A Model-Driven Approach to Continuous Delivery of Cloud Resources

Julio Sandobalin[1,2(✉)]

[1] Escuela Politécnica Nacional, Ladrón de Guevara, E11-253, P.O. Box 17-01-2759, Quito, Ecuador
julio.sandobalin@epn.edu.ec
[2] Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain
jsandobalin@dsic.upv.es

Abstract. DevOps is a paradigm which brings practices and tools that optimize the software delivery time. Cloud-based DevOps processes facilitate the continuous delivery of infrastructure and software applications (i.e. cloud resources). In particular, Infrastructure as Code is the cornerstone of DevOps for automating the infrastructure provisioning based on practices from software development. There exist several Configuration Management Tools (CMTs) that use script languages to define the infrastructure provisioning to be deployed in a particular cloud provider. However, manual setting of the script languages to establish the infrastructure provisioning in a CMT for a particular cloud provider is a time-consuming and error-prone activity. For these reasons, the aim of my PhD research is proposing a model-driven approach to abstract and automate a continuous delivery process of cloud resources through model-driven techniques and DevOps. In addition, this approach seeks to cover the development process of cloud resources in development, testing and production environments.

Keywords: Cloud computing · DevOps · Continuous delivery
Infrastructure as code · Cloud resources · Model-Driven development

1 Introduction and Problem Statement

To succeed in a world where technologies, requirements, ideas, tools and timelines are constantly changing, information must be accurate, readily available, easily found and, ideally, constantly delivery in real-time to all members [1]. To automate the development software process practitioners are using a new paradigm called DevOps [2] (Development & Operations) which is promoting continuous collaboration between developers and operations staff through a set of principles, practices and tools that optimize the delivery software time. In particular, the cornerstone of DevOps is the Infrastructure as Code [3] which is an approach for automating the infrastructure provisioning based on practices from software development. There exist several cloud-based DevOps processes which leverage capacities offered by Cloud Computing and use the Infrastructure as Code for automating the infrastructure provisioning. Moreover, cloud-based DevOps processes facilitate the continuous delivery of

L. Braubach et al. (Eds.): ICSOC Workshops 2017, LNCS 10797, pp. 346–351, 2018.
https://doi.org/10.1007/978-3-319-91764-1_29

infrastructure and software applications (i.e. cloud resources). Configuration Management Tools (CMTs) such as Ansible[1], Puppet[2] or Chef[3] have achieved automate the infrastructure provisioning in the Cloud. Each CMT has its script language to define the infrastructure provisioning to be deployed in a particular cloud provider. However, manual setting of the script languages to establish the infrastructure provisioning in a CMT for a particular cloud provider is a time-consuming and error-prone activity. Although CMTs have a high level of automation in the infrastructure provisioning, it remains a challenge to automate a model-based development process for continuous delivery of cloud resources.

This research is following the guidelines of the Design Science Methodology [4] (DSM) which is oriented to information system and software engineering. The goal of the DSM is obtaining an artifact in a problem context. Therefore, the artifact is:

A model-driven approach to continuous delivery of cloud resources.

The problem context is *cloud-based DevOps* and the stakeholders are *developers and operations staff*. To this end, I aim at answering the following research questions:

RQ1. Which DevOps-based approaches exist for continuous delivery of cloud resources?
RQ2. How can model-driven techniques support the automation of cloud infrastructure provisioning? ***RQ2.1.*** How to abstract the complexity to model the infrastructure of different cloud providers? ***RQ2.2.*** How to abstract the complexity to manage different script languages of the Configuration Management Tools?
RQ3. How to achieve a continuous delivery process of cloud resources based on models? ***RQ3.1***. How to configure a DevOps toolchain for continuous delivery of cloud resources? ***RQ3.2***. How to achieve a cloud resource development process based on models that cover the development, testing, and production environments?

