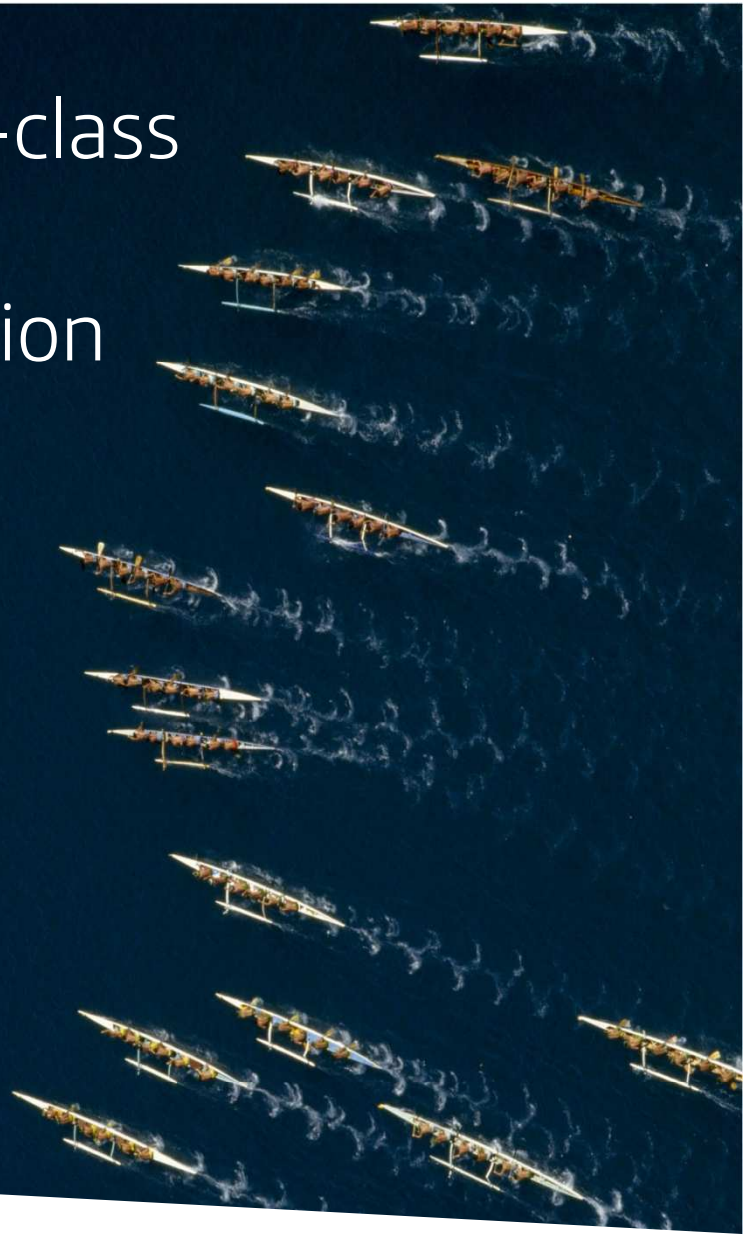
Autoconfiguration of Enterprise-class Deployment in Virtualized Infrastructure Using OVF Activation Mechanisms

