A preliminary analysis of Infrastructure as Code in open source repositories

Luciano Baresi, Elisabetta Di Nitto, Michele Guerriero, Giovanni Quattrocchi
Politecnico di Milano

Infrastructure as Code (IaC) is a novel paradigm that makes system administrators' work quite similar to the work of programmers. In fact, today system administrators (sysadmins) do not have to manually perform the provisioning, configuration, and deployment processes as they can exploit some IaC languages to write pieces of code which automate such processes and can be executed more and more times.

Thanks to this possibility, infrastructural code can become part of the whole codebase associated to a certain software system and can, therefore, be maintained, verified, fixed and optimized using approaches similar to those adopted for traditional software code.

IaC languages are always associated to corresponding engines that interpret the IaC code and execute it. In almost all cases each language is specific to a particular engine. The only exception is TOSCA, which is a standard language supported by a variety of engines, e.g., Cloudify, Opera and Brooklyn (this last one supports also another language).

The combination of IaC languages and the corresponding engines allow people without sysadmin skills, e.g., application logic developers, to execute the set up of a whole software system, thus easily replicating the application operational environment. This capability is an important enabling factor for DevOps practices as it improves the ability of developers to run complete complex systems, get a clue about their actual behavior, and identify possible optimizations. Moreover, it allows the Dev and Ops sub-teams to work in a coordinated way and to be aware of each other's activities.

Since open source software development is often the place where innovative approaches and practices are applied, we have been trying to assess the adoption of IaC in open source projects.

With this objective in mind, we have analysed the situation in GitHub. We have considered the population of active repositories, i.e., the highly rated (ten stars) repositories with at least one commit from 2018, these are 460,155. Within this population, we have searched for those that include in their description some reference to well-known IaC languages. In Table 1, the first and second columns show the results of this analysis with reference to those languages that, according to [1], are the most well-known and used. In total, about 1.3% of the repositories in the initial population includes some reference to IaC in their description. This number can be considered a sort of lower bound for the actual number as we expect that there are repositories storing IaC but not referring to this explicitly in their description. This is because IaC is a support code and, as such, not necessarily mentioned among the most relevant characteristics of an application. Moreover, this percentage, even if small, is still significant considering that IaC is a relatively new technology. To give an idea, according to [2], Matlab is the 10th most used programming language with a share of 2%.

In terms of popularity of the various IaC languages, the analysis confirms the findings published in [1] and based on the results of a survey with 44 respondents. The most referenced languages are Kubernetes (2,269 repositories), Ansible (1,620 repositories) and Terraform (561 repositories) (see the second column of Table 1). Conversely, TOSCA, which is the only standardization effort in the list, still features a limited adoption in the open source context we have considered.

While an in depth analysis of IaC bugs is the subject of our future work, we have started analysing the commit history of some of those repositories that include IaC. Due to the limitations of the querying API offered by GitHub, we could select at most 1,000 repositories for each of the IaC languages we have considered. This has allowed us to consider 4,240 repositories (see the third column in Table 1). These show an average percentage of IaC code that vary from being less than 1% to about 15% (see the seventh column in Table 1), with an average number of LOC for IaC software which is below 200 (see the sixth column in Table 1). It is interesting to see that some maintenance activities are specifically dedicated to IaC, with an average number of commits due to bug fixing on IaC code that varies between the 121 of Puppet (20 commits per year) and the 4 of Brooklyn (0.67 per year) (see the fifth column of Table 1), with most repositories that see at least one commit per year from the time the first IaC file has

appeared in the repository (eighth column) to the current time. Looking at the specific characteristics of the languages we have considered, it seems that configuration languages such as Puppet and Chef need more files compared to those other languages which are focusing on orchestrating a complete deployment process.

Language/tool	Total population per IaC language	Number of extracted repositories	Total number of bug commits	Avg number of bug commits in repo	Avg number of LOC in IaC files	Avg percentage IaC files in repo	First IaC file addition in repo
Salt	74	74	732	13.81	50.20	9.48%	2013-04-26
Brooklyn	14	14	4	4.00	67.45	0.33%	2015-12-16
TOSCA	23	23	27	6.75	111.25	1.34%	2015-10-26
Cloudformation	223	223	105	6.18	219.58	6.53%	2014-04-19
Terraform	561	561	4233	14.35	81.19	9.72%	2015-02-03
Docker Compose	332	332	436	4.89	38.25	3.45%	2015-02-27
Packer	179	179	866	20.14	172.52	4.85%	2013-07-18
Puppet	180	180	18595	120.75	79.91	11.84%	2007-08-07
Kubernetes	2269	1000	5218	64.42	159.29	2.68%	2015-09-01
Chef	290	290	9201	38.49	81.81	14.54%	2009-01-17
Vagrant	364	364	223	4.37	55.43	2.75%	2012-02-14
Ansible	1,620	1000	8212	28.22	61.98	5.63%	2012-10-19
Total	**6129**	**4240**	**47852**	**326.36**	**1178.87**		

Table 1. Analysis of IaC languages adoption in GitHub.

We do not yet know the causes and the implications of the differences we have highlighted, but our analysis certainly shows a significant interest in the open source community around IaC. Also, it shows the presence of a significant number of languages, some of which were born only recently (for instance, from Table 1 we can see that the first Kubernetes file has been included in a repository in 2015, but are already quite well-known and adopted.
The SODALITE project will enter in this scenario with the objective of offering tools that make possible the adoption of multiple IaC languages and tools for managing complex and highly heterogeneous software systems and that, at the same time, allows users to abstract from the peculiarities of such languages by exploiting a high level modeling language and inference mechanisms that guide the IaC definition process.

References

[1] M. Guerriero, M. Garriga, D.A. Tamburri, F. Palomba, “Adoption, Support, and Challenges of Infrastructure-as-Code: Insights from Industry”, International Conference on Software Maintenance and Evolution (ICSME) 2019.

[2] Ahsen Saeed, Here Are the Ten Best Programming Languages to Learn in 2019. Blog post. https://codinginfinite.com/best-programming-languages-to-learn-2019/