# OVF: A few layers short of a full stack

**Lori MacVittie, 2009-21-04**

OVF (Open Virtualization Format) apparently just isn't getting enough mindshare out there in the discussions of cloud computing that focus on portability and interoperability. The goal of OVF is to provide a portable, interoperable non-vendor specific meta-data that describes an application, its virtual container, and the attributes necessary to deploy it in a new environment with minimal human intervention. This will, allegedly, allow it to move seamlessly from cloud to cloud, drifting ever-so-gently and making the entire process appear effortless.

Given that lofty goal, it's no surprise that Jon Oltsik, senior analyst at the Enterprise Strategy Group, wonders why we aren't talking more about it in "Why not talk more about Open Virtualization Format?"

There's a good answer to that, actually, at least from the perspective of an infrastructure provider. And that answer can be found in the description of OVF Jon uses in his article:

> *In geek land, OVF is a way to describe the properties of VMs from the network layer up to the application so they can retain "state" as they are created or moved.*

Well yes, and no. While OVF is great at defining a whole lot of application-specific and some network characteristics for applications residing in a VM, it's a few layers short of describing the a full stack.

---

**THE MISSING PIECE(S) of the PUZZLE**

---

VMWare describes the difference between its VMDK and OVF, giving props to OVF as a more comprehensive solution and inadvertently highlighting the root of the problem from an infrastructure point of view:

> VMDK is a file format that only encodes a single virtual disk from a virtual machine. A VMDK does not contain information about the virtual hardware of a machine, such as the CPU, memory, disk, and network information. A virtual machine may include multiple virtual disks or VMDKs. An administrator who wishes to deploy a virtual disk must then configure all of this information, often manually, using incomplete documentation.

> The OVF format, on the other hand, provides a complete specification of the virtual machine. This includes the full list of required virtual disks plus the required virtual hardware configuration, including CPU, memory, networking, and storage. An administrator can quickly provision this virtual machine into virtual infrastructure with little or no manual intervention. [emphasis added]

OVF is extensible and it certainly provides a robust mechanism for describing all the software-specific details necessary to move a VM from one physical location to another. But cloud  is very much about infrastructure, and architecture, and unfortunately OVF does nothing (and to its credit is not designed to) to forward the cause of interoperability or portability or even collaboration with that infrastructure. There are no policies carried around with OVF – does the application have a specific SLA? Or require external security? Authentication systems? ID management? Aggregated logs? NMS? QoS? Rate shaping? SSL?

An application is not an island, and in the land of cloud computing it must collaborate with its infrastructure to provide feedback on performance and response time and utilization thresholds to the management systems which must, well, manage the application. These things are ***application specific***, and as such these policies and information should be carried with the application as it roams around the Internet.

The reason we aren't talking more about it is that OVF doesn't address all these little configuration necessities, and in fact currently no specification attempts to offer a standard mechanism for sharing these application-specific details either. So the statement that OVF allows an administrator to "quickly  provision this virtual machine into virtual infrastructure with little or no manual intervention" is true only if you define "infrastructure" as being a very narrow set within the data center that does not include any of the network, application network, security, or identity/network and systems' management infrastructure. The claim that OVF allows an organization to "quickly provision an application ***into a new environment*** with little or no manual intervention" is not entirely true because there's a lot of work ***someone*** is doing after the virtual machine is provisioned into a new environment.

Take persistence (server-affinity) as a simple example. Web applications almost always need to store state in the application server session. That session data is important to ensuring the application behaves as expected. But when that application is pushed into a load balanced environment, of which cloud computing is usually one, the user may not be properly connected with the right server throughout their session, which essentially breaks the application. Organizations use persistence – the ability of an intermediary to maintain the client-server relationship during a session – to ensure that applications continue to behave as expected.

The problem is that when you move an application the mechanism for that persistence may very well depend on application-specific data. While load balancers and intermediaries can provide persistence based on other information such as client-IP, this is not always optimal as many clients may be arriving from the same IP (the mega-proxy problem). In traditional environments this persistence is achieved via a unique identifier created by the application – usually ASPSESSIONID, PHPSESSIONID, or JSPSESSIONID. Notice that they are ***not*** the same; they are dependent on the language in which the application was developed. The infrastructure – specifically the load balancer in this case – has no clue how to provide the necessary persistence. It's got to be manually configured, along with application specific meta-data for QoS, rate shaping, compression, caching, response time requirements, and even the maximum number of instances allowed to execute at any given time.

