Abstractions of Abstractions: Metadata to Infrastructure-as-Code

Deslauriers, J., Kovacs, J. and Kiss, T.

Abstractions of Abstractions: Metadata to Infrastructure-as-Code

James DesLauriers
Centre for Parallel Computing
University of Westminster
London, UK
j.deslauriers@westminster.ac.uk

Jozsef Kovacs
UoW, London, UK
j.kovacs@westminster.ac.uk
ELKH SZTAKI, Budapest, HU
jozsef.kovacs@sztaki.hu

Tamas Kiss
Centre for Parallel Computing
University of Westminster
London, UK
t.kiss@westminster.ac.uk

***Abstract*—To support cloud newcomers and empower them with the full suite of benefits afforded by the DevOps toolkit, we propose a solution for generating infrastructure-as-code from metadata. Key-value pairs will describe the containers, volumes, configurations and virtual machines that make up a complex microservices architecture. This metadata will be first compiled down to an intermediate template based on the OASIS TOSCA Specification. There, it will be processed by a deployment and execution engine called MiCADO, which will further compile relevant sections of the template into the respective infrastructure-as-code for tools like Kubernetes, Terraform and Ansible. An implementation of the solution is currently being developed for a European project investigating Manufacturing-as-a-Service and Digital Twins, with the hopes of providing an approachable interface for users who are new to the unfamiliar environment of the cloud.**

***Index Terms*—infrastructure-as-code, devops, TOSCA, cloud, orchestration**

I. Overview

As once cloud-hesitant industries move to adopt cloud computing, the technical teams from these eager companies will do all they can to ensure a soft landing among the clouds. Inexperienced teams, however, will face a steep learning curve as they make the initial transition to cloud - there are cloud consoles and portals to navigate, virtual machines to provision and configure, and applications to containerise and orchestrate. With the advent of infrastructure-as-code (IaC), it became possible to do nearly all of this in a programmatic way, with just a few lines of a code-like language. This empowered experts in the related field of DevOps, but at the same time introduced its own steep learning curve for newcomers. First time cloud users not only had to understand the basic concepts behind configuration management and virtualisation, but now had to learn a flurry of new file formats, syntaxes and APIs.

This paper addresses the technical complexities of cloud adoption, especially as they relate to new-to-cloud industries and their ability to benefit from the tools and IaC that are available in the DevOps toolkit. The proposed approach aims to flatten the learning curve of tools such as Terraform[1], Kubernetes[2], Ansible[3] and Docker[4] by generating their respective IaC from metadata. Users provide this metadata as specific key-value pairs that describe a complex microservices architecture made up of containers, volumes, the required cloud resources and any interrelationships in-between. This metadata will be automatically compiled down to an intermediate language, which is based on the OASIS TOSCA (Topology and Orchestration Specification for Cloud Applications) Specification[5] and known as an Application Description Template (ADT). These ADTs are compatible with an automated deployment engine called MiCADO [1], which will perform its own compilation down to valid IaC for components like Terraform, Kubernetes and Ansible.

In recent years, cloud adoption by Manufacturing SMEs has been accelerating at quite a pace. Efforts to provide an interface to the cloud for Manufacturing have been ongoing in several projects that the University of Westminster has participated in, and these have resulted in several implementations of Manufacturing-as-a-Service. The aim behind these projects has been to provide a system that takes advantage of the latest advances in cloud technology, while offering an interface that demands little knowledge of the cloud from its users. The latest Horzion 2020 project in Manufacturing-as-a-Service for Westminster is DIGITbrain [2], which aims to extend the platform developed during an earlier project called CloudiFacturing [3]. DIGITbrain will enable Manufacturing companies with little technical knowledge of the cloud to reap all the benefits that cloud computing has to offer.

Providers in DIGITbrain will describe Manufacturing-specific software using metadata, building out complex algorithms made up of different microservices. These microservices serve as building blocks that can be combined to realise a specific functionality or computation. End-users can then select individual microservices or algorithms and pair them with their own model, data and cloud resources. The project envisages that providers in DIGITbrain will require only a working knowledge of containers and that end-users need not even understand the concept of cloud.

[1]Terraform. https://terraform.io
[2]Kubernetes. https://kubernetes.io
[3]Ansible. https://ansible.com
[4]Docker. https://docker.io
[5]OASIS TOSCA. https://www.oasis-open.org/committees/tosca/

II. MiCADO

MiCADO is the engine that deploys and executes applications in the DIGITbrain platform. MiCADO brings together DevOps tools such as Ansible, Kubernetes and Terraform to support the deployment, auto-scaling and runtime management of complex microservices architectures. MiCADO was initially developed in the Horizon 2020 Project COLA [4] and has since featured in several other European projects where it severed a variety of use cases as a deployment and execution engine.

