

# Contrail Virtual Execution Platform Challenges in Being Part of a Cloud Federation<sup>\*</sup>

Piyush Harsh, Yvon Jegou, Roberto G. Cascella, and Christine Morin

INRIA Rennes - Bretagne Atlantique,  
Campus Universitaire de Beaulieu, 35042 Rennes, France  
Email: {piyush.harsh, yvon.jegou, roberto.cascella, christine.morin}@inria.fr

**Abstract.** Cloud computing is quickly defining the computing paradigm in the modern networked age. Users can run their large computations online using cloud services at a fraction of the cost compared to setting their own data centers. Clearly cloud computing offers many advantages, and yet many large organizations including governments, financial sector, and health care sector are reluctant in transitioning to cloud computing. Contrail project will address the major concerns behind this reluctance namely mistrust in cloud platforms, lack of Service Level Agreements (SLAs) and Quality of Protection (QoP) of data. Contrail will provide a federation layer support for bringing a multitude of cloud providers, both private and public, together. This will allow multi-tenancy and cloud-bursting capability to end user cloud applications while supporting SLAs and QoP agreements desired by several privacy aware sectors including governments, banks, health care providers to name a few. This paper describes the novel features we are building into the Contrail Virtual Execution Platform (VEP) that will be closely interfaced with the IaaS layer of cloud providers. VEP upgrades the supported cloud providers and brings trust in cloud computing by adding SLAs and QoP features missing at typical IaaS layer. Further this paper outlines challenges faced in being part of a large federation and how VEP will address some of those.

## 1 Introduction

Cloud offerings are becoming more mature with increasingly sophisticated services available to the consumers. End users and corporations have tools available today that enable them to configure and operate their own private cloud services. These tools are available as commercial products, e.g., VMWare's vCloud [13], as well as open source IaaS alternatives such as OpenNebula [11] and OpenStack [12]. In addition, there are public clouds that operate on pay-as-you-go model such as Amazon's EC2 [1], Microsoft's Azure [5], and Google AppEngine [4] into which consumers can launch their applications and services as needed.

---

<sup>\*</sup> This work is supported by the FP7 257438 Contrail integrated project funded by the European Commission.

The above scenario is definitely a huge improvement from traditional data center models of last decade. Even though clouds offer much better price/performance ratio over non-virtualized environments, the true power of cloud computing can only be achieved with seamless federation among different cloud technologies.

A true cloud federation will bring both private and public clouds under the umbrella of one federation where users will have the option to deploy services using resources of multiple providers. The federation will present to the user uniform API/billing/monitoring features regardless of the nature of actual cloud service providers. Moreover, any organization can become part of this federation being both a cloud provider, when its IT infrastructure is not used at its maximal capacity, and a cloud customer in periods of peak activity. Resources that belong to different operators can be integrated into a single homogeneous federated cloud, shown in Figure 1, that users can access seamlessly.

The Contrail project ([www.contrail-project.eu](http://www.contrail-project.eu)) aims to achieve these goals by developing an integrated approach to virtualization, offering Infrastructure as a Service (IaaS), services for federating IaaS clouds, and Platform as a Service (PaaS) on top of federated clouds. This calls for the deployment of a transparent, trusted, and reliable *Contrail federation* with operations governed by Service Level Agreements (SLAs) providing support for strong Quality of Protection (QoP) and authentication.

No cloud implementation, commercial or open source, currently implements all features to be supported by Contrail: federation, IaaS as well as PaaS, security, quality of service and protection enforced by service level agreements. Open source cloud platforms such as OpenStack [12], Nimbus [6] or OpenNebula [11] only support the IaaS layer with no quality of service and protection guarantees. However, some of these platforms such as OpenNebula partially support cloud federation.

Full PaaS support is provided by some commercial products such as VMware vCloud<sup>TM</sup> [13] through the OVF distributed application open format [10]. But these platforms are currently limited to private cloud management only.

A *Contrail federation* integrates both private and public clouds under a common umbrella. User identities, data, and resources are interoperable within the federation, thanks to common supports for authentication and authorization, and to common policy definition, monitoring, and enforcing mechanisms (SLA, QoP, etc.), as well as to a common economic model [14].

