Convenient Deployment of Self-Managed Elastic Clusters on Federated Clouds

Ignacio Blanquer*, Francisco Brasileiro†, Amanda Calatrava*, Thiago Emmanuel Pereira† and Miguel Caballer*
*Instituto de Instrumentación para Imagen Molecular, Universitat Politècnica de València
Valencia, Spain
Email: {iblanque,micafer1,amcaar}@i3m.upv.es
†Departamento de Sistemas e Computação Universidade Federal de Campina Grande
Campina Grande, PB, Brazil
Email: fubica@dsc.ufcg.edu.br, temmanuel@computacao.ufcg.edu.br

***Abstract*—Federated clouds address many problems of scientific and industrial applications, such as legal restrictions and efficient data access. Despite these benefits, its natural geographical distribution and complex software stack challenges the developers and operators aiming to harness this kind of infrastructure. This article shows an architecture to deploy self-managed Kubernetes elastic clusters on federated clouds. Our approach can scale up/down the managed resources and secure communication and execution across different IaaS providers. All the components are available under open source licenses, and the integration is the result of the developments in the ATMOSPHERE project.**

***Index Terms*—Federated cloud, orchestration, horizontal elasticity, containers.**

I. Introduction

Federated cloud infrastructures are becoming a popular solution to deal with Quality of Service, legal restrictions and data access efficiency in scientific and industrial applications. The deployment and reconfiguration of virtual distributed infrastructures on top of geographic distributed federated clouds imply interesting challenges that range from homogeneous interfaces to Infrastructure as a Service (IaaS) providers, federated authentication and authorization, Virtual Machine Images (VMIs) indexing, and private networks spanning multiple cloud providers, a.k.a. federated networks.

This article describes an orchestration platform and a dashboard to manage a federated set of container-based resources. The use of DevOps cloud orchestration tools minimizes the cost of synchronizing (VMIs), by using on-the-fly configuration of vanilla VMIs. The platform provides also measures and adaptive mechanisms to improve performance. Federation is considered at two levels, namely, cloud resources and containers. Several aspects need to be considered at both levels: authentication and authorisation, container image management, and networking. Pooling at the level of cloud resources leverages the Fogbow middleware [5]. The platform is also capable of managing hybrid resources such as specialized devices (e.g., GPGPUs) and facilitates the development of serverless applications.

The first version of the platform includes the capability of the federation of cloud resources from geographically distributed cloud providers, the services to deploy applications, the inventory of services and repositories that are needed to interact with the federated infrastructure. We show the results obtained in the deployment of an elastic Kubernetes [1] cluster on the federated infrastructure.

The paper is structured as follows. First, the ATMOSPHERE architecture is presented in section 2, also describing the main roles involved in. Then, sections 3 and 4 describe an example application and the experiments performed to test the ATMOSPHERE architecture. Finally, section 5 concludes the paper and points out to the future work.

II. ATMOSPHERE Architecture

The architecture is depicted in Figure 1. We identify four roles for the users of the platform:

- Application Developer. The person who will develop the applications running on the federated infrastructure. The Application Developer will require the infrastructure to provide some types of services and resources, such as elasticity, high-availability, secure storage, data persistence and job execution.
- Application Manager. Despite that both roles may be assigned to the same person, an Application Developer may not be in charge of deploying the application on the production infrastructure. The deployment implies the monitoring and management of the resources, services, user accounts and data. The Application Manager will have access credentials to the federated infrastructure and will decide the optimal allocation of the resources.
- Local Resource Manager. The individual federated sites are managed by system administrator who have to install and manage the federated services and properly annotate the special resources.
- End-Users. Users of the applications will not have infrastructure access rights and will be unaware of the underlying services and components. The requirements of the end-users are mainly relevant to the Application Developers.