MiCADO features a container orchestrator for deploying containers across a cluster of nodes. In early versions, this was Docker Swarm, but MiCADO later adopted Kubernetes for this purpose. Two cloud orchestrators are supported in MiCADO for provisioning the worker nodes for the cluster - Terraform and Occopus [5]. Both public and private clouds are supported, including OpenStack, Azure, AWS, Google and Oracle. In the latest version, MiCADO uses Kubernetes to orchestrate containers described by Kubernetes manifests and Terraform and Occopus to provision virtual machines described by Terraform plans and Occopus descriptor files, respectively.

MiCADO supports automated horizontal scaling at both container and virtual machine level. An open-source component called PolicyKeeper [6] was developed during COLA to facilitate this two-level scaling. PolicyKeeper enables users to develop custom scaling logic for their containers and virtual machines. It supports querying Prometheus[6] metrics from the cluster with which the user can define scaling rules using a simple Python[7] script. This script is continuously evaluated at runtime, and the user-defined rules trigger scaling actions in the relevant orchestrators.

A last component, the Security Policy Manager, improves various aspects of cluster security in MiCADO. This includes support for hardened infrastructure and application level secrets, secure inter-node communications over an IPSec tunnel and a container image integrity verification tool. A secure WebUI integrating Prometheus, Grafana[8] and the Kubernetes Dashboard supports users with monitoring the cluster and their application.

The interface to MiCADO is the Application Description Template (ADT) [7] - a domain-specific language based on the OASIS TOSCA specification. Users of MiCADO author an ADT, which describes a complex application made up of containers, volumes and configurations, virtual machines to host those containers, and optional policies for scaling, monitoring and security. MiCADO uses the OpenStack TOSCA Parser[9] to parse the ADT and uses a pluggable set of compilation adapters to compile specific sections of the template down to their respective IaC. The section describing the application is compiled down to a set of Kubernetes manifests, which may include Deployments, Services, ConfigMaps or other Kubernetes workloads. The virtual machine descriptions are compiled down to either Terraform plans or Occopus descriptor files, as desired by the ADT author. Policy descriptions are compiled down to a domain-specific language understood by the PolicyKeeper and API calls for the Security Policy Manager.

To realise the deployment of the described application, MiCADO triggers the execution of the generated IaC by invoking the binary or REST service provided by each of the underlying tools. For Kubernetes and Terraform these are the `kubectl` and `terraform` command-line tools, and for Occopus and the custom developed policy components these are REST services. Once deployed, MiCADO manages the application at runtime for the entirety of its lifecycle, enforcing any policies and terminating the application and infrastructure when requested.

III. Generating IaC in DIGITbrain

DIGITbrain aims to empower SMEs in the manufacturing industry with access to digital twins in the cloud. Digital twins - virtual representations of real-life systems - can support the optimisation of all stages of a system's life cycle by collecting real-time data and applying machine learning techniques to make cost-saving predictions. Digital twins in DIGITbrain are implemented as collections of microservices called algorithms. Algorithms are paired with a model and connected to a data source before being deployed to cloud and edge infrastructure to perform the necessary computations for optimisation and prediction. To encourage re-usability, assets in DIGITbrain are independent of one another and act as building blocks. Users can combine these building blocks just before execution time to realise some specific functionality relevant to their manufacturing line.

MiCADO was selected as the execution engine that would deploy these various assets of DIGITbrain. It can provision cloud resources, deploy the microservices of the algorithm as containers, fetch the necessary model and make the connection to the required data stream. The resources, containers and interconnections would all be described in a single MiCADO ADT, which a user of DIGITbrain could submit to MiCADO at any time to deploy the digital twin.

However, this would mean that providers of the various assets of DIGITbrain - microservice, algorithm, model, data and cloud resources - would require some knowledge of MiCADO, which by association required some knowledge of its underlying tools and supported cloud middleware. This idea was incompatible with the DIGITbrain concept, so an alternative solution was investigated, based around describing assets with metadata. For each asset, a set of metadata was proposed that described it in sufficient detail to programmatically generate the portion of IaC that would enable its deployment.

At the microservice and algorithm level, metadata describes the microservices involved and how they should be configured. This configuration data is supported in a variety of common container configuration formats. Docker-Compose, a

[6]Prometheus. https://prometheus.io

[7]Python. https://www.python.org/

[8]Grafana. https://grafana.com

[9]OpenStack TOSCAParser. https://github.com/openstack/tosca-parser

`docker run` command, a Kubernetes manifest or a Helm chart are the various formats that the platform supports as input for a container configuration.

