Managing Virtual Appliance Lifecycle in IaaS and PaaS Clouds

Michal Kimle[*], **Ľubomír Košarišťan, Boris Parák, Zdeněk Šustr**
CESNET z. s. p. o.
Zikova 4, 160 00, Praha 6, Czech Republic
E-mail: michal.kimle@cesnet.cz, lubomir.kosaristan@gmail.com, boris.parak@cesnet.cz, zdenek.sustr@cesnet.cz

Both the IaaS (Infrastructure as a Service) and PaaS (Platform as a Service) models of providing cloud services rely on virtual appliances. In popular terms, they are "images" of either bare operating systems, typically entailing popular Linux distributions, which can be further contextualized once users instantiate their own virtual resources, or operating systems with applications pre-installed for use in the given platform, which may often consist of a number of complimentary appliances. Such appliances must be offered to users of any cloud service – they are the basic units the users see and select from when they decide to procure resources in the cloud. Understandably, cloud service providers are often expected to offer a variety of appliances. Even in a simple IaaS scenario, users expect to see a range of OS distributions and flavours. With PaaS, the variety is even greater. Obviously a range of appliances can be obtained from cloud marketplaces, but that only offsets rather than solves the problem since the challenges of maintaining their appliances are the same for local cloud site administrators and marketplace maintainers alike. This, inevitably, means that cloud site or marketplace administrators must not only offer a selection of appliances, but also manage them throughout their life cycle, keep them secured and updated, and eventually discontinue them when the time comes. It is not only cumbersome but also inherently insecure to leave updates to the user instantiating the given appliance. On top of that, the ability to always offer "fresh" appliances to its users is a competitive advantage a cloud site may wish to exploit. This paper introduces a concept of automated periodic appliance updates in a federated cloud environment, alongside actual tools developed to perform that task. It also sums up up-to-date experience with operating such tools in the European Grid Initiative's Federated Cloud.

International Symposium on Grids and Clouds (ISGC) 2016
13 – 18 March, 2016
Taipei, Taiwan

[*]Speaker.

1. Introduction

This article introduces a concept of periodic cloud appliance updates and redistribution to cloud sites integrated into a federated cloud environment, alongside tools that actually implement such appliance life cycle in the real world. Firstly, Section 2 is going to outline the ground rules of virtual appliance preparation and maintenance, and introduce COMFY – a tool developed to automate that task. Then, Section 3 will show what additional tasks must be performed to make such appliances available within a cloud site, or even across a heterogeneous cloud infrastructure, and use tools developed for the EGI Federated Cloud platform as examples. Finally, plans for future development will be introduced in Section 4 and the impact of the work done will be summed up in Section 5.

2. Appliance Preparation Principles

Whether it is IaaS or PaaS, appliances became the cornerstones of today's clouds. Cloud users expect an ability to choose from a variety of ready-to-use appliances, prepared by the cloud providers for them. This is a trend that has to be accepted and embraced by cloud providers. Cloud providers are therefore presented with challenges brought by these requirements.

2.1 Requirements for Cloud Appliances

Both cloud appliances for IaaS and PaaS share the same foundation – an operating system. More and more popular choice of operating system among cloud users is Linux and its derivatives. Linux is an open source operating system kernel well known mainly among more technical audience. Thanks to its openness, throughout the years, users and companies created numerous so-called distributions – collections of software easily accessible and usable by users. There are hundreds of Linux distributions in existence but only a few have a steady place on the market and are considered to be the leaders. Hence, cloud providers are basically forced to support at least these few leading distributions to provide users with a choice. Despite the fact that all the distributions are based on the Linux kernel, each distribution is a bit different with different installation process and configuration. That said, creation of only a few of virtual machine images with just a base operating system can be a complicated and time-consuming task, especially when done repeatedly and manually.