In this scenario, the application manager will be the entry point for end-users to the ATMOSPHERE platform. They will interact with the ATMOSPHERE dashboard[1], a web service

[1]http://servproject.i3m.upv.es/ec3-atmosphere/

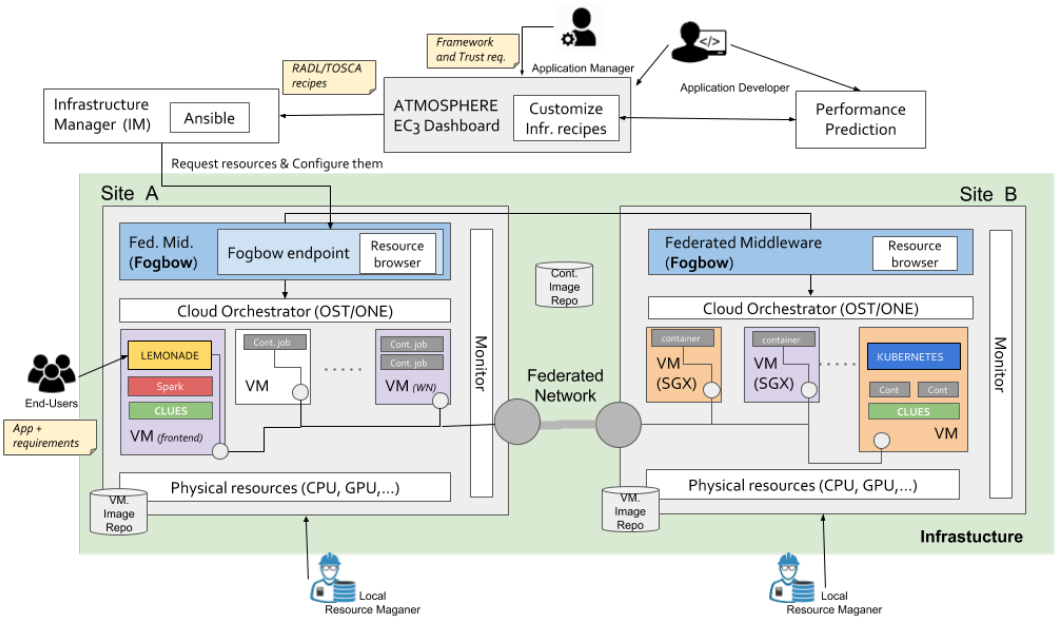

Fig. 1. Architecture design of the ATMOSPHERE Cloud and Container federated infrastructure and associated services.

based on the Elastic Cloud Computing Clusters (EC3) tool [7], to deploy the required infrastructure for a desired framework, i.e. LEMONADE (Live Exploration and Mining of a Non-trivial Amount of Data from Everywhere) a platform for visual creation and execution of data analysis workflows [9]. EC3 is a tool that deploys self-managed elastic virtual clusters on top of IaaS Clouds. EC3 interacts with the IaaS resources through Infrastructure Manager (IM) [6], which is a cloud orchestrator that deploys applications described in standardized documents that supports multiple cloud providers, including Fogbow. Fogbow will be used as the middleware layer to federate sites in the cloud platform to abstract the infrastructure layer to the upper services. Thus, all the resources (general resources like a VM or specific resources like a GPGPU) would be accessed the same way through Fogbow. Dedicated resources (such as GPGPUs or SGX capabilities) will be accessed only if the application manager has indicated this requirement. The ATMOSPHERE dashboard has a customization service, where software, hardware and trustworthiness requirements are informed and mapped into the RADL [6]/TOSCA [3] recipes. These recipes are used by IM to deploy the virtual resources that will compose the application's customised infrastructure. The configuration of the deployed resources is done automatically at run-time using a DevOps tool as Ansible. It uses the recipes specified in the RADL/TOSCA input document that describe the required steps to configure the machines.

EC3 manages the horizontal elasticity of the clusters by means of Cluster Energy Savings (CLUES) [4], a framework that plugs to local resource managers such as Apache Mesos or Kubernetes to adjust the number of resources required according to the workload, applying green computing techniques. Thus, CLUES monitors the working nodes and intercepts the job submissions to the LRMS, enabling the system to dynamically manage the cluster size transparently to the LRMS and the user, scaling in and out on demand (self-management).

