Ingeniería, investigación y tecnología
ISSN: 1405-7743
ISSN: 2594-0732
Facultad de Ingeniería, UNAM

García-Perellada, Lilia Rosa; Vega-Gutiérrez, Sergio; Fé-Herrero, José Manuel de la; Rodríguez-De Armas, Yalina; Garófalo-Hernández, Alain Abel

Driver LXC Development for OpenNebula

Ingeniería, investigación y tecnología, vol. XIX, no. 1, 2018, January-March, pp. 63-76
Facultad de Ingeniería, UNAM

DOI: https://doi.org/10.22201/fi.25940732e.2018.19n1.006

Available in: https://www.redalyc.org/articulo.oa?id=40458280006

Ingeniería Investigación y Tecnología
volumen XIX (número 1), enero-marzo 2018 63-76
ISSN 2594-0732 FI-UNAM artículo arbitrado
Información del artículo: recibido: 7 de octubre de 2016, reevaluado: 12 de marzo de 2017, aceptado: 3 de julio de 2017

DOI: http://dx.doi.org/10.22201/fi.25940732e.2018.19n1.006

Driver LXC development for OpenNebula
Desarrollo de un driver LXC para OpenNebula

García-Perellada Lilia Rosa
José Antonio Echeverría Higher Polytechnic Institute, La Habana, Cuba
Electrical Engineering Faculty
Telecommunication and Telematics Department
E-mail: lilianrosa@tele.cujae.edu.cu

Rodríguez-De Armas Yalina
José Antonio Echeverría Higher Polytechnic Institute, La Habana, Cuba
Faculty of Electrical Engineering
Information and Communication Technologies Services Department
E-mail: yalina@electrica.cujae.edu.cu

Vega-Gutiérrez Sergio
José Antonio Echeverría Higher Polytechnic Institute, La Habana, Cuba
Electrical Engineering Faculty
Telecommunication and Telematics Department
E-mail: sergiojvg92@gmail.com

Garófalo-Hernández Alain Abel
José Antonio Echeverría Higher Polytechnic Institute, La Habana, Cuba
Electrical Engineering Faculty
Telecommunication and Telematics Department
E-mail: aagarofal@gmail.com

De la Fé-Herrero José Manuel
José Antonio Echeverría Higher Polytechnic Institute, La Habana, Cuba
Electrical Engineering Faculty
Telecommunication and Telematics Department
E-mail: mdelafe92@gmail.com

Abstract

Operating system level virtualization is a technology that has recently emerged into the cloud services paradigm. It has the advantage of providing better performance and scalability than para-virtualized or full virtualization hypervisors. This solution is getting acceptance into cloud infrastructures. Nowadays public cloud Infrastructure as a Service providers offer applications based in Docker containers deployed on virtual machines. Only a few bring Infrastructure as a Service on a bare metal container infrastructure. In the private cloud scenario, however, it hasn't had a wide acceptance. Private cloud managers, like OpenStack, OpenNebula and Eucalyptus, don't offer good support for it. OpenNebula is a flexible cloud manager, which has been gaining a lot of market over the last years, so it seemed a good idea to strengthen the operating system virtualization support in this cloud manager. This will contribute to achieve better interoperability, performance and scalability in OpenNebula clouds. Therefore, the objective of the present work was to implement a driver to support Linux Containers for OpenNebula. The driver has several features such as: the ability to deploy containers on File Systems, on Logical Volume Managers and on Ceph; it's able to attach and detach network interface cards and disks while the container is on; and it's able to monitor and limit container's resources usage.
Keywords: containers, LXC, OpenNebula, operating system virtualization.

Resumen

La virtualización de sistemas operativos es una tecnología emergente en el paradigma de la Computación en la Nube, presentando mejores índices de desempeño y escalabilidad que los hipervisores soportados por la virtualización completa o por la para-virtualización. Actualmente abre paso en las infraestructuras de Nubes. Proveedores de Infraestructura como Servicio brindan servicios basados en contenedores sobre máquinas virtuales, con soluciones como Docker. Pocos brindan Infraestructura como Servicio sobre una plataforma de contenedores bare-metal. En las Nubes Privadas, sin embargo, los gestores de infraestructuras como OpenStack, OpenNebula y Eucalyptus, le brindan muy poco soporte, o nulo, a esta tecnología. OpenNebula, gestor con aceptación en el mercado, dada su flexibilidad, modularidad, interoperabilidad, usabilidad y ligereza, podría enriquecerse con la integración de una solución de contenedores, lo que les añadiría a su infraestructura mayor eficiencia. Es por ello que el objetivo trazado en el presente trabajo fue el desarrollo de un driver para OpenNebula, que le permitiese soportar LinuX Container, una de las principales soluciones de virtualización de sistemas operativos actualmente. El driver obtenido soporta funcionalidades como el despliegue de contenedores sobre Sistemas de Ficheros, Volúmenes Virtuales y Ceph, adicionar y eliminar interfaces de red, y discos a los contendores en caliente.
Descriptores: OpenNebula, contenedores virtuales, LXC, virtualización de sistemas operativos.

DOI: http://dx.doi.org/10.22201/fi.25940732e.2018.19n1.006

Introduction

The majority of the widely deployed virtualization platforms in data centers and cloud infrastructures are based in full and para-virtualization technologies. Such is the case of Xen, *Kernel-based Virtual Machine* (KVM), VMware ESXi and Hyper-V hypervisors (Arceo *et al*., 2015). On the other hand, the *Operating System Level Virtualization* (OSLV) technology is getting acceptance into cloud infrastructures with solutions like: Docker, *LinuX Container* (LXC) and LXC's new interface LXD (LXC/LXD), which have their roots in the OSLV pioneer solution OpenVZ (Arceo *et al*., 2015; Agarwal, 2015; Wallner, 2015; 2014).

OSLV can be considered as a lightweight alternative to full and para-virtualization technologies. The main difference is that OSLV eliminates the hypervisor layer, redundant OS kernels, binaries and libraries needed to run workloads in *Virtual Machines* (VMs). Hypervisors abstract hardware, which results in overhead in terms of virtualizing hardware and virtual device drivers. A full OS is typically run on top of this virtualized hardware in each VM instance. In contrast, containers implement isolation of processes at the OS level, thus avoiding such overhead. These containers run on top of the same shared OS kernel of the underlying host machine, and one or more processes can be run within each container. Due to the shared kernel, as well as the OS libraries, container based solutions can achieve higher density of virtualized instances with better performance compared to hypervisor based solutions, bringing thus better efficiency in a *Data Center* (DC) infrastructure (Arceo *et al*., 2015; Agarwal, 2015; Wallner, 2015; Morabito *et al*., 2015; Graber, 2015a; Petazzoni, 2015).

