Programming trustworthy Infrastructure As Code in a Secure Framework

Juncal Alonso[1], Christophe Joubert[2], Leire Orue-Echevarria[1], Matteo Pradella[3,4], Daniel Vladušič[5]

[1] TECNALIA, Basque Research and Technology Alliance (BRTA), Parque Científico y Tecnológico de Bizkaia, Astondo bidea,700, E-48160 Derio, Spain
[2] PRODEVELOP S.L., Paseo Ciutadella, 13 - entresuelos 2,3 y 4, 46003 Valencia, Spain
[3] Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Piazza Leonardo Da Vinci 32, 20133 Milano, Italy
[4] IEIIT, Consiglio Nazionale delle Ricerche, via Ponzio 34/5, 20133 Milano, Italy
[5] XLAB d.o.o., Pot za Brdom 100, 1000 Ljubljana, Slovenia
juncal.alonso@tecnalia.com, cjoubert@prodevelop.es, leire.orue-echevarria@tecnalia.com, matteo.pradella@polimi.it, daniel.vladusic@xlab.si

Abstract. Infrastructure-as-Code (IaC), enables the automation of several deployment, configuration and management tasks. IaC has a lot of potential in cloud computing as it results in a significant saving of time when an application needs to be redeployed on a different set of resources, even running on different infrastructures. Unfortunately, IaC still suffers from some important issues, such as the large variety of competing tools or the strong orientation toward the cloud, leaving aside e.g. the edge. Also, trustworthiness and security aspects of are often left for the end of the cycle, where errors and vulnerabilities are often too late or too expensive to correct. We present here the PIACERE project, which provides tools, methods and techniques for the Infrastructure-as-Code approach. The project will make the creation of IaC more accessible to designers, developers and operators, increasing the quality, security, trustworthiness and evolvability of infrastructural code while ensuring its business continuity by providing self-healing mechanisms anticipation of failures and violations.

Keywords: Secure software engineering, DevOps, Infrastructure as Code, DevOps Modelling Language, self-learning and self-healing mechanisms, optimization algorithms, security analysis

1 Introduction

The virtualization revolution that has taken place in the last years along with the advent of the cloud computing continuum (combination of cloud and edge), has allowed for an increase in the use of software to build, control and configure entire virtual data centers and the entire infrastructure layer. The use of such tools and APIs has only stressed the importance of software in the infrastructure arena. Moreover, the significance of software has recently grown to the point of merging the role of software developers with that of infrastructure operators, whose main focus is the automation of infrastructure-related activities through the use of software, giving birth to the *Infrastructure as Code* trend, the engine of the DevSecOps [1, 2, 4] movement, which is an organizational change that consists of using software engineering tactics that reduce the technical and organizational distance between development and operation, leading to the creation of a single, well-coordinated team of people.

Infrastructure-as-Code (IaC) [3], enables the automation of several deployment, configuration and management tasks that otherwise would have to be performed manually. IaC has a lot of potential in a cloud computing context as it results in a significant saving of time when an application needs to be redeployed on a different set of resources or needs to be extended with new components, even possibly running on different cloud infrastructures. In these cases, the infrastructural code can be reused, adapted, if needed, and then run for recreating very quickly the new or extended software instance. As such, IaC has represented a very important progress that has dramatically changed the work organization of many IT-intensive organizations (e.g. Netflix[6]).

Unfortunately, IaC still suffers from these five main issues:

1. a large variety of competing tools requiring the adoption of different programming languages for writing infrastructural code,
2. the fact that all these tools and languages are focusing on a single or a small set of automation steps and of types of resources (e.g. VMs),
3. they mostly focus on cloud computing leaving aside other computational resources such as the edge;
4. they focus on certain phases of the lifecycle of the IaC such as provisioning, configuration or deployment but there is not really an end-to-end solution covering the Devs and the Ops;
5. trustworthiness and security aspects of the IaC are often left for the end of the cycle, for once the code is already in operation, when it is already too late, the errors and vulnerabilities are expensive to correct and can affect the business continuity of the application.

