FIWARE Lab: Managing Resources and Services in a Cloud Federation supporting Future Internet Applications

Theodore Zahariadis , Andreas Papadakis , Federico Alvarez , Jose Gonzalez , Fernando Lopez
Federico Facca , Yahya Al-Hazmi

Abstract — **Real-world experimentation facilities accelerate the development of Future Internet technologies and services, advance the market for smart infrastructures, and increase the effectiveness of business processes through the Internet. The federation of facilities fosters the experimentation and innovation with larger and more powerful environment, increases the number and variety of the offered services and brings forth possibilities for new experimentation scenarios. This paper introduces a management solution for cloud federation that automates service provisioning to the largest possible extent, relieves the developers from time-consuming configuration settings, and caters for real-time information of all information related to the whole lifecycle of the provisioned services. This is achieved by proposing solutions to achieve the seamless deployment of services across the federation and ability of services to span across different infrastructures of the federation, as well as monitoring of the resources and data which can be aggregated with a common structure, offered as an open ecosystem for innovation at the developers' disposal. This solution consists of several federation management tools and components that are part of the work on Cloud Federation conducted within XIFI project to build the federation of cloud infrastructures for the Future Internet Lab (FIWARE Lab). We present the design and implementation of the solution-concerned FIWARE Lab management tools and components that are deployed within a federation of 17 cloud infrastructures distributed across Europe.**

Index Terms — **cloud computing; federation management; open innovation.**

I. INTRODUCTION

Open Innovation [1] is one of the most active trends in innovation processes during the last years. In short, Open Innovation is about creating and profiting from technologies by providing open ecosystems or platforms that foster the easy and open exchange of ideas to produce innovative products or solutions. As shown by many recent attempts [2], cloud capacity and services are of great value to support Open Innovation and facilitate start-up incubation through infrastructure resources. How to create a large distributed cloud to boost innovation across European regions? This requires either large investments by a single player or an agreement among many players that team-up together to build such a cloud. Of course, when different players team up together, they do not want to lose all the control on their own infrastructure. For example, a given infrastructure owner may not want to share all its monitoring data, or may prefer to use a specific solution to provide network connectivity to the user accessing it, or may request the user to be registered in an infrastructure proprietary Identity Management solution. These are some of the challenges and motivations behind a federated cloud architecture [3].

This paper presents the work on Cloud Federation conducted within XIFI project [4] to build the federation of cloud infrastructures for the Future Internet Lab (FIWARE Lab) [5]: an open innovation platform to develop Future Internet applications. FIWARE Lab offers a rich catalogue of services available either in SaaS or PaaS modality, the so called Generic Enabler implementations (GEis) and a wide offer of Future Internet facilities (e.g. sensor networks, 4G networks, etc.). The GEis, developed within FIWARE project [6], are conceived to be building blocks that can be easily composed to create complex applications. In this perspective, the Future Internet facilities provide advanced experimental capacities to developers that are able to link their applications with actual infrastructures and test them in real world settings.

Currently, the cloud federation backing FIWARE Lab consists of a community of 17 cloud infrastructures spanning multiple regions across Europe [7]. The variety and heterogeneity of such context raise several challenges to the harmonization of the cloud offering to developers. To reap the rewards of adopting a cloud federation approach, the appropriate management of such complex environment and the associated resources are key elements in the provision of cloud services. This paper presents some innovative components that, working in an orchestrated manner, perform core operations within the federated management of FIWARE Lab. More specifically, in the context of this paper we present the components that deal with the management of resources and services from an application developer viewpoint, in particular: i) the capacity to deploy multi-tier applications across the federation; ii) the provisioning of real-time information by the services (GEis)

offered by the community, supporting prompt responses and accommodating different accessing policies; iii) the design and implementation of a centralized mechanism to handle large-scale monitoring data gathered via the federated underlying resources, by establishing a well-defined and standardized API for storing, aggregating and publishing such data.

The rest of the paper is structured as follows: Section II provides current state-of-the-art solutions and challenges that have been tackled for the realization of FIWARE Lab. Section III presents an overview of the cloud federation management architecture adopted in FIWARE Lab, along with the description of the components functionality. Moreover, Section IV presents a real-case scenario, discussing appropriate actions for managing the whole life-cycle of a multi-tier application within FIWARE Lab. Finally, the main conclusions and future work are presented in Section V.

II. State of the ART and Challenges

Plenty of academic research and industry standardization activities are related to the concept of cloud federation. The IEEE intercloud is an initiative that focuses currently on standardization for cloud interoperability and federation [8]. Additionally, the OpenStack as a widely-used cloud management framework, adopted by FIWARE Lab, has some means of federation aspects focusing on the federation of identity management across the federated clouds through the Keystone [9]. The survey in [10] discusses many aspects motivating the concept of the cloud interoperability or federation, categorizes and identifies multiple possible scenarios and architectures for cloud interoperability. Papazoglou in [11] presents a reference architecture for cloud integration together with an approach for integration and management environment capable of offering integration-as-a-service functionality. Moreover, the authors in [12] show the added value of the federation of multiple independent cloud providers, through the adoption of a layered cloud service model (of SaaS, PaaS, IaaS), mediated by a broker, specific to the concerns of the parties at that layer. Moreover, several European activities focus on cloud federation such as EGI federated cloud [13], Contrail [14] and BonFIRE [15].

