Challenges Towards Modeling and Generating Infrastructure-as-Code

Galia Novakova Nedeltcheva†
DEIB, Politecnico di Milano
Milan, Italy
galia.nedeltcheva@polimi.it

Bin Xiang
CNRS@CREATE
Singapore
bin.xiang@cnrsatcreate.sg

Laurentiu Niculut
HPE CDS
Rome, Italy
laurentiu.niculut@hpe.com

Debora Benedetto
HPE CDS
Rome, Italy
debora.benedetto@hpecds.com

ABSTRACT

The infrastructure-as-code (IaC) is an approach for automating the deployment, maintenance, and monitoring of environments for online services and applications that developers usually do manually. The benefit is not only reducing the time and effort but also the operational costs.

This paper aims at describing our experience in applying IaC in cloud-native applications, mainly discussing the key challenges towards modeling and generating IaC faced in the ongoing project Programming Trustworthy Infrastructure-As-Code in a Secure Framework (PIACERE). The concluding insights could spur the wider adoption of IaC by software developers.

CCS CONCEPTS

• Computer systems • Cloud Computing • Software and its engineering

KEYWORDS

Infrastructure-as-Code (IaC), Modeling IaC, Generating IaC, DevOps, Challenges

ACM Reference format:

Galia Novakova Nedeltcheva, Bin Xiang, Laurentiu Niculut and Debora Benedetto. 2023. Challenges Towards Modeling and Generating Infrastructure-as-Code. In *the Companion of the 14th ACM/SPEC International Conference of Performance Engineering (ICPE'23), April 15–19, 2023, Coimbra, Portugal. ACM, New York, NY, USA, 5 pages.* https://doi.org/10.1145/3578245.3584937

1 INTRODUCTION

The infrastructure-as-code (IaC) is a method of deploying environments for applications, as well as optimizing infrastructure management and deployment time.

ICPE '23 Companion, April 15–19, 2023, Coimbra, Portugal.

ACM ISBN 979-8-4007-0072-9/23/04. https://doi.org/10.1145/3578245.3584937

The IT infrastructure includes Virtual Machines (VMs) and associated configuration resources. IaC requires three elements to function: resource pooling, software-defined intelligence, and a unified API. Software development tools, such as deployment orchestration, version control systems, and automated testing libraries are used to manage the infrastructure [1], [2]. Moreover, the development and maintenance of IaC should follow the software engineering methodologies and best practices [3].

Also, IaC defines where new infrastructure is deployed (ex: public or private cloud), the type of service it will run on, and the settings and security that should be enabled. The deployment models are repeatable and can be changed and tested to make the deployment and management of the infrastructure consistent [4], [5].

Cloud computing, containers, virtualization, orchestration, and networking applications are used to streamline IT operations. With that in mind, software provisioning, configuring, and maintenance should require less time and effort. Other than that, the problems should be promptly identified and resolved [1], [6].

The DevOps culture and practices are evolving rapidly. DevSecOps addresses security vulnerabilities while leveraging automation. Various new tools are being built and there is no single way of applying DevOps practices by practitioners. In fact, the DevOps and IaC models necessitates a high level of technical competence. This is the reason why IaC services are often outsourced in order to improve their automation process in terms of time, cost, and quality of the IT infrastructure. Overall, IaC is an optimal approach to managing modern IT environments and supporting successful DevOps [1], [4], [7].

After pointing to a summary of the related work and positioning our contributions in terms of identifying solutions to issues, the paper continues with a discussion on the key challenges faced in the ongoing project PIACERE towards modeling and generating IaC. The proposed approach and discussed IaC concerns relate to the field of computing continuum, from Cloud to IoT

We try to report our experience and provide a meaningful perspective of the issues that might be tackled within similar innovation and research projects as the PIACERE.