OSLV has been around for over 18 years, but its adoption has been hindered in DC infrastructures because of the shared kernel approach and the mechanisms to achieve the resource isolation, which can be an issue for multitenant security. Nevertheless, nowadays, OSLV supports a variety of technologies to mitigate most security concerns, removing this drawback. The main tools are: namespaces, specially user namespaces, control groups (cgroup) and *Linux Security Modules* (LSMs). Namespaces give the containers their own view of the system, limiting what containers can see and therefore use, while cgroup limits how much they can use, achieving resource isolation. User namespaces define by default unprivileged containers, which are safe by design. The container uid 0 is mapped to an unprivileged user outside of the container and only has extra rights on resources that it owns itself. Thus LSMs like SELinux, AppArmor and Seccomp are not necessary, although solutions like LXC and LXC/LXD use them to add an extra layer of security which may be handy in the event of a kernel security issue. Cgroup, restricts the use of physical resources like the *Central Processing Unit* (CPU), the *Random Access Memory* (RAM) and storage devices, through the establishment of quotas and priorities to containers, avoiding potential *Denial of Services* (DoS) attacks. (Petazzoni, 2017; Graver 2014a: 2016a). These alternatives are supported by leading OSLV exponents today like LXC, LXC/LXD and Docker. In addition, *Cloud Service Providers* (CSPs) like Joyent, Kyup and ElasticHosts, are offering *Infrastructure as a Service* (IaaS) based on bare metal container infrastructures (Graber, 2014b; 2017a, b, c; 2016b, c). So it is time to exploits the advantages of the OSVL in DC infrastructures, especially in *Small and Medium-sized Enterprises* (SMEs).

Nowadays public cloud IaaS providers, like Amazon (Graber, 2015b), offers applications based in Docker containers deployed on VM. Joyent, however, brings IaaS on a bare metal containers infrastructure, which completely enjoys the OS virtualization's advantages (Graber, 2015c). Joyent's solution for deploying containers is named Triton (Graber, 2015b; Cantrill, 2014). It is free and open source (Cantrill, 2014). In the private cloud scenario, however, the OS virtualization technology hasn't a wide support. Private cloud managers, like OpenStack, OpenNebula and the Elastic Utility Computing Architecture for Linking your Programs to Useful Systems (Eucalyptus), don't offer good support for this technology, if they support it at all.

OpenStack stands out because it gives support to LXC, Docker and LXD, but this support is on early stages (Cantrill, 2017a, b, c). However, OpenStack is not the best solution for all the entities. It is a complex *Cloud Management Platform* (CMP) with a steep learning curve and with high hardware requirements for its deployment compared with others like OpenNebula (Chilipirea *et al*., 2016a). OpenStack in a basic ready production deployment with high availability, suggests at least seven physical hosts for supporting its controller services, which are recommended to not be virtualized (Cantril, 2017a; Chilipirea *et al*., 2015). Hosts minimum features proposed are 32GB of RAM, 2x Intel® Xeon® CPU E5-2620 0 @ 2.00 GHz and two Network Interface Cards (NICs) at 10Gbps (Chilipirea *et al*., 2015a). On the other hand OpenNebula requires only two VMs with 2GB of RAM, two CPUs and two NICs for a production private cloud (Chilipirea *et al*., 2016b). So, a lightweight, flexible, scalable and easy to use CMP could be the efficient solution for SMEs, especially for those with restricted budgets,

DOI: http://dx.doi.org/10.22201/fi.25940732e.2018.19n1.006

that do not require high compute resources for supporting their *Information Technology* (IT) services, but do need the benefits of the private cloud paradigm.

OpenNebula is an open-source solution, used in several types of environments. Leading organizations, like *National Aeronautics and Space Administration* (NASA) and the Tokyo Institute of Technology in Supercomputing field, use OpenNebula to build enterprise private clouds, hosting, public cloud services, high performance computing and science clouds. Some features that makes OpenNebula a wise choice are: a powerful user security management, support of multi-tenancy with group management, on-demand Provision of *Virtual Data Centers* (VDCs), control and monitoring of virtual and physical infrastructures, distributed resource optimization, management of multi-tier applications, standard cloud interfaces and simple provisioning portal for cloud consumers; broad commodity and enterprise platform support, such as a broad hypervisor and storage technologies support; and easy extension and integration, it is a modular and extensible architecture able to fit into any existing DCs with customizable drivers. It is a fully open-source technology available under Apache License and new drivers can be easily written in any language (Chilipirea et al., 2015b, c).

OpenNebula's features directly contribute to achieve the functional and nonfunctional requirements of a cloud DC. However, its interoperability, adaptability, feasibility and contribution to an efficient infrastructure with good levels of performance and scalability, can be improved with a reliable support of an OSLV solution. Two drivers were developed for the supporting of OpenVZ and LXC, one by China Mobile and the other by Valentin Bud respectively. Both had deficient features and functionalities, and poor support (Chilipirea *et al*., 2012; 2016c).

Therefore, the objective of the present work was to implement a driver for supporting one of the main OS virtualization solutions by OpenNebula. The driver developed was for LXC. It supports the following features:

- Deploy containers on *File Systems* (FS), *Logical Volume Manager* (LVM) and Ceph.
- Stop, shutdown, reboot, suspend and resume containers.
- Live attach and detach NICs and disks.
- Monitor hosts and containers.

This paper treats the following topics: reasons for choosing LXC to integrate to OpenNebula; OpenNebula's components and interfaces needed for the driver development; the creation of the driver LXCoNe and; the proofs of concept done to demonstrate its functionalities.

Why LXC?

LXC was selected by the authors of the present paper for being integrated to OpenNebula because it's a stable solution that supports the majority of the most important features of Linux containers; and because it has a big community developing new functionalities that improve its usability, security, performance and fault tolerance isolations, which can be proved with LXD (Chilipirea *et al*., 2015d; e; f; 2011; 2015g). LXC is a derivation of OpenVZ, supported by Canonical Ltd., which shares many of the same developers as OpenVZ (Wallner, 2015). It is a collection of user-level tools that assists in the creation, management and termination of containers, which is included in most Linux distributions. In most cases installation is as simple as selecting it in the package manager, removing one of the main OpenVZ detractions (Aderholdt *et al*., 2014). LXC has other advantages like the supporting of:

- A liblxc library and *Application Programming Interfaces* (APIs) for Ruby, Python 2 and 3, Haskell and Go.
- Different storage systems: *Network Attach Storage* (NAS), *Direct Attach Storage* (DAS) (ext4 and B-tree FS (btrfs)) and *Storage Area Networks* (SANs).
- Any disk format.
- Virtual SANs: *Distributed Replicated Block Device* (DRBD) 9, Ceph and GlusterFS.

