An Application of Kubernetes Cluster Federation in Fog Computing

Francescomaria Faticanti*,‡, Daniele Santoro*, Silvio Cretti*, Domenico Siracusa*

***Abstract*—This demonstration aims at showcasing an application of a cluster federation to increase the elasticity and resilience of a Fog Computing system. Federation is performed by means of the Kubernetes Cluster Federation (KubeFed), a framework we augmented with a two-phase workload placement mechanism that smartly distributes applications' microservices among the federated infrastructure. Despite KubeFed has been generally used in a multi-cloud environment for workloads split on different cloud providers avoiding the lock-in, in this demonstration we show that it can also be used for implementing a decentralized control plane in a highly distributed architecture where networking issues should be taken into account.**

I. Introduction

Fog Computing [1] aims to extend the concept of Cloud Computing offering cloud services and functionalities, such as computational, storage and network resources, throughout the whole Cloud-to-Things continuum, *i.e.,* among all different tiers between the world of the IoT devices and the traditional Cloud Computing ecosystem. The Fog paradigm offers new challenges and increases the complexity of the management and the orchestration of such a distributed computing infrastructure with respect to the traditional centralized Cloud Computing environment. In this context, application deployment, resource allocation and workload placement must be conceived in a heterogeneous, widespread environment where locality, context-awareness and network performances must be taken into account.

FogAtlas [2] is a Fog Computing software platform, based on Open Source technologies, such as OpenStack and Kubernetes [3], [4], offering (i) service-aware workload placement, (ii) zero-touch deployment and (iii) negotiation of resources (both computational and network ones) along the Cloud-to-Things continuum. However, the ability of FogAtlas to support not only distributed but also decentralized systems has yet to be demonstrated. In this paper we will describe how the Kubernetes Cluster Federation, namely KubeFed [5], can be used to achieve a good degree of decentralization and robustness in a Fog environment. Indeed, the use of KubeFed results to be innovative in this context given its sole usage in cloud. In fact, such an approach is frequent to guarantee multi-cloud distribution of the workload, high availability of applications, and support for hybrid clouds. Furthermore, all the existing cloud technologies such as Kubernetes [4] present high availability mechanisms that can recover from failures within a given cluster and/or when clusters' resources are interconnected with a stable and robust network. However, in case of highly and remotely distributed resources interconnected with unstable and low quality networks (frequent conditions in Fog Computing), such mechanisms can not guarantee a reasonable degree of robustness.

The demonstration compares two different scenarios, the first one where FogAtlas manages a single Kubernetes cluster covering the entire Fog infrastructure, and the second one where the infrastructure is split, according to the geographical location, in different and independent Kubernetes clusters federated among each other. In the first scenario, we show that the single centralized control plane is not able to guarantee the continuity of the operations in case of a fault in the control plane or in the networking side. On the other hand, the second scenario, while still offering the possibility to coordinate the entire multi-cluster system (it would not be possible in case of multiple isolated/not federated clusters), is also able to guarantee a higher level of resilience and availability. In fact, despite the "host cluster" (the cluster that exposes the KubeFed API and runs the KubeFed control plane) could undergo a disruption of the service, still the "member clusters" (the clusters registered to the federation) can continue to operate locally on their own resources.

The main contribution of this demonstration is represented by: (i) the introduction of a federation-based approach to subdivide the infrastructure; (ii) the conception of a new application modeling method allowing the split of an application into different chunks that can be deployed on different clusters of the federation; (iii) a two-phase placement scheme compatible with the cluster federation. In this manner, we obtain a more scalable and resilient system for dealing with a highly distributed scenario such as the Fog Computing one.

The rest of the paper is organized as follows. The next section describes the architectural view of the proposed solution. Sec. III outlines the demonstration setup and workflow, and a concluding section ends the paper.

II. Architecture

We introduce some fundamental concepts of the system:

Federation: Framework that allows to coordinate the configuration of multiple Kubernetes clusters [5].

Host Cluster: Cluster containing the control plane of the Federated Kubernetes.

Member Cluster: Cluster that takes part in the federation.

*Fondazione Bruno Kessler, Italy, ‡University of Trento, Italy. This work has received funding from the EU H2020 R&I Programme under Grant Agreement no. 815141 (DECENTER: Decentralised technologies for orchestrated Cloud-to-Edge intelligence).