For the data and model assets, this metadata describes the location of the asset (URL) or in the case of a stream, details on how to connect to it (protocol, URI, credentials). Further metadata describes how these assets can be used by specific microservices in the algorithm.

Metadata for cloud resources describes the number and configuration details of the virtual machines and edge devices that are required by the digital twin. The DIGITbrain platform connects to the cloud via the CloudBroker[10] platform, which supports a variety of cloud middleware and enables billing and usage statistics. The metadata for a desired virtual machine consists of the CloudBroker UUIDs that specify cloud middleware, region, instance type, firewall settings and SSH key pairings.

Given metadata for a complete set of DIGITbrain assets, the next step is to generate the IaC that MiCADO can use to facilitate the deployment of the described digital twin. For this task, a new DIGITbrain component called the ADTGenerator was designed and implemented. The ADTGenerator is a REST service that takes as input a complete set of DIGITbrain metadata and returns a valid, ready-to-deploy MiCADO ADT.

Because of the independent nature of assets, the ADTGenerator does not produce a classic, single-file ADT for MiCADO. Instead, each asset is compiled down to a respective descriptor file based on a subset of the ADT. When a user is ready to combine microservices into an algorithm and pair it with data, model and cloud resources, a multi-file ADT will be generated that leverages TOSCA's substitution mappings[11] to link together the individual descriptor files and describe the overall architecture of the application to be deployed by MiCADO. MiCADO then parses this multi-file ADT and deploys the digital twin to the cloud.

IV. Conclusion

Generating complete and valid IaC from metadata is not without its challenges or restrictions. Kubernetes manifests and Terraform plans both overlay incredibly detailed APIs and it would be impossible to capture the full functionality of either via a flat map of key-value pairings. The implementation described above works well because there are certain constraints imposed by the DIGITbrain platform. Neither providers nor users of DIGITbrain are required or expected to have a deep knowledge of Kubernetes, Terraform or any other DevOps tool, so the solution can take on some responsibility for making architectural choices on their behalf. For example, DIGITbrain microservices compile to Kubernetes Deployments, container mounts compile to Kubernetes HostPath volumes, and virtual machines descriptions rely directly on UUIDs instead of using Terraform's Data Sources search.

We envisage supporting more flexibility in the future and expect this solution to grow and develop over the course of the DIGITbrain project to support the diverse experiments that will join the project over its lifetime. We imagine re-using this solution for future projects where the target applications would benefit greatly from cloud deployment, but whose users are not cloud savvy.

[10]CloudBroker. http://cloudbroker.com/
[11]TOSCA Specification v1.3

Acknowledgment

This work was funded by the *DIGITbrain - Digital twins bringing agility and innovation to manufacturing SMEs, by empowering a network of DIHs with an integrated digital platform that enables Manufacturing as a Service*, project, No. 952071, European Commission (EU H2020).

Availability of Data

MiCADO is open-source. You can find all of the various MiCADO components under the `micado-scale` organisation in GitHub, with the deployment specification located at the following URL: https://github.com/micado-scale/ansible-micado. The DIGITbrain work, with examples and an under-development implementation of the compile and generate functions mentioned in this paper can be found in the `UoW-CPC/ADTGenerator` repository, or at the following URL: https://github.com/UoW-CPC/ADTGenerator.

References

[1] T. Kiss, P. Kacsuk, J. Kovács, B. Rakoczi, A. Hajnal, A. Farkas, G. Gesmier, and G. Terstyanszky "MiCADO—Microservice-based Cloud Application-level Dynamic Orchestrator," Futur. Gener. Comput. Syst., Volume 94, May 2019, Pages 937-946

[2] "DIGITbrain European Project", [Online]. Available: https://digitbrain.eu/ [Accessed: 17-December-2021].

[3] T. Kiss, "A Cloud/HPC Platform and Marketplace for Manufacturing SMEs, 11th International Workshop on Science Gateways, IWSG 2019. Ljubljana, Slovenia 12 - 14 Jun 2019.

[4] COLA - Cloud orchestration at the level of application, H2020 EU Project (2020). Available: https://project-cola.eu/ [Accessed: 17-December-2021].

[5] Kovács, J., Kacsuk, P.: Occopus: a multi-cloud orchestrator to deploy and manage complex scientific infrastructures. Journal of Grid Computing 16(1), 19–37 (2018)

[6] Kovács, J., 2019. Supporting programmable autoscaling rules for containers and virtual machines on clouds. Journal of Grid Computing, 17(4), pp.813-829.

[7] Pierantoni, G., Kiss, T., Terstyanszky, G., DesLauriers, J., Gesmier, G., Dang, H.-V.: Describing and processing topology and quality of service parameters of applications in the cloud. Journal of Grid Computing, 1–18 (2020)