Nevertheless, in federated clouds, many challenges concerning resource provisioning, orchestration, monitoring require thorough investigation. Authors in [16] introduce a data-centric approach for orchestration of cloud resources where resources are modeled as structured data that can be queried by a declarative language as well as updated with well-defined transactional semantics. In [17], the authors propose technology-neutral interfaces and architectural additions for handling placement, migration and monitoring of Virtual Machines (VMs) in federated cloud environments. However, this work does not take into consideration the specificities of gathering and consuming data of the federation through a uniform API. In [18], the advantages of adopting a heterogeneous federation approach are discussed and a high-level architecture is presented regarding the Future Internet Experimentation Facilities, where the monitoring component are introduced to cater for providing information about the infrastructure resources for experimenters. As it is an ongoing work, this federation approach is still in its specification and implementation phase. The BonFIRE multi-site cloud testbed facility [15] supports large-scale testing of applications, services and systems over multiple, geographically distributed, heterogeneous cloud testbeds. In order to ease the setup and deployment of an experiment in BonFIRE, the JSON-formatted experiment descriptor is used [19]. A user submits a single document to the BonFIRE experiment manager interface that is able to orchestrate the cloud resources provisioning taking into consideration their dependencies. However, the aim of this federated infrastructure is not to provide a production environment for cloud applications; this makes the adoption of this architecture improper for the needs of FIWARE Lab. Last but not least, a cloud federation platform demands a standard API to manage, in a homogeneous manner, the performance monitoring data collected from the multiple resources spread across the federated domains. To accomplish such a duty, the data-set to be handled requires uniform representation at federation level. We already give in our prior work [20] an overview on the state-of-the-art on cloud monitoring and introduced the concept of Federation Monitoring together with its description, which is enhanced in this document.

From the efforts introduced so far, it is evident that a number of challenges regarding the management of cloud federation still remain unsolved. Additionally, particularly in the case of FIWARE Lab, one has to deal with specificities introduced by the adoption of the GEi concept that are not addressed by cloud platforms, such as OpenStack which has been selected by FI-Lab as the basis for its cloud federation environment. It is highlighted that through the selection of OpenStack, FIWARE Lab is also aligned with the standardized Open Cloud Computing Interface (OCCI) specifications [21]. In the following, we present the challenges related to the simplification of activities to be performed from an application developer's point of view, covering the whole life-cycle of an application, as well as the proposed solutions adopted by FIWARE Lab to address these challenges:

- Deploying an application in a cloud federation is not a trivial task for an application developer, due to several reasons. First, he must be capable of deploying multi-tier application, given that due to regulatory or security reasons each tier may need to be deployed in different infrastructures comprising the cloud federation. Second, he must be able to select among a set of configuration tools that will help him to minimize chronophage operations, such as upgrade, etc. To resolve this issue, a Platform-as-a-Service instantiation has been developed, namely Pegasus, which is a reference implementation of the PaaS developed within FIWARE and offered in FIWARE Lab, catering for the minimization of the effort needed for deploying and adapting applications, while offering the ability to deploy multi-tier applications within the cloud federation in an easier manner.
- The application developer must be given the opportunity to check the GEis already available in the cloud federation, in order to be able to combine them and develop complex applications timely and easily. The Deployment and Configuration Adapter (DCA) is a component, deployed within FIWARE Lab that gathers and correlates information

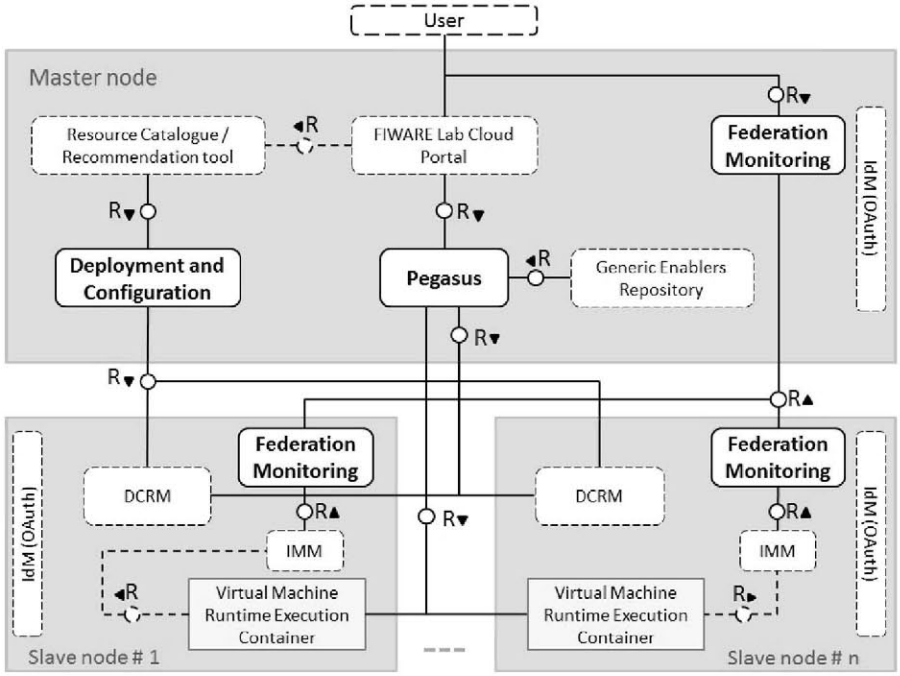

Figure 1 - Architecture overview and federated management approach.

on GEis offerings throughout the cloud federation, providing a standardized, RESTful API to execute complex queries regarding the provisioned services across the cloud federation.