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



TELEFÓNICA DIGITAL

fi-ware

Telefónica

Outline

01 Introduction

02 OVF-based Activation

03 Use Case

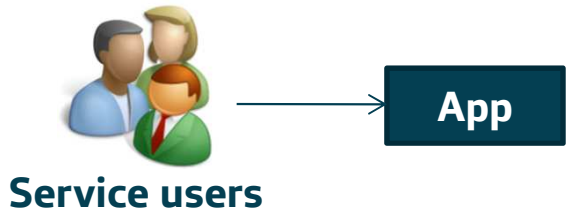
04 Lessons Learnt and Conclusions

01

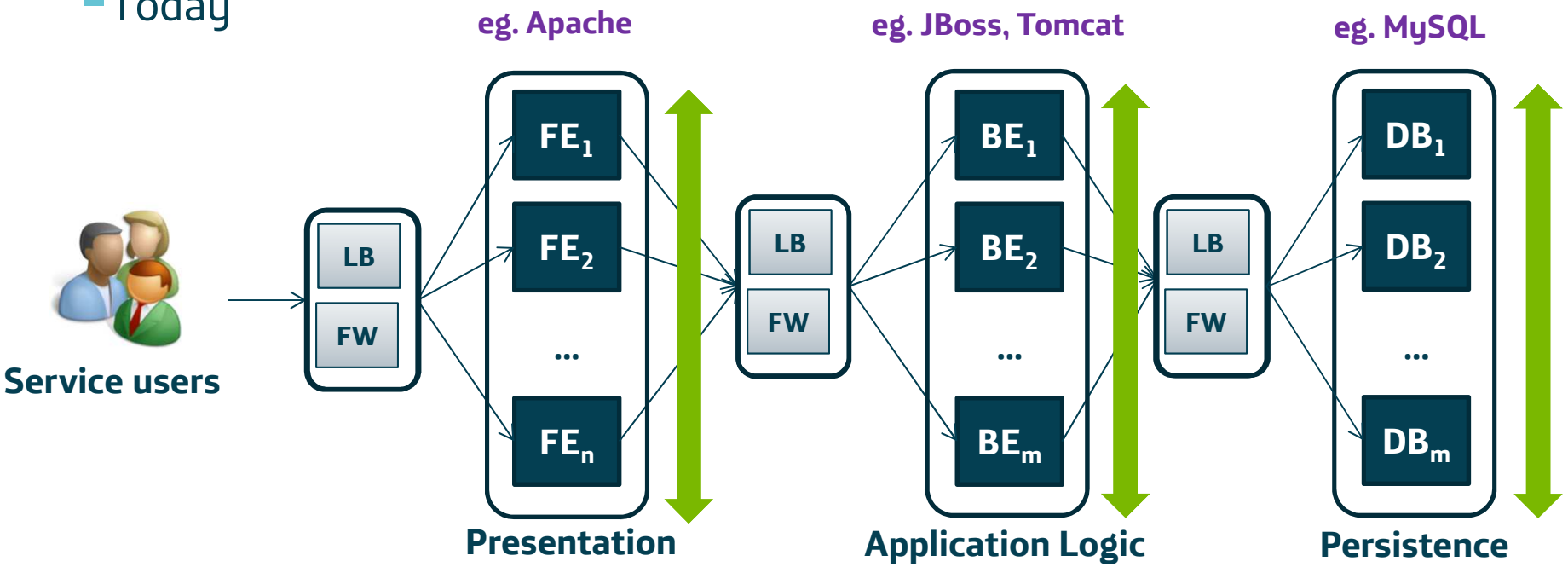
Introduction

IT-based Service Evolution

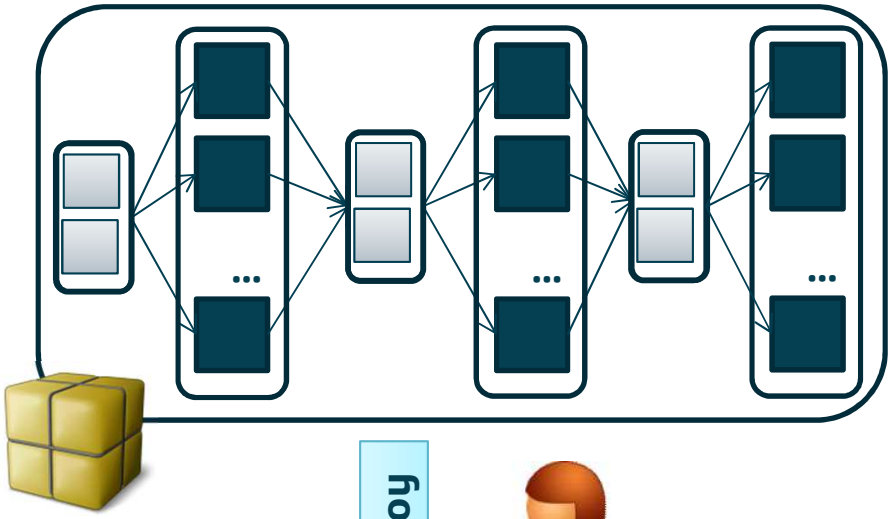
- ~10-15 years ago

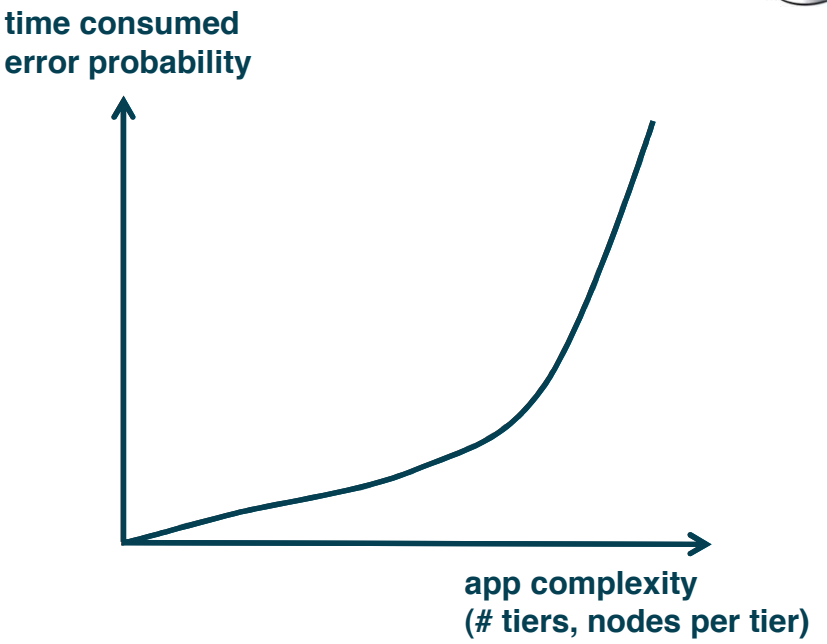


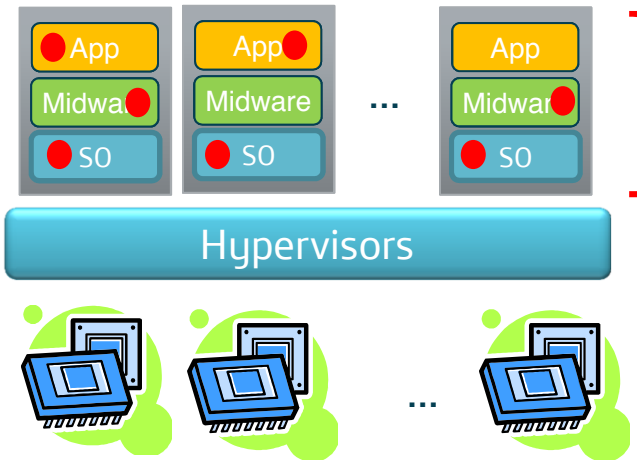
- Today



Problem Statement



