Autoconfiguration of Enterprise-class Deployment in Virtualized Infrastructure Using OVF Activation Mechanisms

Fermín Galán, Miguel Gómez, Fernando de la Iglesia, Ignacio Blasco, Daniel Morán

6th DMTF System and Virtualization Management Workshop (SVM) at 8th Int’l Conference on Network and Service Management (CNSM)

October 26th, 2012
Las Vegas, Nevada, USA
Outline

01 Introduction

02 OVF-based Activation

03 Use Case

04 Lessons Learnt and Conclusions
01

Introduction
IT-based Service Evolution

~10-15 years ago

Service users

App

Today

Service users

eg. Apache

Presentation

FE₁

FE₂

... 

FEₙ

eg. JBoss, Tomcat

Application Logic

BE₁

BE₂

... 

BEₘ

eg. MySQL

Persistence

LB

FW

LB

FW

LB

FW

LB

FW

DB₁

DB₂

... 

DBₘ
Problem Statement

![Diagram showing IT Staff deploying applications and middleware to hypervisors. The diagram illustrates the relationship between time consumed, error probability, and app complexity (number of tiers and nodes per tier).]

Many manual configuration points = Many potential failure points

- network configuration
- middleware/app parameters
Solution outline

- IT staff introduces configuration parameters
  - The only one manual operation
- Configuration is injected in VMs and applied
  - Automatically
OVF-based Activation
OVF 1.1 Introduction (structure)

DMTF Standard for Virtual Appliance (VA) packaging
## OVF 1.1 Introduction (main sections)

<table>
<thead>
<tr>
<th>Section</th>
<th>Location</th>
<th>Cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiskSection</td>
<td>Envelope</td>
<td>0..1</td>
</tr>
<tr>
<td>NetworkSection</td>
<td>Envelope</td>
<td>0..1</td>
</tr>
<tr>
<td>DeploymentOptionSection</td>
<td>Envelope</td>
<td>0..1</td>
</tr>
<tr>
<td>ResourceAllocationSection</td>
<td>VSC</td>
<td>0..1</td>
</tr>
<tr>
<td>StartupSection</td>
<td>VSC</td>
<td>0..1</td>
</tr>
<tr>
<td>AnnotationSection</td>
<td>VSC/VS</td>
<td>0..1</td>
</tr>
<tr>
<td>ProductSection</td>
<td>VSC/VS</td>
<td>0..*</td>
</tr>
<tr>
<td>EulaSection</td>
<td>VSC/VS</td>
<td>0..*</td>
</tr>
<tr>
<td>OperatingSystemSection</td>
<td>VS</td>
<td>0..1</td>
</tr>
<tr>
<td>InstallSection</td>
<td>VS</td>
<td>0..1</td>
</tr>
<tr>
<td>VirtualHardwareSection</td>
<td>VS</td>
<td>1..*</td>
</tr>
</tbody>
</table>
Product Section

- Specifies information about a **product**
  - The notion of product is quite ample in OVF
  - `product = OS | middleware | application | any-stuff-inside-a-VM`

- **Structure**
  - General information
    - Product name, vendor, version, product URL, icon, etc.
  - Configuration properties (key-value)
    - Two basic types
      - `userConfigurable="false"` → value established at package **building time**
      - `userConfigurable="true"` → value established at package **deployment time**
    - Values can be derived from parent VSC properties using a macro language
      - `{$<name_of_the_property_in_VSC>}`
(Simplified) ProductSection Example

Product identification

```
<ProductSection ovf:class="org.mysql.db" ovf:instance="1">
  <Info>Product customization for the installed MySQL Database Server</Info>
  <Product>MySQL Distribution Z</Product>
  <Version>5.0</Version>
  <Property ovf:key="queryCacheSizeMB" ovf:type="uint16" ovf:value="32">
    <Description>Buffer to cache repeated queries for faster access (in MB)</Description>
  </Property>
  <Property ovf:key="maxConnections" ovf:type="uint16" ovf:value="500" ovf:userConfigurable="true">
    <Description>The number of concurrent connections that can be served</Description>
  </Property>
  <Property ovf:key="waitTimeout" ovf:type="uint16" ovf:value="100" ovf:userConfigurable="true">
    <Description>Number of seconds to wait before timing out a connection</Description>
  </Property>
  Property ovf:key="hostname" ovf:type="string" ovf:value="${appHostname}"
</ProductSection>
```

