1 What is VEP?

2 Interoperability, Portability and Open Standards

3 VEP features

4 Status Report from Y3 and Roadmap

5 How to follow VEP’s progress?
Why VEP?

Our Requirements

- Open Source
- Uses Open Standards to promote interoperability
- Supports Service Level Agreements
- Supports multiple Open Source IaaS softwares

While in essence we wanted to create a service similar to vCloud, these requirements prompted us to create our own software.
What is VEP?

In short:

- A contrail independent middleware which allows a cloud provider to manage its datacenters.
- A software which can be used directly by the end user to manage his applications, providing support for Service Level Agreements between a user and the provider, with an appropriate monitoring to exploit the SLA.
- While able to work independently, it is also an important part of the Contrail Stack.
VEP in Contrail

Contrail federation

SLA management and enforcement

VEP: Virtual Execution Platform

IaaS
What is VEP?

Interoperability, Portability and Open Standards
- Open Virtualization Format
- Cloud Infrastructure Management Interface

VEP features
- Constrained Execution Environment
- Constraints
- Scheduler
- oAuth
- REST API
- REST GUI

Status Report from Y3 and Roadmap

How to follow VEP’s progress?
Challenges

- Each software provides its own API
- Resources are not represented in the same way
- About portability: How can we transfer applications between cloud providers?

How can we use open standards to solve these challenges?
Interoperability and VEP

VEP strives to become interoperable with 3rd party cloud tools by supporting open standards. Using an open standard to describe your cloud application further makes it portable.
A short overview of Cloud Standards

- Major cloud standardization bodies
  - OGF - Open Grid Forum
  - Distributed Management Task Force
  - Storage Network Industry Association

- Key Cloud Standards
  - OCCI - Open Cloud Computing Interface (OGF)
  - Open Virtualization Format (DMTF)
  - Cloud Infrastructure Management Interface (DMTF)
  - Cloud Data Management Interface (SNIA)

- De-Facto Cloud Standards
  - Amazon EC2 API
Open Virtualization Format - Overview

It is an Open standard by Distributed Management Task Force (DMTF), an industry non-profit organization. Includes big-wigs such as Intel, HP, IBM, Cisco, Vmware, Microsoft, US DoD etc.

With approval of major players in the cloud industry, it is more likely to succeed.
OVF - More details

- provides a standard way of describing a virtual application
- ability to describe a VM hardware specifications
- ability to specify the network and storage parameters
- individual VM contextualization support
- provision for controlling VM start-up order
- container description for a self-contained application in a single .ova package
- support for elasticity since the 2.0 update
Contrail Virtual Execution Platform
Interoperability, Portability and Open Standards
Open Virtualization Format

OVF - Example

First part

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://schemas.dmtf.org/ovf/envelope/1"
  xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
  xmlns:osn="http://schemas.dmtf.org/ovf/environment/1"
  xmlns:ovai="http://schemas.dmtf.org/ovf/environment/1"
  xmlns:cim="http://www.dmtf.org/schemas/CIM/1/CIM-schema-c15"
  xmlns:raed="http://schemas.dmtf.org/ovf/environment/1">
  <!-- References to all external files -->
  <References>
    <File ovf:id="ubuntu-1110" ovf:href="/var/lib/images/ubuntu_1110.img" ovf:size="180114672" />
  </References>
  <!-- Describes meta-information about all virtual disks in the package. This example is encoded as a delta-disk hierarchy. -->
  <DiskSection>
    </Info>
  </DiskSection>
  <!-- Describes all networks used in the package -->
  <NetworkSection>
    <Info List of logical networks used in the package>
      <Network name="test-net">
        <Description>Network used to link the web server node and the engine frontend node.</Description>
      </Network>
    </Info>
  </NetworkSection>
</Envelope>
```
OVF - Example part 2

Second part