The way the Contrail federation is conceived and its open source deployment nature will enable seamless access to provider resources, avoiding potential vendor lock-in for the end users. These users can now deploy their distributed applications on demand on different providers by negotiating resources with the federation, which will choose the best providers based on the application requirements.

In such a scenario, Contrail technology is able to satisfy the user needs for the deployment of elastic and salable applications guaranteeing performance dependability. The run time performance of the application can scale with the number of assigned resources, the user only needs to specify the desired perfor-

mance level and the Contrail's automatic provisioning system can add resources on demand during the execution of the application in order to guarantee the desired performance (thus providing elasticity).

A challenging aspect of Contrail is the management of computing resources to guarantee this performance dependability and QoS of active virtual execution platforms. Indeed, cloud providers might have different efficiency policies given that the individual resources being contributed to this federated cloud will be highly heterogeneous in their hardware configuration and system-level organization, thus making their interoperability challenging.

The support for efficient cooperation and resource sharing within cloud federation is critical for the Contrail technology. This is achieved through Contrail Virtual Execution Platform (VEP), an open source technology implementing standards that exploits resource virtualization to provide virtualized distributed infrastructures for deployment of end-user applications independently from the underlying platform. VEP operates above IaaS layer as shown in the figure 1. In Contrail, each cloud provider will run a copy of VEP software which in turn will seamlessly integrate its resources with the Contrail federation.

VEP offers reliable application deployment platform that is resilient to operational failures and which supports secure management of user data with a strong guarantee for QoS. These requirements are critical to make the provider resources trustworthy so that users can obtain performance guarantees of their application, bring trust in cloud computing solutions making them suitable to run the users' businesses safely.

This paper presents the detailed information on VEP, which is one of the components of Contrail service stack deployed at the cloud resource providers end in order to seamlessly integrate the resources under Contrail federation.

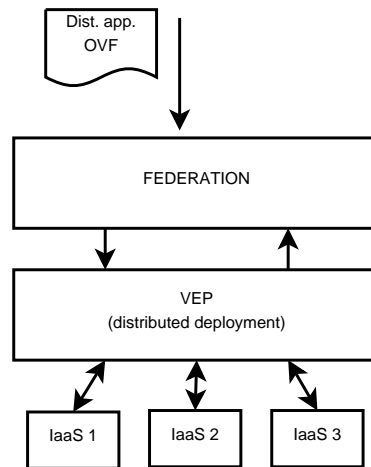
The remainder of this paper is organized as follows - section 2 outlines the key challenges inherent in a large cloud federation, section 3 describes the role of VEP in cloud computing, section 4 describes the components and the design rationale of the Contrail VEP development effort, and finally section 5 describes the Contrail progress report, VEP development road map and recap of important contribution and key points presented in this paper.

## 2 Challenges of a large Federation

In a federation many independent members operate together governed by the agreed upon federation laws. Just like in a society where a federation is successful only if the individual members operate properly, the same set of challenges and opportunities exist in a federated cloud. In this section we will list few major issues that may arise in a federation of clouds. Later on we will see how VEP helps address some of these concerns.

### 2.1 Authenticity of reported statistics

An end user of a federation of clouds may not know a priori which provider her application will run on. The user agrees to pay the resources used by her application and in turn may require some form of validation of the reported resource



**Fig. 1.** A simple cloud federation

usage. The provider may fudge the reported statistics in order to fleece the end users. It would be the task of the federation to ensure such a case does not arise. Clear code of conduct and penalties for violation must be defined. Even then, the enforcement could be problematic. Surprise audits of provider logs by federation officials could be one deterrent against frauds by providers. But, the logs themselves may be modified. Fool-proof implementation of anti-fraud measures may be impossible to achieve. Further, for a successful federation, the terms of participation by individual provider must not be too stifling and must not favor large providers against small providers. Finding the correct participation terms and condition balance may prove a key for the success of the federation and yet this may be the most difficult balance to achieve.