- An application developer must be able to monitor the performance of the federation resources that he is willing to use, regardless of the heterogeneity arising from different monitoring tools installed by each infrastructure node. In order to accomplish the challenge of a unified and scalable monitoring framework across the cloud federation, a Federation Monitoring system (along with specific adapters) has been implemented and integrated in FIWARE Lab, offering a standard API to manage, in a homogeneous manner, aggregated and real-time monitoring data.

The aforementioned components are described in detail in the next section.

III. Architecture Overview and Federated Management of Resources and Services

This section presents an overview of the cloud federation management architecture and the connections between corresponding components, comprising the federation layer of FIWARE Lab, as depicted in Figure 1. It is noted that although the architecture includes a plethora of components, this paper will focus on those related to application developers and end-users. In the next paragraph, the main functionality and the interactions of the main components are exemplified through a typical deployment and monitoring scenario. Prior to describing the basic functionality of each component depicted in Figure 1, it is important to mention that the FIWARE Lab platform architecture consists of a Master Node, that includes the components responsible for supporting the federation management and the provisioning of resources and services across the federation in a unified fashion, and several Slave Nodes, i.e. the different Cloud infrastructures taking part to the federation, offering their resources and services.

As aforementioned, in this paper, we focus on the perspective of an application developer, in the FIWARE Lab context, as a user of the cloud federation platform. To access the cloud resources, the developer, after authenticating through the Federated Identity Manager (IdM-OAuth), logs into the FIWARE Lab Cloud Portal. The portal enables the user to see the list of available services and resources according to their location in the federation or even retrieve on demand information (in the form of ratings) regarding the provided services already deployed and used by other users, via the Resource Catalogue. At this point, the user, willing to deploy his own application in the cloud federation, may select one or more services (including all GEis stored in the Generic Enablers Repository), either through pre-built images or dynamically composed recipes and blueprints, provided through the Pegasus which represents a PaaS GEi. Pegasus offers the ability to the application developer to deploy a multi-tier application where each tier is deployed in a different Slave Node (multi-node deployment of multi-tier applications). To support the deployment, Pegasus communicates with the Data Center Resource Management (DCRM) GEi, i.e. the Cloud IaaS manager, and deploys the requested appliance, either being a simple VM, a single GEi or multiple GEis. When the deployment of the new application is completed, there are two activities that take place (in an automated fashion, transparent to the user) within the cloud federation environment: the first activity is the real-time gathering of information regarding the newly deployed

application by the Deployment and Configuration Adapter (DCA), that maintains information related to all services deployed within the cloud federation environment and provides an API for information retrieval by the Resource Catalogue; the second activity is the collection of monitored data information from each federated cloud infrastructure through the Infrastructure Monitoring Middleware (IMM) component and a set of probes, enabling application developers to access monitoring data of their deployed applications through a properly defined API, offered by the Federation Monitoring component.

Although the components in the dotted boxes have been depicted for the sake of completeness, they will not be further discussed in this paper. However, the interested reader may refer to [22] for extended description of their functionality. In the next section, we describe in detail the functionality of the components accommodating the application developers' needs, depicted in solid boxes in Figure 1, namely, Pegasus, DCA and Federation Monitoring.

A. Platform-as-a-Service Component (Pegasus)

Pegasus [23] orchestrates the provisioning of the required virtual resources at infrastructure level and the installation and configuration of the whole software stack of an application, taking into account the underlying virtual infrastructure. It provides a flexible mechanism to perform the deployment, enabling multiple deployment architectures: all components in a single server, in several servers, or elastic architectures based on load balancers and different software tiers. Collectively, Pegasus enables a user to deploy easily any kind of application, be a single VM, a single GEi or multiple GEis in the FIWARE Lab federated infrastructures. Most importantly, Pegasus offers the opportunity to the user to deploy multi-tier applications, where each tier can be accommodated by a different infrastructure of the federation as shown in Figure 2.

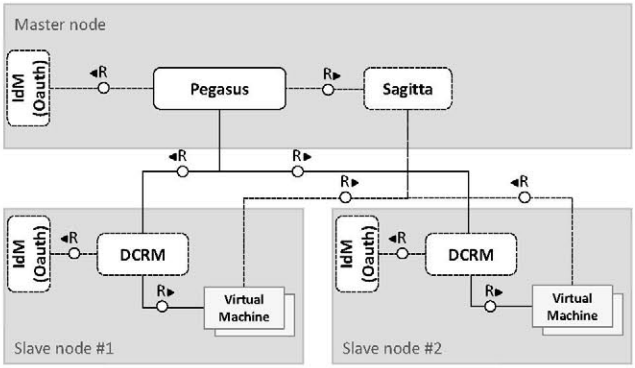

Figure 2 - Pegasus architecture overview.

