Package your existing applications for automatic deployment into clouds.

The vast majority of IT professionals today understand that cloud computing constitutes a fundamental paradigm shift in how the software solutions are marketed, distributed, and deployed. I contend that there are two fundamental features of cloud computing that will guarantee its success in the marketplace:

1. Self-service
2. Automation

The concept of *self-service* empowers the end-users. They can browse a catalog of prepackaged software solutions and decide when to deploy the workloads that satisfy their requirements. In this scenario, the data center provides the required services much in the same way as your public utilities company delivers electricity to your home. The data center administrators are no more the gatekeepers; they become transparent to the end-users.

With a high level of *automation,* the companies are able to lower their data center operational expenses (OPEX) by orders of magnitude. To achieve these high levels of automation, however, the IT shops need to adopt a new deployment model. In the traditional model, a solution provider such as an ISV would send a specialist to a customer location to install or upgrade their solution. Often, it is a cumbersome, time-consuming, tedious process that needs

to be repeated over and over again at each customer location. Cloud computing simplifies this problem by implementing a solution known as a software virtual appliance (software VA, for short).

A software virtual appliance is a self-contained and self-described entity that consists of one or more virtual machine images and a metadata descriptor that describes the content of the virtual appliance and the required features of the target deployment platform.

IBM—along with many other vendors in the virtualization space—have adopted the Distributed Management Task Force (DMTF) Open Virtualization Format (OVF) specification as a standard way of building and deploying software virtual appliances.

Why Implement Software Virtual Appliances?

The software virtual appliances can encourage a quicker transition to the cloud deployment model by dramatically lowering the initial requirements for in-depth virtualization skills and addressing the key issues that have previously hindered adoption of the virtualized solutions:

- Software Licensing—The software license agreement can be included in the software virtual appliance so that the deployer must accept its terms and conditions before the appliance can be successfully activated.
- A standard way to accompany the executable image with contextual information—The metadata descriptor, called the OVF file, contains detailed information about the target system Hardware Abstraction Layer (HAL) requirements, such as how many virtual CPUs are needed, what the networking setup is, etc. In addition, the OVF file can contain precise descriptions of the software components that constitute the appliance. For example, there might be separate sections describing the attributes of the operating system, the middleware, and the end-user

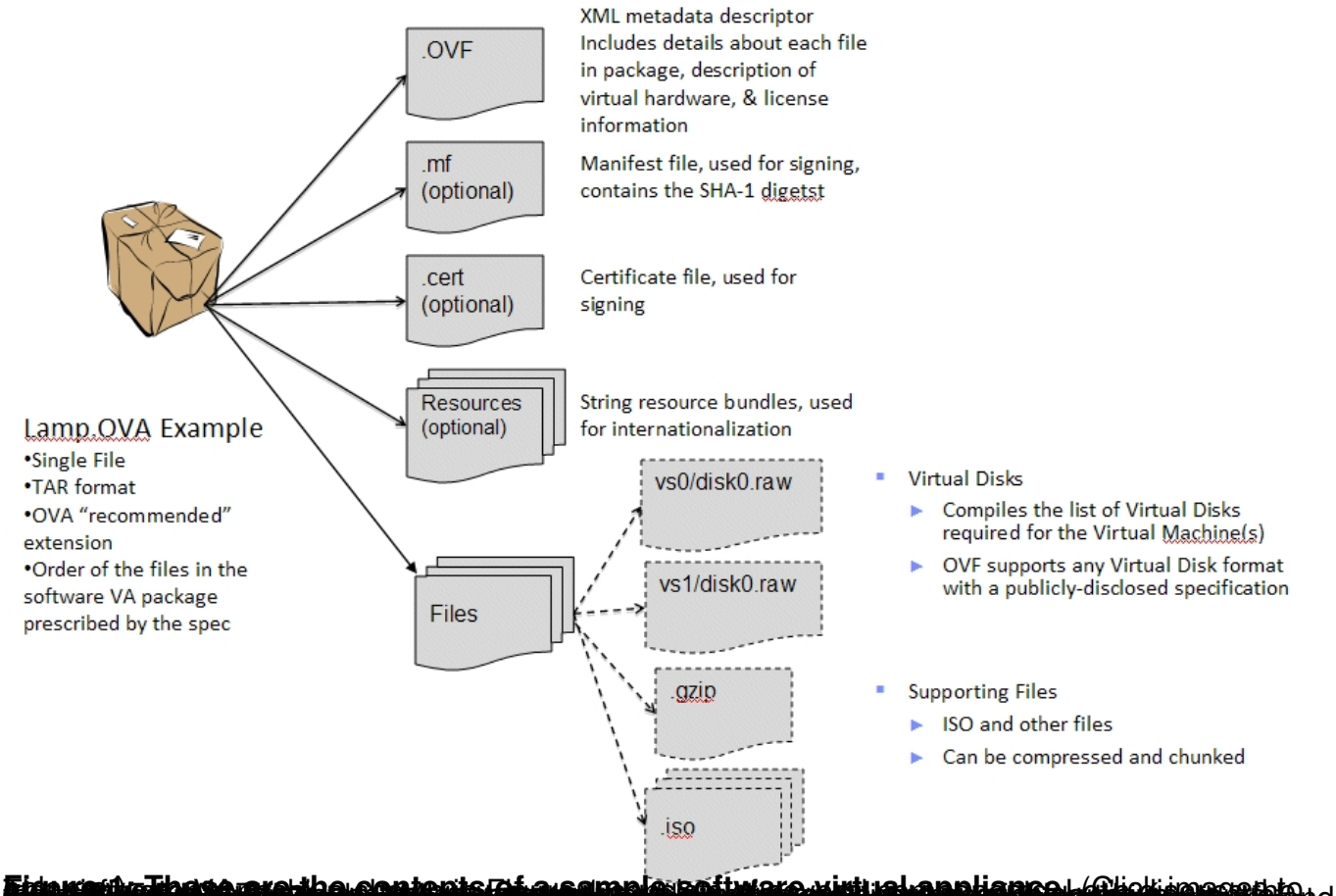
applications.