## 2.2 Verification of SLA adherence

Whenever there is a contract agreed between two parties, the contract enforcement becomes a legally binding obligation. Service Level Agreements (SLAs) in the context of cloud computing could contain agreements on the operating environment and placement restrictions of virtual machines (VMs) at the provider's location. An example SLA object could contain condition that some VMs in an application must not be placed on the same cluster rack as others. There might be CPU load conditions, memory requirements to name a few SLA parameters that could be present.

In a large federation where negotiated SLAs are supported between the user and the provider, the verification of the SLA contract adherence could prove to be challenging. If the user deploys her own VMs as part of the application, the user could embed her own verification code to periodically check if the execution environment is within the negotiated SLA parameters or not. The provider must maintain adequate logs for auditing if need arises.

Typically, initial SLA adherence is not a major issue because the provider checks for locally available resources before accepting an OVF and corresponding SLA object for provisioning. If adequate resources are not available at deployment stage, the SLA contract and OVFs are rejected. Ideally, such rejection should take place at the federation level itself before the deployment request reaches a provider. But the SLA enforcements for dynamic characteristics such as CPU usage should best be done by SLA enforcement agent based on periodic monitored data.

### 2.3 Scalability of interface APIs

A single private cloud operator if operating by itself may not have huge customer base, but becoming part of a federation could suddenly change that. The provider APIs throughput that were once sufficient to deal with influx of requests from non-federation customers may prove a serious bottleneck if not properly provisioned for the large federations. As an example, if the provider supports REST interfaces, it would be advisable to stress test the REST APIs and check for scalability and other issues of being part of a large cloud federation. In Contrail, since VEP will be at the interface between the provider and the federation layer, all federation request will first come at VEP and therefore the VEP REST and other APIs will be stress tested for sudden influxes from a large federation customer base.

### 2.4 Authorization/Authentication nightmare

Authorization and authentication are required whenever a provider resource needs to be accessed. Primarily it is required to keep track of resource usage for billing and accounting purposes, but also for keeping the compute and storage resources secure from malicious access. Private providers can easily maintain user accounts for authentication/authorization of their users. But once being part of a larger federation, it is not possible to keep account details of all possible users. The provider now must support means to allow resource access to federation users and at the same time keep at bay the malicious users.

One solution is to adopt a two-tier approach and keep the local authentication system intact for the non-federation users, and incorporate a federation authentication module in order to be able to validate the federation users' credentials before allowing access to local resources to federation users.

Other solution could be to elevate all local users as federation users, incorporate local accounts into the federation and provide all local users their new federation account details. But in this approach, the provider risks to lose the business of local users to competing providers under the federation while gaining the simplicity of one unified authentication mechanism.

Regardless of what solution one adopts, there are bound to be headaches in the integration process with respect to users' authentication and authorization with the (likely heterogeneous) providers in a large cloud federation.

## 2.5 Issue of trust in the federation

A federation of cloud providers brings several disparate cloud providers under one umbrella. Every cloud provider has different guiding principles and motives. The issue of trust in the federation is very important for proper operation of the system. The cloud providers must trust the federation. They have certain expectations from the federation, they expect the federation to properly filter out malicious users, they expect the federation to be honest in its accounting and billing practices. In the similar manner, the federation expects the participating cloud providers to adhere to a minimum code of conduct, be truthful in reporting resource usage, properly honor any negotiated SLA contracts, etc. The federation and cloud providers can both maintain some sort of trustworthiness rating system. It need not be uniform across all participating parties, each provider can adopt any suitable rating system (or not).

## 3 Need for VEP in cloud computing

With the growing importance of cloud based services in the computing of tomorrow, it is only a matter of time when there will be a real need for merging disparate IaaS platforms in order to support increasingly complex applications of the future.

The Virtual Execution Platform (VEP) could provide that aggregating glue for bringing different cloud platforms under one umbrella. The standard APIs exported by the VEP software would allow other developers to quickly and easily develop tools for making their own federated platforms tuned for specific tasks that other federations including Contrail may not support.

There are several open source tools already available for accessing public clouds (Amazon EC2), integrating such tools together with VEP would allow quick and easy solution for bringing public and private clouds together. Thus a well designed VEP solution could usher in the next wave of cloud research and open up new vistas for cloud innovations. Keeping this motivation for developing a quality VEP, let us next look at Contrail's VEP design in detail.