The infrastructure, that an application needs to execute their logic, can be manually deployed in the cloud. But this is a time-consuming and repetitive operation ad eternum. Opposite to it, Pegasus helps to improve the efficiency of IT resources and processes that form part of any application delivery. It facilitates the creation of applications and services without the complexity and cost of provisioning of the required resources and the management of the traditional application platform stack. Just in a few minutes, the developer can use his applications by instantiating the infrastructure template defined for his applications. In the same time, Pegasus configures the VMs deployed in order to allow the monitoring of its resources and help to identify the GEi installed on it with the configuration of metadata associated to it (see section IV for more details) together with the identification of the infrastructure in which it is deployed. This configuration is made through the use of the metadata service functionality offered by OpenStack [24]. All this information is important in order to provide traceability of the deployment of the different GEis. The Deployment and Configuration Adapter (DCA) takes the information of these metadata in order to identify which GEis have been installed around the federation (more details about DCA can be found in section III.B).

All these automatic operations help to increase the benefits of Agile Software Development [25] and DevOps [26] methodologies through the reduction of expended time to delivery software application projects. Once the developer defines the infrastructure of his applications through the definition of its template, it can be reused each time the developer needs to deliver a new instance of it. Additionally, and due to the use of DevOps concepts, Pegasus reduces the architectural complexity and allows adapting the applications and IT services to answer quickly to new conditions market and organization changes. Any changes on them is promptly translated to the description of the application infrastructure via the redefinition or update of the infrastructure template.

But the functionality offered by Pegasus does not end with these characteristics. Pegasus prepares the images to connect automatically with the federated monitoring architecture and install the required software on it. There are two alternatives to resolve this issue. The first one consists in the installation of any client that the developer could need on his VM, such as Chef Client, Puppet Agent, Monitoring client. The problem of this solution is that one needs to have all Operating System versions with all the preconfigured clients already installed which brings an exponential growth in complexity due to the different number of versions and operating systems. The second alternative is to make use of the cloud-init [27] scripts. It is a set of python scripts and utilities that allows the developer to install everything that he might need. It is the genuine multi-distribution package that manages early initialization of a VM instance and all FIWARE Lab images have it pre-installed for relaxing developer from time-consuming configuration activities.

The next step is the installation and execution of the Chef or Puppet recipes, through the cloud-init. The VM should connect to the Sagitta in order to know the recipes that should be installed on this VM. Sagitta [28] is another FIWARE Lab component, out of scope of this article that allows registering which software should be installed on each VM, using puppet or chef recipes. This GEi allows the automatic installation and modifications to a system through a configuration management system and system's rules as code which resolves lots of manual intervention of IT manager [29]. Pegasus informs Sagitta about which software should be installed on a VM instance in order to have control of what the developer needs to install on each VM. Additionally, the monitoring recipe

includes a post-installation section which is used to inform which operations should be done in order to connect the monitoring client with the Orion CB and so inform of the state of the new VM instance.

Finally, the last functionality that Pegasus offers is the automatic provisioning of VPN between different nodes when a template is deployed in different clouds. This operation is performed in a transparent way where the FIWARE Lab user does not need to know anything about network configuration between different infrastructures. Nowadays, we have seen references to Andromeda (Google) [30] that attempts to offer the same functionality for their well-known cloud services.

B. *Deployment and Configuration Adapter (DCA)*

Cloud federation management is not only related to the deployment of applications, but also includes the provisioning of the appropriate components offering the opportunity, on one hand to service providers to publish available services (GEis), along with their detailed descriptions, and on the other hand to application developers to discover and utilize these services. However, in order to be able to advertise and make use of these services (GEis), all required information has to be gathered from the cloud federation environment, leveraging on the advantages offered by the Pegasus service deployment. In this perspective, Deployment and Configuration Adapter (DCA) is the component that caters for the persistency of all pertinent information related to the whole lifecycle of services (GEis), seen as an advanced cloud federation caching mechanism, providing the following functionalities:

- Capable of storing the deployment commands performed by the application developers and the respective responses of the cloud federation and correlating it with a particular GEi,
- Properly collects and correlates all services deployed and offered in the cloud federation, even in the case that the application developer has not used the functionality of DCA to deploy an application, but instead used other FIWARE Lab components (e.g. FIWARE Lab Cloud Portal),
- Offers to application developers and other interested users real-time responses to complex queries related to information and statistics of available services (e.g. where a specific GEi is deployed, what is the down-time of this specific GEi, how many users are using it, which services are offered by a specific infrastructure, etc.).

However, in order to fulfil the above functionality, two technical barriers had to be overcome.