- Support for software topologies that require multiple virtual machines—The realistic multi-tier software architectures can be properly represented with each tier being mapped to a separate virtual machine image. Note that in the DMTF specification a virtual machine is called a virtual system.
- Vendor and platform independence—The appliances can be ported across management stacks, making a migration among virtualization vendors and among data centers fairly easy and non-disruptive.
- Industry standard content verification and integrity checking support—The appliance payloads are digitally signed, ensuring that the content was not tempered with or modified.

The Anatomy of a Software Virtual Appliance

As mentioned, a software virtual appliance can consist of one or many virtual systems (VSes). Each VS can contain one or many virtual disks. The relationships between components as well as their attributes are described in the OVF file. All of the components, including binary images of the virtual disks along with the corresponding OVF file, are packaged in a *tar* archive. This tar archive becomes a software VA. In its most basic form, a software VA consists of a single virtual server that contains just one virtual disk image and a corresponding OVF.

Figure 1 illustrates the contents of a software VA targeting KVM as the deployment hypervisor.



The OVF File

The OVF file is a critical part of every DMTF-compliant software VA. The cloud deployment management system parses its content to retrieve the information that is necessary for a successful activation of the software solution contained in a given appliance. The OVF file is an XML document that contains a number of sections. Each section describes a different aspect of the appliance.

Let's examine the most important parts of a typical OVF file:

Disk Section

The Disk Section defines all virtual disk images at the global (virtual appliance) level. Here's an XML snippet that shows a typical Disk Section for a simple single-VS, single-disk appliance:

```
<ovf:References>

    <ovf:File ovf:href="vs0/disk0.raw" ovf:id="disk0"

ovf:size="10737418240" />

</ovf:References>

<ovf:DiskSection>

    <ovf:Info>Describes the set of virtual disks</ovf:Info>

    <ovf:Disk ovf:capacity="10737418240"                    [1]

ovf:capacityAllocationUnits="byte"

ovf:diskId="vs0/vd0"
```

```
ovf:fileRef="disk0"                                [2]

        ovf:format=" http://www.ibm.com/xmlns/ovf/diskformat/qemu.raw "  [3]

        ovf:populatedSize="10737418240" />

</ovf:DiskSection>
```

In the sample above, at [1] the virtual disk capacity is specified, and at [2] a file reference to the disk image location is defined. The deployment platform uses this info to locate the file within the appliance tar archive that contains the binary image of a given virtual disk. At [3] a link to the disk format specification is provided.