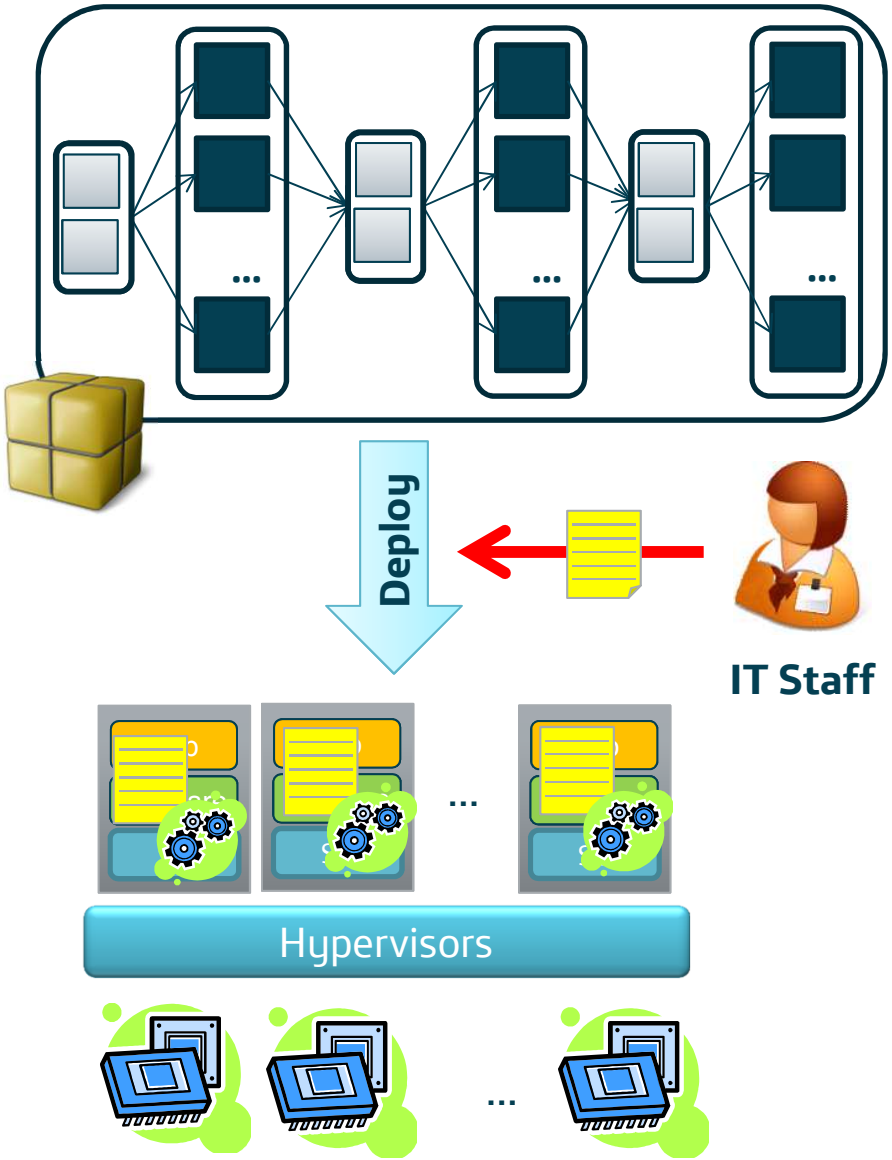




Many manual configuration points = Many potential failure points

- network configuration
- middleware/app parameters

Solution outline



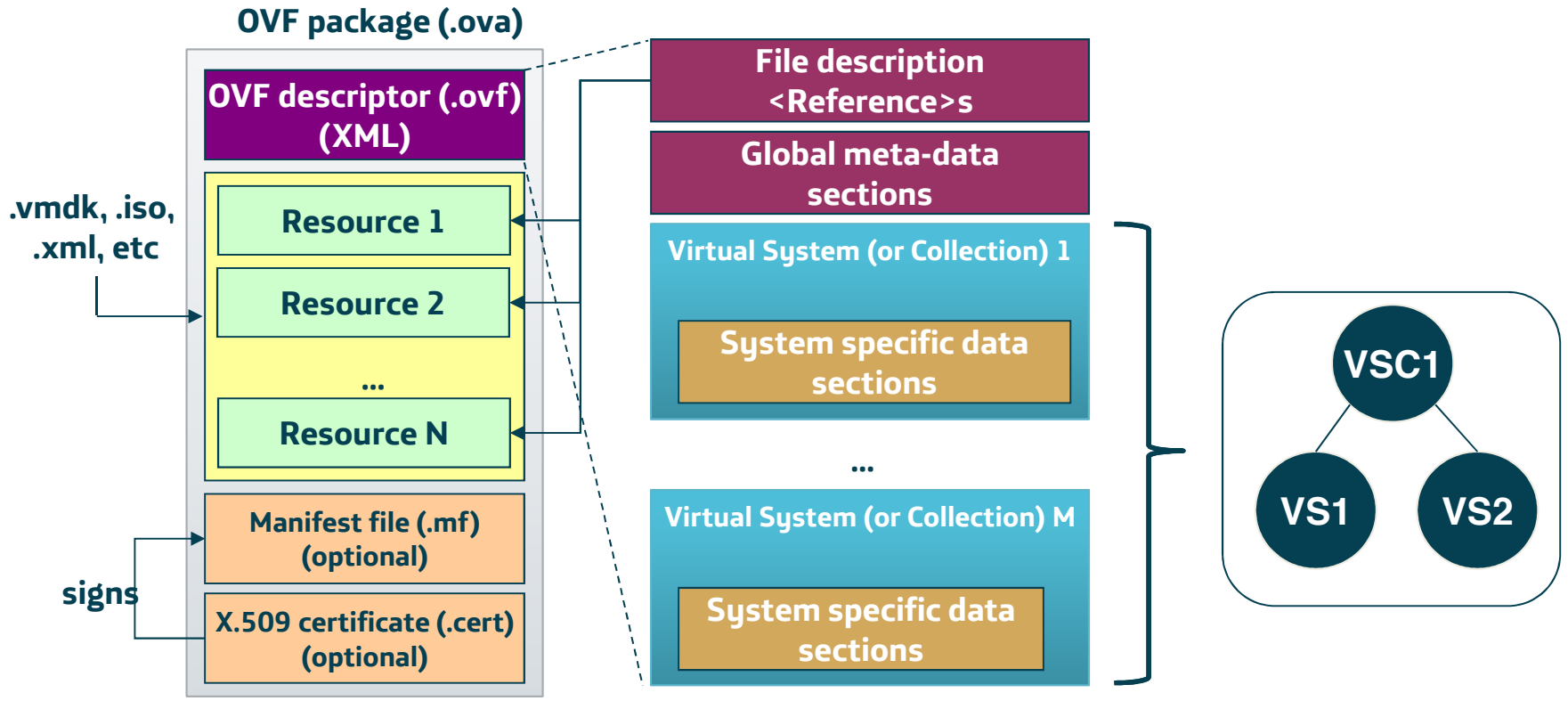
- IT staff introduces configuration parameters
  - The only one manual operation
- Configuration is injected in VMs and applied
  - Automatically

02

OVF-based Activation

OVF 1.1 Introduction (structure)

DMTF Standard for Virtual Appliance (VA) packaging



OVF 1.1 Introduction (main sections)

Section	Location	Cardinality
DiskSection	Envelope	0..1
NetworkSection	Envelope	0..1
DeploymentOptionSection	Envelope	0..1
ResourceAllocationSection	VSC	0..1
StartupSection	VSC	0..1
AnnotationSection	VSC/VS	0..1
ProductSection	VSC/VS	0..*
EulaSection	VSC/VS	0..*
OperatingSystemSection	VS	0..1
InstallSection	VS	0..1
VirtualHardwareSection	VS	1..*

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

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

```
<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>
```