The first one was to find a technical solution that would allow the DCA component to access each individual infrastructure, as depicted in Figure 3, in order to collect information regarding offered services and resources, respecting potential stringent access policies applied by the infrastructure owners (Slave Nodes) not willing to provide administrative privileges to external parties. In order to satisfy this requirement, a software component has been developed (Python script), that is able to collect respective information through the OpenStack Nova and Glance components. This is achieved by offering the opportunity to the infrastructure administrator to edit and customize the parameters of the Python script before installing it on the controller of the infrastructure (DCRM) and thus allowing infrastructure owners to preserve access privileges. Of course, in the case that infrastructure owners are willing to provide administrative access privileges to DCA, then OpenStack Keystone API (through the respective endpoint) can be used directly by DCA.

The second one was a FIWARE Lab specific issue, related to the unambiguous identification of the deployment of a specific GEi in several Slave Nodes. As an example, one can consider that the same GEi (presenting the same service) has been deployed in two separate FIWARE Lab nodes. How one will be able to identify that the same GEi is deployed in these two nodes? The first option is to use the name of the VM that the user gave during deployment, but this option is not applicable in this case since the user is free to use any name. The second option is to correlate the same GEi (service) by retrieving the image reference identifier, but again this option is not appropriate since in a federated environment, each OpenStack Glance module might use its own image identification numbering scheme. So, since the options so far do not solve this problem, DCA, leveraging on Pegasus functionality, followed a different approach by inserting a specific value (called NID) in the Glance metadata that uniquely and unambiguously allows for GEi identification across the cloud federation. The interested reader may refer to [31].

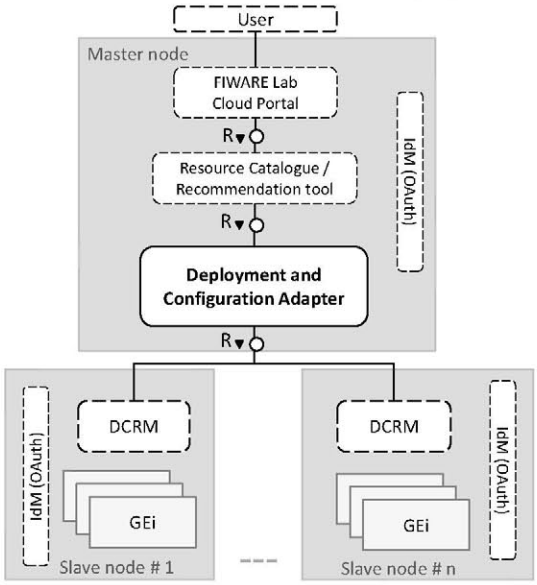

Figure 3 - Deployment and Configuration Adapter architecture.

Collectively, DCA is a flexible component that accommodates different accessing policies, following infrastructure owners' requirements, while it provides all needed information to the end-users. Additionally, DCA exposes a well-defined RESTful API [32] that can be used by interested users to collect all needed information regarding GEis available in the cloud federation and utilize them appropriately.

C. *Federation Monitoring*

Monitoring the performance of a single cloud infrastructure, both in terms of compute and network resources, is an important task in any standard data center. Such activity allows administrators to check the current status of the infrastructure, determine where and when a fault occurred in order to resolve it, and even prevent from forthcoming (undesired) events.

Normally, such monitoring activity is performed by a private Network Management Systems (NMS) [33] that constitutes an integrated vertical solution. However, it cannot be assumed that each single infrastructure of the federation leverages the same NMS. Since one of the main goals of the FIWARE Lab federation is to accommodate new infrastructures to offer their resources in the cloud federation, this requirement implies that the overall architecture —and the monitoring in particular— must be legacy-compliant. Hence, the establishment of a unique NMS is not a recommended approach.

In order to accomplish a unified and scalable monitoring framework capable of spanning multiple cloud infrastructures within the FIWARE Lab federation, authors in [20] propose an extended architecture where two layers are embedded in the traditional single-domain monitoring approach.

Attached to the private monitoring systems configured by each cloud infrastructure administrator, a cross-domain adaptation mechanism, denoted as Infrastructure Monitoring Middleware (IMM), standardizes —through a collection of common Application Programming Interfaces (APIs) —the format of and the accessibility to the data collected from the multiple probes and/or systems beneath. Such data reveal the performance of the compute and network-based resources of the different infrastructures involved. Hence, such abstraction layer establishes the basis of the common monitoring data model for the whole federation. In addition, the IMM instances include a distinctive feature which is not feasible with isolated NMSs; a component capable of determining the inter-connectivity status among the federated nodes. Such functional block is not within the scope of this document, but a more detailed description can be found in [34].

On top of this abstraction layer, the Federation Monitoring component fulfils the next operational layer in the enhanced architecture proposed. Such layer is in charge of storing and publishing the unified data-set by defining a Federation Monitoring API [35]. This layer is able to elaborate the data by leveraging on Big Data analysis techniques and providing aggregation features.

This Federation Monitoring does not represent a stand-alone system but a compendium of distributed modules as illustrated in Figure 4. As a matter of fact, each single node hosts a part of the Federation Monitoring system, as well as the IMM, but only the above-mentioned Master Node hosts specific federation-aware functionalities.

