual intervention. As all the updates are defined by source code in scripts, the changes can be applied in test environments as much as necessary to certify the production environments will be modified correctly.
- Self-documented systems and processes: Instead of creating documents with instructions to be executed by humans, scripts can contain the documentation of the systems and processes and the commands that need to be applied in the infrastructure. Besides, they are more precise and consistently than humans when executing these instructions.
- Version control: all the code need to be kept in a version control system like Git or Github. This way the infrastructure can be versioned. Every configuration and every change is kept to help diagnose problems that might happen.
- Continuous test: like we have seen recently on modern software system, tests are essential to rapidly find errors, even in infrastructure source code. Like any other kind of software, we can set up continuous integration pipelines to test and guarantee the quality of our code. The continuous integration will support the continuous deployment and delivery practices.
- Continuous deployment: an update with many simultaneous changes on the infrastructure can present a high number of issues. Continuous deployment pipelines can help to mitigate this problem, once making small updates it is faster to find errors and fix them.
- Continuous delivery: with this practice it is possible to decrease the downtime of the systems on upgrades or fixes, using techniques like Blue-Green Deployment and Parallel Change to apply small updates.

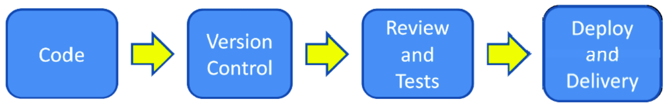


Figure 2. Infrastructure-as-Code workflow.

Adopting these practices, we can follow a well-defined workflow to provide infrastructure using IaC, as shown in Figure 2.

III. RELATED WORKS

In this section authors present a literature review and related works, made by other authors done in the field of infrastructure-as-code. These authors applied it to different areas of the software industry and scientific applications.

Garcia and Castillo [9] made a presentation about the main characteristics of the Cloud computing and virtualization techniques, discussing how benefit these features could be when used in scientific applications. In this paper, these authors highlight that neither virtualization nor cloud resources are commonly used in scientific computing environments due an idea that virtualization techniques have a negative impact on this kind of environments. They discuss the viability of the Infrastructure-as-a-Software cloud paradigm to attend the requirements of the computational science and the main issues that need to be addressed by the cloud players to provide the needed conditions to obtain the maximum benefit from this type of infrastructure.

Cole and Moore present in [10] a guide for creating biomedical workflows based on cloud computing environments. They show some cloud computing tools and characteristics that can help to increase reproducibility, scalability, resilience, fault-tolerance, security of software applications. Also they highlight how this paradigm can be cost and time efficient. They provide also an overview on how researchers can make the transition of their traditional biomedical informatics workflows to cloud environments.

In [11], Parnin et al. present a research work about the use of IaC in continuous deployment. Professionals from 10 different companies were interviewed by the authors. They reported the practice of IaC have changed the way how IT companies are managing their infrastructure. They also reported that the most frequent automation is related with unit testing, staging, and branching. When development focuses on delivery speed is essential to adopt continuous deployment, but some aspects like architecture and safety can have a decrease in quality, and they need a special attention.

In [12] members of the UC Berkeley D-Lab, Statistical Computing Facility (SCF), and Berkeley Research Computing (BRC) present strategies to reduce the complexity on creating scientific computing environments based on DevOps concepts. They start by showing some training, research use-cases and tools that support their strategies and help to create environments with more accessibility, productivity, reuse, and reproducibility. After, they present a DevOps framework that supported the creation of the Berkeley Common Environment (BCE). The BCE provides a standard reference that explains how to run a variety of projects on different platforms. This environment is available to be deployed on VirtualBox and Amazon EC2 environments, but it can be also provisioned for other platforms. It is expected to achieve identical results across any platform.

Howe [13] makes a presentation about the main characteristics of the virtualization on cloud computing platforms, focusing the use of virtual machines to provide computational environments for scientific research projects. He refers that a virtual machine can contain the entire working environment of the research project, including its data, all software, notes, logs and scripts, facilitating the reproducibility of the research. He presents a discussion about the consequences and benefits of the use of such a model, and shows the adaptations that are needed to use this approach in scenarios where of the reproducibility is more complex.

Boettiger presents in [14] the issues related with the reproducibility of the source code developed for a specific research project. He points the reasons why the code cannot be executed or extended by other researchers with success. Also, he provides a review on different approaches like virtual machines and workflow systems, showing their limitations. The Docker technology is analyzed by the author, that shows the advantages of the containerization like portability, reusability, versioning and cross-platform, and how it can address those challenges that are related to the provisioning of computational environments for scientific research.

IV. IaC for Scientific Computing Environments

Based on the state-of-the-art and the review of the literature presented in the above section, in this topic it will be highlighted the advantages of the adoption of the IaC practices and techniques, on scientific environment provisioning. Some typical issues faced by researchers will be shown and it will be discussed how IaC can help to solve them.