General information

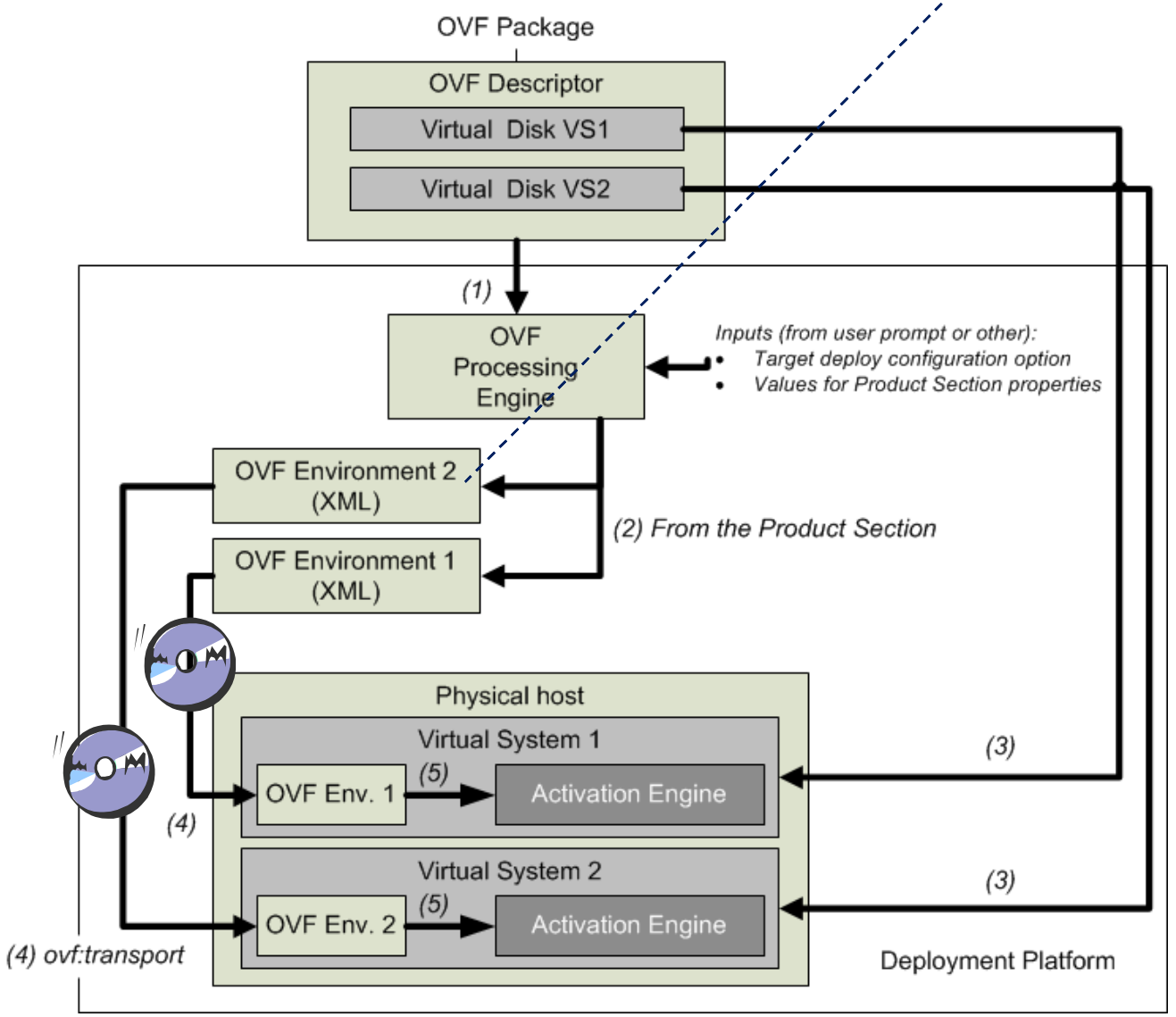
deployment-time properties

derived value, from the "appHostname" property value in the parent VSC (not shown here)

Q: How are these properties "injected" in the VMs in the end?

OVF-Based Activation

```
<Property ovf:key="org.mysql.db.cacheSize.1" ovf:value="32"/>
<Property ovf:key="org.mysql.db.maxConn.1" ovf:value="500"/>
<Property ovf:key="org.mysql.db.waitTime.1" ovf:value="100"/>
```