## 4 Contrail Virtual Execution Platform

Virtual Execution Platform (VEP) is a Contrail service that sits just above IaaS layer at the service provider end of the Contrail cloud federation. Some major design goals were kept in mind while designing the VEP architecture. Support for open virtualization standards including Open Virtualization Format (OVF) has been one of the major requirement. Scalability of the API interface in order to support potentially large customer base, interoperability with various open IaaS technologies in order to bring several kinds of cloud providers under the Contrail umbrella, and providing support for elasticity in accordance with the negotiated SLAs have been other major requirements that VEP will satisfy.

It will have selectable built in drivers for various open source IaaS platforms such as OpenNebula and OpenStack. The VEP layer will integrate clouds composed using different technologies under the ambit of Contrail federation. VEP will be very closely integrated with the IaaS layer and will expose uniform APIs to higher layers in the Contrail architecture. It will enable seamless deployment of OVF applications over any underlying supported cloud platform.

In case the federation module suspects that none of the participating Contrail cloud providers can satisfy a user's application resource requirements, it may enable part-deployment of the appliance over public clouds such as Amazon EC2 (cloud-bursting) using user credentials for such public clouds. This capability will be handled by the federation layer and not VEP directly.

Since support for Service Level Agreements (SLAs) and providing Quality of Protection (QoP) to end users is a major feature that Contrail aims to provide, the VEP layer has also been designed keeping the larger Contrail requirements in the picture. The following subsections describe major VEP components and supported features in some detail.

#### 4.1 OVF deployment subsystem

Open Virtualization Format (OVF) [10] is an open standard that is used to compose virtual applications. It allows inclusion of multiple VM templates, allows description of interconnect between VMs, individual VM's contextualization to name a few features provided by the OVF standard. It has been developed by the Distributed Management Task Force (DMTF) [3]. It is an up-and-coming standard with increasing vendor support.

VEP will provide full support for OVF v 1.1 standard [15]. The support for OVF in VEP will enable deployment of virtual applications described in OVF XML document over supported IaaS platforms regardless of their native support for OVFs.

OVF deployment subsystem in VEP is responsible for deploying user appliances described within OVF XML document. Figure 2 shows the simplified design of the VEP's OVF deployment module. The user's OVF file is received from the federation layer using REST protocol (refer figure 2 subcomponent a). The user's credentials and authentication parameters are sent as part of HTTP headers in the Contrail REST scheme. Once the request for OVF deployment has been verified, the OVF document along with the corresponding SLA object is sent to the OVF deployment subsystem of VEP stack (refer figure 2 subcomponent b).

Once the OVF document and the corresponding SLA object is received at the VEP layer, the OVF document is validated for correctness against DMTF OVF standard schema document (currently VEP supports OVF v 1.1)(refer figure 2 subcomponent c). If the OVF document fails the validation process, the deployment effort is terminated and a suitable response is reported to the federation layer.

Upon successful validation, the OVF document is parsed (refer figure 2 subcomponent d) and individual elements deconstructed. The deconstructed OVF