However, it doesn't support some features like OpenVZ: live migration and storage *Quality of Service* (QoS) (Chilipirea *et al*., 2015g). But LXC has LXD, a new project of Canonical Ltd., aimed at revitalizing the use of LXC (Scott, 2015a). It is intended (Aderholdt *et al*., 2014; Scott, 2015a, b; Banerjee, 2014; Ectors, 2014a, b; 2016; 2014):

- To make LXC-based containers easier to use through the addition of a back-end daemon supporting a *Representational State Transfer APIs* (REST APIs) and a straightforward *Command Line Interface* (CLI) client that works with both the local daemon and remote daemons via the REST API.
- To be image based. No more distribution templates, only good, trusted images.
- To support live-migration and snapshotting.
- To support vSecure by default, with AppArmor, user namespaces and Seccomp.

LXD isn't a rewrite of LXC, in fact it's building on top of LXC to provide a new, better user experience. Under the hood, LXD uses LXC through liblxc and its Go binding to create and manage the containers. It's basically

DOI: http://dx.doi.org/10.22201/fi.25940732e.2018.19n1.006

an alternative to LXC's tools and distribution template system with the added features that come from being controllable over the network (Scott, 2015b). So, why not LXD? While LXC is stable, LXD is still undergoing a rapid development. Some features haven't been implemented yet and the documentation is still a bit on the light side (Aderholdt *et al*., 2014; Scott, 2015a, b).

Docker, although it is focused on being the universal container for applications (Ectors, 2014c), was not selected because it is considered by the community and by the authors of the present paper a solution suited for providing *Platform as a Service* (PaaS) (Banerjee, 2014b), not IaaS, for the main following reasons:

- It restricts the container to a single process only. The default Docker base image OS template is not designed to support multiple applications, processes or services like init, cron, syslog and Secure SHell (SSH). This introduces a certain amount of complexity for day to day usage scenarios, since current architectures, applications and services are designed to operate in normal multi process OS environments (Banerjee, 2014b, 2015a; Wallner, 2015).
- It separates container storage from the application, which eliminates one of the biggest features of containers for end users, easy mobility of containers across hosts (Banerjee, 2014b, 2015a; Wallner, 2015).

Besides, Docker was initially based on the LXC project, although it has now developed its own implementation libcontainer that uses kernel namespaces and cgroups directly (Ectors, 2014c). This makes Docker not a virtualization solution, but one that automates the deployments of applications inside containers, by providing an additional layer of abstraction and automation of the OSLV.

Components and APIs of OpenNebula Needed for Integrating the LXC Driver

In order to achieve the goal of the present work, it was necessary to identify what OpenNebula offers to cloud integrators. The main strength is the modular and extensible architecture of OpenNebula, which has been designed to be easily adapted to any infrastructure and easily extended with new components (Banerjee, 2015b). Figure 1 shows the OpenNebula's architecture and the components and interfaces used for the driver development. The Virtualization (VM driver) is in charge of all the interaction with the hypervisors. This driver had to be created by the authors of the present paper in order to manage LXC containers through OpenNebula. The Monitoring (IM driver) makes it possible to acquire real time information of the host and the VM deployed. This driver had to be created by the authors of the present paper in order to monitor LXC containers through OpenNebula. The VM-API and IM-API were needed for the driver to interact with the rest of the OpenNebula's sections.

LXC Driver Requirements

LXC driver for OpenNebula should be able to perform several actions, some of them mandatory and others optional but desirable. The following actions must be supported by the LXC driver:

- Deploy LXC containers.
- Limit container's resources usage: disk quotas, In/Out (I/O) rate limiting, RAM limits, CPU quotas and network isolation.
- Reboot, reset and shutdown containers.
- Monitor hosts and containers.
- Create, delete and revert snapshots.
- Provide support for DAS FS such as ext4 and btrfs.
- Provide support for SAN networks implemented with Ceph, *Internet Small Computers System Interface* (iSCSI) or *Fiber Channel* (FC).

The following actions are considered by the authors of this paper optional, but desirable:

- Provide support for NAS devices over the Network File System (NFS) and ZFS. This could provide compatibility.
- Hot attach/detach NICs and disks. This could provide elasticity, performance and usability.

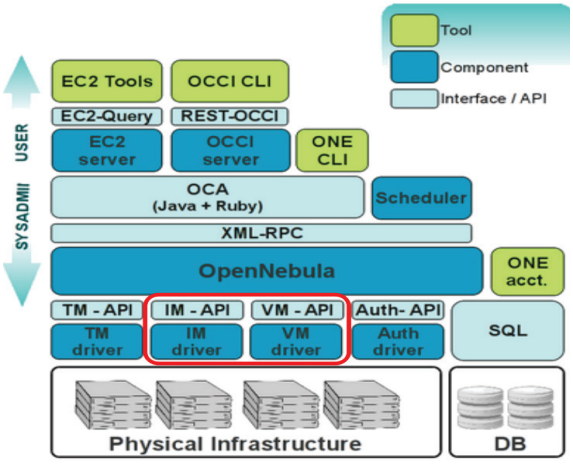

Figure 1. Components used in the OpenNebula's architecture (Banerjee, 2015b)

DOI: http://dx.doi.org/10.22201/fi.25940732e.2018.19n1.006

- Live snapshot. This could improve usability and availability.
- Live migrate containers. Besides usability and availability, this action could also provide elasticity.

Integrating LXC in Data Centers and Cloud Managers, Previous Work

Today's free and open source cloud managers that have worked in the support of LXC are OpenStack and OpenNebula. Proxmox 4.x, a free and open source DC manager for small enterprises, is supporting LXC too. So, the drivers of these solutions were analyzed in order to be aware of their advantages and drawbacks.

Two previous LXC drivers for OpenNebula were found, one made by China Mobile and the other by Valentin Bud:

- The driver from China Mobile is not accessible (Banerjee, 2014c). However, its developers showed the bugs that had been found, such as the driver can't implement reboot, shutdown and restart operations, and explained that the reason behind them could be the use of libvirt. This was useful, because their experience gave reasons to use liblxc instead of libvirt. Besides, their community announced that the driver was only able to monitor hosts, deploy and delete containers (Chilipirea, 2012).
- On the other hand, the driver from Valentin Bud was implemented directly over LXC (Bud, 2015a). It however only has support for LVM data stores and with very limited features. Some of these features does not work well, like container's monitoring. It has poor documentation and almost a year without any support (Chilipirea, 2016c). However the work from Valentin Bud gave an example about how to write the driver and how to organize it. It also showed how some features like monitoring and support for LVM could be implemented.

OpenStack, a cloud manager with a great market share, is actually developing its LXC driver. OpenStack allocates its LXC driver inside "Group C". Drivers inside this group "have minimal testing and may or may not work at any given time. Use them at your own risk" (Bud, 2015b). For this reason and because this driver was built over libvirt, it wasn't used as a reference.