Once the appliance image is prepared and the base operating system installed, it has to be configured for a smooth run within a cloud platform. Again, different distributions require different sets of configurations but all the appliances share the same concepts of what should be pre-configured so that users can be provided with a practical starting point. One necessary functionality that has to be configured is remote access, allowing users to access the virtual machine once it is instantiated from the image. Users need to gain control over the virtual machine either via SSH (accessing command line), VNC (accessing graphical interface) or any other way that is desirable and acceptable by users. In case of PaaS clouds, users may be restricted to use only appliance-specific tools via some specialized interface, e.g., a web console. Granting users access rights to the virtual machine is doubtless a necessity but it may open the appliance to security threats. Firewall configuration is therefore another crucial part of the appliance preparation process.

The last step in the appliance building process is usually the installation of appliance specific software. In most cases, this part applies only to PaaS clouds but there may be situations where an IaaS cloud provides virtual machines with pre-installed tools. Software installation may once again differ for various distributions, which renders the whole appliance production process unmaintainable when done manually.

Most of the cloud platforms today come with a feature called contextualization, which allows users to make certain configuration modifications within the appliance prior to its instantiation. This is usually achieved by various methods from simple bash scripts run on virtual machine's first start to complex solutions, e.g., cloud-init [1] that can easily manage users, configuration and services of the launched virtual machine. Support of this feature, especially with mechanisms like cloud-init, needs to be incorporated into the appliance preparation process so it can be ready when users want to utilize such a component.

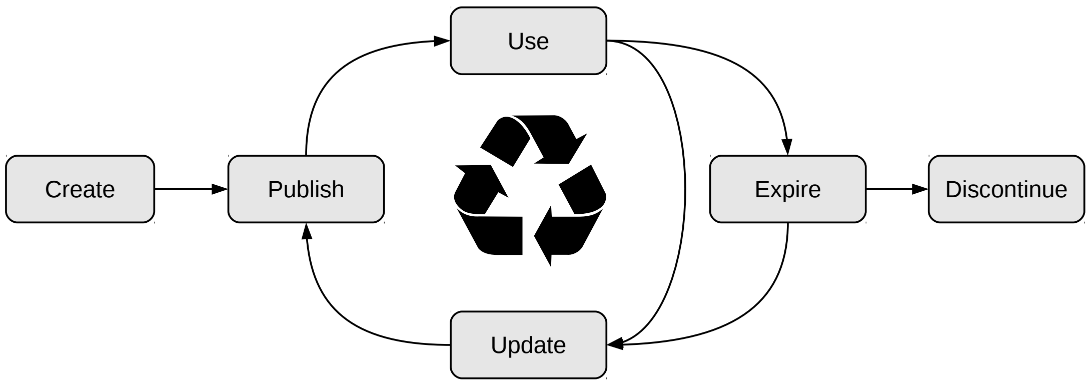

Figure 1: Appliance life cycle

This section makes it obvious that appliance generation is a complex process which can introduce nontrivial problems even for a low number of appliances. Furthermore, it is a process that has to be performed repeatedly and frequently (Fig. 1). All these facts lead to the conclusion that this process cannot be maintained manually but has to be fully automated. Automation using proper tools can solve multiple problems presented herein and increase overall productivity. There are multiple tools available to address one or more phases of appliance preparation. The next section will introduce a tool titled COMFY, which provides an all-in-one solution for the appliance production process.

2.2 Automated Cloud Appliances Production

As mentioned in Section 2.1, automation is a must in order to manage adequate appliance production process. To achieve this goal, a tool named COMFY [2] has been designed and developed. The main purpose of COMFY (which is an acronym for Cl<u>o</u>ud I<u>m</u>age <u>F</u>actor<u>y</u>) is to easily create virtual appliance images of multiple Linux distributions with only a little or no configuration. COMFY is currently capable of delivering images of most popular cloud distributions, namely Debian, Ubuntu, CentOS and Scientific Linux, which are pre-configured for effortless use in a cloud environment.

An image build is guided by appliance description, partly consisting of the overall COMFY configuration (for common options such as disk size or image format), and partly of appliance description files, which are specific for each appliance. Appliance description files provide more advanced options that can be used to optimize the building process and define custom modifications including disk partition layout and options for virtualisation platforms. COMFY ships with sensible default configuration so that adjustments are in most cases not necessary.