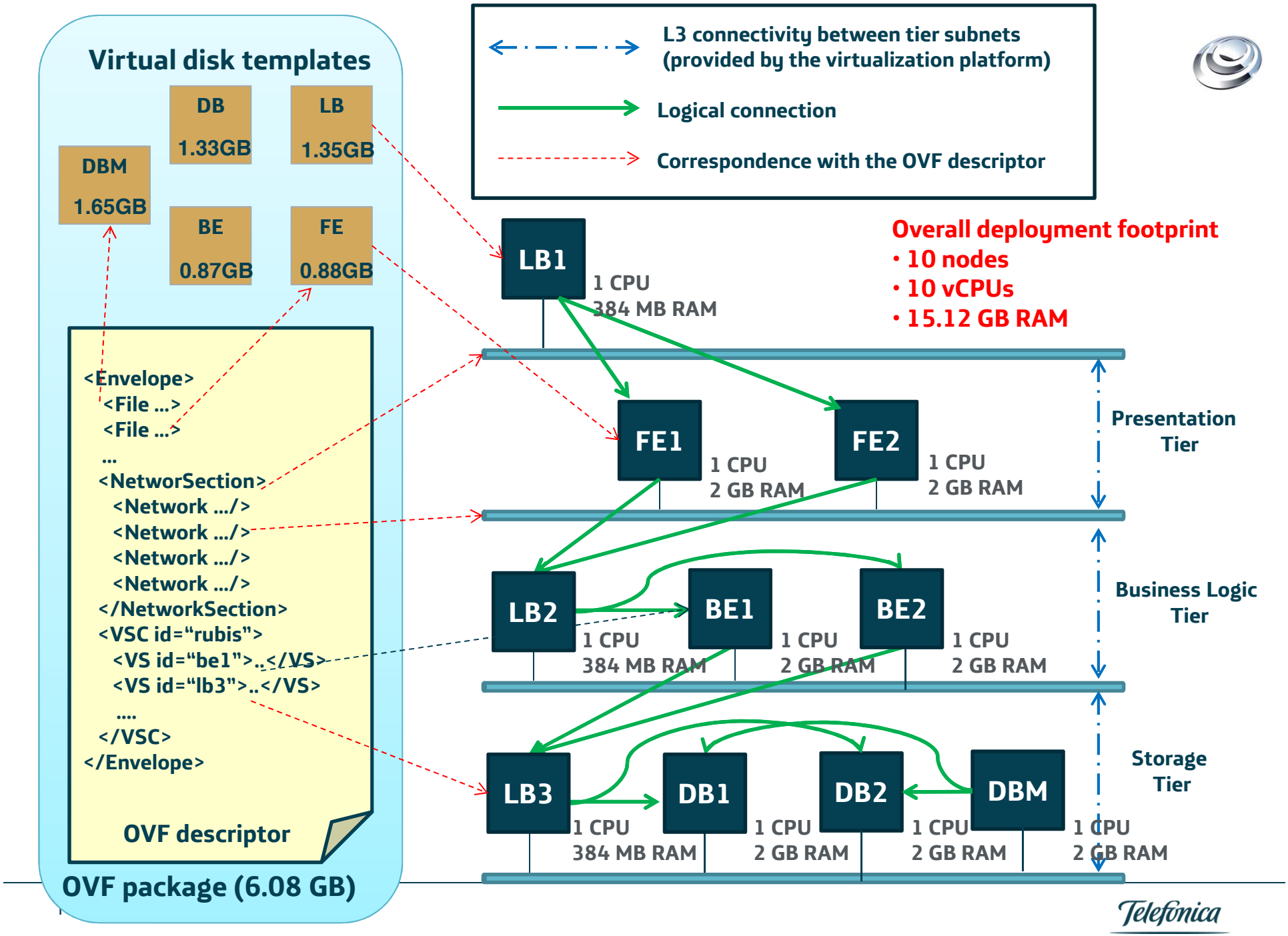


03

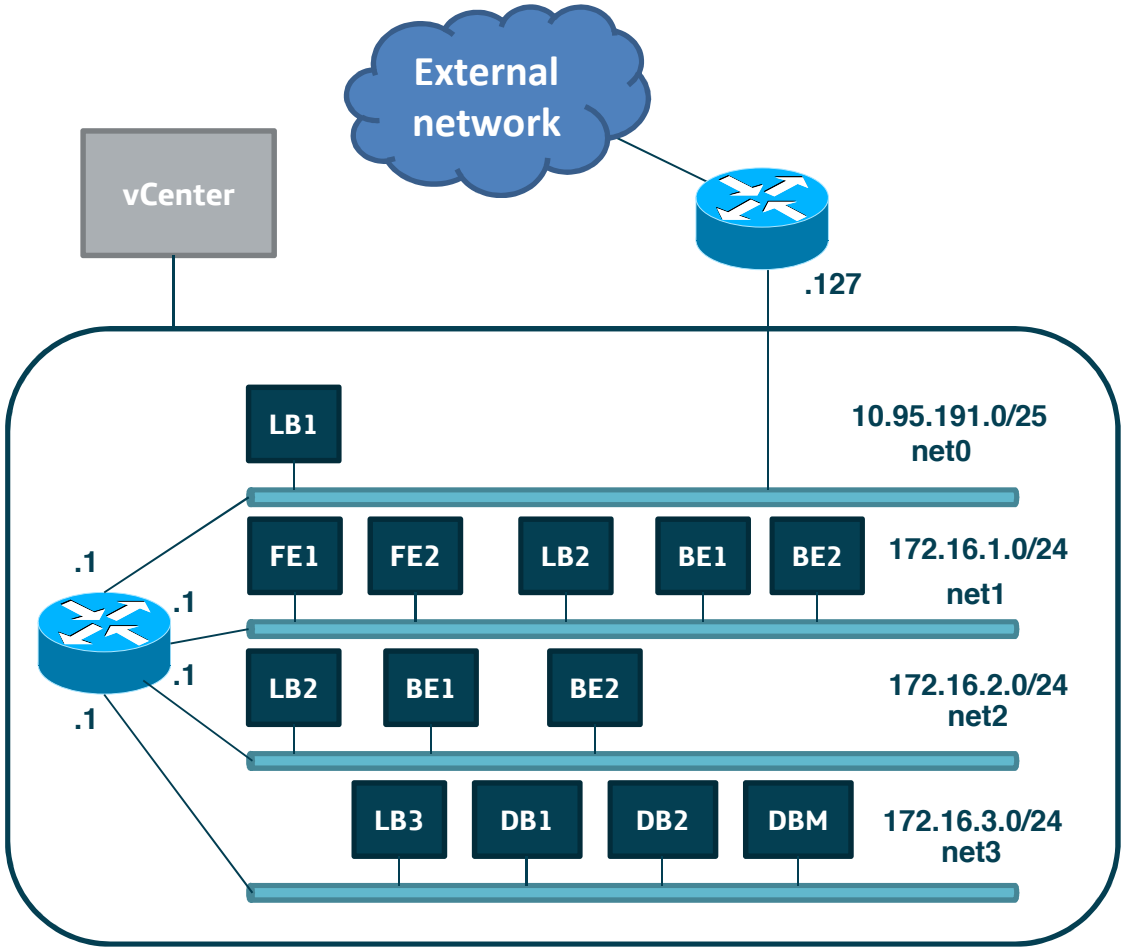
Use Case

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

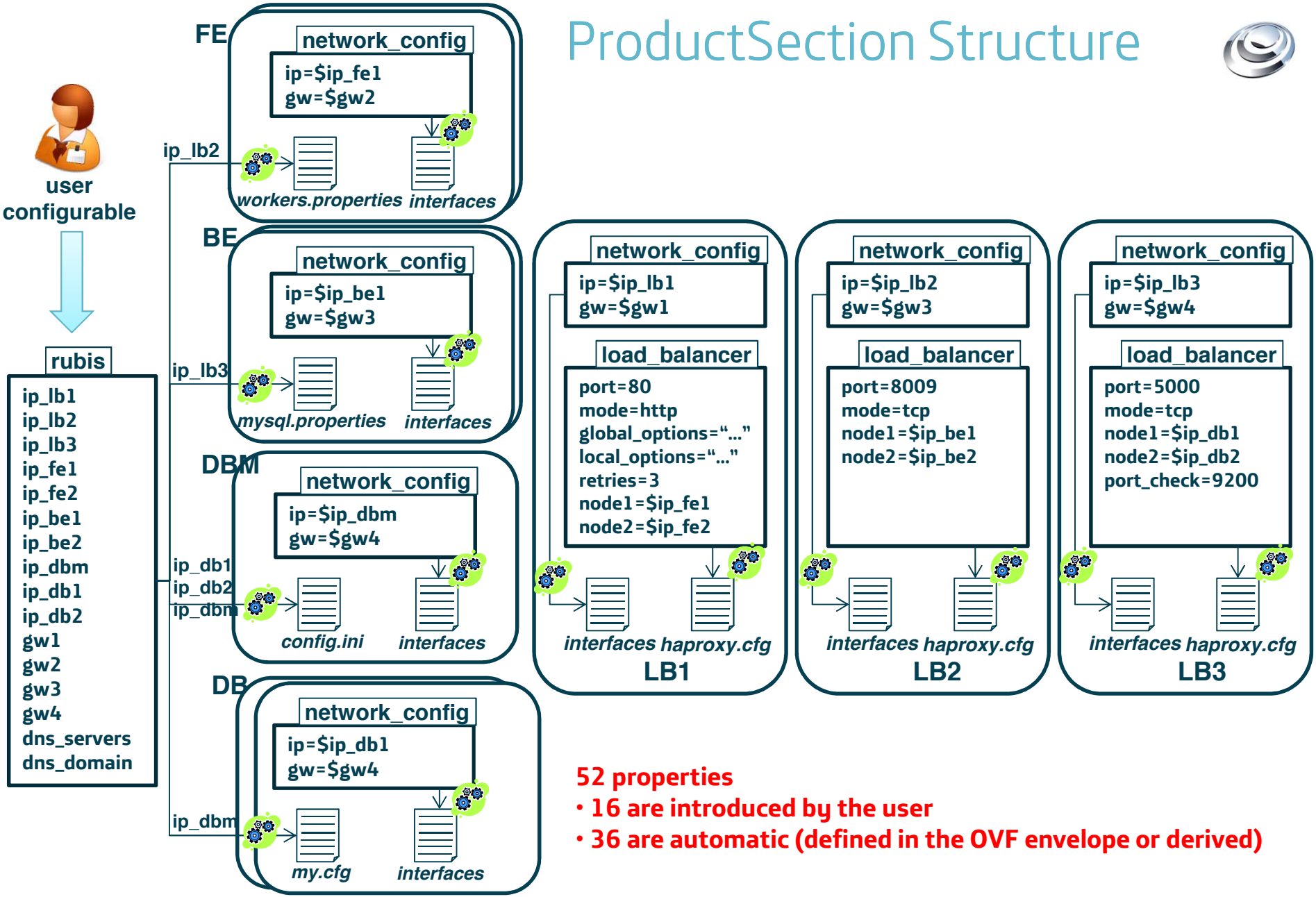

Virtual disk templates
DB
1.33GB
LB
1.35GB
DBM
1.65GB
BE
0.87GB
FE
0.88GB
<Envelope>
<File ...>
<File ...>
...
<NetworSection>
<Network .../>
<Network .../>
<Network .../>
<Network .../>
</NetworkSection>
<VSC id="rubis">
<VS id="be1"..</VS>
<VS id="lb3"..</VS>
....
</VSC>
</Envelope>
OVF descriptor