Proxmox 4.x supports LXC. It's able to deploy, reboot, reset, shutdown and monitor containers. It can use DAS, SAN and NAS storages. It has support to: live snapshots, limit containers resources and live migration, although the latter is in the experimental phase yet. It has limitations such as it: only supports rootfs resizing through the *Graphical User Interface* (GUI) and does not support hot disk attach/detach (Bud, 2015c; d; e; 2016a) Proxmox 4.x was the reference for the authors of the present paper, for the development of the CPU limitations in the LXC driver for OpenNebula, and for its *Virtual Network Computing* (VNC) implementation.

(Bud, 2016b) states that libvirt-lxc is not generally recommended due to a lack of Apparmor protection for containers. This recommendation, together with the China Mobile experience, caused that the LXC driver for OpenNebula had been developed with liblxc instead of libvirt.

LXC Driver Development for OpenNebula, LXCoNe

OpenNebula manages the hypervisor underneath by running scripts on the host. Each operation that OpenNebula is able to perform over a hypervisor consists of a specific script located at **/var/lib/one/<driver-name>/<script-name>**. For example, the deploy action is a script called **deploy**, and for hot-attach a NIC there is a script called **attach_nic**. The name of the script is always suggestive. The most important and difficult action in this driver is to deploy a new container.

Figure 2 shows the script's basic blocks. The first step is to read all the information that will be used from the containers template and store it in variables. Once this is done, a folder that will contain all the necessary files for the container is created with the right permissions. This folder is created inside the folder configured as the default container location in the lxc user tools, for example **/var/lib/lxc** in Ubuntu. In this way the containers created by OpenNebula are shown at the output of **lxc-ls** command, which is necessary for the driver to be able to monitor containers. Then the configuration information for the NIC is extracted, arranged and prepared inside a variable in a format that LXC's configuration file understands. This process is simple and is shown in Figure 3. Now, the driver will proceed to configure the root storage as explained in Figure 4.

Because this driver supports three different storage types: FS, LVM and Ceph, it needs to find out which type it is and act accordingly. In case it is FS and LVM, the only thing that needs to be done is to indicate to LXC a path to the image. It is important to remember that OpenNebula uses images, either raw or qcaw, as virtual disks. LXC supports regular FS and LVM, so nothing else needs to be done, but in case the image is stored as a block device in Ceph, this image will need to be mapped in the host and provide LXC the route to where it was mapped. The reason why LXC doesn't

