METAMODELING ENVIRONMENT IN CLOUD

Marian LACATUSU[1], Anca Daniela IONITA[2]

Nowadays, moving everything to the Cloud is a trend that is still rising, due to many advantages, like the efficient resource management. In the case of modeling tools, moving to Cloud increases the collaborative potential, alongside with the accessibility to the common user that comes along with this approach. The paper presents the deployment of a metamodeling environment in a public Cloud, based on technologies such as Docker and Kubernetes. The solution offers modeling services in Cloud with a private collaboration capability, for creating modeling environments specific to various domains. The paper gives an example for modeling sensor networks, by migrating an on-site modeling environment to a web-based one, hosted in Cloud.

Keywords: Modeling Environments, Cloud Computing, Containers

1. Introduction

Modeling has been an important endeavor in science and engineering, and it has been realized with a large variety of paradigms, in respect with the application domains, like geosciences [1], electrical systems [2] and many others. Model Driven Development (MDD) is a concept that was first talked about in the nineties, with the main goal to evolve the ways of application development, by improving the collaboration within the development team and by making the crossing from writing code manually to generating source code [3]. This approach resulted in a multitude of desktop-based modeling tools that fulfilled the needs of software developers, such as the integration with the most popular programming languages. With the evolution of technology and the movement to the Cloud, the modeling tools have evolved with this trend too. As a result, new approaches and features have arisen with this transition. Many of them are related to the Cloud service models, such as Modeling as a Service [4], which may come on top of Software as a Service and propose the offering of a modeling service in Cloud. Furthermore, Model as a Service [5] sits a layer above Modeling as a Service, as an on-demand modeling service with the purpose of generating and providing models at the request of users. Other features and benefits stand in the collaboration ability that permits a better workflow for development teams. Moreover, the generated/created

[1] PhD Student., Automation and Industrial Informatics Department, University POLITEHNICA of Bucharest, Romania, e-mail: marian.lacatusu@aii.pub.ro

[2] Prof., Automation and Industrial Informatics Department, University POLITEHNICA of Bucharest, Romania, e-mail: anca.ionita@upb.ro

models have to be stored somewhere, so the approach of model repositories is widely spread alongside the initiative of moving a modeling environment to the Cloud.

Moving to the Cloud has important advantages, such as cost savings, disaster recovery and collaboration. This phenomenon is also present in the modeling community, were blockers such as scalability of the models and incompatible changes to the model [6] made the movement to Cloud-based modeling environments very important. As a result, WebGME (Web-based Generic Modeling Environment) [7] was developed, alongside other tools, such as AToMPM (A Tool for Multi-Paradigm Modeling) [8], addressing the previously described blocker (i.e. collaboration).

Our goal was to combine the advantages of WebGME with the benefits given by Cloud Computing. WebGME in Cloud can offer many advantages, because it is a privately managed environment with a private collaboration system and user-managed availability of the modeling environment. Another advantage of this approach is that the system is not dependent of an infrastructure offered by WebGME.org, or of a local infrastructure. As a result, the users of WebGME in Cloud can benefit from the disaster recovery implementation for the Kubernetes cluster, the high availability and the data security and compliance sustained by the selected public Cloud.

The work presented in this paper was based on services such as Docker and Kubernetes, along with their constituent elements. A comparison was also done between WebGME and its desktop version - Generic Modeling Environment (GME). WebGME deployed in the IBM Cloud on a Kubernetes cluster offers as main advantage the collaboration facilities. Thus, modeling projects can be maintained and modified by multiple users, as service availability and data persistence are guaranteed by the Cloud. In addition, two new deployments for Kubernetes were realized, for WebGME and MongoDB. A new image for WebGME was also developed with a Dockerfile specific for the IBM Cloud requirements.

Section 2 describes the background and the related work. Section 3 presents the actual implementation of the modeling environment in Cloud (Section 3.1), followed by an example that consists in a paradigm migration from GME to WebGME in Cloud, via the Paradigm Importer plugin (Section 3.2) and a discussion (Section 3.3).