As an example of the problems just mentioned, software configuration can be performed by writing Chef recipes or Puppet code, just to name two well-known languages, while deployment orchestration can be defined by relying on TOSCA, Terraform, Brooklyn or many other less known approaches. Moreover, Software-Defined Networking (SDN) exploits specific protocols, such as OpenFlow or NETCONF, each with its own data modeling language, storage mechanisms, and based on a large number of proprietary protocols and configuration languages. The same occurs if the monitoring stack, the load balancer or the auto-scaling mechanism are to be configured using proprietary languages and associated tools. Such a diversity implies that adopting the IaC approach today requires a variety of specialized skills and DevSecOps environments that, in many cases, companies do not have, and struggle to find.

The main objective of the PIACERE project is thus to provide means (tools, methods and techniques) to enable most organizations to fully embrace the Infrastructure-as-Code approach, through the DevSecOps philosophy, by making the creation of such infrastructural code more accessible to designers, developers and operators (DevSecOps

[6] https://medium.com/netflix-techblog/how-we-build-code-at-netflix-c5d9bd727f15

teams), increasing the quality, security, trustworthiness and evolvability of infrastructural code while ensuring its business continuity by providing self-healing mechanisms anticipation of failures and violations, allowing it to self-learn from the conditions that triggered such re-adaptations.

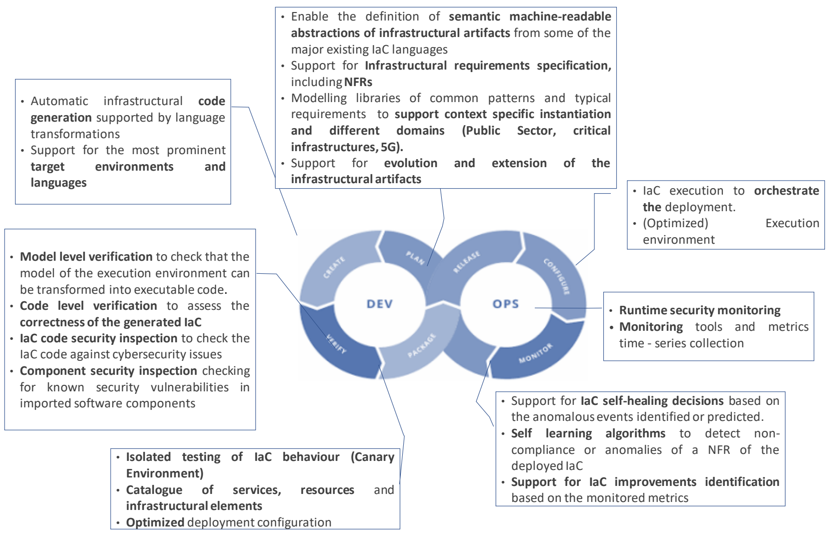
• Enable the definition of **semantic machine-readable abstractions of infrastructural artifacts** from some of the major existing IaC languages
• Support for **Infrastructural requirements specification,** including **NFRs**
• Modelling libraries of common patterns and typical requirements to **support context specific instantiation and different domains (Public Sector, critical infrastructures, 5G).**
• Support for **evolution and extension of the infrastructural artifacts**
• Automatic infrastructural **code generation** supported by language transformations
• Support for the most prominent **target environments and languages**
• IaC execution to **orchestrate the** deployment.
• (Optimized) Execution environment
• **Model level verification** to check that the model of the execution environment can be transformed into executable code.
• **Code level verification** to assess the **correctness of the generated IaC**
• **IaC code security inspection** to check the IaC code against cybersecurity issues
• **Component security inspection** checking for known security vulnerabilities in imported software components
CREATE
PLAN
RELEASE
CONFIGURE
DEV
OPS
VERIFY
PACKAGE
MONITOR
• **Runtime security monitoring**
• **Monitoring** tools and metrics time - series collection
• Support for **IaC self-healing decisions** based on the anomalous events identified or predicted.
• **Self learning algorithms** to detect non-compliance or anomalies of a NFR of the deployed IaC
• **Support for IaC improvements identification** based on the monitored metrics
• **Isolated testing of IaC behaviour (Canary Environment)**
• **Catalogue of services, resources** and **infrastructural elements**
• **Optimized** deployment configuration

Fig. 1. Relationship between PIACERE and the DevOps cycle.