The assessment service is used to evaluate a posteriori if an application run was affected by contention and then adapt the trustworthiness score together with the monitoring framework. Application developers would facilitate the end-user to define a set of functions and a set of rules that will trigger the functions' executions. For example, with LEMONADE, users can program their workflow from a graphic interface, and application developers can also use the privacy policies of the data management layer when accessing the databases. The Fogbow middleware is used to abstract the specificities of the different underlying cloud providers, as well as to implement federated services, such as Authentication, Authorisation and Auditing at the federation level, and federated networking services across multiple cloud providers.

In ATMOSPHERE, the working nodes of the clusters may be deployed in different sites of the federated cloud. In this case, they must be connected with a cross site private network that can assure connectivity and confidentiality among all the nodes. Thus, all datacenters of the federated infrastructure must be configured with some SDN/VPN technology enabling the creation of private networks that can connect computing nodes instantiated in different providers. These technologies must ensure connectivity among all the resources belonging to the same virtual cluster infrastructure, and also ensure security on the communications. Fogbow will provide the unifying API that will allow the creation of such private networks across cloud providers potentially running different cloud orchestrators. Moreover, ATMOSPHERE deployments may have a predefined number of fixed working nodes or dedicated nodes, such as datanodes, that the CLUES elasticity manager will not manage.

End users of the platform will connect directly to the framework chosen by the application manager. For example, they might connect to the LEMONADE user interface and

use the applications previously developed by the application developers. In this sense, end users do not need to know the low-level details of the infrastructure they are using.

A. ATMOSPHERE's Infrastructure

The infrastructure of the project is comprised of multiple cloud providers that are geographically distributed. These cloud providers use different cloud orchestrators - e.g OpenStack and OpenNebula - to manage their resources. The seamless federation of these providers is implemented through the use of the Fogbow middleware [5]. Fogbow was originally conceived in the context of the EU-Brazil Cloud Connect project[2]. In the context of the ATMOSPHERE project the middleware is being enhanced in several directions. Firstly, its architecture has been redesigned from a monolithic one, to one based on microservices. Secondly, a number of additional features have been added, and other features should be incorporated until the end of the ATMOSPHERE project. Fogbow provides a RESTful API that applications can use to seamlessly manage resources in any of the IaaS providers that belong to a particular federation, no matter which cloud orchestrator middleware are used by these providers. Currently, there is support for the most popular cloud orchestrator middleware available (eg. OpenStack, CloudStack and OpenNebula). Moreover, adding support to other cloud middleware is simply a matter of developing the appropriate interoperability plugins. For those users that want to interact directly with the system to manage their resources, Fogbow also provides command line and graphical interfaces.

In the ATMOSPHERE project we will support two infrastructures. The first one, called *atm-prod*, is the production infrastructure that is going to be used by all partners to test their developments, and, eventually, to deploy the use case applications. The second infrastructure, called *atm-test*, is the validation infrastructure used to test the software components that implement the Cloud and Container Services Management Layer, including Fogbow, IM and EC3. Currently, the following sites are available on each infrastructure:

TABLE I
ATMOSPHERE PRODUCTION AND TEST INFRASTRUCTURE DETAILS.

Site	Underlying Cloud manager	Location	Num. nodes	Num. CPUs	Memory Size (GB)	Dedicated HW
UFCG	OST	Brazil	20	64	45	None
UFCG	OST	Brazil	1	1	2	SGX
UPV	OST	Spain	10	20	50	None
UFCG	OST	Brazil	20	64	45	None
UFCG	OST	Brazil	20	64	45	None
UFCG	OST	Brazil	2	2	4	SGX

Figure 2 shows the new interface for quota of Fogbow for the production deployment. The quota of the whole federation is aggregated, and the drop list is used to show the quota of a particular member (OST stands for OpenStack [2]).

III. APPLICATIONS

Federated clouds can deal with distributed applications that require executing part of the workload on different sites.

[2]http://eubrazilcloudconnect.eu/.