In the sequel, we briefly describe the components comprising the federation monitoring solution adopted towards the realization of FIWARE Lab:

- Context Broker (CB): FIWARE Lab's Context Broker is based on the Generic Enabler implementation of Orion Publish/Subscribe CB [36]. This component enables publication of context information by producer entities via standard interfaces, so that published context information becomes available to other consumer entities. In [34] authors address how a specific adapter within the IMM is meant to be in charge of standardizing raw monitoring data into NGSI context format, and notifying such information into the Context Broker. A dedicated CB instance shall be deployed in each node of the federation to handle the data-set associated to such domain, and update the metrics into the specific Hadoop instance. Nevertheless, this CB also requires to be linked with the instance deployed in the Master Node when real-time data are requested by the API Server.

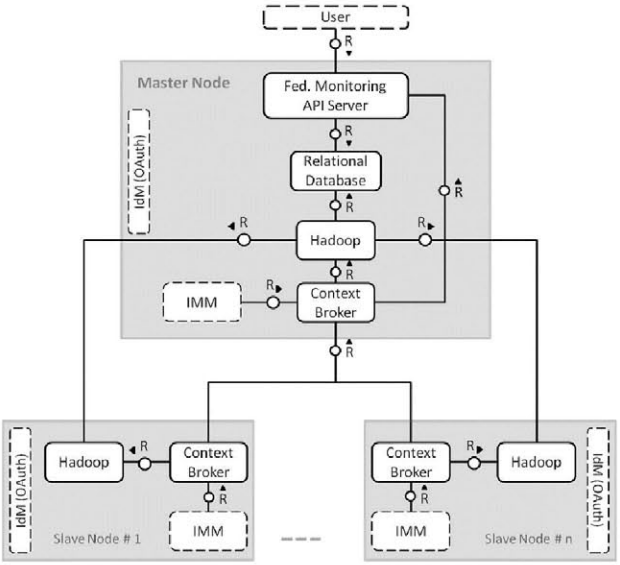

Figure 4 - Federation Monitoring architecture.

- Hadoop: Apache Hadoop [37] is a framework that provides scalable, reliable and distributed data processing and storage, designed to span large infrastructures. By distributing storage and computation across many servers, the combined storage resources can grow with demand while remaining economical at every size. Rather than rely on hardware to deliver high-availability, the system itself is designed to detect and handle failures at the application layer. Every FIWARE Lab node will host a full-fledged Hadoop deployment, which includes the Hadoop Distributed File System (HDFS) providing high throughput access to application data, and MapReduce for processing vast amount of data in a reliable and fault-tolerant manner. Each one of these Hadoop instances shall perform aggregating operations to the data coming from the subscription with the Context Broker, and all of them will be federated in order to maintain the service in High Availability. However, only the Hadoop deployed in the Master Node is allowed to perform operations with the relational database.
- Relational Database: A relational database is required to store the elaborated data provided as previously described. The API Server cannot afford, in terms of delay, to fetch the data from all over the federated Hadoops. Hence, this relational database (e.g. MySQL in the case of FIWARE Lab) is meant to store the already processed data-set and provide it in an easier and faster manner to the server.
- Federation Monitoring API Server: implementation of the API [35] for accessing, from a user-driven layer in the FIWARE Lab architecture, the processed monitoring data stored in the relational database, and real-time data from the federation. These resources are defined by a common monitoring data model and point, among others, to VMs, physical hosts and network elements. These API are capable of performing diverse operations, ranging from listing all the objects in a given node to retrieve the attributes of an inter-domain connectivity link.

IV. Managing Resources in a Multi-Tier Application: A Real-Case Scenario

This section presents the functionalities provided by the architecture discussed in the previous session through a real-case scenario.

Consider an application developer willing to offer a weather service by allowing subscribed users to check, via a website, weather conditions or be automatically informed of sudden weather changes. Weather information are collected through a network including several environmental sensors. To minimize the development and deployment effort, the developer browses the FIWARE Lab Cloud Portal to discover possible GEis that may assist him to create the weather service: he selects the Orion Context Broker (Orion CB) [36] to publish the data collected by the sensor; the Complex Event Processing (CEP) GEi to process the information from sensors and to create events through customized threshold; and finally the WireCloud GEi to properly display this information in a webpage, through the use of specific widgets.

Both WireCloud and CEP implementations are based on Ubuntu OS, while Orion CB is based on CentOS. Considering the architecture depicted in Figure 1, in order to reduce data processing latency and to comply with national regulatory constraints, the developer decides to deploy WireCloud on a VM in Slave Node 1, and Orion CB as well as CEP on two VMs in Slave Node 2 (two-tier, two-infrastructure application). Finally, the developer decides to deploy WireCloud and CEP through Glance images, and Orion CB using Chef [36] or Puppet [39] recipes.

When these three GEis are deployed in the respective infrastructures within the FIWARE Lab cloud federation, DCA becomes aware of the NIDs of these deployments, being able to unambiguously identify each GEi. However, there is a difference between deployment through images and using Chef recipes. In the former case, the NID of each GEi is pre-configured during the process of populating the Glance module of each infrastructure with the GEis, using the following command:

```
$ glance image-create --name cep-image-R2.3 --
disk-format qcow2 --container-format ovf --size
4028891136 --min-disk 0 --min-ram 0 --is-public
True --is-protected False --property nid=146 --
file <name of the file of the corresponding CEP
image>
```

The same command is used by the infrastructure administrator for each image in the Glance. In this perspective, NID value is stored for each GE image (Table 1) and can be retrieved by DCA, as explained in Section III. In the latter case, after the instantiation of the VM using a recipe, Pegasus is using the metadata service of OpenStack to introduce the NID metadata of Orion CB.