The basic component of COMFY's building process is a software developed by HashiCorp called Packer [5]. Packer is a tool for creating virtual machine images for multiple platforms from a single source configuration. It can use multiple virtualisation tools, e.g., VirtualBox [6], QEMU [7], OpenStack [8] and others to create a virtual machine and setup an installation process. If needed, a local Web server is launched to serve files required by distribution installation process.

At the end of the installation, before the virtual machine shuts down, Packer provides a provisioning feature allowing contextualization of the appliance just created. Again, multiple forms are supported from simple bash scripts run on a virtual machine to complex solutions, e.g., Puppet [10] or Chef [11] recipes. This way, software can be installed and fundamental configuration made in the appliance. COMFY currently uses provisioning via simple bash scripts but, as stated in Section 4, transition to Puppet recipes is in progress.

The last part of the appliance building process is the generation of appliance metadata in the form of a JSON-formatted descriptor. This descriptor contains information about the appliance's version, identifier, memory and CPU requirements, installed distribution and associated image files or any other arbitrary data. The main purpose of these metadata is an easy integration with OpenNebula cloud platform described in Section 3.1, but can be simply parsed by any other software managing appliance.

3. Appliance Distribution and Life Cycle Management

With the appliance (re)generated, further steps must be taken to make it available either as a new product, or a replacement for an older version. That task requires additional automation even in a single cloud site setup (Subsection 3.1) and becomes even more complicated if the appliance is to be made available on different cloud sites relying on different cloud management frameworks (Subsection 3.2).

3.1 Cloud Site Considerations – an OpenNebula-based Case Study

Appliance preparation is only a first step in the appliance life-cycle management. For an appliance to be made available to users, it has to be uploaded and registered either within a cloud platform, allowing users to instantiate the appliance, or to other image distribution platform providing users with access to bare images. But that is not the end of the life-cycle. Appliances need to be regularly updated, whether because of the new release of installed software or to patch security issues, and when the time comes, they also need to expire and be discontinued. Clearly, such a process can be time-consuming and requires automation.

The solution proposed to address this issue is a software product called NIFTY [12]. NIFTY stands for OpenNebula Image File Synchronization Utility and as the name suggests, it was designed to be used with the OpenNebula cloud platform. NIFTY is able to upload and register appliance images into OpenNebula cloud as well as handle appliance updates and expiration.

Despite its name and the fact that OpenNebula is currently the only supported cloud platform, NIFTY's modular design allows only a loose binding with OpenNebula and makes future support of other platforms simple enough without having to modify the core functionality of the application.

When started, NIFTY reads an appliance descriptor (described in Sections 2.2 and 3.2) to determine what is the next course of action. In case of a new appliance, an appliance image is uploaded into OpenNebula where it is registered and a virtual machine template is automatically created so that users can immediately instantiate virtual machines based on the new appliance. The image uploading process may vary for different platforms or even for different setups of the same platform. Because of this, NIFTY uses a concept of modular transport methods, each dealing with image upload differently and the one used is selected in NIFTY's configuration.

Should the appliance expire, NIFTY disables the appliance image and deletes its virtual machine template from OpenNebula, so that users are no longer able to instantiate the expired appliance. An appliance update is therefore a combination of expiring the old version and registering the new one. As described in Section 2, appliances expired for a longer period of time are unlikely to get updated and have to be removed. NIFTY includes a concept of a grace time period after which expired appliances are automatically deleted from the cloud.

3.2 Solutions for Large-Scale Heterogeneous Infrastructures

Large infrastructures add an additional level of complexity to the automation process as appliances must be shared among providers, distributed from one or more approved sources and made available in a uniform way across the whole infrastructure.

3.2.1 EGI Federated Cloud Image Distribution Mechanism

The EGI Applications Database [4] is a centralized service that stores information about software tools integrated with the EGI infrastructure. Among all the various types of items stored there, EGI AppDB contains a set of appliances which allow the community to use scientific software and other relevant tools easily in cloud. The Cloud Marketplace section stores images for the cloud to be made available as appliances. At first, however, there needs to be a mechanism that distributes these appliances across the EGI Federated cloud sites.