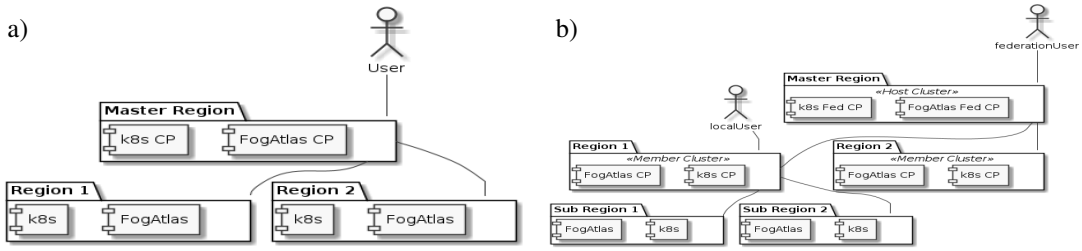

Fig. 1. a) Scenario 1: FogAtlas handles a single Kubernetes cluster covering the entire infrastructure; b) Scenario 2: The infrastructure is divided in different independent and federated Kubernetes clusters.

Region: Set of computational resources in the same geographical location.
Master Region: For the non-federated scenario it is the region where the control plane is hosted. For the federated scenario it is the region containing the Host Cluster.

Figure 1 depicts the different scenarios mentioned in the previous section. Figure 1a) shows a single cluster scenario where Kubernetes and FogAtlas control planes are located in the Master Region (generally hosted in cloud). The user can interact with the system only through the Master Region and the other regions trust it for operations related to the control plane. As shown in [2], in this scenario FogAtlas can help the user in modelling a cloud-native application, orchestrating its workload on the distributed Fog infrastructure, and monitoring its performance. This solution, based on the Kubernetes Custom Resource Definition (CRD) extension point, has a limited overhead and works well in all the cases where we have a reliable connection between the centralized control plane and each region. However, in case of faults in the network connecting the region hosting the control plane with the other regions, it is no more possible to operate on the remote regions.

Figure 1b) shows a multi-cluster federated scenario where the Master Region (generally the one hosted in cloud) plays the role of Host Cluster while the other remote regions play the role of Member Clusters. The control plane of the federation (Fed CP) is hosted on the Master Region but each Region hosts the control plane of its Kubernetes and FogAtlas cluster. In this scenario we leverage two different levels of workload orchestration, one at the federation level in order to select the cluster candidate to host a given workload and another at cluster level in order to select the sub-region/worker nodes[1] where to deploy the workload. The FogAtlas implementation of such a multi-layered orchestrator is again based on the Kubernetes CRD extension. In this case the CRD FedFAApp is the representation of a "federated", cloud-native application composed by a set of application pieces, namely *application chunks*, and *DataFlow* between these different *chunks*. Each chunk is a set of one or more microservices that should be placed on a given cluster of the federation. Such a placement is managed by the federated FogAltas controller. An application chunk is represented by a federated version of the FADepl CRD [2] that in turn is ingested by the corresponding FogAtlas controller for a fine tuned and intelligent (based on resource availability and constraints imposed) placement inside the selected cluster. Figure 2 shows the FedFAApp models.

[1]Inside a region/cluster, the computational nodes can be grouped in order to create smaller sets of resources, i.e. sub-regions.

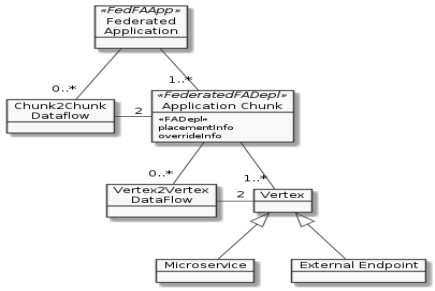

Fig. 2. Models underlying a FedFAApp.

In terms of resilience, this federated solution is more convenient than the single cluster one. In fact, in case of a service interruption in the Host Cluster or a network failure, the users can continue to interact directly with the control plane of the region where their applications are deployed with limited disruption of the services offered.

III. Setup and Workflow

Environment and Setup. We set up two infrastructures, corresponding to the scenarios mentioned above:
1) The single-cluster infrastructure is composed by two regions all belonging to a single Kubernetes cluster;
2) The multi-cluster federated infrastructure is composed by two clusters and one of them is composed by two sub-regions.
Figure 3 depicts these two different scenarios in an emulated environment representing an application deployment around the city of Trento. The application used for the experiment, as depicted in Figure 3, is a simple and generic IoT data-intensive application composed by three microservices:
Data Collector (DC): It collects data from a sensor/camera and elaborates them obtaining some processed result.
Repository (REP): Database storing processed data.
Webserver (WS): It allows the access to processed data.
Storyboard and Workflow. The main objective of the demonstration is to show a new placement scheme for the applications in the new federated scenario, and the resiliency of the federated environment in response to failure situations like networking faults. The new placement scheme consists of two steps: the first one where the application's microservices are