DOI: http://dx.doi.org/10.22201/fi.25940732e.2018.19n1.006

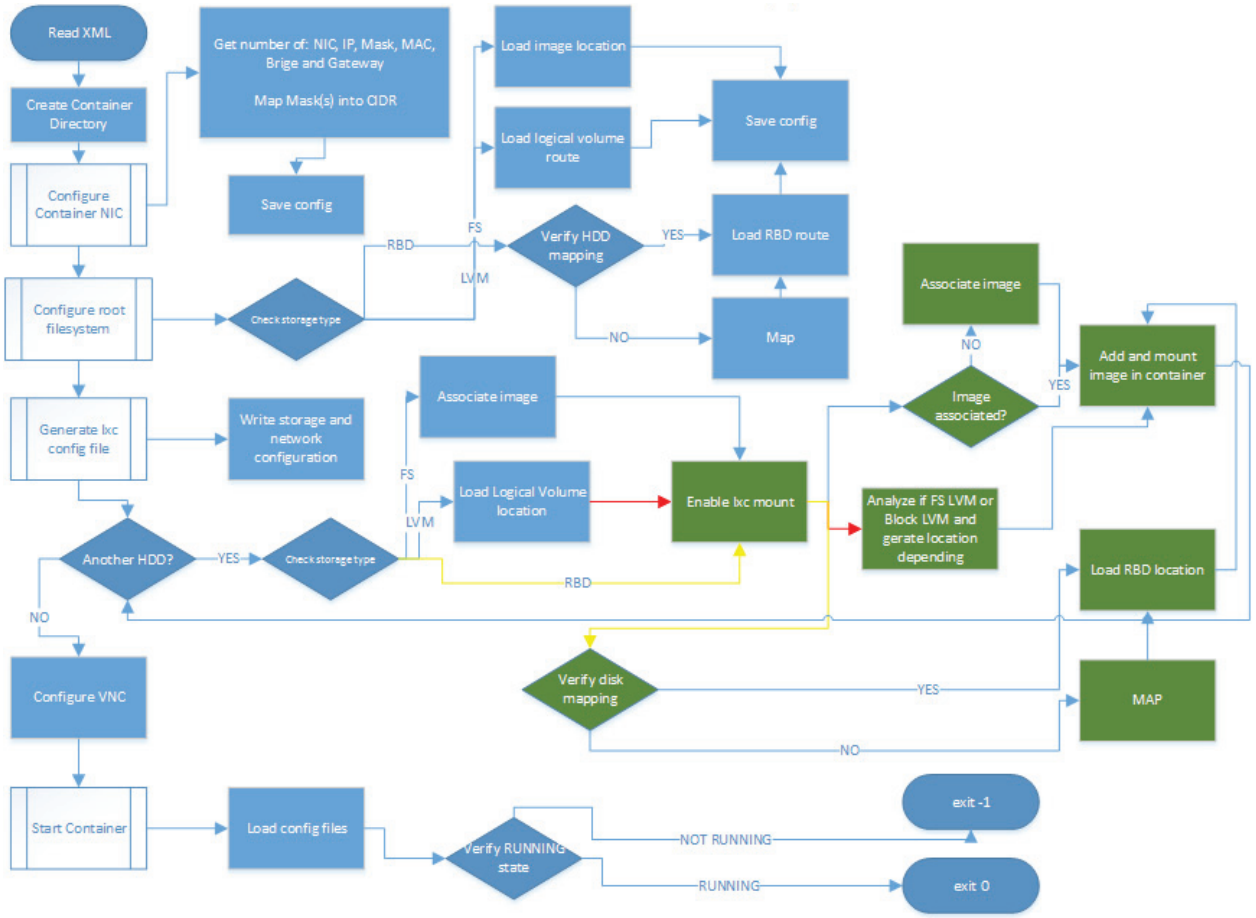

Figure 2. LXCoNe's main work-flow diagram

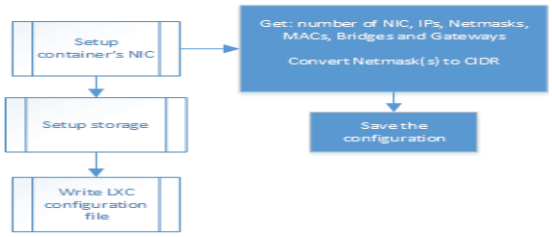

Figure 3. NIC Configuration

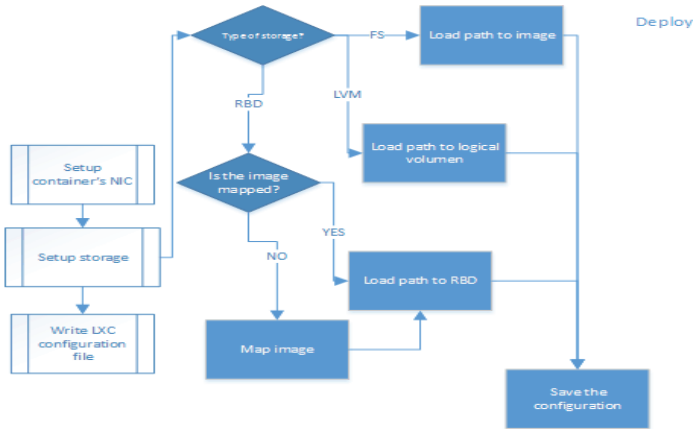

Figure 4. Root storage set up process

DOI: http://dx.doi.org/10.22201/fi.25940732e.2018.19n1.006

support Ceph's block devices is mainly because it can't perform this mapping action by itself, so it needs to be done by something else before the container starts. A possible approach here could be to perform this operation from OpenNebula itself. This makes sense because OpenNebula is already executing code, so the only thing that needs to be done is to write a line that tells the host to map the image. The problem with this approach is that, in case of an electrical failure or any other issue that could cause the physical host to crash, the containers that were running will not be able to start automatically. An administrator will need to manually redeploy them from OpenNebula. One of the goals wanted is precisely to avoid this, so another approach was used. The required instructions to map the root image must be executed once the host's OS initializes. Write it inside **/etc/rc.local** is a possibility. Also, containers must be configured to start automatically once the OS initializes.

The next step will be to generate LXC's configuration file. Inside this file will be located all the container's parameters, like NIC information, route to root storage, and resources limit. At this point the container is ready to start, but first must be checked if the user added another disk(s) to the container. If this results to be the case, the driver must be capable of attaching this disk(s) to the container. LXC allows to mount locations inside the container specified in the configuration file, but this is only useful to mount images when the container is going to be started, so hot-attach is not possible using this approach. A method that allowed mounting images inside the container while it was on needed to be found. Once a solution was found, it was implemented for the hot-attach disk action and also to add extra disks defined by the user before starting the container. The reason behind this was to have a single method to address both situations. LXC allows managing devices in running containers by using **lxc-device**. With this tool, a block or loop device from the host can be added to a running container, so it will be able to see it and mount it. Now, it could be possible to instruct OpenNebula to start the container and then use the previous defined method to attach any extra disk specified by the user. This solution will definitely work, but it has a major drawback, containers will only be able to be started and managed from OpenNebula. One of the goals wanted is to be able to manage containers either from OpenNebula, liblxc, a ssh session with the container or any other way after they were created by OpenNebula. The solution found was to use LXC's hooks. The instructions to add the device to the container and then mount it are not executed by OpenNebula, but written to the start hook of LXC. The script that represents this start hook is executed by LXC before it starts the container. These last two steps are explained in Figure 5.

Extra disks attachment is not the only thing that is configured in this hook at this point, neither the start hook is the only one used. The VNC session is configured in the start hook so it will be started with the container, and then used by OpenNebula. LXC's post-stop hook is also used. It will run at the node's namespace after the container has been shut down. With the help of this hook a cleanup process will occur. After a container is shut down, files like LXC container's configuration and hooks are left behind. Even these files are small they could cause problems once they accumulate after

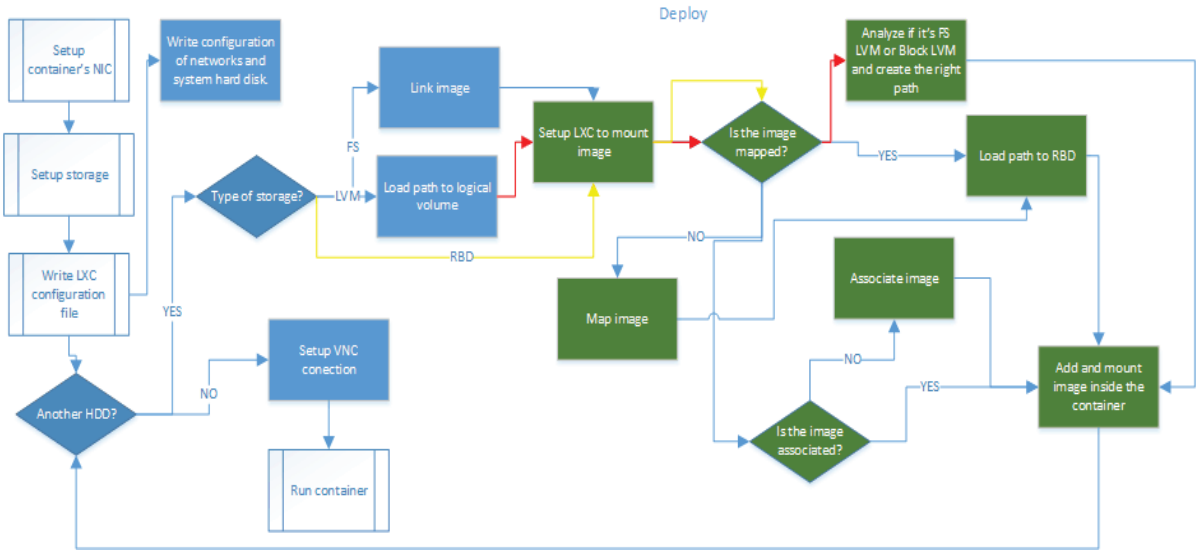

Figure 5. Last steps before the container is ready

DOI: http://dx.doi.org/10.22201/fi.25940732e.2018.19n1.006

some time. This cleanup process is in charge of erasing this files and any other remains of the container. Once this two hooks are created, OpenNebula will instruct LXC to start the container. After this, OpenNebula will check the container's status for a short while assuring it was successfully started, and no error occurred. In case of any error, OpenNebula will notice it, will change the status to **FAILURE** and will log it.

Because every configuration in the node that LXC might need to be able to successfully start the container is set inside the hooks, and liblxc allows performing operations such as **shutdown**, **suspend**, **reboot** and **resume** over containers, they will be easy to implement on OpenNebula. A simple command will usually be enough. The only remaining operation is hot-attach/detach NICs. Hot-attach can be accomplished easily by creating a virtual Ethernet interface and moving it to the container's namespace. Hot-detach will be achieved by deleting the interface.

This algorithm was implemented in bash at the first place. It was deployed in the Private Cloud infrastructure of the José Antonio Echeverría Higher Polytechnic Institute (CUJAE)'s DC, supporting the Information and Communications Technology (ICT) of the university. A stable release has been made public on github: https://github.com/OpenNebula/addon-lxcone/, together with the guidelines for supporting its deployment.