For this purpose, the vmcaster and vmcatcher are used [18]. Vmcaster is a simple tool for managing and updating a published virtual machines image lists, following the HEPiX image list format. More information on vmcaster can be found at its home GitHub repository page [14]. With this tool, Virtual Machine Image Lists with appliances from EGI AppDB are generated. Usually, image lists used in the infrastructure are so-called VO-wide lists. This means that the administrators of a virtual organization choose relevant appliances and create an image list for them.

If sites want to use images from a given image list, they need to subscribe to it. This subscription is provided by the second tool mentioned – the vmcatcher [15].

Vmcatcher allows users (for example site administrators) to subscribe to the given image list, cache the images referenced in it, verify the image list source with x.509-based public key cryptography, validate listed images against sha512 hashes given in the list and finally emit events

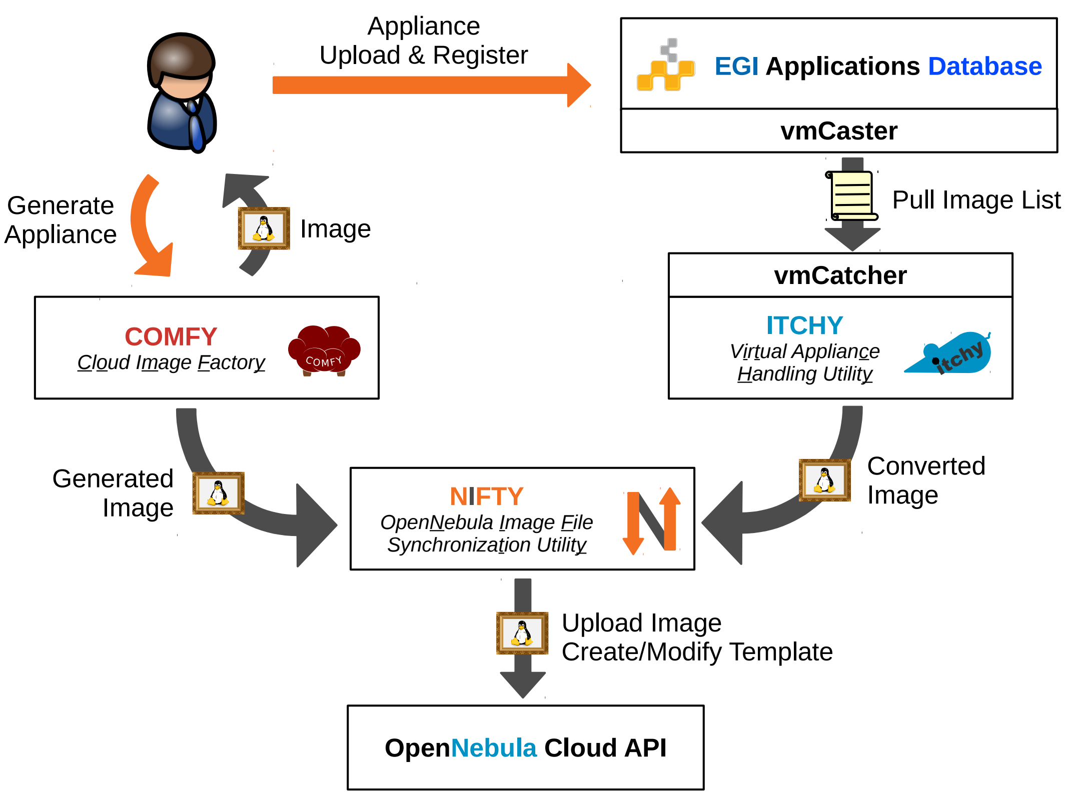

Figure 2: Overall design of the virtual appliance distribution and life cycle management system, showing individual components, their tasks and interactions

to be handled by subsequent applications that will appropriately process, update or expire virtual machine images without having to validate them themselves again.