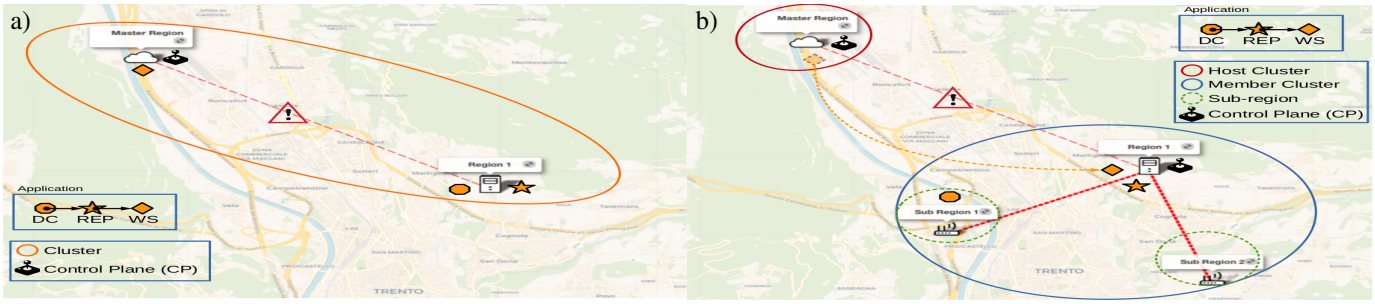

Fig. 3. a) First part of the demo with a single cluster; b) Second part of the demo with two federated clusters.

divided into application chunks and each chunk is assigned to a specific cluster, and the second step where each microservice of each chunk is deployed within the specific cluster assigned to the chunk in the previous step. This last step, within a specific cluster, is performed placing each microservice among the subregions composing the cluster.

The initial phase of the demonstration consists of the application composition where the user specifies the computational requirements for each microservice of the application, and the networking requirements, in terms of throughput and latency, for each link between the application's microservices. After this preliminary step, the demonstration consists of two parts.

In the first part of the demo, the single cluster infrastructure is used and the application described above is deployed on top of it. As shown in Figure 3a), the application's microservices are split among the regions in the single cluster. According to the FogAtlas placement algorithm, one instance of WS is deployed on Cloud (Master Region), while DC and REP are deployed on the Edge region (Region 1) taking into account the application's requirements. A network failure that isolates the Cloud region causes a disruption of the service of the application since the microservice placed in Cloud is no more reachable by the ones in Region1. Furthermore, the FogAtlas/Kubernetes control plane is no more available to recover from this problem.

In the second part of the demo, the multi-cluster infrastructure is used. In this case the application is divided into two chunks: one in the cloud cluster (Host Cluster), and the second in the Region1 cluster (Member Cluster). Within each cluster the second phase of placement mimics the allocation obtained in the previous case with the DC and REP modules deployed among the sub-regions of the Region1 cluster. However, in case of a network failure between the two clusters, causing a disruption of the application services, a recovery process can be initiated requesting the FogAtlas control plane of the Region1 cluster to deploy another instance of WS microservice on the same cluster. In this way the WS can access the REP data and the application services can be temporarily restored thanks to this new placement, waiting for the recovery of the network services.

Measurements. Finally, we report some measurements of the overhead, in terms of time, added by FogAtlas controllers in a single and in a multi cluster configuration for different numbers of *microservices/application chunks* to be placed, with respect to Kubernetes vanilla. The measurements are taken on a master node equipped with 2 vCPU and 2 GB RAM and consider only the time needed by the FogAtlas controllers in order to decide the placement. Please note that the reported overhead includes the time taken by the call to the Kubernetes API for submitting the resources to be scheduled, for which we experienced a sharp increase when the number of microservices to be deployed goes over 5 units. This aspect will be further analysed in future works. As shown in Table I, the overhead increases smoothly until a certain amount of microservices/chunks (i.e. 5) but then the increment is more than linear due to the saturation of the computational resources on the master node. Reasonably, the overhead in the multi-cluster case is bigger since it represents the sum of times of the two phases of placement. However, the overhead for deploying a mid sized application is around a second in the worst case.

TABLE I
MEASUREMENTS

Microservices/chunks	1	2	5	10
Single cluster (ms)	180	255	479	2141
Multi cluster (ms)	552	686	1071	4572

IV. CONCLUSIONS

We proposed a new application of KubeFed in Fog Computing showing a more resilient system to face network faults that can compromise the application services on the infrastructure. We set up a demo where two different scenarios are compared: a single cluster scenario, and a federated multi-cluster one. The latter, thanks to the presence of a control plane for each cluster, is able to easily adapt to network faults situation with a small overhead with respect to classical solutions. Future works will investigate clusters configuration methods and new application placement mechanisms.

REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.
[2] FogAtlas. https://fogatlas.fbk.eu/.
[3] OpenStack.https://www.openstack.org/.
[4] Kubernetes. http://kubernetes.io/.
[5] KubeFed. https://github.com/kubernetes-sigs/kubefed.