Table 1 - List of NID's per GEi.

GEi	NID
Orion Context Broker GEi	344
WireCloud GEi	513
Complex Event Processing (CEP) GEi	146

Each GEi deployment, either through images or recipes, embeds a pre-installed monitoring component which is part of the Infrastructure Monitoring Middleware (IMM) that caters for automatically publishing performance data. Thus, once the GEis under consideration are operational, the developer may wish to inspect the current performance of a specific VM —for example the one hosting the WireCloud— to check whether such resource of the federation works in a proper manner. In order to accomplish this task, the developer may request from DCA to provide all VMs created by a specific user with the following command [32]:

```
GET http://ip-
address:port/dca/servers/user/{user_id}
```

By issuing this command, the developer will be informed about all VMs that he has developed, together with all related information as shown below:

```
  [{
      "id":"6fda6be8-0e70-4d08-9731-
c858d6a27f50",
      "name":"CEP",
      "nid":"146",
      "imageRef":"90d4865d-5e7b-4d95-af2c-
69753e1740d6",
      "flavorRef":"2",
       ...
      "created":1390051880000,
      "tenantId":"xxxxxx",
      "userId":"john-smith",
      "region":"slave_node_2"
   },
   {
      "id":"f85c7ce6-e125-4d90-b5dd-
134eae56fc5a",
      "name":"WireCloud",
      "nid":"513",
      "imageRef":"82d4877f-4c7b-4a45-a222-
62643e1110e5",
      "flavorRef":"2",
       ...
      "created":1230151137000,
      "tenantId":"xxxxxx",
      "userId":"john-smith",
      "region":"slave_node_1"
   },
   {
      "id":"135755e3-e378-6a63-b5ef-994e4e22c5a",
      "name":"Orion",
      "nid":"344",
      "imageRef":"22d4337f-4d1a-8cca-a885-
24412d1991a7",
      "flavorRef":"2",
       ...
      "created":1230151155000,
      "tenantId":"xxxxxx",
      "userId":"john-smith",
      "region":"slave_node_2"
   }]
```

Hence, the developer can identify the unique identification number of the VM hosting WireCloud software and retrieve the monitoring information of this appliance by leveraging the Federation Monitoring API [35], e.g.:

```
GET
/monitoring/regions/slave_node_2/vms/f85c7ce6-
e125-4d90-b5dd-134eae56fc5a
```

The response obtained includes some relevant performance attributes, such as the memory utilization, free hard disk drive and processor load:

```
{...
"regionid": "slave_node_1",
"vmid": "f85c7ce6-e125-4d90-b5dd-134eae56fc5a",
"ipAddresses": [
      {
         "ipAddress": "192.168.0.70"
      }
 ],
 "measures": [
     {
        "timestamp" : "2014-06-20 12.00",
         "percCPULoad": {
            "value": "25",
            "description": "Current percentage of
CPU Load"
         },
         "percRAMUsed": {
            "value": "50",
            "description": "Current percentage of
RAM consumed"
           },
           "percDiskUsed": {
             "value": "10",
             "description":  "Current  percentage
of Disk consumed"
           },
...}
```

A wider plethora of monitoring data can be retrieved from the Federation Monitoring API, although not shown in this example due to space limitation. Once the deployment, configuration and monitoring of the components are completed, the weather service can be advertised through the FIWARE Lab Cloud Portal and rated from users.

V. Conclusions and Future Work

There are several challenges in the development of a cloud federation to support generic services to be used by developers. In this paper, we addressed some solutions to overcome the seamless deployment of services across the federation and ability of services to span across different members of the federation, the monitoring of the resources and data which can be aggregated with a common structure, and of course to be offered as an open ecosystem for innovation at the developers' disposal. The presented modules and the architecture allow the management of the resources of the federation and their control, a key element in the federated cloud we are presenting in the paper. To deploy multi-tier applications across the federated platform is one of the cases presented, which is supported by the proposed architecture and modules, such as the federated monitoring as a centralized mechanism to handle large-scale monitoring data gathered throughout the federated underlying resources, Pegasus as a specialized PaaS, or the deployment and configuration adapter. In the future it is planned to increase the number of cases where we apply these modules and improve the capabilities to offer an improved federated cloud to offer Future Internet services useful for developers.

References

[1] H.W. Chesbrough, "Open Innovation: The new imperative for creating and profiting from technology", Harvard Business Press, Boston, 2003.

[2] IBM, Catalyst Startup Program. http://www.softlayer.com/catalyst.

[3] A. Celesti, F. Tusa, M. Villari, A. Puliafito, "How to Enhance Cloud Architectures to Enable Cross-Federation," in Proceedings of the 3rd IEEE Intern. Conf. on Cloud Computing, pp.337-345, July 2010.