2. Background and related work

The Generic Modeling Environment offers tool support for creating metamodels and modeling environments for application domains characterized by those metamodels, including model editors and model interpreters. The metamodels

realized with GME include the domain concepts and the relationships between them – expressed through their abstract syntax. Nonetheless, one can also configure the resulted tools, and can change the implicit concrete syntax with customized notations [7].

GME is object-oriented and integrated with various programing languages, such as Java and C++. It also supports add-ons and decorators. It is a desktop tool designed for the creation of small and medium-sized models, limited by the impossibility of multiple users to contribute to the same project.

WebGME addresses the limitations of GME, being a more modern approach that supports collaboration, API and JavaScript integrations, alongside metamodels and domain-specific models that are stored into the cloud. Maroti et al. wrote an in-depth description of the WebGME git-like collaborative features, alongside a very detailed comparation between the two modeling environments – the desktop and the web-based ones [6].

As previously mentioned, this trend to offer web-based modeling tools is a part of a more comprehensive approach, called Model as a Service (MaaS), which lends itself to an implementation through various types of Cloud environments [5]. This also comes with disadvantages like security threats, but it includes various capabilities, from 'running' executable models just like programs and delivering the results of their execution, to giving access to environments that allow users to conceive and represent their own models.

As an example of putting in practice the MaaS approach, AToMPM [8] has the same collaborative features as WebGME and a multi-view mode that permits users to work simultaneously at the same project. Another solution that stands at a starting point of our work is a Cloud deployment of GME on a set of virtual machines, used for educational scope; the environment, presented in [9], is specific for modeling sensor networks, and it provided the source metamodel for the experiment of migration from GME to WebGME presented in this paper.

3. Generic Modeling Environment in Cloud

3.1. Deployment of WebGME in Cloud

It is very important to create a private modeling environment in which all participants to a modeling project can contribute to the project at the same time. WebGME integration with IBM Cloud is appropriate to meet this criterion, using Docker and Kubernetes. A first step of this integration was the creation of a Kubernetes cluster with a single worker, using the IBM Cloud academic license. There is a pod that manages at least a container; for containerization purposes we used Docker, based on a custom WebGME image created with a Dockerfile. Before creating a WebGME docker file, it was necessary to prepare a configuration file

that it will use. This configuration file refers to the WebGME container link to the MongoDB container.

To create the WebGME image automatically, we created a Dockerfile that sequentially runs all the commands that compose it (Fig.1). Using the docker build command, one can create an image from that Dockerfile. The Dockerfile may have only one *CMD* statement, i.e. the condition for which the container is running.

```
FROM node:latest
RUN apt-get update
RUN apt-get install -y git
WORKDIR /usr/app
COPY . .
RUN npm install
RUN npm install webgme
ENV NODE_ENV docker
EXPOSE 8888
RUN cp config.docker.js ./node_modules/webgme/config
WORKDIR /usr/app/node_modules/webgme
CMD ["node", "app.js"]
```

Fig 1. WebGME Dockerfile

WebGME is integrated with Git, so it must be installed alongside the required node-modules. After these procedures, the WebGME is installed, and the WebGME port is exposed. The container will remain operable as long as *script app.js* is running.

After creating the WebGME image, it must be added to the IBM Cloud container registry. This image storage method is private and integrated with the Kubernetes service.

The Kubernetes pods are based on deployments that are instructions for pods and replica sets. The pod is the smallest object that can be deployed in Kubernetes; it encapsulates storage, a unique IP and container barrier options.

The pods can be used in two ways [10]:

- Using a single container - the most common in Kubernetes, who manages the pod,
- Managing multiple containers - a pod encapsulates an application staying on multiple containers that communicate with each other; such a container can form a single service that may be used; the pod encompasses the container resources and uses them as a single entity.

The two deployments from Fig. 2 follow the steps described below [10]:

- A deployment called *webgme-deployment* is created; this is indicated by the *metadata* field.