```xml
<VirtualSystem ovf:id="Contrail_Test_Application">
  <Name>Contrail Test Application</Name>
  <VirtualSystem ovf:id="Ubuntu MySQL DB">
    <Info>Ubuntu 18.04.1604</Info>
    <ProductSection>
      <Property ovf:key="ip" ovf:value="192.168.1.10" ovf:type="string"></Property>
      <Property ovf:key="v_principal" ovf:value="virtual王子" ovf:type="string"></Property>
      <Property ovf:key="v root" ovf:value="/kuser" ovf:type="string"></Property>
    </ProductSection>
    <VirtualHardwareSection>
      <Info>2GB MB, 1 CPU, 1 disk, 1 naś virtual machine</Info>
      <System>
        <read:ElementName>Virtual Hardware Family</read:ElementName>
        <read:InstanceID>0</read:InstanceID>
        <read:VirtualSystemType>vmware-3.4</read:VirtualSystemType>
      </System>
      <read:description>Number of virtual CPUs</read:description>
      <read:ElementName>virtual CPU</read:ElementName>
      <read:InstanceID>0</read:InstanceID>
      <read:ResourceType></read:ResourceType>
      <read:VirtualQuantity>1</read:VirtualQuantity>
    </Item>
    <Item>
      <read:AllocationUnits>Bytes * 2^14</read:AllocationUnits>
      <read:ElementName>512 MB of memory</read:ElementName>
      <read:InstanceID>0</read:InstanceID>
      <read:ResourceType></read:ResourceType>
      <read:VirtualQuantity>512</read:VirtualQuantity>
    </Item>
    <Item>
      <read:AddressOnParent>6</read:AddressOnParent>
      <read:ElementName>Hard disk 1</read:ElementName>
      <read:InstanceID>0</read:InstanceID>
      <read:ResourceType></read:ResourceType>
    </Item>
    <Item>
      <read:ElementName>Ethernet adapter on "test-net"</read:ElementName>
      <read:InstanceID>0</read:InstanceID>
    </Item>
  </VirtualSystem>
</VirtualSystem>
```
OVF Centric View of VEP
Lifecycle of an OVF described application on VEP

1. Contextualization
2. Deployment
3. Elasticity
4. Snapshot through the creation of a new OVF file
5. Partial deployment for the federations
Cloud Infrastructure Management Interface

CIMI is a DMTF standard defining a model and a protocol for the management of interactions between IaaS clouds and users of IaaS services.

- CIMI system comprises of network, volumes, and machines
- System can be instantiated from templates supplied by cloud providers and/or users
- specification for generating a CIMI system template from OVF
- process for generating an OVF from a deployed application snapshot is described

OVF and CIMI standards works seamlessly with each other!
1. What is VEP?

2. Interoperability, Portability and Open Standards

3. VEP features

4. Status Report from Y3 and Roadmap

5. How to follow VEP’s progress?
### Salient features

- Standard OVF application support
- Constrained Execution Environment
- Datacenter representation
- Scheduler with constraints support
- OAuth controlled certificate delegation
- CIMI compatible REST API, REST GUI
What is a CEE?

Constrained Execution Environment

A CEE defines a virtual infrastructure integrating:

- Resource Handlers such as Network, Storage and VM Handlers
- Different types of constraints on these resources
- Configuration of the monitoring systems

OVF applications are deployed inside CEEs

- OVF items such as VirtualSystems are mapped to Resource Handlers
Contrail Virtual Execution Platform

- **VEP features**
- **Constrained Execution Environment**

**CEE Constraints**

- largeVM
- smallVM
- largeVolume
- L2net

- sameSwitch
- NFS
- mySQL
- load balancer
- Apache server
- HTTP1
- HTTP2
- HTTP3

- 10 Gb

**CEE handlers**
Contrail Virtual Execution Platform

- VEP features
- Constrained Execution Environment

Contrail Application

Hypervisor

L2net

smallVM

largeVM

largeVolume

sameSwitch

10 Gb

CEE Constraints

Resource Handlers

CEE handlers

Apache server

load balancer

MySQL server

NFS server

network

OVF Application
Contrail Virtual Execution Platform

- VEP features
- Constrained Execution Environment

### CEE Constraints
- Resource Handlers
  - LargeVM
  - SmallVM
  - LargeVolume
  - L2net

### Virtual Resources
- HTTP3
- HTTP2
- HTTP1
- Load balancer
- MySQL
- NFS
- NFS server

### CEE handlers

### Contrail Application

### OVFs Application
Contrail Virtual Execution Platform

- VEP features
- Constrained Execution Environment

CEE Constraints
Resource Handlers
Virtual Resources
OVF Application

CEE handlers

Contrail Application