2 RELATED WORK

Some of the challenges in front of IaC have been discussed in the literature, but are not limited to, include the following ones [8]:

- *Configuration:* The IaC development process is usually automated however, some of the process steps need to be executed manually, e.g. the generation of the parent code. In this respect, a risk of technical faults due to the man effort involved could be always expected [1].
- *Code execution:* Rather than managing physical system configurations, developers can instead use scripts written as code executed on the infrastructure [2]. Also, it is recommendable that the configuration is separated from the code [3].
- *Dependencies between the different components*: They are manually passed as configuration parameters, leading to a complex set of templates and pipelines, which is difficult to maintain [3].
- *Access management:* Developers often require privileged access to specific systems however, this creates a risk of unnecessary entry into mission-critical systems. Cloud administrators generally enforce access management using IAM (Identity Access Management) [2].
- *Security of the generated IaC code:* Security risks in configuration include poor configuration, human input, applications undergoing unintended changes, and manual edits into the cloud terminal [2].
- *Integration of the existing target code in the model* or use in the generated IaC code: If some other resources exist like VMs, it is important how to reuse them in the models and in the IaC code.

In the following section 3, we provide a discussion and indicate how the ongoing PIACERE project adds to what is already available on the topic. In particular, we advance the related state-of-the-art by adopting an abstract modeling approach and trying to generalize the common concepts from different cloud applications and infrastructures.

3 KEY CHALLENGES TOWARDS MODELING AND GENERATING INFRASTRUCTURE-AS-CODE

The PIACERE project aims to increase the productivity of the DevOps teams in the development and operation of IaC through the provisioning of an integrated DevSecOps framework. DevOps teams can program IaC as if they were programming any software application. [8]

In view of the challenges indicated in Section 2, the PIACERE project has been an insightful experience for us in dealing with the modeling and generating of IaC. While we have managed to cope with most of the challenges described in the Introduction and discussed in this section, we foresee some more of those to tackle towards the end of the project.

In the following subsections, we present the PIACERE DevOps Modeling Language (DOML) deployment approach, as well as the approach to generating IaC.

3.1 The PIACERE DOML approach

Modeling IaC means supplying IT environments with machine-readable definition files and deploying them in a matter of minutes. During the design time, in the PIACERE DOML approach, we model the different elements of the application (as data, and infrastructure) by making use of abstractions.

DOML is the end-user language enabling the modeling of deployment and configuration of complex infrastructural software in a way that can then be transformed by the Infrastructure Code Generator (ICG) in executable IaC. Such a language allows DevSecOps teams to select and combine abstractions with the purpose of creating a proper infrastructure provisioning, configuration, deployment, and self-healing model [8], [5].

Generalization of IaC approaches: Modern IaC languages (such as Terraform, TOSCA, Ansible, Chef, Puppet, etc.) facilitate the provisioning, deployment, and configuration process of cloud applications. However, building IaC models is not trivial work since it requires in-depth knowledge about both the language itself and the characteristics of the operating environment. The PIACERE DOML [8] aims to generalize different IaC languages and provide a user-friendly language model to manage the development and operation process [9].

During the design time, we model the different elements of the application, like data of the application and infrastructure, by making use of abstractions. The problem that we are addressing is to improve the ability of (non-)expert DevSecOps teams to model provisioning, deployment, and configuration needs in complex contexts by providing a set of abstractions of execution environments and composing them into machine-readable representations [8].

For example, Terraform is using a declarative approach while Ansible follows a hybrid one. In comparison to those two languages, DOML models the application in a multiple-layer approach, e.g., application layer, abstract, and concrete infrastructure layers. The advantage of this approach is to separate the efforts of developing the application and maintaining a clear model structure. In this way, different developers can be dedicated to different layers, and different layers of code can also be reused.

Despite this, there are still a couple of challenges to be tackled with furtheron. DOML adopts an abstract modeling approach and tries to generalize the common concepts from different cloud applications and infrastructures. While the underlying different techniques and architectures make it difficult to abstract the common concepts, for instance, the concepts related to different cloud providers vary a lot from each other.

The current compromised approach adopted in DOML is to incorporate those special elements as properties. However, since they are quite general concepts and have no standard, the information cannot be easily passed to downstream tools, e.g., ICG, unless some particular agreements are reached separately between DOML and ICG.

Integrating existing codes and handling existing resources: In some cases, to successfully provision infrastructure resources, deploy and configure applications, DOML still needs the support of existing IaC languages, e.g., Terraform, Ansible, etc. However, there is no unified and clean way to include the external codes due to the different information required to be passed. Meanwhile, it will introduce new problems in the execution order as it is discussed in section 2.2.