- The deployment creates a single replica, indicated by the *replicas* field.
- The *selector* field defines how the deployment recognizes the pods to be managed; in our case it is set as the *Webgme* app.
- The *template* field configures the pod to run the *Webgme* container, and we use the image loaded into the IBM Cloud container registry.

The Kubernetes cluster (Fig.3) and all of the deployed components can be monitored from the IBM Cloud Kubernetes Dashboard. Some minor settings, such as the replica set value, or the user rights, can also be managed from here.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webgme-deployment
  labels:
    app: webgme
spec:
  replicas: 1
  selector:
    matchLabels:
      app: webgme
  template:
    metadata:
      labels:
        app: webgme
    spec:
      containers:
      - name: webgme
        image: us.icr.io/marian/webgme/webgme
        ports:
        - containerPort: 8888
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo-deployment
  labels:
    app: webgme
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongo
  template:
    metadata:
      labels:
        app: mongo
    spec:
      containers:
      - name: mongo
        image: mongo:latest
        ports:
        - containerPort: 27017
```

Fig 2. WebGME and Mongo DB Deployments

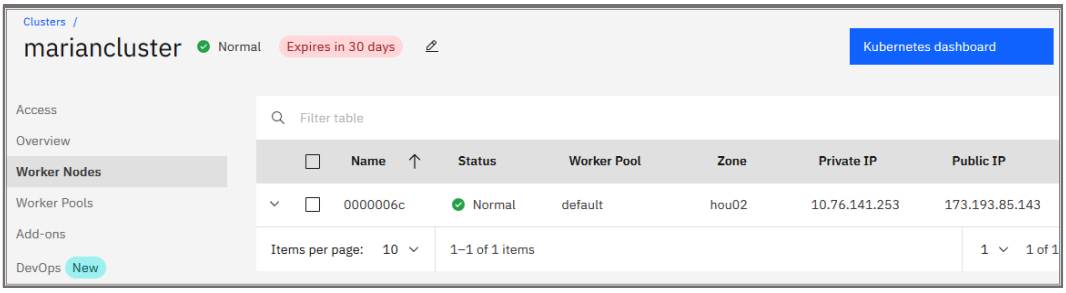

Fig 3. Kubernetes Cluster in IBM Cloud

3.2. On-site Modeling Environment Migrated to the Cloud

The WebGME Cloud implementation was tested for a metamodel that we have previously presented in [9]. Fig. 4 shows a part of the abstract syntax of this

metamodel, conceived for creating a graphical modeling language specific for sensor networks, which have become important to many systems for the data collection and fusion capabilities [11]. The metamodel was represented in GME - the on-site metamodeling environment studied here. A *SensorNetwork* is composed of several *SensorNetworkUnit* elements, which can be units for communication, processing or sensing – the last ones potentially composed of multiple sensors. There may also be connections between these units, as well as dependencies between the software units that are part of the processing units. In addition to the elements represented in Fig. 4, the metamodel contains further details in regard with the communication, memory, power and sensor aspects. Based on this metamodel, we created models for testing purposes, in order to be able to find the information necessary for a side-by-side comparison.

The paradigm created with GME was imported in WebGME deployed in IBM Cloud, using the MetaGME importer plugin, via the .xmp file. Such an import is characterized, however, by several limitations [12], like the impossibility to import roles, constraints, cardinality, aspects and visual properties, as well as the usage of kinds instead of roles. The result of this export for the sensor network metamodel is illustrated in Fig. 5.