What do CEE bring to VEP?

**Administrator point of view**

While allowing a user to upload his own OVFs, it makes resource scheduling easier through the use of specific, static VM handlers, similarly to what EC2 does with its predefined instances.

**User point of view**

Once a CEE is created, it acts as a framework, a user can create multiple applications from multiples OVFs using a chosen set of Handlers and Constraints, without negotiating the CEE creation again.
Datacenter representation

Clusters, Racks and Hosts are represented in our DB, which allows an administrator to specify at all times which resources are available, and how they are linked to each other.
Supported constraints

Types of Constraints

- **IN/NOT IN COUNTRY**: Allows a user to specify where he wants his application deployed
- **IN SAME RACK**: Increase communication speed between instances
- **NOT IN SAME HOST**: Increase reliability, compatible with the previous constraint
VEP Scheduler

Why a specific scheduler?

VEP is meant to allow an administrator to connect different IaaS architectures to the same VEP instance, to create a common scheduling environment, we created our own scheduler

Requirements

- Handle specific constraints as described previously, on heterogeneous providers
- Allow a user to reserve resources in advance
OAuth integration

**Delegated Certificates**

- VEP supports Delegated Certificates through OAuth
- All VM instances created through VEP include a user delegated certificate to allow the user to connect to his resources
- VEP uses the Contrail OAuth client developed to suit our needs
CIMI Compatible API

- Currently supports a part of the fully designed API, described in one of our deliverables
- Allows a User to create his own CEE and attach OVF afterwards
- Supports elasticity, allowing a user to start, stop and suspend VM instances as he wishes
Contrail Virtual Execution Platform

VEP features

REST GUI

CEE Creation GUI

Welcome to the Contrail Virtual Execution Platform (VEP) Management Portal. You can manage your deployed applications and update existing CEE elements from this page. If you are not sure, use the default CEE mapping rules for deployment.

You can either add a new CEE or select an existing CEE from the list.

Available CEE elements:
- VM Handlers
- Storage Handlers
- VNet Handlers
- Cloud Connections

Click on the listed items above to add that element to your current CEE. You can then process and deploy the element to see more details.

process and update
Administrator GUI

Welcome to VEP Service Management Tool
admin, you wish to manage your datacenter?

Summary view of your datacenter resources:
- datacenter
- cluster
- rack
- L2 switch
- storage
- host
- cloud network

Datacenter settings management panel:
- Select a host to add to the form.
- Please check the box if modifying an existing entry. If left unchecked, the submit action will create a new host entry for your datacenter.
- Check box if modifying an entry.
- Select the entry to modify.
- CPU Frequency (in MHz)
- 600
- Number of Cores
- 6
- RAM (in KB/byte)
- 409432
- Disk Capacity (in MB)
- 100
- Hostname
- ghostwheel42
- CPU Architecture
- x86_64
- Virtualization
- kvm
- Connected to L2 switch:
- Not selected
- Belongs to rack:
- Not selected
- Belongs to cloud:
- Not selected
- last Host ID
- 17

Submit
Video

Application Management through REST GUI Screencast
http://www.youtube.com/watch?v=KJ-K1CCUAPM
1 What is VEP?

2 Interoperability, Portability and Open Standards
   - Open Virtualization Format
   - Cloud Infrastructure Management Interface

3 VEP features
   - Constrained Execution Environment
   - Constraints
   - Scheduler
   - OAuth
   - REST API
   - REST GUI

4 Status Report from Y3 and Roadmap

5 How to follow VEP’s progress?
VEP Today

- OpenNebula 2.2 and 3.6+ support
- OVF and CIMI standards support
- Datacenter representation (cluster, racks, hosts)
- Advance Reservation and constraints-based scheduling
- REST API and GUI available since version 2.0
Roadmap

- VIN (Networking) and GAFS (Distributed Storage) integration in the coming weeks
- OpenStack support
- Snapshot support through the creation of new OVF documents
How to follow VEP’s progress?

Useful links

- VEP wiki: http://vep.gforge.inria.fr/
- VEP website: https://project.inria.fr/vep/
- Contrail: http://contrail-project.eu/