Other than that, DOML should have a mechanism to handle the existing resources. Different from the standard IaC languages which have defined their specific way to declare the existing sources in the model, DOML starts from a more abstract viewpoint and tries to generalize the way to integrate the existing resources of the different providers nonetheless, facing certain difficulties. Particularly, there should be some spot to store the status of the existing resources, and some way to query or keep track of the information.

3.2 Generating Infrastructure-as-Code: The PIACERE ICG approach

Generating IaC means producing machine-readable definition files from a user intent expressed in a more abstract language (DOML in the PIACERE project), to achieve this the PIACERE ICG uses a template-based approach for code generation.

Despite the advantages of improved testability, reliability, repeatability, versioning control, provisioning, and configuring on demand, as well as proactive recovery from failures [6], [10], [11] IaC faces also some challenges, such as the ones acknowledged in our ongoing project, as follows [1]:

IaC template configuration: Adopting declarative and imperative approaches and tools like Terraform, Ansible, and AWS Cloudformation, etc., the transformation of complex and interrelated objects and their dependencies into code is a cumbersome task. Deploying IaC requires time and coordination with others in the team, especially those responsible for security and compliance. In the process of IaC adoption, it is necessary to figure out where and how the resources are delivered and managed [1], [12].

In the PIACERE, one of the objectives is to allow the end user to seamlessly transition from one environment to another and be capable of deploying an equivalent infrastructure. To allow for this capability, each resource is categorized and standardized, and code is produced for each resource on each provider.

This standardized code though is subject to obsolescence so, it needs to be upgraded when new functionalities are available, also vulnerabilities may appear in time if not updated.

Security of the generated IaC code: It is not always possible to rely only on security measures in the IaC environment. Moreover, it may take many cycles to check the supplied resources and ensure they are operating properly with IaC while using conventional security tools. In fact, IaC is more dynamic than the provisioning practices, as such it has the potential either to be utilized optimally or misused easily [1].

To handle this complexity, the PIACERE framework was enriched with multiple tools (see Figures 1-2) dedicated to validating the code, starting from the validation of the initial model up to the scan for potential vulnerabilities, and the monitoring of the generated infrastructure.

The main tools covering this functionality in the PIACERE framework are as follows:

- The Model Checker which first validates the integrity of the initial design also checks for any possible circular dependencies.
- The IaC Scan Runner is tasked with the vulnerability assessment of the images and libraries listed in IaC files.
- The Monitoring agents actively monitor the generated infrastructure and can take also self-healing actions when necessary.

Obscuration of the sensitive data is another relevant challenge that we faced. This is handled in the PIACERE by having the data stored in a secure Vault [13]. During the execution, the data are instantiated and referenced in the IaC files as environment variables. Once the execution is finished, the data are cleaned up leaving no trace of them. We will extend this as the project proceeds and experience is gathered.

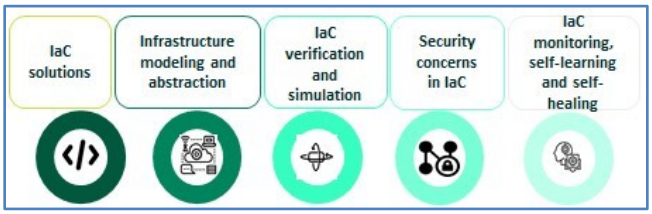

Figure 1. The PIACERE Innovations [8]

Order of Execution: The execution of the IaC files created needs to take care of the order of their execution. Actually, the infrastructure resources probably have to be created before the configuration and deployment of the applications. Moreover, the applications themselves sometimes have to follow some specific order for their release. For example, the database should be installed before the application is going to connect to it.

This challenge is typically more related to the use of tools that apply an imperative approach such as Ansible, while tools that use a declarative approach are more resilient.

The PIACERE ICG [14] component currently solves this challenge following the assumption that the infrastructure resources are deployed before the applications and create configuration files describing the order of execution. The ICG is a code generator, built in the context of PIACERE, which will transform the models into infrastructural code from different IaC languages [6], [15].

Indeed, the ICG IaC files are organized in folders, i.e. there is one dedicated folder for each application and another one for the infrastructure resources. The root folder contains a configuration file that describes the order of execution of all folders.

As for the definition of the infrastructure resources, tools that use a declarative approach are chosen in the PIACERE, this takes away the complexity of having to define the correct order of deployment of the infrastructural elements.