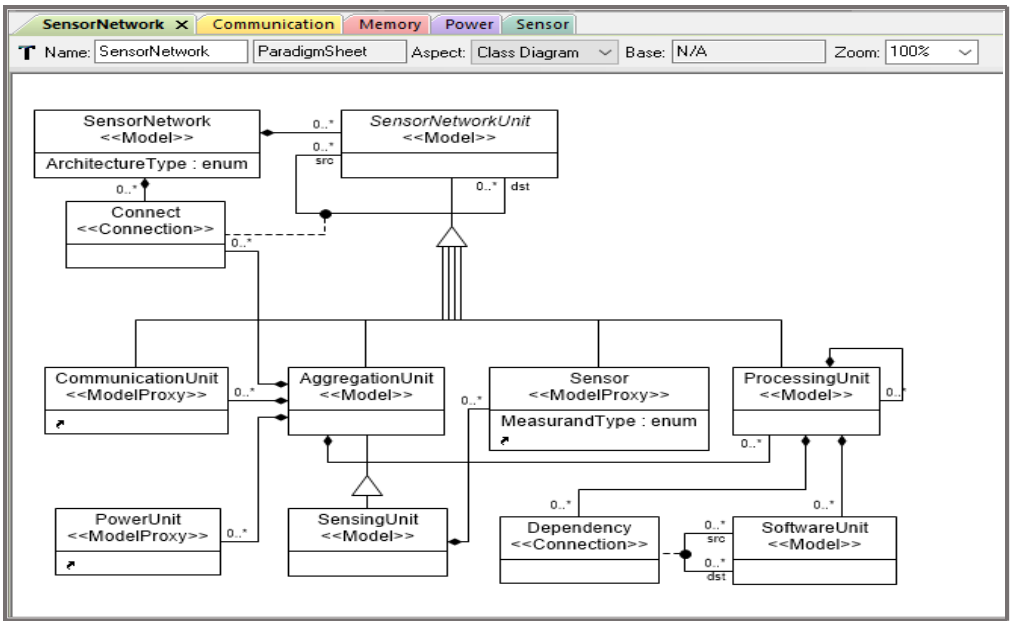

Fig 4. Part of the metamodel from the on-site metamodeling environment (after [9])

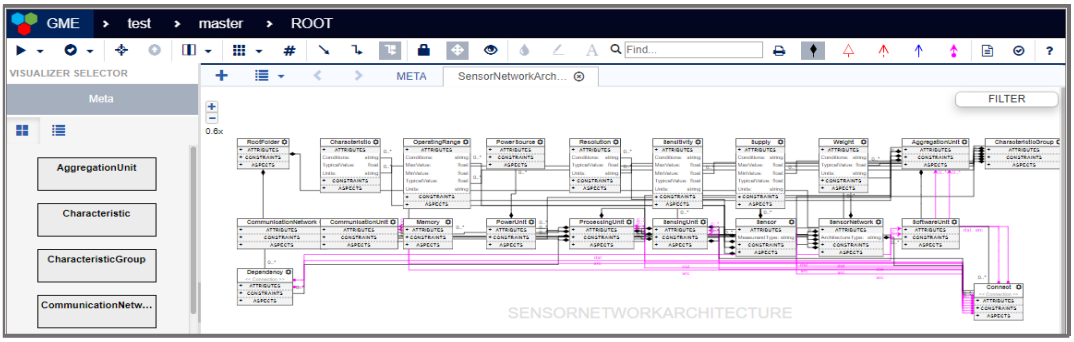

Fig 5. Modeling environment migrated to the Cloud

3.3. Discussion

The microservices Cloud infrastructure for WebGME was implemented using Kubernetes and Docker. As a result, WebGME can be privately accessed via a public IP of the Kubernetes worker and privately managed on a public Cloud. For these activities, two deployments and two new services were realized.

A new configuration for the WebGME docker image has been introduced, in order to suit the MongoDB connectivity. WebGME in Cloud is expected to work better than other desktop and web-based solutions (classic WebGME and AToMPM), according to the following criteria: high availability, disaster recovery, private collaboration, resource management, and private modeling environment. However, they are influenced by the IBM Cloud offerings and Worker performance. *High availability* means the ability of a system/application to run without a failure for a large amount of time. For our work, *disaster recovery* was considered the capacity of the system to be still available if a natural/non-natural disaster occurs. *Private collaboration* and *private modeling environment* refer to the fact that the Git-like features that contain the modeling process are accessible and visible to certain users. *Resource management* is a criterion that is inherited from the Cloud implementation, where resources can be deployed/deleted as needed.

All of the presented criteria represent strengths that the private implementation of WebGME in Cloud has in comparison with similar tools such as classic WebGME and AToMPM. These improvements can help teams in achieving teamwork goals and private environment management alongside cost savings guaranteed by the Cloud provider costs plans.

Regarding performance monitoring for this solution, a specialized monitoring tool like ELK Stack, New Relic, or Dynatrace needs to be added for proper performance track at the pod and application level. This addition represents a much-needed improvement for a production-ready Kubernetes cluster.

4. Conclusion

This paper presented an implementation of the WebGME modeling environment in IBM Cloud and an example of metamodel migration, from a source metamodel for sensor networks initially developed in GME - the on-site variant of this environment. WebGME in Cloud is accessible to all the members of a development team at the same time, and the modifications on the same project can be made by them on any device.