Proofs of concept

Different proofs of concept were done in the OpenNebula Private Cloud of the CUJAE's DC to check the effectiveness of LXCoNe. Figure 6 shows the logical design of the CUJAE's network, and Figure 7 shows the infrastructure's compute nodes. The Frontend was deployed on an LXC container inside Node-0 using OpenNebula 4.14. The Frontend was able to manage containers inside its own node, Node-0, and six other nodes. The first five

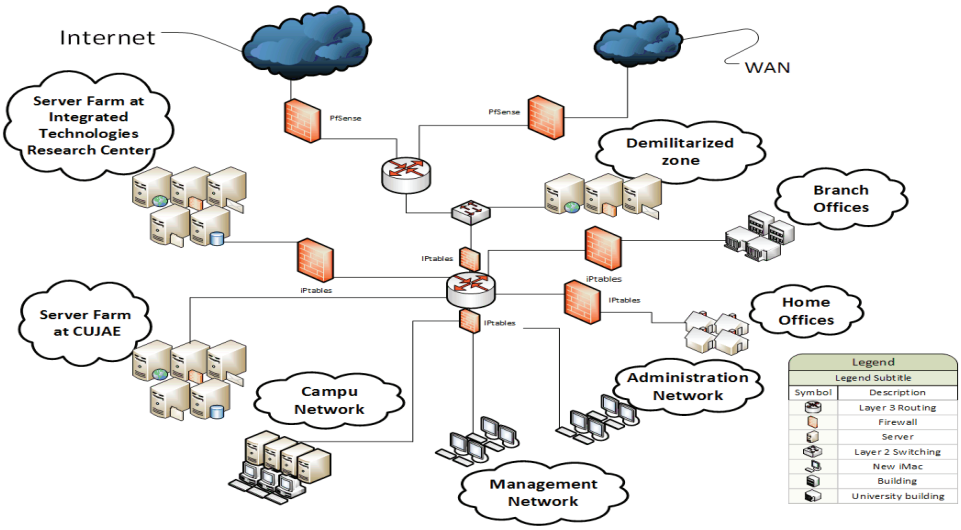

Figure 6. CUJAE data center's logical design

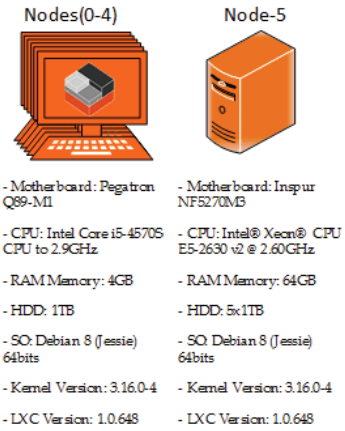

Figure 7. Infrastructure's compute nodes at ISPJAE/CUJAE's data center

DOI: http://dx.doi.org/10.22201/fi.25940732e.2018.19n1.006

nodes were commodity hardware with identical characteristics. The 6th. node was an Inspur professional server. Two different storage systems were used at the same time, Ceph and LVM. The tools used in the proofs of concept were the OpenNebula cloud manager and its Sunstone administration interface.

The proofs of concept demonstrated the capacity of LXCoNe to:

- Deploy, shut down, suspend and reset of LXC containers. The container "**r_nucleo1.cujae.edu.cu**" was configured in the TEMPLATE view and deployed in the VM view. Figure 8 shows that the container was in the RUNNING state at the end of the test.
- Attach and detach disks and NICs. These procedures were done at the Network/Storage tab in the VM view. Figures 9 and 10 show the successful operations.

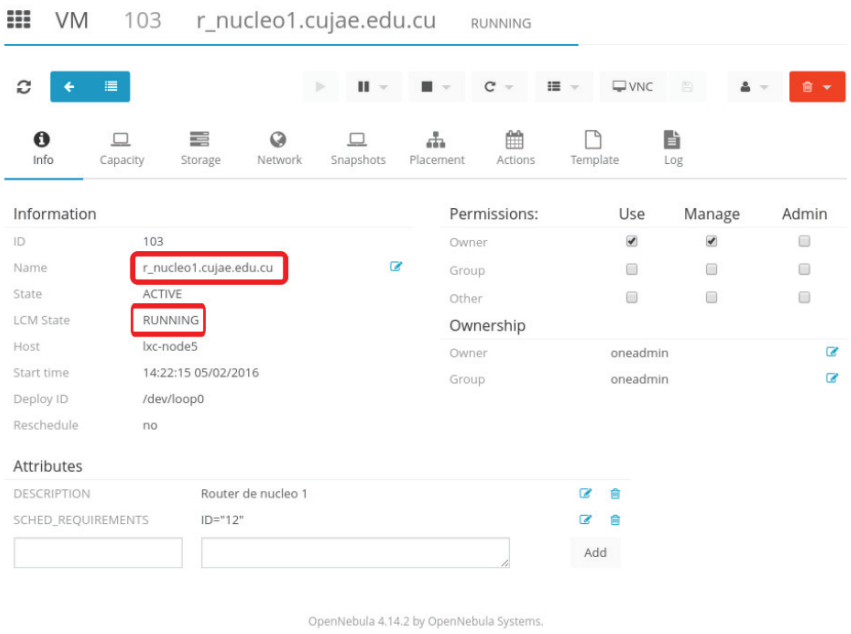

Figure 8. VM view. Container in RUNNING state

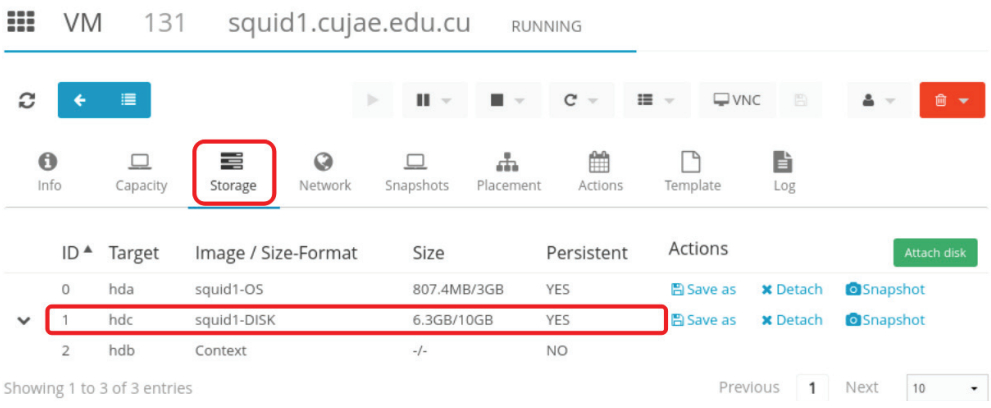

Figure 9. VM view. Extra HDD attached

DOI: http://dx.doi.org/10.22201/fi.25940732e.2018.19n1.006

- RAM limit per container. Figure 11 shows the Capacity tab inside the VM view, in which it can be seen that the RAM provisioned to the container was 4GB. Figure 12 shows that the Stress tool was configured in the container to fill the RAM up to 5GB. That was the only container running in the host. Figure 13 shows that the amount of RAM consumed didn't get over 4GB.
- Let OpenNebula monitors nodes and LXC containers. Figure 14 shows that the OpenNebula CLI was used for checking the monitoring.
- Support LVM and Ceph. Figures 15 and 16 show containers with different types of storage, LVM and Ceph respectively. Figure 17 confirms that containers were in RUNNING state.