This approach solves just part of the problem but does not consider the order of execution of the applications themselves. The solution implemented in the PACERE to define the proper order of execution of the applications is to do a thorough analysis of all the possible dependencies between the applications during the modeling process and have them properly defined. To this end, in DOML there are multiple terms and structures dedicated to handling dependencies, given that DOML is generated and new and better solutions to handle dependencies are implemented with every new DOML version.

Selection of the target language for a specific task: There is a wide selection of IaC tools that focus on a single or small set of automation steps. For example, one can set up the infrastructure using Terraform or Chef, and configure it using Ansible or Puppet, or either release the framework on a virtual Kubernetes environment.

The current version of the PIACERE ICG supports the two most popular IaC tools: Terraform for automation of the provisioning infrastructure, and Ansible for its configuration. This approach is strictly bound to the definition of the DOML model which defines the "infrastructure" and the "application" layers. The ICG parses the DOML model and chooses the IaC tool depending on which layers the resources belong to. This is the first implementation of the ICG, and further releases will support other IaC languages such as Docker and Kubernetes.

Another approach that can be adopted is to leave the user to choose the target language to be used. This approach requires a higher level of knowledge of the IaC languages by the user, i.e. it is up to the user to choose which one fits better for the specific task. On the other hand, this procedure gives much more flexibility to the user.

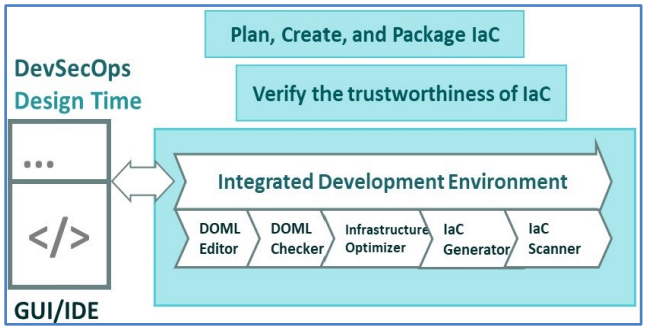

Figure 2. The Design Time of DevSecOps [11]

Integrating existing resources and existing target code: The deployment and configuration of new infrastructure can leverage existing resources. In the PIACERE, it comes out that the provisioning of a use case relies on some existing private resources. In that example, the VMs are linked to a network and the storage is provided by another team in the company. Thus, the user defines in the DOML model the VM to be created and the Network to be used. The IaC file generation must take care of this distinction: the resources to be created have a different IaC code than the existing ones. In the PIACERE, we solve this challenge by introducing in DOML the "preexisting" concept. For the resources labeled as "preexisting", the ICG is going to generate IaC, mostly dedicated to retrieving the related data and connecting these existing resources to the new ones.

Such a challenge raises also for the existing target code: the user may have defined some configuration or IaC files to be executed which are required for the complete application deployment. In the PIACERE, the current version of the ICG takes as input the user's IaC folder and imports it into its code-generated folder. The ICG knows the location of these external files thanks to the DOML model definition and fulfills the information for its execution.

Table 1. Summary of the challenges in the modeling and generating of IaC addressed by the PIACERE solution

The PIACERE key challenges towards IaC	
DOML approach	Generalization of IaC approaches
	Integrating existing codes and handling existing resources
IaC approach	IaC template configuration
	Security of the generated IaC code
	Obscuration of the sensitive data
	Order of execution
	Selection of the target language for a specific task
	Integrating existing resources and existing target code
	Access management
	Standardization
	Reliable IaC implementations

4 CONCLUSION

In the development of the PIACERE solution, we faced a number of challenges and issues (see Table 1) which led to a couple of useful insights for the developers and researchers [3]:

- *Configuring the IaC templates* in a standardized manner allows for easier conversion from one environment to another.
- *The generation of the IaC code* faces risk vulnerabilities at multiple stages. Thus, having appropriate tools to check the consistency of the code at each stage is important.

- *The IaC resources may have dependencies*, due to this fact, a need to handle the order of execution could arise. The definition of proper structures can simplify this issue.
- *There are different IaC tools to automate* the different release steps of the application. To avoid an inexperienced user to study all of them, the PIACERE framework can choose the appropriate IaC tool to use for each step.
- *The IaC generation may need to integrate existing target code or resources.* In that case, the modeling process has to provide adequate functionalities.
- *Developing clean deployment templates.* Using an integrated developer environment (IDE) such as Eclipse to enhance the *development.*
- *Technology.* The technology choice is an important one to make, based on the experience and demand of the development teams [3]. For instance, Terraform can provide advantages with regard to readability and maintainability.