There are various event types that can be provided by vmcatcher:

ProcessPrefix	Vmcatcher has started updating its cache.
AvailablePrefix	An image will be available soon – there is an ongoing attempt to retrieve it.
AvailablePostfix	An image was successfully validated as being available and placed in the cache directory.
ExpirePrefix	This image will no longer be available in the cache directory.
ExpirePosfix	This image is no longer in the cache directory.
ProcessPostfix	Vmcatcher has finished updating its cache.
SubscriptionImageNew	A new image exists in a subscribed image list.

3.2.2 Automated Distribution

So far, only NIFTY and vmcatcher were discussed. But there is a gap between these two tools, prohibiting their cooperation on automated distribution of appliances from AppDB to cloud storage. Images must be available in a proper format, with all the necessary information for successful registration in cloud storage. A tool titled ITCHY [13] aims to bridge this gap.

ITCHY stands for vIrtual applianCe Handling utilitY. It basically converts heterogeneous output from vmcatcher and prepares all necessary files for upload an registration into cloud storage.

ITCHY consists of two cooperating tasks. One serves for archiving vmcatcher events, second one for further processing. This concept was needed for two reasons. First reason is that vmcatcher runs regularly as a cron job. There is a possibility that image processing time would be greater than time period between two vmcatcher runs. In this case, the whole process would fail. The second reason is that it is impossible to interrupt vmcatcher event processing and start over. Therefore, the first part of ITCHY running with vmcatcher as event handler is as simple as it can be. The other acts separately and can be restarted or delayed if there is a need to do so since it cannot affect vmcatcher.

ITCHY's first task is archiving. As mentioned earlier, vmcatcher is designed to use an event handler, which should take the form of software hooked to vmcatcher's event processing. In our case, it is the `itchy archive`. This task takes information about vmcatcher events and stores it in JSON format for later processing.

The second task, `itchy process`, is more complex. It iterates trough all stored events and (as is easily guessed from its name) process them. From the list mentioned in Section 3.2.1, there are only two events that are interesting for its purpose. These are *AvailablePostfix* and *ExpirePostfix*. Other events are just detected, but do not require more attention.

The results of both events are almost the same – there is information about required changes in cloud datastore. Then, it is easy for another tool (in this case NIFTY) to apply these changes in the cloud.

The *AvailablePostfix* event is probably the most common. There is an image stored in vmcatcher's cache directory and it needs to be processed. ITCHY has to locate proper image file and check it. If it is in proper format, it is just copied to the output location, otherwise the original image file is transformed. If it is already in simple virtual disk format (*raw*, *qcow*, etc.), the conversion is rather easy. Numerous images are, however, saved as archives (*ova* or *tar*) and their processing is more complicated.

After an image is processed, its description file is created. This file contains all the necessary information required for successful appliance registration in the cloud. The information is loaded from the JSON event file created earlier by the ITCHY archive job. In the end, there is the description file stored with a link to the corresponding image file.

When an *ExpirePostfix* event arrives, ITCHY needs to create the information that the given image will be unavailable and needs to be disabled in the cloud datastore. Therefore a description file for expiration is created. This file contains information on how and which appliance should be disabled.

4. Future Work

All the tools described in this paper are rather new, hence the obvious next step is their field testing. At the time of writing this paper, both NIFTY and ITCHY are being deployed on sites engaged in EGI Federated Cloud initiative. This will hopefully bring in the much needed feedback and will help discover and solve any unforeseen issues. As for COMFY, appliance images generated by this tool have already been employed in CESNET's MetaCloud [9] for several months and multiple flaws were already discovered and fixed.

It is important to keep up with new trends and demands, therefore the next phase of the development plan, apart from fixing reported problems, is the addition of new features. In case of NIFTY, emphasis will be put on supporting other cloud and image distribution platforms, e.g., OpenNebula Marketplace [17] or EGI Application Database [4]. This will not affect the basic functionality but it will be merely an independent extension allowing NIFTY to operate with other platforms.