To achieve the IaC DevSecOps concept, PIACERE will provide an integrated DevSecOps framework to develop, verify, release, configure, provision, and monitor infrastructure as code. The extensible architecture and modular approach of PIACERE will support the different DevSecOps activities. Using a single integrated environment (IDE) to develop infrastructural code will unify the automation of the main DevSecOps activities and will shorten the learning curve for new DevSecOps teams. PIACERE will allow DevSecOps teams to model different infrastructure environments, by means of abstractions, through a novel DevOps Modeling Language (DOML), thus hiding the specificities and technicalities of current solutions. Moreover, PIACERE will provide an extensible Infrastructural Code Generator (ICG), translating DOML into source files for different existing IaC tools, to reduce the time needed for creating infrastructural code for complex applications. The provided extensibility mechanisms (DOML-E) shall ensure the sustainability and longevity of the PIACERE approach and tool-suite (new languages and protocols that can appear in the near future).

2 The PIACERE Approach

The PIACERE workflow (Figure 2) starts from the specification of qualitative requirements such as infrastructure, security, networking and software characteristics. These

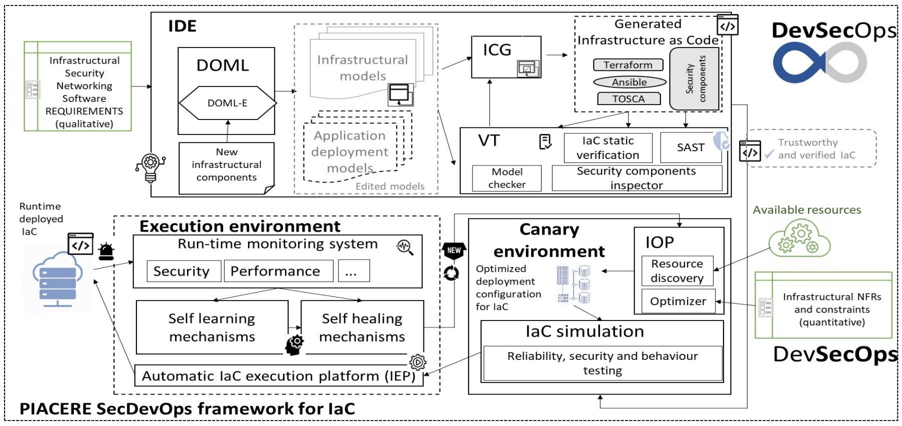
IDE
DOML
DOML-E
New infrastructural components
Infrastructural models
Application deployment models
Edited models
ICG
Generated Infrastructure as Code
Terraform
Ansible
TOSCA
Security components
VT
IaC static verification
SAST
Model checker
Security components inspector
Infrastructural Security Networking Software REQUIREMENTS (qualitative)
DevSecOps
Trustworthy and verified IaC
Available resources
Infrastructural NFRs and constraints (quantitative)
DevSecOps
Runtime deployed IaC
Execution environment
Run-time monitoring system
Security
Performance
...
Self learning mechanisms
Self healing mechanisms
Automatic IaC execution platform (IEP)
NEW
Canary environment
Optimized deployment configuration for IaC
IOP
Resource discovery
Optimizer
IaC simulation
Reliability, security and behaviour testing
PIACERE SecDevOps framework for IaC

Fig. 2. The PIACERE workflow.

trigger the development of both the application and infrastructural software. Among the others, they identify those non-functional requirements for application software that become functional ones (need for provisioning certain services, for configuring certain network functions and the like, security requirements, etc.) for the infrastructural code. Based on these requirements, the DevSecOps team will focus on the development of the infrastructural code. Through DOML, the DevSecOps Modeling Language, the topology and the properties of the infrastructure to be created such as computation, memory, networking, software and services resources, security requirements and rules, will be defined. This will allow the DevSecOps team to model an infrastructure with its configuration abstracting the peculiarities of the provisioning and configuration of complex execution environments, without being limited by the lack of knowledge on the large variety of needed IaC languages and protocols.

After the modeling stage, the DevSecOps team will check if the resulting model can be turned into executable code (verification at model level), using the Verification Tool (VT), a composition of tools that works both at the level of models and infrastructural code.