As mentioned in the above Introduction section, one of the most common issues reported by researchers on reproducing results is related with the computational environment. Less than 50% of software can be built or installed successfully. It is necessary for a huge effort to recreate the original environment and get similar results [15]. IaC fits perfectly in this scenario. Once the environment is defined in shell-script files and these files are stored in version control repositories, researchers can reproduce exactly the original environment anytime. Besides, the environment can be versioned and the results of the research can be related to each version.

Another issue reported is related to the documentation, and on how to install and run the code associated with a published work. This problem is related not just with the software environment but with the code produced by the researchers as well. Gilbert et al. [16] published a study where just 30% of the analyzed researches had their results reproduced due imprecise documentation. This kind of problem can be easily addressed using IaC practices once all the documentation is embedded in the definition files combining instructions and code in the same place.

A problem known as "code rot" is a kind of issue that affects the results of the reproducible research due updates of the computational environments. Before to start creating a new system, developers have to decide which will be the Operating System, the development language, the libraries and the database technology that will be used in the project. Practically, the new software will have many dependencies of this environment that are not static [14]. A simple update to fix bugs on the operating system or replace deprecated features in the libraries can change the results produced by the code. To mitigate this kind of concern, there are some practices used in IaC like containerization and virtualization that permit to create and store an image of the entire environment. In this way, the results can be reproduced by other researchers using the same infrastructure created by the original researchers [17].

V. A Case Study

In this section will be described, in a brief way, all the steps and resources needed to provisioning an environment to support a real scientific experiment on computer vision. This environment was used for the development of a software application capable of identifying canine dysplasia cases, from the analysis of x-ray images. This research involved researchers from the Agrarian and Veterinary Sciences School and the Science and Technology School of the University of Trás-os-Montes and Alto Douro, Portugal.

Initially, it was defined that the environment should be provisioned as a virtual machine. In this way, it could be recreated at any time, by any of the researchers, in any available platform, more specifically, stand-alone, on-premise or cloud platform. For this project it was decided empirically that the VM should have 4 GB of RAM and 50 GB of disk space. In Table I are shown which software applications were defined to compose the environment.

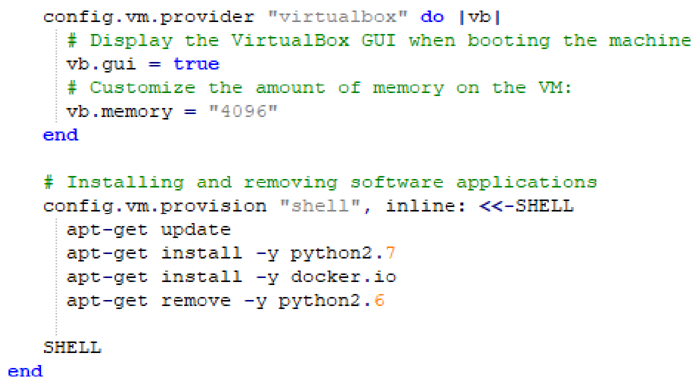
TABLE I. List of software applications defined to compose the environment.

Type	*Software*	*Version*
Operating system	Linux Ubuntu Server	Xenial v16.04 (LTS)
Programming language	Python	v2.7.12
Container platform	Docker	v18.09.0
Deep learning framework	Tensorflow	v1.7

The flow to provide this environment started with a tool called Vagrant v2.2.5. All the software and hardware needed to create the environment were described on Vagrant that generated a script with all these definition, and executed it creating a VM. The Figure 3 shows part of the script source code.

The first version of the script was uploaded to Github repository. The VM were used to execute tests on the installed software and their respective versions. After the test, the Vagrant script was updated with new commands to remove all versions of Python different of the version chosen by the researchers to avoid conflicts. After more tests, a new version of the script was uploaded to Github.

The last step was to validate if the code produced by the researchers could be compiled and ran properly on this environment, generating consistent results. Having a tested version of the script on Github, all the researchers could download it and create the same environment on their own equipment to reproduce the experiment, getting the same results.



```
   config.vm.provider "virtualbox" do |vb|
     # Display the VirtualBox GUI when booting the machine
     vb.gui = true
     # Customize the amount of memory on the VM:
     vb.memory = "4096"
   end

   # Installing and removing software applications
   config.vm.provision "shell", inline: <<-SHELL
     apt-get update
     apt-get install -y python2.7
     apt-get install -y docker.io
     apt-get remove -y python2.6

   SHELL
end
```

Figure 3. Script with the definitions of the environment.

This is a very simple example case study, but that can be used to show that the use of Infrastructure-as-Code can be very helpful to obtain reproducible research.

VI. CONCLUSION AND FUTURE WORK

In this paper the authors made a presentation of the state-of-the-art and a literature review on Infrastructure-as-Code and how it can be used to obtain reproducible research. Some aspects related to practices and techniques that are used by other researchers were discussed, and a workflow based on these items was proposed.