OVF package (6.08 GB)
L3 connectivity between tier subnets
(provided by the virtualization platform)
Logical connection
Correspondence with the OVF descriptor
Overall deployment footprint
• 10 nodes
• 10 vCPUs
• 15.12 GB RAM
LB1
1 CPU
384 MB RAM
FE1
1 CPU
2 GB RAM
FE2
1 CPU
2 GB RAM
Presentation Tier
LB2
1 CPU
384 MB RAM
BE1
1 CPU
2 GB RAM
BE2
1 CPU
2 GB RAM
Business Logic Tier
LB3
1 CPU
384 MB RAM
DB1
1 CPU
2 GB RAM
DB2
1 CPU
2 GB RAM
DBM
1 CPU
2 GB RAM
Storage Tier
Telefónica

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

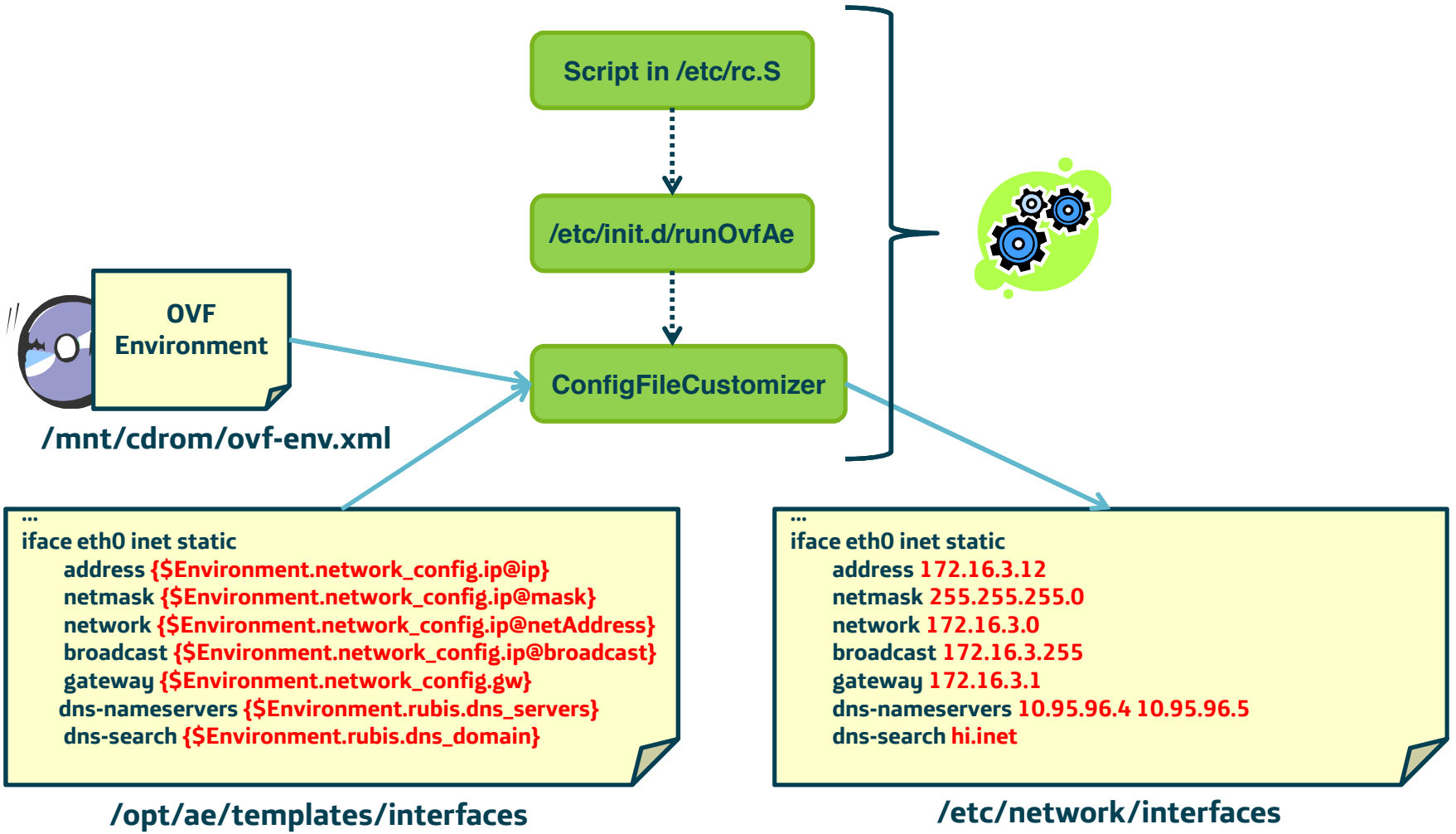
ProductSection Structure



52 properties

- **16 are introduced by the user**
- **36 are automatic (defined in the OVF envelope or derived)**

Activation Engine



04

Lessons Learnt and Conclusions

Lessons Learnt

Value	Lesson
☺	Adaptation to the network topologies existing in the Deployment Platform
☺	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 assignation (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


Telefónica