Once the infrastructure, network, and execution environment models are ready and verified, the DevSecOps team will generate the infrastructural code from them through the PIACERE Infrastructure Code Generator (ICG). The currently most prominent target IaC environments and languages (e.g. Terraform, Ansible, TOSCA) will be supported by ICG, in terms of provisioning and deployment orchestrators, configuration management environments, monitoring platforms, and network APIs. The adequacy of the generated infrastructural code will be checked, ranging from syntactic correctness to consistency and ability to fulfill specific non-functional properties such as performance of applications after their deployment. Special focus on security will be included. The VT will contain an IaC code Security Inspector that will check the IaC code against the known cybersecurity issues offering a Static Analysis Security Testing. Additionally, the VT will support the testing of the specific security libraries and middleware to be used within the framework to build the security inclusion policy. This whitebox fuzzing testing will be executed by the Security Components Inspector. All these components (from DOML to VT) supporting the IaC DevSec process will be integrated in an IDE which will offer the IaC developers an integrated framework to support the design, development, generation and verification of the infrastructural code.

Once the code is verified, PIACERE will support the simulation of the conditions of the production environment through the Canary Sandbox Environment, enabling isolated execution and testing of the IaC behavior (reliability and behavior testing and cybersecurity threats) while identifying potential vulnerabilities and bottlenecks before the code is deployed. As part of this pre-deployment phase, PIACERE will offer the IaC optimized execution platform (IOP), which will include both a catalog describing the available services (e.g. IaaS, XaaS) and infrastructural elements (e.g. computation, networks) to deploy the IaC, and an Optimizer that will provide the best combination of those services with such resources, based on a set of constraints (e.g., types of infrastructural elements, NFRs, and so on).

The following step is the execution at runtime of the created IaC through the IaC execution platform (IEP). The main task of the execution platform is to create a deployment plan by splitting the work into separate tasks, ordering them according to their interdependencies, and distributing them to the specialized subsystems that perform the actual provisioning (e.g. creating virtual machines using proper IaaS connector, installing software packages or adjusting application configuration using Ansible). Tracking dependencies also encompasses information transfer between interdependent tasks, which is vital for producing functioning end-results. The PIACERE Monitoring platform will log the whole IaC execution run, making metadata and metrics about the creation of resources available to the rest of the PIACERE components. This will enable insight into the effects of the specific IaC code, enabling feedback not only on the application's own runtime, but also on its provisioning, deployment. In such a complex system, PIACERE will also verify any security violation at runtime through Runtime security monitoring. This data (security-, performance-, uptime-related) will feed the PIACERE Self-learning and self-healing mechanisms to ensure that the conditions of the QoS are met at all times and that a failure or non-compliance of NFRs is not likely to occur. If any of these occurs an alert to the DevSecOps teams will be sent and the most appropriate IaC deployment configuration will be sought and selected by the optimizer again. PIACERE will support the automatic re-deployment of the selected IaC to ensure that their infrastructural code is always conforming to the SLAs committed with the end-user even if the environmental situation changes.

3 Conclusion

The main aim we envision for the PIACERE project is to enable organizations to embrace the Infrastructure-as-Code (IaC) approach, through the DevSecOps philosophy, by making the creation of such code more accessible to designers, developers and operators (DevSecOps teams), increasing the quality, security, trustworthiness and evolvability of infrastructural code while ensuring its business continuity by providing self-healing mechanisms anticipating to failures and violations, and self-learning from the conditions that triggered such re-adaptations.

The envisioned PIACERE benefits will be assessed and demonstrated on its Use Cases (5G, Public Administrations, Critical Infrastructures), with the focus on the utility and suitability of the PIACERE approach and toolset in real cases from relevant application domains and industrial sectors (telco, IoT, services).

Acknowledgments This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101000162.

References

1. Bass, L.J., Weber, I.M., Zhu, L.: DevOps - A Software Architect's Perspective. SEI series in software engineering, Addison-Wesley (2015)
2. Mohan, V., Othmane, L.B.: Secdevops: Is it a marketing buzzword? - mapping research on security in devops. In: 2016 11th International Conference on Availability, Reliability and Security (ARES). pp. 542–547 (2016)
3. Morris, K.: Infrastructure as Code: Managing Servers in the Cloud. O'Reilly Media, Inc., 1st edn. (2016)
4. Myrbakken, H., Palacios, R.C.: DevSecOps: A multivocal literature review. In: Mas, A., Mesquida, A.L., O'Connor, R.V., Rout, T., Dorling, A. (eds.) Software Process Improvement and Capability Determination - 17th International Conference, SPICE 2017, Palma de Mallorca, Spain, October 4-5, 2017, Proceedings. Communications in Computer and Information Science, vol. 770, pp. 17–29. Springer (2017), https://doi.org/10.1007/978-3-319-67383-7_2