The majority of future development is planed for the COMFY component. Appliance creation is a complex process and there is much to improve to provide a simple but powerful building tool. As already mentioned in Section 2.2, provisioning with bash scripts will be replaced by more robust Puppet recipes. This transition will contribute to reusability of similar installation directives among multiple distributions as well as to an ability to easily separate unique ones. Since Puppet recipes are a wide-spread and popular concept users are well familiar with, it may lead to an increase of custom-made appliances. Furthermore, COMFY will support a concept of appliance hierarchy and inheritance, allowing custom appliance builders to inherit from other appliances and make modifications without having to create an appliance from scratch. The same method is used by Docker [19] images and is proven to be a favourable one.

In relation to Puppet recipes, future releases are planned to support retrieval of information on appliances to be built from *git* [20] repositories. This will introduce a more modular approach and will decrease the size of distributed software, because necessary files will be downloaded on demand during the building process. Such a solution is popular in open-source communities and many open-source projects are built in a similar way.

5. Summary

In this paper, a concept of Virtual Appliance Lifecycle Management has been described. This paper has discussed the basic requirements of appliance principles and their usage in cloud. In addition, a fully functional complex system of tools, used for automation of creating, distribution and life cycle management, was presented, together with some possible improvements scheduled for the future.

The principles and tools outlined herein are partly already in production, partly in the process of being deployed in a large-scale heterogeneous cloud federation.

6. Acknowledgements

This work is co-funded by the EGI-Engage project (Horizon 2020) under Grant number 654142.

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

References

[1] *cloud-init*, [Online] Available: https://cloudinit.readthedocs.org/en/latest/ [Accessed: November 11, 2016].

[2] *COMFY*, [Online] Available: https://github.com/CESNET/comfy [Accessed: November 11, 2016].

[3] European Grid Infrastructure, *Federated Cloud*, [Online] Available: https://www.egi.eu/infrastructure/cloud/ [Accessed: November 11, 2016].

[4] European Grid Infrastructure, *Applications database*, [Online] Available: https://www.egi.eu/services/catalogue/appdb.html [Accessed: November 11, 2016].

[5] HashiCorp, *Packer*, [Online] Available: https://www.packer.io/ [Accessed: November 11, 2016].

[6] *VirtualBox*, [Online] Available: https://www.virtualbox.org/ [Accessed: November 11, 2016].

[7] *QEMU*, [Online] Available: http://wiki.qemu.org/Main_Page [Accessed: November 11, 2016].

[8] *QEMU*, [Online] Available: https://www.openstack.org/https://www.openstack.org/ [Accessed: November 11, 2016].

[9] MetaCentrum, *HPC Cloud interface*, [Online] Available: http://www.metacentrum.cz/en/cloud/index.html [Accessed: November 11, 2016].

[10] Puppet Labs, *What is Puppet?*, [Online] Available: https://puppetlabs.com/puppet/what-is-puppet [Accessed: November 11, 2016].

[11] *Chef*, [Online] Available: https://www.chef.io/chef/https://www.chef.io/chef/ [Accessed: November 11, 2016].

[12] *NIFTY*, [Online] Available: https://github.com/CESNET/nifty [Accessed: November 11, 2016].

[13] *ITCHY*, [Online] Available: https://github.com/CESNET/itchy [Accessed: November 11, 2016].

[14] *vmcaster Source*, [Online] Available: https://github.com/hepix-virtualisation/vmcaster [Accessed: November 11, 2016].

[15] *vmcatcher Source*, [Online] Available: https://github.com/hepix-virtualisation/vmcatcher [Accessed: November 11, 2016].

[16] HEPIX Virtualisation Working Group, August 26, 2012, [Online] Available: http://grid.desy.de/vm/hepix/vwg/doc/pdf/Book-a4.pdf [Accessed: November 11, 2016].

[17] C12G, *OpenNebula Marketplace*, [Online] Available: http://marketplace.c12g.com/appliance [Accessed: November 11, 2016].

[18] HEPIX Virtualisation Working Group, August 26, 2012, [Online] Available: http://grid.desy.de/vm/hepix/vwg/doc/pdf/Book-a4.pdf [Accessed: November 11, 2016].

[19] *Docker*, [Online] Available: https://www.docker.com/ [Accessed: November 11, 2016].

[20] *git-scp*, [Online] Available: https://git-scm.com/ [Accessed: November 11, 2016].