Several instances of the same product can be distinguished within the same VS/VSC (optional)

General information

Q: How are these properties “injected” in the VMs in the end?
OVF-Based Activation

Inputs (from user prompt or other):
- Target deploy configuration option
- Values for Product Section properties

(1) OVF Processing Engine

(2) From the Product Section

(3) Deployment Platform

(4) ovf:transport

(5) OVF Env. 1, OVF Env. 2

Virtual System 1, Virtual System 2

Activation Engine
03 Use Case
Target application: RUBiS

- Benchmark application
  - Sample online auction service (eBay-like)

- Characteristics
  - Presentation tier
    - Apache 2.2.16 + mod_jk
    - stateless
    - 2 nodes
  - Application logic tier
    - JBoss EAP 5.1
    - stateless
    - 2 nodes
  - Persistence tier
    - MySQL Cluster 7.1
    - 1 management node (DBM) + 2 worker nodes (API & Data)
  - Per-tier HTTP LB
    - HA Proxy 1.3.15
Virtualization Platform (Deployment Platform)

- VMware vSphere 4.1 ESXi
  - 4 Xeon E5310 CPU @ 1.6 GHz
  - 4 GB RAM
  - 150 GB HD

- 4 subnets
  - Supporting several deployment topologies for the application
52 properties
- 16 are introduced by the user
- 36 are automatic (defined in the OVF envelope or derived)
Activation Engine

/etc/init.d/runOvfAe

Script in /etc/rc.S

ConfigFileCustomizer

OVF Environment

/mnt/cdrom.ovf-env.xml

/etc/network/interfaces

/opt/ae/templates/interfaces

```bash
iface eth0 inet static
    address {Environment.network_config.ip@ip}
    netmask {Environment.network_config.ip@mask}
    network {Environment.network_config.ip@netAddress}
    broadcast {Environment.network_config.ip@broadcast}
    gateway {Environment.network_config.gw}
    dns-nameservers {Environment.rubis_dns_servers}
    dns-search {Environment.rubis_dns_domain}
```

```bash
iface eth0 inet static
    address 172.16.3.12
    netmask 255.255.255.0
    network 172.16.3.0
    broadcast 172.16.3.255
    gateway 172.16.3.1
    dns-nameservers 10.95.96.4 10.95.96.5
    dns-search hi.inet
```
04

Lessons Learnt and Conclusions
# Lessons Learnt

<table>
<thead>
<tr>
<th>Value</th>
<th>Lesson</th>
</tr>
</thead>
<tbody>
<tr>
<td>😊</td>
<td>Adaptation to the network topologies existing in the Deployment Platform</td>
</tr>
</tbody>
</table>
| 😊 | All VMs used the same generic idempotent AE  
This works for RUBiS, but might not work for other enterprise-class apps |
| 😞 | Is it not possible to “factorize” all the nodes in a tier using just one  
<VirtualSystem>. OVF 2.0 solves this with (<ScalingSection>) |
| 😞 | Weak typing in <ProductSection> properties, e.g. “IP address” type can not be specified, just plain strings |
| 😊 | Scale running applications through “delta” packages |
| 😞 | Dynamic IP assignment (e.g. DHCP) doesn’t work in this approach. AE-AE communication mechanisms are needed |
| 😞 | CDMI is not supported natively by OVF, so disk images cannot be stored in storage clouds based on that standard |
| 😊 | Suitable for deployment in IaaS clouds with full OVF support |
Conclusions

- **Main takeaways**
  - The manual deployment of large multi-tiered enterprise-class applications is **complex**
    - Error prone and time consuming for IT staff
  - OVF-based activation is a very convenient mechanisms to alleviate this situation
  - We have successfully proved the feasibility of the approach with such a large and complex enterprise-class application

- **Future work lines**
  - Test with other virtualization platforms, e.g. XenServer, RHEV, etc.
  - Runtime scaling through “delta” packages
  - Obtaining environmental parameters (such as IPs) and implementing AE-AE communication mechanisms