The future challenges that we foresee to deal with by the end of our ongoing project are as follows:

- *Affidability and resilience of the code*: Improper interactions of the user with the IaC code or not documented changes to the infrastructure may cause unwanted results during the IaC deployment. Having appropriate checks inside the IaC code can help to avoid these circumstances.
- *Access management*: Allowing the developers to have access only to the functionalities required by their role is important to limit possible misuse of IaC.
- *Standardization*: IaC can assure reducing maintenance efforts, increased stability, and security by using standardization [1].
- *Reliable IaC implementations:* IaC requires consistent implementation [3].

This paper contributes to the context of the challenges in our DevOps practice, toward the modeling and generating of IaC. We tried to provide a meaningful perspective of what issues we have tackled during the ongoing project PIACERE. As discussed above, IaC utilization requires awareness of security concerns, and not following the best practices can introduce security risks to the infrastructure. In fact, insecure IaC creates cloud environments that could result in compliance violations and cloud data breaches [4].

Also, the paper aims to foster discussion and collaboration among practitioners and researchers from the computing continuum, from Cloud to IoT. Despite the discussed difficulties, the consistent use of IaC and other emerging DevOps approaches is gaining more popularity in the practice [6].

ACKNOWLEDGMENTS

This work is partially funded by the EU Commission in the Horizon 2020 research and innovation programme under grant agreement No. 101000162.

REFERENCES

[1] "Infrastructure as Code: Challenges and How to Deal With Them," [Online]. Available: https://www.iotforall.com/infrastructure-as-code-challenges.

[2] M. Langford, "Top 5 Infrastructure as Code Security Challenges," [Online]. Available: https://www.trendmicro.com/en_us/devops/22/g/infrastructure-as-code-iac-security.html.

[3] Pingen, Rene, "A reflection on the perceived benefits of Infrastructure as Code," Compact, KPMG Advisory, 2021.

[4] "3 Tips for Success with Infrastructure as Code (IaC)," [Online]. Available: https://www.techadv.com/blog/3-tips-success-infrastructure-code-iac.

[5] D. Linthicum, "Will the private cloud disrupt your 'infrastructure as code' practice?," [Online]. [Accessed 22 January 2023].

[6] S. Clauirton, "From theory to practice: the challenges of a DevOps infrastructure as code implementation," Porto: Portugal, July 2018.

[7] Artac, M., Borovssak, T., Di Nitto, E., Guerriero, M., & Tamburri, D. A., "DevOps: introducing infrastructure-as-code," in *IEEE/ACM 39th International Conf. on Software Engineering Companion (ICSE-C)*, 2017.

[8] PIACERE project website:, [Online]. Available: https://piacere-project.eu/. [Accessed 2023].

[9] Bin Xiang, Elisabetta Di Nitto, Galia N. Nedeltcheva, "Deliverable D3.1 PIACERE Abstractions, DOML, and DOML-E – v1," PIACERE Project, 2022.

[10] C. Rong, "OpenIaC: open infrastructure as code-the network is my computer."," *Journal of Cloud Computing*, vol. 11.1 (2022), pp. 1-13.

[11] Juncal Alonso, et al., "Embracing IaC through the DevSecOps philosophy: Concepts, challenges, and a reference framework," *IEEE Software*, vol. 1, pp. 56-62, 2022.

[12] Juncal Alonso, et al., "PIACERE: Programming Trustworthy Infrastructure-As-Code in a Secure Framework," in *CEUR-WS Workshop Proceedings*, 2021.

[13] Microsoft, "Use Azure Key Vault to pass secure parameter value during deployment," 17 December 2020. [Online]. Available: https://docs.microsoft.com/en-us/azure/azure-resource-manager/templates/key-vault-parameter?tabs=azure-cli.

[14] Lorenzo Blasi (HPE), "Deliverable D3.5 Infrastructural code generation - v2," PIACERE Project, 2022.

[15] Osaba Eneko, et al., "PIACERE project: description and prototype for optimizing infrastructure as code deployment configurations," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2022.