2 Related Work

Currently, there is much interest in cloud-based DevOps research. Research efforts have focused on infrastructure provisioning and applications deployment in the Cloud. In this context, below are described the main research works that aim this approach.

TOSCA [5] is a standard for Topology and Orchestration Specification for Cloud Application which allows modeling nodes (virtual or physical machines) and orchestrates the deployment of Cloud applications. TOSCA uses DevOps provisioning tools such as Chef for infrastructure provisioning and Juju[4] for deployment of cloud-based applications.

MORE [6] is a model-driven approach that focuses on automating of initial deployment and dynamic configuration of a system. MORE defines a topology of

[1] https://www.ansible.com.
[2] https://puppet.com.
[3] https://www.chef.io.
[4] https://jujucharms.com.

infrastructure to specify system structure and transforms this topology into executable code for Puppet tool to get virtual machines, physical machines and containers.

MODAClouds [7] is a European project undertaken to simplify the cloud services usage process. One of its goals is to support system developers in building and deploying applications and related data to multi-clouds spanning across the full cloud stack. MODAClouds includes automated infrastructure provisioning platform using Puppet's modules.

On the other hand, research works that provide guidelines for continuous delivery have focused their efforts on code-based software delivery process. The main approaches in this context are following.

Soni [8] present a research work that focuses on the necessities of the insurance industry. This approach proposes a proof of concept for designing an effective framework for continuous integration, continuous testing, and continuous delivery to automate the source code compilation, code analysis, test execution, packaging, infrastructure provisioning, deployment and notifications using build pipeline concept.

Rathod and Surve [9] propose a framework for automated testing and deployment to help automated code analysis, test selection, test scheduling, environment provisioning, test execution, results from analysis and deployment pipeline.

3 Proposed Solution

Answering the research question ***RQ2***, to support the automation of cloud infrastructure provisioning through model-driven techniques I have developed ARGON [10] (*An infRastructure modelinG tool for clOud provisioNing*) tool. ARGON has two main components: (a) Domain Specific Language (DSL) to model the infrastructure in the Cloud (***RQ2.1***), and (b) Transformation engine which creates scripts to manage different Configuration Management Tools (***RQ2.1***).

ARGON has an abstract syntax which is defined through an *Infrastructure Metamodel* [10] (see Fig. 1a). The metamodel abstract the capacities of the cloud computing instead of focus on infrastructure provisioning tools. Thus, I can distinguish four groups according to the cloud capacities: (1) ***Computing*** capacity allows the creation of *Virtual Machines* with one or more *Security Groups* that perform as a firewall. Each *Security Group* enables a *Virtual Machine* access through ports as *Inbound* rules and *Outbound* rules. *Load Balancer* allows distributing incoming application traffic between multiple *Virtual Machines* and with an input rule or *Listener* checks the connection requests. In addition, I can assign a *Static IP* address to a *Virtual Machine*. (2) ***Storage*** capacity allows the creation of *Databases* and *File* servers. (3) ***Elasticity*** capacity allows the creation of templates or *Launch Configuration* where features of a *Virtual Machine* are specified. Templates are used to configure the creation of groups of *Virtual Machines* by means of *Auto Scaling Group*. Creation or elimination of *Virtual Machines* is done based on *Scaling Policy* which is executed by an *Alarm* that monitor a metric in a period of time. (4) ***Networking*** capacity is represented by associations among metaclasses.

The metamodel only defines the abstract syntax, but not a concrete notation of the graphical language in ARGON. In order to use graphical notation to render the model