[4] XIFI EU Project, https://www.fi-xifi.eu/home.html.

[5] FIWARE Lab, http://lab.fi-ware.org.

[6] FIWARE, http://www.fi-ware.org.

[7] FIWARE Lab Federation Members: https://www.fi-xifi.eu/about-xifi/federation-members.html.

[8] IEEE P2302 - Standard for Intercloud Interoperability and Federation (SIIF), Online: oasis-open.org/committees/download.php/46205/p2302-12-0002-00-DRFT-intercloud-p2302-draft-0-2.pdf.

[9] OpenStack Federated Keystone, https://wiki.openstack.org/wiki/Keystone/Federation/Blueprint.

[10] A. N. Toosi, et al., „Interconnected Cloud Computing Environments: Challenges, Taxonomy, and Survey", ACM Computing Surveys (CSUR), Vol. 47, Issue 1, July 2014.

[11] M. P. Papazoglou, "Cloud Blueprints for Integrating and Managing Cloud Federations", Software Service and Application Engineering, Springer LNCS, Volume 7365, pp. 102-119, 2012.

[12] D. Villegas et al., "Cloud federation in a layered service model", Journal of Computer and System Sciences, Vol. 78, Issue 5, pp. 1330–13, 2012.

[13] E. Carlini, et al., "Cloud Federations in Contrail", in Proceedings of the Euro-Par Workshops (1), 2011, pp.159-168.

[14] EGI Federated Cloud, https://www.egi.eu/infrastructure/cloud.

[15] A.C. Hume, et al., "BonFIRE: A Multi-cloud Test Facility for Internet of Services Experimentation", in Proceedings of the 8th Intern. ICST Conf., TridentCom 2012, Thessaloniki, Greece, pp. 81-96, June 2012.

[16] C. Liu, et al., "Cloud Resource Orchestration: A Data-Centric Approach", in Proceedings of the 5th Biennial Conference (CIDR'11), Asilomar, California, USA, pp. 241-248, January 2011.

[17] E. Elmroth and L. Larsson, "Interfaces for Placement, Migration, and Monitoring of Virtual Machines in Federated Clouds", in Proceedings of the 8th IEEE International Conference on Grid and Cooperative Computing (IEEE GCC), pp. 253-260, August, 2009.

[18] W. Vandenberghe, et al., "Architecture for the Heterogeneous Federation of Future Internet Experimentation Facilities", Future Network and Mobile Summit, Lisbon, Portugal, July 2013.

[19] K. Kavoussanakis, et a.. "BonFIRE: the Clouds and Services Testbed", in Proceedings of the 5th IEEE Intern. Conf. on Cloud Computing Technology and Science, Bristol, UK, pp. 321-326, December 2013.

[20] Y. Al-Hazmi, et al., "Unified Representation of Monitoring Information Across Federated Cloud Infrastructures" in Proceedings of the Workshop on Federated Future Internet and Distributed Cloud Testbeds (FIDC), Karlskrona, Sweden, September 2014.

[21] OCCI, http://occi-wg.org/2012/07/18/occi-in-openstack.

[22] XIFI Components - http://wiki.fi-xifi.eu/Public:Software_Components

[23] Pegasus, http://catalogue.fi-ware.org/enablers/paas-manager-pegasus.

[24] OpenStack Metadata Service. http://docs.openstack.org/admin-guide-cloud/content/section_metadata-service.html.

[25] T. Dingsøyr, T. Dybå, N.B. Moe, "Agile Software Development. Current Research and Future Directions", ISBN: 978-3-642-12574-4 (Print) 978-3-642-12575-1 (Online), 2010.

[26] M. Sacks, "DevOps - Pro Website Development and Operations", "DevOps Principles for Successful Web", pp. 1-14, 2012.

[27] Cloud-Init, http://cloudinit.readthedocs.org/en/latest.

[28] Sagitta, http://catalogue.fi-ware.org/enablers/software-deployment-configuration-sagitta.

[29] D. Spinellis, "Don't Install Software by Hand", IEEE Software, Vol. 29, Issue 4, pp. 86-87, July-August 2012.

[30] Google Andromeda, http://gigaom.com/2014/04/02/google-launches-andromeda-a-software-defined-network-underlying-its-cloud.

[31] Deployment and Configuration Adapter - http://wiki.fi-xifi.eu/Public:Deployment_and_Configuration_Adapter.

[32] DCA API - http://docs.dca.apiary.io.

[33] R. Khan, et al, "An Efficient Network Monitoring and Management System", International Journal of Information and Electronics Engineering, Vol. 3, No. 1, January 2013.

[34] J. Gonzalez, et al., "Inter-domain Monitoring and Software-Defined Network Connectivity for Federated Infrastructures Management", The IEE EuCNC, Bologna, Italy, June 2014.

[35] FIWARE Lab Fed. Mon. API, http://docs.federationmonitoring.apiary.io

[36] Orion Context Broker- http://catalogue.fi-ware.org/enablers/publishsubscribe-context-broker-orion-context-broker

[37] Apache Hadoop. http://hadoop.apache.org.

[38] Chef, http://www.getchef.com.

[39] Puppet, http://puppetlabs.com.