The Cloud-based solution is beneficial in comparison with the web-based and the on-site variants, i.e. WebGME and GME. The integration of WebGME with IBM Cloud was made possible by technologies such as Kubernetes and Docker. This approach came with the advantages offered by containers, such as rapid deployment, simplicity, continuous deployment and isolation.

R E F E R E N C E S

[1]. *Z. Li, Q. Yang, K. Liu, M. Sun, J. Xia*, "Building Model as a Service to support geosciences", in Computers, Environment and Urban Systems, **vol. 61**, Part B, 2017, pp. 141–152

[2]. *D.I. Dogaru, I. Dumitrache*, "Modelling the dynamic electrical system in the context of cyber attacks", in UPB Scientific Bulletin, Series C: Electrical Engineering and Computer Science, **vol. 80**, iss. 2, 2018, pp. 3-16

[3]. *M. Maróti, R. Kereskényi, T. Kecskes, P. Völgyesi, A. Ledeczi*, "Online Collaborative Environment for Designing Complex Computational Systems", in Procedia Computer Science, **vol. 29**, 2014, pp. 2432-2441

[4]. *S. Popoola J. Carver and J. Gray*, "Modeling as a Service: A Survey of Existing Tools", in Proceedings of the Model-Driven Engineering Languages and Systems Conference, 2017

[5]. *E. Cayirci*, "Modeling and simulation as a cloud service: A survey," 2013 Winter Simulations Conference (WSC), Washington, DC, 2013, pp. 389-400

[6]. *M. Maróti T. Kecskes, R. Kereskényi, B. Broll, P. Völgyesi, L. Jurácz, T. Levendoszky, A. Ledeczi*, "Next generation (Meta)modeling: Web- and cloud-based collaborative tool infrastructure", CEUR Workshop Proceedings, **vol. 1237**, 2014, pp. 41-60

[7]. GME: Generic Modeling Environment, http://www.isis.vanderbilt.edu/projects/GME, Accessed March 26 2019

[8]. *E. Syriani, H. Vangheluwe, R. Mannadiar, C. Hansen, S. Van Mierlo, H. Ergin*, "AToMPM: a web-based modeling environment", in Joint Proceedings of MODELS'13 Invited Talks, Demonstration Session, Poster Session, and ACM Student Research Competition, **vol. 1115**, CEUR-WS.org, 2013, pp. 21–25

[9]. *A. Ionita, F. Anton, A. Olteanu*, "Sensor Network Modeling as a Service", in Proceedings of the 8th Int. Conference on Cloud Computing and Services Science (CLOSER 2018) 2018, pp. 346-353

[10]. Kubernetes Concepts: https://kubernetes.io/docs/concepts/, Accessed March 26 2019

[11]. *M. Kenyeres, and J Kenyeres*: "Multi-Sensor data fusion by average consensus algorithm with fully-distributed stopping criterion: comparative study of weight designs.", UPB Scientific Bulletin, Series C: Electrical Engineering and Computer Science, **vol. 81**, iss. 2, 2019, pp. 27-42

[12]. MetaGMEParadigmImporter: https://github.com/webgme/webgme-engine/tree/master/src/plugin/coreplugins/MetaGMEParadigmImporter, Accessed March 26 2019