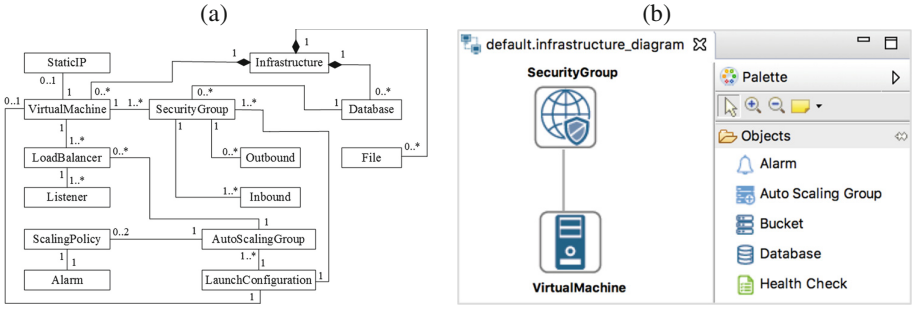

Fig. 1. (a) Infrastructure Metamodel. (b) Infrastructure Model.

elements in the modelling editors, I use a concrete syntax developed by using EuGENia [11]. EuGENia facilitates to generate the models needed to implement a GMF editor in Eclipse Modeling Framework [12]. ARGON uses this DSL to create an *Infrastructure Model* [10] (see Fig. 1b) representing the infrastructure with its provisioning requirements of hardware and software.

The transformation engine creates configurations files or scripts that have the full instructions to create hardware and its settings and install the underlying software. The transformation engine abstracts the features of scripting languages of Configuration Management Tools (CMTs) to create transformation rules which represent modules to build cloud elements. The transformation engine uses an infrastructure model to apply on it the model-to-text transformations and generates scripts for CMTs. It is worth mentioning that both DSL and transformation engine are JARs (Java ARchives) which can be used with the Eclipse packages.

On the other hand, as a first approach to answering the research question ***RQ3***, I have configured an end-to-end automated toolchain for infrastructure provisioning in the Cloud based on DevOps community tools [13] (***RQ3.1***) and ARGON [10]. This approach takes advantage of Infrastructure as Code concept to apply DevOps practices by supporting to a systematized and automated *infrastructure provisioning pipeline* [13] (see Fig. 2) (***RQ3.2***). I use ARGON tool to model an infrastructure model with its provisioning requirements of hardware and software. Subsequently, I will take this infrastructure model and push it toward a *Version Control* system in order to retain and provide access to every version of every infrastructure model that has ever been stored on it. Moreover, this approach allows teams with infrastructure models across different places to work collaboratively. An *Artefact Repository* is used to provide software libraries, such as the model-to-text transformation engine and ARGON's Domain Specific Language. I use an artefact repository in order to provide an only one provider of libraries or software artefact in phases of development, build, and testing.

Every infrastructure model must be checked into a single version control repository to begin the *Continuous Integration* stage. Continuous integration requires that every time developers or operations staff commits a change, the entire application is built and a compressive set of automated tests run against it [2]. The transformation engine is used as a plugin in the continuous integration server to create scripts for provisioning tools. After, a set of the automated test proposed by Morris [3] is run against the scripts.

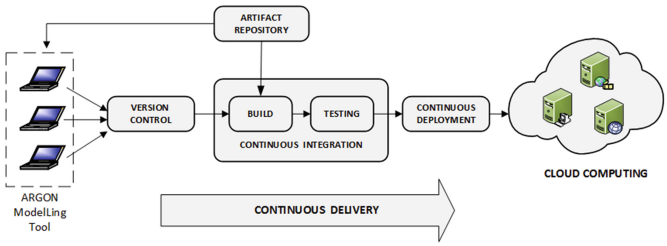

Fig. 2. Overview of the infrastructure provisioning pipeline.

First, *Syntax Check Tests* are executed for the verification of the structure of the scripts. Second, *Static Code Tests* are performed by parsing code or definition files without executing them in the Cloud.

Scripts that have been built and have overcome a set of automated tests are ready to be used in CMTs. *Continuous Deployment* stage takes these scripts built in the previous stage and automatically use them in CMTs to orchestrate the infrastructure provisioning and software deployment in the Cloud. Finally, once the infrastructure provisioning is finished, the *Infrastructure Tests* are executed towards ensuring the correct functioning of the infrastructure and the software deployed in the Cloud.