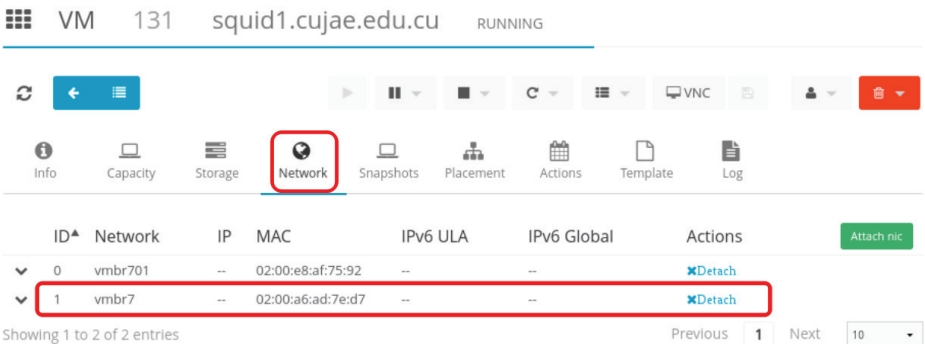

Figure 10. VM view. Extra NIC attached

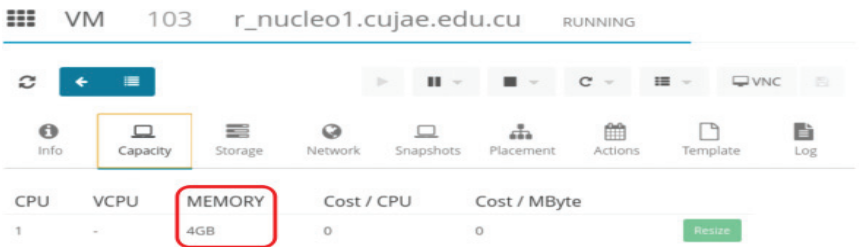

Figure 11. VM view. 4 GB of RAM assigned to the container

```
# stress --vm 1 --vm-bytes 5G
```

Figure 12. Fill the container's RAM. With stress tool

```
              total        used        free      shared  buff/cache   available
Mem:           5.7G        4.0G        0.8G        86M        982M        2.4G
Swap:            0B          0B          0B
```

Figure 13. Host's maximum used RAM

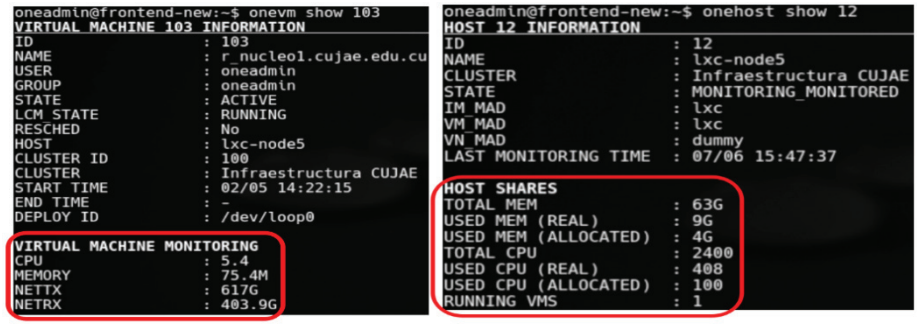

Figure 14. Host's maximum used RAM

DOI: http://dx.doi.org/10.22201/fi.25940732e.2018.19n1.006

Figure 15. Image view. LVM

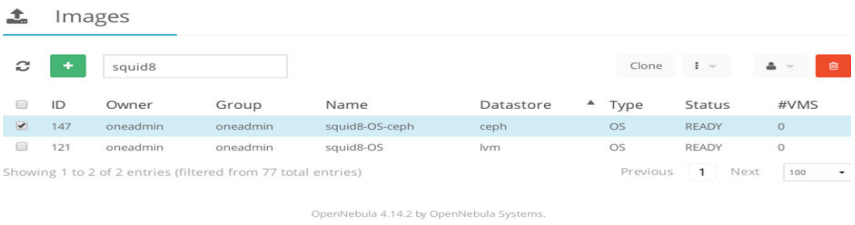

Figure 16. Image view. Ceph

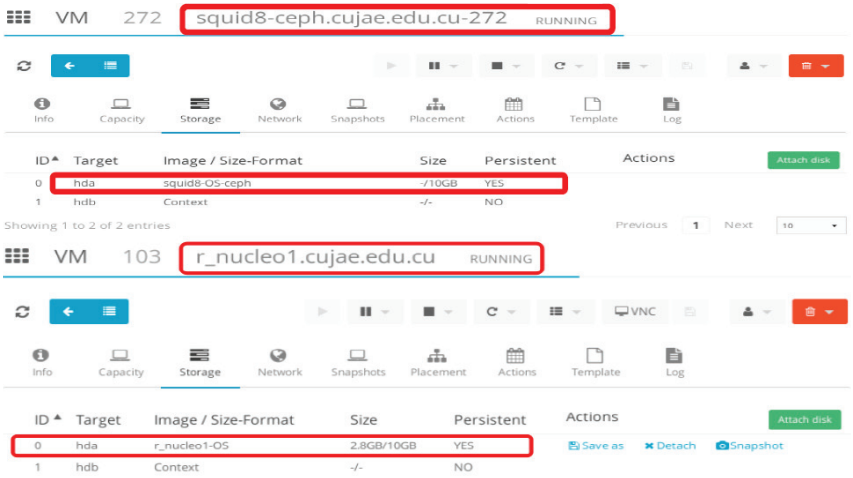

Figure 17. Container's state after the deployment

This driver was tested on two flavors of Linux: Debian 8 (Jessie) and Ubuntu 14.04 (Trusty Tahr) (De la Fé, 2016).