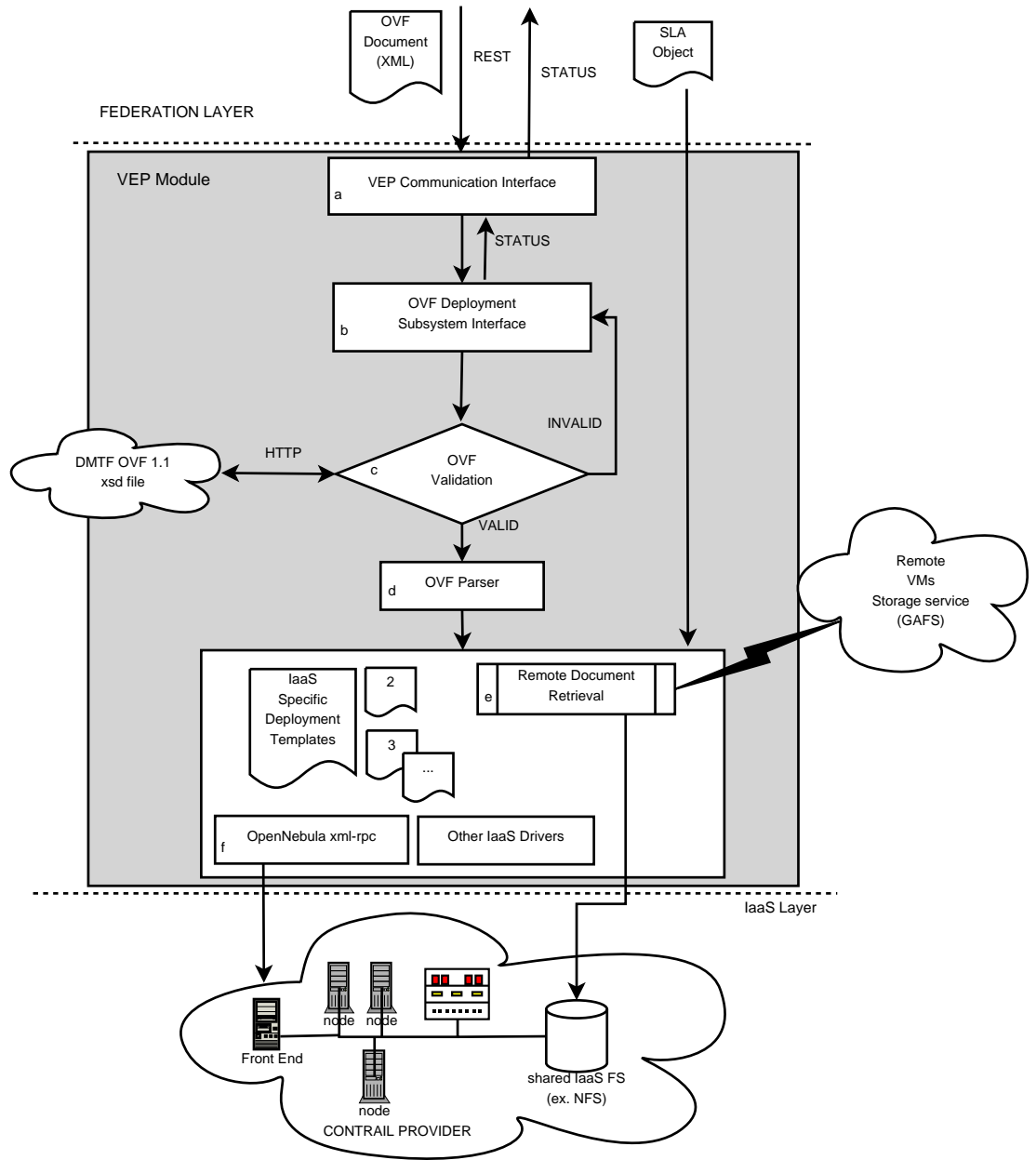


Fig. 2. OVF subsystem in Contrail VEP (simplified version)

components including VM elements, network elements etc. are then linked to their corresponding SLA contracts deconstructed from the SLA object. These linked elements are then used as input in the IaaS translation function which, depending on the type of IaaS environment, generates the template/deployment files for the underlying IaaS platform. For the initial release, VEP aims to provide support for OpenNebula IaaS clouds (refer figure 2 subcomponent f). Later on support for other open source cloud technologies will be developed.

If a virtual machine is stored at a remote location, the OVF deployment subsystem has modules for authenticated remote file transfer (refer figure 2 subcomponent e). The remotely hosted resources (VMs) will be transferred onto the cloud provider's shared file system before the actual virtual appliance deployment takes place at the cloud provider. The Contrail project will incorporate a Global Autonomous File System (GAFS) based on the XtremOS project. The users' virtual machine image files will be hosted at remote GAFS shares and the OVF subsystem will log on behalf of the user and transfer image files from remote GAFS locations to the cloud provider shared file system.