4 Conclusions and Future Directions

At the end of the second year of my PhD research[5], my work has focused on demonstrating the feasibility of how to support the continuous delivery of cloud resources through model-driven techniques and DevOps. First, ARGON tool provides support to model the cloud infrastructure elements and then generate scripts to manage Configuration Management Tools. Second, an end-to-end automated DevOps toolchain brings support for continuous delivery of infrastructure in the Cloud based on infrastructure models developed by ARGON tool.

The next step of my PhD research[6] is to design a first experiment to validate the ARGON tool. The propose of the experiment is to obtain the effectiveness, efficiency, perceived ease of use, perceived usefulness and intention to use the ARGON tool from the point of view of software engineering, in the context of undergraduate and post-graduate students in Computer Science. On the other hand, ARGON tool provides support for Amazon Web Services. Thus, I am going to extend the ARGON tool to support infrastructure modeling for different cloud providers such as Microsoft Azure and Google Computing Engine. Additionally, I am going to extend the DevOps toolchain to support the continuous delivery process for different cloud providers. Finally, I am going to generalize the findings for proposing a model-driven method for continuous delivery of cloud resources.

[5] I started my PhD research in September 2015.

[6] The next academic year 2017/2018 is the third year of my PhD in which I have to finish my research.

Acknowledgments. This research is supported by the Value@Cloud project (TIN2013-46300-R).

References

1. Cois, C.A., Yankel, J., Connell, A.: Modern DevOps: optimizing software development through effective system interactions. In: IEEE International Professional Communication Conference (IPCC) (2015)
2. Humble, J., Farley, D.: Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, 1st edn. Addison-Wesley Professional, Upper Saddle River (2010)
3. Morris, K.: Infrastructure As Code: Managing Servers in the Cloud, 1st edn. O'Reilly Media Inc, Sebastopol (2016)
4. Wieringa, R.: Design Science Methodology for Information Systems and Software Engineering. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43839-8
5. Wettinger, J., Breitenbücher, U., Kopp, O., Leymann, F.: Streamlining DevOps automation for Cloud applications using TOSCA as standardized metamodel. Future Gener. Comput. Syst. **56**, 317–332 (2015)
6. Chen, W., et al.: MORE: a model-driven operation service for cloud-based IT systems. In: Proceedings - 2016 IEEE International Conference on Services Computing, SCC 2016, pp. 633–640 (2016)
7. Di Nitto, E., Matthews, P., Petcu, D., Solberg, A.: Model-Driven Development and Operation of Multi-Cloud Applications. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-46031-4
8. Soni, M.: End to end automation on cloud with build pipeline: the case for DevOps in insurance industry, continuous integration, continuous testing, and continuous delivery. In: Proceedings - 2015 IEEE International Conference on Cloud Computing in Emerging Markets, CCEM 2015, pp. 85–89 (2016)
9. Rathod, N., Surve, A.: Test orchestration a framework for Continuous Integration and Continuous deployment. In: 2015 International Conference on Pervasive Computing: Advance Communication Technology and Application for Society, ICPC 2015 (2015)
10. Sandobalin, J., Insfran, E., Abrahao, S.: An infrastructure modelling tool for cloud provisioning. In: Proceedings - 14th IEEE International Conference on Services Computing, SCC (2017)
11. Kolovos, D.S., García-Domínguez, A., Rose, L.M., Paige, R.F.: Eugenia: towards disciplined and automated development of GMF-based graphical model editors. Softw. Syst. Model. **16**(1), 229–255 (2015)
12. Steinberg, D., Budinsky, F., Merks, E., Paternostro, M.: EMF: Eclipse Modeling Framework. Addison-Wesley Professional, Lebanon (2008)
13. Sandobalin, J., Insfran, E., Abrahao, S.: End-to-End automation in cloud infrastructure provisioning. In: Proceedings - 26th International Conference on Information Systems Development, ISD (2017)