What's needed – or at least what would greatly forward the interoperability and automation of provisioning in cloud environments via OVF - is application-specific data that is needed by the infrastructure to properly configure it in real-time. Somewhere in the OVF meta-data wrapping that application there needs to be some entry indicating what data value or HTTP header should be used for persistence, among other attributes and properties necessary for the deployment of an application.

Response-time thresholds, quality of service, and even access rights to the application are also ***application specific*** policies that are traditionally applied at strategic points in the data path – on intermediary infrastructure. What about context? What about policies that differentiate based on not only application, but user and location? This meta-data should be carried along with the application and a mechanism made available for infrastructure to interpret and apply appropriate policies. Without this meta-data and a method of automatically communicating it, provisioning of the infrastructure is still a manual process. And it's more than "a little", at that.

---

**OVF and INFRASTRUCTURE 2.0**

OVF certainly appears capable of providing support for the type of meta-data necessary to assist infrastructure in understanding how to properly deliver the application it describes. OVF is designed to be extensible, so the addition of these infrastructure-focused application-specific details should be not be problematic.

But simply adding infrastructure-focused meta-data won't ensure interoperability. What's necessary is a standardized mechanism for describing that meta-data. This need for more structure around application meta-data is one that is heavily tied to the success of Infrastructure 2.0. It's the difference between connected and collaborative; a dynamic infrastructure requires collaboration, not just more connections. OVF can certainly be one path to helping Infrastructure 2.0 solutions apply their dynamic nature in a way that benefits the portability process, and enables a more seamless deployment even within a cloud environment (internal or external, public or private) as applications move from one resource to another.

This may be a more difficult task than it first appears. Differences in terminology across infrastructure solutions cause more than enough confusion today (pool? farm? cluster? factotum?) and would certainly be problematic in defining what should be used within OVF to describe meta-data. There is also the issue of communicating with the intermediary infrastructure that it can automatically create and/or provision the appropriate policies. There is no standard method of this, either, which only further complicates the matter.

Without an agreed upon standardized mechanism for describing this meta-data, a single application descriptor in OVF could become bloated as each vendor adds product-specific language. That's not something that will be beneficial to the user. Making application images larger by including all the possibilities for infrastructure support would not help anyone but cloud providers that benefit from the additional revenue collected based on bandwidth transfers.

So if some of us aren't talking more about OVF, it might be because right now OVF isn't enough to support true portability across cloud environments. It might be because we just haven't gotten that far yet. It might be that pundits keep telling people they don't need to care about the infrastructure, that's all taken care of for them. This attitude is actually hindering efforts to raise the status of infrastructure in any cloud-based or virtual environment to its proper significance. If you tell people often enough they don't have to worry about something, they won't. After all, they have enough on their plates without adding to it with concerns over provisioning and automation and infrastructure.

Do we need to talk more about OVF? Perhaps. If it can support Infrastructure 2.0 requirements – the connectivity intelligence and meta-data necessary for that level of collaboration – then we (infrastructure players) definitely need to start talking about it, at least in terms of how we might leverage to include the basis for automating a fully dynamic infrastructure.

If not, then we need to start talking about what *can* support those requirements.

- Who owns application delivery meta-data in the cloud?
- More on the meta-data menagerie
- Interoperability between clouds requires more than just VM portability
- Cloud interoperability must dig deeper than the virtualization layer
- Making Infrastructure 2.0 reality may require new standards
- DMTF – VMAN (OVF Specifications)

- Dynamic Infrastructure: The Cloud within the Cloud
- The Context-Aware Cloud
- Infrastructure 2.0: As a matter of fact that isn't what it means

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

| | | | |
|---|---|---|---|
| F5 Networks, Inc.<br>Corporate Headquarters<br>info@f5.com | F5 Networks<br>Asia-Pacific<br>apacinfo@f5.com | F5 Networks Ltd.<br>Europe/Middle-East/Africa<br>emeainfo@f5.com | F5 Networks<br>Japan K.K.<br>f5j-info@f5.com |