#### 4.2 REST APIs for other Contrail modules

REST [16] stands for Representational State Transfer and is a simple yet elegant way to support API development. Using REST, one can provide simple CRUD - Create, Read, Update, and Delete operations over identified resources. REST provides a very clean way of developing APIs and enables quick adoption from cloud developers. Since we desire VEP to be easily integrable in other projects, VEP will provide clear REST APIs for developers. In addition VEP will have clearly defined RESTful APIs for other contrail modules to communicate with it. All requests from other modules including federation layer to VEP layer will be made using REST. The details of the APIs are beyond the scope of this paper. The REST APIs are built over Hyper Text Transfer Protocol (HTTP) protocol messages with support for HTTP GET/PUT/POST/DELETE messages only. Not all methods are available for all users and /or resources. Depending on the nature of the request and the access rights of users' a few or all methods are made available. The VEP REST interface will be stress-tested for scalability.

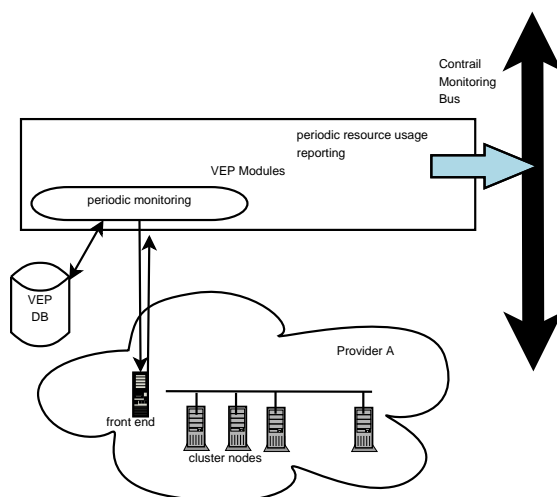
#### 4.3 OCCI support

Open Cloud Computing Interface (OCCI) [8] is a set of open standards [7] for managing cloud resources that have been developed through Open Grid Forum (OGF) [9]. The VEP component in the Contrail project will have an extended OCCI support. Again the need to incorporate OCCI in VEP arises from our requirement to be extensible and integrable in other projects. Additionally, as a result of our work and the experiences gained in providing support for OVFs and SLAs, Contrail project will propose an extension to the current OCCI draft for inclusion in the standardization effort within the OCCI community. These extensions will allow the OCCI framework to support OVF deployments along with providing support for SLAs which the standard lacks in its current format.

The support for OCCI in VEP will hopefully allow the platform to be integrated in future cloud projects and will allow the open source community to develop feature enhancements and add-ons to the VEP layer in the federation of clouds.

#### 4.4 Application monitoring and reporting module

Application monitoring subsystem is a very integral part of the VEP as it provides support for the proper enforcement of the SLA contracts. Every application that is deployed on provider's IaaS infrastructure is monitored by the VEP module periodically. The collected data is stored in the VEP database for on-demand retrieval on user's behalf by the federation, but few statistics are pushed periodically on the monitoring bus to be used by the SLA enforcement module to check for compliance and for making elasticity decisions. Figure 3 shows the simplified



**Fig. 3.** Reporting subsystem in Contrail VEP (simplified version)

schematic of the monitoring subsystem in the VEP module.

Each deployed application is assigned a Universally Unique Identity (UUID) that is assigned by the Contrail federation. The same UUID is included as part of the reported data that helps various modules in the SLA/monitoring system in the federation link the data with the correct application and the negotiated SLA object for verification. The federation monitoring bus over which the periodic data is sent will be a distributed messaging system (example: Apache ActiveMQ [2]) that supports Java Messaging Service (JMS) APIs.

#### 4.5 IaaS drivers

The VEP software is envisioned for use with multiple open source IaaS clouds. Contrail VEP software eventually plans to support several major IaaS platforms including OpenNebula, OpenStack, Nimbus scientific cloud [6], etc. We will provide appropriate drivers that can be enabled by the cloud providers as desired for interfacing with appropriate IaaS platforms.

#### 4.6 Federation authentication/authorization support

The VEP layer helps a cloud provider become part of the larger Contrail cloud federation. The VEP layer handles all communication to/from the cloud provider and the federation. All requests on behalf of the federation user to the provider comes from the federation layer. Each such request has to be properly authenticated and only authorized requests should be forwarded to the IaaS layer. This check is necessary to prevent resource abuse and is also required for accounting and billing purposes.