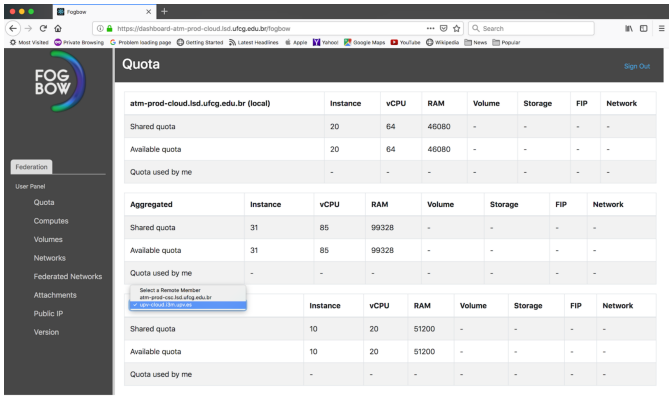

Fig. 2. Quota panel of the Fogbow dashboard (production deployment)

Performance constraints, such as the inefficiency of moving large datasets or the availability of special resources, or legal constraints, as the restrictions on moving sensitive data out of country borders can be fulfilled through this federated model.

In this section, an example of an application to be executed in the ATMOSPHERE platform is presented. We have used a Kubernetes cluster to deploy the application templates, described in YAML. The Dashboard of EC3 can deploy a Kubernetes application description along with the elastic Kubernetes Cluster. The application topology used in this article is described in figure 3, and it is available in a GitHub repository[3]. It comprises several Docker Images, publicly available in the ATMOSPHERE Docker Hub repository [4].

The application addresses the capability of processing a set of data that is available in a persistent volume from a user friendly interface over a distributed cluster (see application topology in Figure 3). For this purpose, we will instantiate the following elements:

- A persistent volume and a sshfs server as a Docker container that exposes it in the private Flannel [8] overlay network sharing the secrets and configMaps Kubernetes objects.
- A Jupyter notebook that mounts such remote volume and the local files necessary to interact with the kubernetes cluster to scale it up and down.
- A set of worker containers running the services to form an ipython cluster. All of them run inside Docker containers and access the shared space through sshfs.

IV. EXPERIMENTS

The deployment of the Kubernetes cluster is made though the IM. It uses an Ansible role [5] to configure the resources to behave as a Kubernetes cluster. A vanilla Ubuntu 16.04 image has been used in the tests. Due to this restriction, all the VMs have been finally allocated at the UFCG site as it is the only one that supports this image. The tests performed consisted in four different steps:

[3]https://github.com/eubr-atmosphere/d42_applications/tree/master/integrated

[4]https://hub.docker.com/r/eubraatmosphere/eubraatmosphere_autobuild

[5]Kubernetes Ansible role https://github.com/grycap/ansible-role-kubernetes

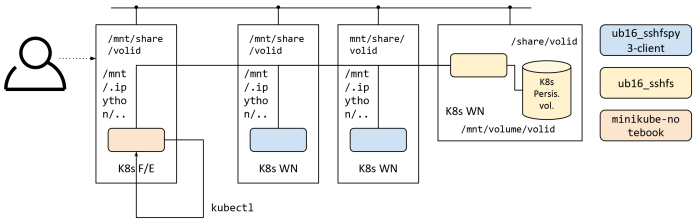

Fig. 3. Final application topology. The coloured boxed on the right refer to the tags of the Docker containers used.

- Initially only a Kubernetes front-end node is launched: this process is the most time consuming step as it implies the installation of Ansible in the master node plus the installation and configuration of the Kubernetes software. This step took 13 min. 31 sec. This time implies an average times of 1 min. 39 sec. to boot the VM until the VM is reachable by SSH; 6 min. 20 sec. to install Ansible, and 5 min. 32 sec. to install and configure the Kubernetes front-end.
- In the second step, 2 working nodes are added to the infrastructure. This step implies the total configuration of the working nodes and the reconfiguration of the front-end to correctly accept the new added nodes. This step took 6 min 9 sec. This time implies an average times of 1 min. 39 sec. to boot the VM, and 4 min. 30 sec. to install and configure Kubernetes Working Node (WN).
- Third step removes one of the previously deployed WNs. This step took 1 min. 26 sec. This step is the less time consuming as it does not deploy and configure any new VM and only reconfigure the rest of nodes needed to maintain the infrastructure correctly working.
- Finally the last step performed adds 5 WNs to the infrastructure to get a final infrastructure of 1 front-end and 6 WNs. This step took 10 min. 32 sec. Considering the average time of 1 min. 39 sec. to boot the VMs, the time needed to reconfigure the infrastructure, including the fully configuration of the 5 new nodes, took an average time of 8 min. 53 sec.