Network Section

The Network Section provides all network definitions at the global level, listing the required private and public networks used in the software VA.

Virtual System Collection

The Virtual System Collection contains one or many virtual system definitions. This concept allows the deployment platform to manage the collection of the virtual systems as a unit of deployment. A successfully deployed Virtual System Collection becomes a workload that can be started, stopped, and migrated as a unit.

Virtual System

The virtual system defines the metadata for each of the virtual machines that make up the appliance. The virtual system section contains the following sections:

- Eula Section—This is the license agreement for the given virtual server.
- Product Section—This defines application, service, and static info that needs to be configured at the deployment of the appliance. For example, the local host name needs to be changed in the DB2 server configuration.
- Hardware Section—This defines the virtual hardware to be used per virtual server. This needs to be in the scope of the Hardware Abstraction Layer (HAL) of the hypervisor chosen for deployment. Typically, this section describes the following aspects of HAL: network (virtual or physical), capacity of virtual servers that will boot from images (memory, CPU), storage controller, and devices.

The Magic of the Virtual Appliance Self-Activation

The true value of the OVF-compliant software VAs lies in the ability to automatically install and configure all of the software components, including the operating system, middleware, and end-user applications. Unfortunately, the prevailing approach of many appliance architects is to ship them "half-baked" without automating the activation of the entire software stack. Often, appliance architects use, say, the VMWare tools to ensure the reconfiguration of the OS-level settings, such as host name, domain name, and root password. They tend not to address the reconfiguration needs of the software components above the OS at the initial deploy time. For example, they do not reconfigure the database settings. Consequently, at deployment time, the host name at the OS level gets set to the proper value, while the database configuration shows

the original host name. This may lead to unpredictable results and often requires manual debug and repair. This approach is problematic for several reasons:

- The need for a manual configuration of deployed workloads defeats the key value proposition of cloud computing—namely, automation.
- The deployer (a cloud admin) may not have the necessary domain knowledge to properly reconfigure a third-party application or middleware or the inter-dependencies of a multi-tiered application construct.

A preferred solution to address these issues is to take advantage of the IBM Virtual Solutions Activation Engine (VSAE). The Activation Engine is a software tool that is used during the appliance creation and deployment phases. The appliance architect installs the Activation Engine in each virtual system image during the appliance construction phase. Then, when the appliance is deployed, the Activation Engine runs in each of the virtual systems and reconfigures the system and the installed applications.

Essentially, the VSAE is a scripting engine that starts on the first boot before the application services are activated. For example, the WebSphere Application Server (WAS) data source configuration requires the DB2 server host name, the port on which DB2 is listening, the database name, and DB2 user credentials. Please note that these settings may be different for each instance and are runtime environment–specific. They need to be set before the WAS application startup to prevent conflicts and security exposure.

The VSAE supports a pluggable architecture where the configuration tasks for a specific aspect of a software stack are performed by the specialized activation programs. These activation programs are invoked by the Activation Engine in a predefined sequence.

The OVF standard recommends that the runtime parameters are passed to the Activation Engine using an XML configuration file (default name is ovf-env.xml) contained in a virtual CD drive. The virtual CD drive is attached to a given virtual system at the boot time. The Activation Engine parses the XML configuration file to retrieve the parameters and then calls a specific activation program that sets points of variability (change points) in the application stack.

These concepts are illustrated in Figure 2:

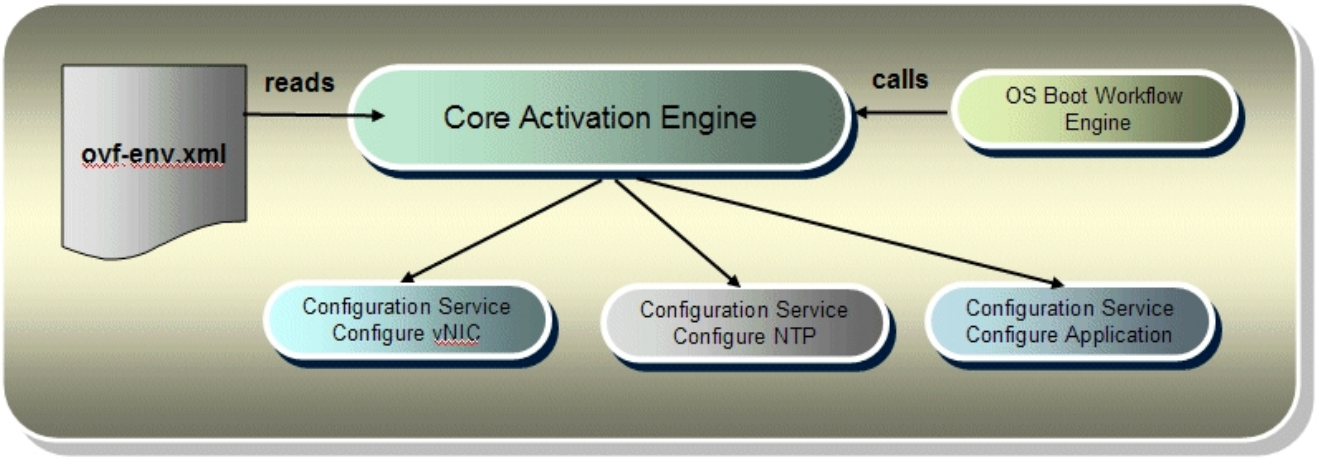


Figure 2: These are the Activation Engine components.

Let's analyze the activation flow in greater detail. For example, an activation program that reconfigures a DB2 instance requires a number of runtime-specific parameters that need to be passed by the deployment platform using the ovf-env.xml transport mechanism. Here's an excerpt from the ovf-env.xml file that illustrates how the DB2 runtime parameters are passed to the Activation Engine:

```
<PropertySection>

<Property ovfenv:key="com.ibm.vsae.2_1.db2-config.db2inst1_password"

                ovfenv:value="" />

<Property ovfenv:key="com.ibm.vsae.2_1.db2-config.db2fenc1_password"

                ovfenv:value="" />

<Property ovfenv:key="com.ibm.vsae.2_1.db2-config.dasusr1_password"

                ovfenv:value="" />
```

<Property ovfenv:key="com.ibm.vsae.2_1.db2-config.db2inst1_username"

ovfenv:value="db2inst1" />

<Property ovfenv:key="com.ibm.vsae.2_1.db2-config.db2inst1_group"

ovfenv:value="db2iadm1" />

<Property ovfenv:key="com.ibm.vsae.2_1.db2-config.db_name_new"

ovfenv:value="SAMPLE" />

<Property ovfenv:key="com.ibm.vsae.2_1.db2-config.db_action"

ovfenv:value="do-nothing" />

<Property ovfenv:key="com.ibm.vsae.2_1.db2-config.db2_license_type"

ovfenv:value="paid" />

<Property ovfenv:key="com.ibm.vsae.2_1.db2-config.db2_hostname" [1]

```
                ovfenv:value="vs102" />

/PropertySection>
```

In the example above, at [1] the db2_hostname parameter is passed with the value of "vs102". Note that I use the fully qualified parameter name that will allow the activation engine to locate the proper activation program to which to pass the hostname parameter and thus avoid the name conflicts for the same-named parameters used by different activation programs. For instance, the WAS activation may also require a db2_hostname parameter to properly reset the server name in the data source object.

There are two methods for setting the parameters:

1. At the appliance construction time—The appliance architect can use the OVF Product Section for DB2 to preset the parameters to their default values. Consider the following excerpt from the corresponding OVF file:

```
<!-- db2_hostname = localhost -->
```

```
<ovf:Property ovf:key="db2_hostname" ovf:type="string"

    ovf:userConfigurable="true" ovf:value="localhost" password="false">   [1]

        <ovf:Label ovf:msgid="ConfigDB2.db2_hostname.label"></ovf:Label>

        <ovf:Description>Local Hostname</ovf:Description>

</ovf:Property>
```

In the snippet above, at [1] the appliance architect sets the db2_hostname to a default value of "localhost". He or she also indicates that the value is user-configurable, meaning that it can be changed by the deployer.