Since VEP communicates to federation through REST APIs, each request carries with it the user authorization credentials embedded in the HTTP headers. User certificates and a trusted chain of certificates are used for verification of authorization credentials. The federation maintains the registered user accounts and the resource access rights and other details. The VEP is only interested in finding whether the presented credentials are authentic or not. Once the credentials are verified, the provider's resource access request is forwarded to the proper IaaS cluster.

The VEP also maintains a temporary access control table for active users and their applications that have been deployed through itself. An additional check against this table may also be performed if deemed necessary against the type of access requested.

#### 4.7 Virtual Organization

The Contrail VEP supports the notion of a Virtual Organization (VO). The cloud provider / administrator will have the ability to configure the parameters of the virtual organization including list of hosts, networking connections, storage. The VO users' can submit OVF's for deployment making use of already provisioned VO resources. The billing for such resources will be sent to the VO and not the individual VO users. The VEP will support VOs even if the underlying IaaS platform does not support such a notion of organization. If a supported IaaS platform supports VOs natively (example OpenStack), VEP will make an effort to utilize the supported feature as much as feasible. Thus VEP will provide a uniform notion of VO to end users regardless of the underlying IaaS support of VOs.

## 5 Concluding remarks

In this paper, we presented a detailed design of Contrail VEP implementation and rationale for our design. We have discussed the Contrail cloud federation briefly. We have explored the major challenges any cloud provider would face, and must address if they plan to become part of a larger federation. Contrail VEP software proposed support for a broad set of open cloud platforms would help significant number of private cloud operators to seamlessly join Contrail federation. Further, VEP's support of open standards and well defined API set will enable cloud researchers and developers to easily extend its features to suit their future requirements and thus making VEP somewhat future proof.

VEP's full integrated support for Contrail federation authentication and authorization mechanisms, and fully stress tested REST and OCCI interface will address some of the challenges (see 2.3, 2.4) we outlined for being part of a federation. The open source mindset and access to the source code would allow independent verification of monitoring modules and other features and thus would help build some level of trust in VEP and Contrail federation (see 2.5).

The Contrail project road map is well defined, in the initial release the VEP effort will provide complete integration with OpenNebula IaaS platform. Support for other cloud platforms including OpenStack will follow in the subsequent releases. There will be basic scheduling support in the first release, but we plan to provide a full resource scheduling feature in the second release. A full support for OCCI will also be provided in subsequent releases. The first release will also see basic OVF support with more complete support in subsequent releases.

## References

1. Amazon Elastic Compute Cloud (Amazon EC2), <http://aws.amazon.com/ec2/>
2. Apache ActiveMQ, <http://activemq.apache.org/>
3. Distributed Management Task Force, Inc., <http://www.dmtf.org/>
4. Google App Engine, <http://code.google.com/appengine/>
5. Microsoft Azure, <http://www.microsoft.com/windowsazure/>
6. NIMBUS, <http://www.nimbusproject.org/>
7. OCCI Specification, <http://occi-wg.org/about/specification/>
8. Open Cloud Computing Interface, <http://occi-wg.org/>
9. Open Grid Forum, <http://www.ogf.org/>
10. Open Virtualization Format (OVF), <http://www.dmtf.org/standards/ovf>
11. OpenNebula.org - The Open Source Toolkit for Cloud Computing, <http://www.opennebula.org/>
12. OpenStack.org - Open source software for building private and public clouds, <http://www.openstack.org/>
13. VMware vCloud, <http://www.vmware.com/products/vcloud/overview.html>
14. Consortium, C.: Requirements on Federation Management, Identity and Policy Management in Federations (March 2011), Contrail Deliverable - D2.1
15. Crosby, S., Doyle, R., Gering, M., et. al.: Open Virtualization Format Specification. DMTF, 1.1.0 edn. (January 2010)
16. Fielding, R.T.: Architectural styles and the design of network-based software architectures. Ph.D. thesis (2000), AAI9980887