In the section about the work done by in this field, we could follow how IaC can have a very positive impact in software development development, both for enterprise and scientific applications. Also, this work discussed some typical issues related to scientific computational environments and showed how IaC can help to address them.

As future work, we propose to implement a scientific computational environment applying the practices of IaC, and compare it with traditional methods of computer infrastructure provision, in terms of costs and time.

The above presented methods will also be very helpful in other Data Center applications, in Virtual Machine provisioning, by defining the base infrastructure for other scientific projects under development in the authors Research Centres and University, or even to define the basic infrastructure to be used by students in their projects.

ACKNOWLEDGMENT

This work is supported by National Funds by FCT - Portuguese Foundation for Science and Technology, under the project scopes UID/CEC/00319/2019 and UID/AGR/04033/2010.

REFERENCES

[1] D. L. Donoho, A. Maleki, I. U. Rahman, M. Shahram, and V. Stodden, “Reproducible Research in Computational Harmonic Analysis,” Computing in Science Engineering, vol. 11, no. 1, Jan 2009, pp. 8–18.

[2] M. Schwab, N. Karrenbach, and J. Claerbout, “Making scientific computations reproducible,” Computing in Science Engineering, vol. 2, no. 6, Nov 2000, pp. 61–67.

[3] L. A. Barba and G. K. Thiruvathukal, “Reproducible Research for Computing in Science Engineering,” Computing in Science Engineering, vol. 19, no. 6, November 2017, pp. 85–87.

[4] A. Rahman, R. Mahdavi-Hezaveh, and L. Williams, “Where Are The Gaps? A Systematic Mapping Study of Infrastructure as Code Research,” Information and Software Technology, vol. 18, April 2018, pp. 65–77.

[5] M. de Bayser, L. G. Azevedo, and R. Cerqueira, “ResearchOps: The case for DevOps in scientific applications,” in 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), May 2015, pp. 1398–1404.

[6] D. Chapman. Introduction to DevOps on AWS. [Online]. Available: https://d1.awsstatic.com/whitepapers/AWS_DevOps.pdf [retrieved: September, 2019]

[7] Introduction to DevOps on AWS. [Online]. Available: https://d1.awsstatic.com/whitepapers/AWS_DevOps.pdf [retrieved: September, 20179]

[8] K. Morris, Infrastructure as Code: Managing Servers in the Cloud, 1st ed. O’Reilly Media, Inc., 2016.

[9] Á. L. García and E. Fernández-del-Castillo, “Analysis of Scientific Cloud Computing requirement,” in Proceedings of the 7th Iberian Grid Infrastructure Conference, Madrid, Spain, September 2013, pp. 147–158.

[10] B. S. Cole and J. H. Moore, “Eleven quick tips for architecting biomedical informatics workflows with cloud computing,” PLOS Computational Biology, vol. 14, no. 3, March 2018, pp. 1–11.

[11] C. Parnin, E. Helms, C. Atlee, H. Boughton, M. Ghattas, A. Glover, J. Holman, J. Micco, B. Murphy, T. Savor, M. Stumm, S. Whitaker, and L. Williams, “The Top 10 Adages in Continuous Deployment,” IEEE Software, vol. 34, no. 3, May-June 2017, pp. 86–95.

[12] D. Clark, A. Culich, B. Hamlin, and R. Lovett, “BCE: Berkeley’s Common Scientific Compute Environment for Research and Education,” in Proceedings of the 13th Python in Science Conference, Austin, USA, July 2014, pp. 5–12.

[13] B. Howe, “Virtual Appliances, Cloud Computing, and Reproducible Research,” Computing in Science Engineering, vol. 14, no. 4, July-August 2012, pp. 36–41.

[14] C. Boettiger, “An Introduction to Docker for Reproducible Research,” ACM SIGOPS Operating Systems Review - Special Issue on Repeatability and Sharing of Experimental Artifacts., vol. 49, no. 1, Jan. 2015, pp. 71–79.

[15] C. Collberg, T. Proebsting, G. Moraila, A. Shankaran, Z. Shi, and A. M. Warren. Measuring Reproducibility in Computer Systems Research, Technical Report. [Online]. Available: http://reproducibility.cs.arizona.edu/tr.pdf [retrieved: Septeber, 2014]

[16] K. J. Gilbert, R. L. Andrew, D. G. Bock, M. T. Franklin, N. C. Kane, J.-S. Moore, B. T. Moyers, S. Renaut, D. J. Rennison, T. Veen, and T. H. Vines, “Recommendations for utilizing and reporting population genetic analyses: the reproducibility of genetic clustering using the program STRUCTURE,” Molecular Ecology, vol. 21, 2012, pp. 4925–4930.

[17] J. Ooms, “Possible Directions for Improving Dependency Versioning in R,” The R Journal, vol. 5, 2013, pp. 197–206.