This leads to the second method for parameter modification:

2. At the deployment time—The deployment platform identifies all the parameters that have the userConfigurable attribute set to true. The parameters will be presented to the deployer at one of the deployment steps, at which time they can be changed to the values that reflect the current runtime environment. For example, the db2_hostname gets set to "vs102". Note that the

deployer can be a human being or the cloud management software, such as the appliance placement services.

The actual code within the activation program responsible for resetting the host name may look as shown below:

```
echo "Update the DB2 hostname to $db2_hostname"

set -x

$DB2_PATH/adm/db2set -g db2system=$db2_hostname

set -

echo
```

Following the methodology described in this section, a number of software VAs were built for IBM partners. These software VAs fully automated the deployment of the real-life, sometimes-quite-complex solutions.

Building Software Virtual Appliances

By now you should be excited about the tremendous possibilities offered by the software Virtual Appliance deployment model. You may ask yourself, "How do I package *my* applications into this format so that they become cloud-ready?" Well, there are several options:

1. Build the software VA manually using the open-source tooling. For example, you could install on your Linux workstation the open-source hypervisor KVM, along with the virtualization tools, such as virsh, virt-manager, and virt-viewer. You would then use the virt-manager to create the virtual machine images, boot those images on KVM, and install all the necessary software components. A simple text editor could then be used to create or modify an OVF file to describe the content of the virtual machine images you created. The final step would be to create a tar archive that contains the images and the OVF. This is, of course, easier said than done. The manual VA creation requires significant virtualization and OVF skills and probably is appealing only to geeks. So let's explore other options.

2. Use VMWare tooling. VMWare Studio and the OVF Toolkit are aimed at simplifying the process of virtual appliance creation. The virtual appliances created in VMWare Studio can be easily imported and deployed using vSphere Client. The process is well-documented in the VMWare publications. Please keep in mind that VMWare tools do not support open-source hypervisors such as KVM or architectures other than Intel.

3. Leverage Virtual Appliance Factory Service (VAF Service). With this no-fee service, IBM Business Partners and ISVs get assistance and resources to build virtual images and software virtual appliances. The VAF Service is a set of Web 2.0 tools, services, and processes that dramatically simplify and automate the process of ISV solutions "cloudification." Currently, this

service is hosted by the IBM Innovation Center in Waltham, Massachusetts. Some of the benefits of the VAF Service include these:

- Allows entry into cloud space with minimal up-front investment
- Hides complexity of the virtualization infrastructure, meaning that partners can focus on their knowledge domain, which is multi-tier application composition rather than infrastructure setup
- Creates a virtual appliance deployable into DMTF OVF–compatible data centers
- Delivers on two key value propositions of cloud computing: automation and self-service (industry-strength IBM VSAE can be bundled with the VAs)

Please contact the author for further details on the VAF Service.

Closing Comments

The software virtual appliance model governs the packaging and deploying of software solutions into cloud environments. It does *not* have dependencies on any specific cloud architecture or cloud services. I believe that the software VA standard as codified by the DMTF OVF spec has been the best option for easing the pains of moving from the traditional deployment model to the cloud deployment model. It speeds up the process of "cloudifying" a range of applications. On the one end of the spectrum, you can take an existing matured, hardened, and market-tested COBOL app and with little effort package it as a single-image, single-disk software VA, which immediately becomes deployable into the cloud. On the other end of the spectrum, you can package a newly implemented, highly distributed, service-oriented application. In this case, the software VA may consist of multiple virtual systems, each requiring multiple disks and sophisticated activation logic. This newly created software may, in fact, integrate with the services provided by the cloud, such as database services or queuing services. Please note, however, that this integration with the cloud services is neither required nor governed by the DMTF OVF spec. Which underlying services to use is the solution architect's choice much in the same way as she or he may choose to use J2EE rather than .NET. This has nothing to do with how the solution is packaged.

I hope this helps clarify the scope and the benefits of the software VA deployment model.

Additional Material

The DMTF OVF specification Web site:

http://www.dmtf.org/standards/ovf

The IBM Web site for Partners who wish to participate in the Cloud Specialty program:

https://www-304.ibm.com/partnerworld/wps/servlet/ContentHandler/isv_com_spe_cloud_index

as/400, os/400, iseries, system i, i5/os, ibm i, power systems, 6.1, 7.1, V7, V6R1