The time consumed deploying the working nodes of the cluster can be improved suspending the VMs instead of destroying them. Another possible solution is the usage of golden images, a feature of EC3 that creates a VMI when the first node is configured and uses it later on to accelerate the following deployments,

These tests show that, in a reasonable time, using a vanilla image (without any pre-installed application) a Kubernetes cluster can be deployed in a federated infrastructure using the components shown in the architecture.

V. Conclusion

This article describes a service for the convenient deployment of self-managed elastic clusters on a hybrid federated cloud offering. The federated cloud infrastructure consists on a federated deployment of several sites at both sides of the Atlantic ocean that integrates a small amount of resources, enough for the validation of the deployment services. This first prototype is based on a reworked new microservice-oriented version of Fogbow and a set of PaaS services that enable the deployment of applications on top of the federated infrastructure.

Currently, the platform enables deploying virtual appliances with multiple interconnected VMs in a federated environment supporting the execution of applications for data analytics on top of it. We provide a simple, user friendly interface to deploy two virtual appliances models and several applications as containers on top of them.

The next steps will focus on the integration of heterogeneous resources (GPGPUs and SGX) through the information system of Fogbow and the extension of the applications descriptions to include such resources.

We foresee that before the end of the ATMOSPHERE project, fogbow will also integrate resources in the main public cloud providers (Microsoft Azure and Amazon Web Services), as well as clouds orchestrated by vRA, VMWare's solution for cloud management, which will be also supported by the cloud deployment service.

Acknowledgment

The work in this article has been co-funded by project ATMOSPHERE, funded jointly by the European Commission under the Cooperation Programme, Horizon 2020 grant agreement No 777154 and the Brazilian Ministério de Ciência, Tecnologia e Inovação (MCTI), number 51119.

References

[1] Kubernetes web site. https://kubernetes.io, accessed: 29-12-2018
[2] OpenStack web site. https://www.openstack.org/, accessed: 29-12-2018
[3] TOSCA Simple Profile in YAML Version 1.0. http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/TOSCA-Simple-Profile-YAML-v1.0.html, accessed: 29-12-2018
[4] de Alfonso, C., Caballer, M., Alvarruiz, F., HernÃąndez, V.: An energy management system for cluster infrastructures. Computers & Electrical Engineering **39**(8), 2579 – 2590 (2013). https://doi.org/https://doi.org/10.1016/j.compeleceng.2013.05.004
[5] Brasileiro, F., Vivas, J.L., d. Silva, G.F., Lezzi, D., Diaz, C., Badia, R.M., Caballer, M., Blanquer, I.: Flexible federation of cloud providers: The eubrazil cloud connect approach. In: 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA). pp. 165–170 (March 2016). https://doi.org/10.1109/WAINA.2016.128
[6] Caballer, M., Blanquer, I., Moltó, G., de Alfonso, C.: Dynamic Management of Virtual Infrastructures. Journal of Grid Computing **13**(1), 53–70 (2015). https://doi.org/10.1007/s10723-014-9296-5
[7] Calatrava, A., Romero, E., Moltó, G., Caballer, M., Alonso, J.M.: Self-managed cost-efficient virtual elastic clusters on hybrid Cloud infrastructures. Future Generation Computer Systems **61**, 13–25 (2016). https://doi.org/10.1016/j.future.2016.01.018
[8] CoreOS.: Flannel web site. https://github.com/coreos/flannel/, accessed: 29-12-2018
[9] Santos, W., Carvalho, L.F.M., de P. Avelar, G., Silva, Jr., A., Ponce, L.M., Guedes, D., Meira, Jr., W.: Lemonade: A scalable and efficient spark-based platform for data analytics. In: CCGrid (2017)