Conclusions

The LXC integration in OpenNebula contributes to develop more efficient solutions with high flexibility and interoperability levels in Cloud infrastructures. It has made possible an easier adaptation to the client economics restrictions, the human resources and the initial IT technologies of the client. The present work has as a main result, the development of a driver for OpenNebula to allow the deployment and the monitoring of the LXC virtualization platform. The driver has several basic features such as: to deploy, shutdown, suspend, reset LXC containers; to attach and detach disks and NICs to LXC containers; to support LVM and different FS for the storage; to limit RAM and CPU resources and to monitor containers and hosts. The next steps will be aimed to integrate other features which guarantee security, high availability and improvement of performance and scalability in the infrastructure.

Acknowledgments

The authors wish to thank to the IT managers of the CUJAE's datacenter and network for all their support and patience. This work was supported also by the Telecommunication and Telematics Department, and by the IT Services Department, both of the CUJAE University.

References

Aderholdt F., Caldwell B., Hicks S., Koch S., Naughton T., Pelfrey D., Pogge J., Scott S.L., Shipman G., Sorrillo L. Review of

DOI: http://dx.doi.org/10.22201/fi.25940732e.2018.19n1.006

enabling technologies to facilitate secure compute customization, Oak Ridge, Tennessee, USA, OAK Ridge National Laboratory, ORNL/TM-2015/210, 2014.

Agarwal K. *A Study of Virtualization Overheads*, (thesis master of Science in Computer Science), United State of America, Stony Brook University, 2015, pp. 55 [on line]. Available on: http://animal.oscar.cs.stonybrook.edu/papers/files/KavitaAgarwalMSThesisSubmission.pdf.

Arceo Feria A., Montejo-Ricardo G., García-Perellada L.R., Irigoyen-Saumel A., Garófalo- Hernández A.A. Propuesta de pruebas, parámetros y métricas para comparar plataformas de virtualización, on: XVI Convención de Ingeniería Eléctrica (CIE 2015) (16th., 2015, Santa Clara, Cuba). XVI Convención de Ingeniería Eléctrica CIE 2015, Santa Clara, Cuba, Martha Abreu University, 2015, pp. 605-612.

Banerjee T. LXC vs LXD vs Docker-Making sense of the rapidly evolving container ecosystem. Flockport, 2014a [on line]. Available on: https://www.flockport.com/lxc-vs-lxd-vs-docker-making-sense-of-the-rapidly-evolving-container-ecosystem/.

Banerjee T. Understanding the key differences between LXC and Docker. Flockport, 2014b [on line]. Available on: https://www.flockport.com/lxc-vs-docker/.

Banerjee T. China Mobile Releases OpenNebula-based Public Cloud, OpenNebula | Blog", OpenNebula Systems, 2014c [on line]. Available on: http://opennebula.org/blog/?author=52.

Banerjee T. Dockerfile reference, Docker, 2015a [on line]. Available on: https://docs.docker.com/reference/builder/.

Banerjee T. Scalable Architecture and APIs — OpenNebula 4.12.1 documentation, OpenNebula Systems 2015b [on line]. Available on: http://docs.opennebula.org/4.12/integration/getting_started/introapis.html.

Bud V. LXC Drivers for OpenNebula. GitHub, Inc., 2015a [on line]. Available on: https://github.com/OpenNebula/addon-lxc.

Bud V. HypervisorSupportMatrix-OpenStack. OpenStack Foundation, 2015b [on line]. Available: https://wiki.openstack.org/wiki/HypervisorSupportMatrix.

Bud V. How to add and/or resize a LXC disk. Proxmox Support Forum, XenForo Ltd., 2015c [on line]. Available on: https://forum.proxmox.com/threads/how-to-add-and-or-resize-a-lxc-disk.23792/.

Bud V. Lxc - How to resize a linux container in proxmox-Stack Overflow. Stack Overflow, Stack Exchange Inc., 2015d [on line]. Available on: http://stackoverflow.com/questions/32370052/how-to-resize-a-linux-container-in-proxmox.

Bud V. Problem with LXC disk resize. Proxmox Support Forum, XenForo Ltd., 2015e [on line]. Available on: https://forum.proxmox.com/threads/problem-with-lxc-disk-resize.24658/.

Bud V. Doubts with LXC File system and LXC disk size. Proxmox Support Forum, XenForo Ltd., 2015f [on line]. Available on: https://forum.proxmox.com/threads/doubts-with-lxc-file-system-and-lxc-disk-size.23124/.

Bud V. Roadmap-Proxmox VE. Proxmox VE, 2016a [on line]. Available on: http://pve.proxmox.com/wiki/Roadmap.

Bud V. LXC. Ubuntu, 2016b [on line]. Available on: https://help.ubuntu.com/lts/serverguide/lxc.html.

Cantrill B. Smar tData Center and Manta are now open source-Blog-Joyent. Joyent, Inc., 2014 [on line]. Available on: https://www.joyent.com/blog/sdc-and-manta-are-now-open-source.

Cantrill B. Operations Guide Release Version: 15.0.0. OpenStack, 2017a [on line]. Available on: https://docs.openstack.org/ops-guide/.

Cantrill B. Linux Containers-LXD-Getting started-OpenStack. Canonical Ltd., 2017b [on line]. Available on: https://linuxcontainers.org/lxd/getting-started-openstack/.

Cantrill B. Feature Support Matrix — nova 15.0.0.0rc2.dev705 documentation. OpenStack Foundation, 2017c [on line]. Available on: https://docs.openstack.org/developer/nova/support-matrix.html.

Chilipirea C., Laurentiu G., Popescu M., Radoveneanu S., Cernov V., Dobre C. Paper-Linux-VServer. GNU Free Documentation License 1.2, 2011 [on line]. Available on: http://linux-vserver.org/Paper.

Chilipirea C., Laurentiu G., Popescu M., Radoveneanu S., Cernov V., Dobre C. OpenNebula LXC Driver Plugin (OneLXC)–OpenNebula, OpenNebula Project, 2012 [on line]. Available on: http://opennebula.org/opennebula-lxc-driver-plugin-one-lxc/.

Chilipirea C., Laurentiu G., Popescu M., Radoveneanu S., Cernov V., Dobre C. OpenStack architecture design guide. OpenStack Foundation, 2015a [on line]. Available on: http://docs.openstack.org/arch-design/arch-design.pdf.

Chilipirea C., Laurentiu G., Popescu M., Radoveneanu S., Cernov V., Dobre C. An Overview of OpenNebula—OpenNebula 4.14.0 documentation, OpenNebula Systems, 2015b [on line]. Available on: http://docs.opennebula.org/4.14/design_and_installation/building_your_cloud/intro.html.

Chilipirea C., Laurentiu G., Popescu M., Radoveneanu S., Cernov V., Dobre C. Features—OpenNebula 4.14.0 documentation, OpenNebula Systems, 2015c [on line]. Available on: http://docs.opennebula.org/4.14/release_notes/release_notes/features.html#features.

Chilipirea C., Laurentiu G., Popescu M., Radoveneanu S., Cernov V., Dobre C. Linux Containers-LXC-News. Canonical Ltd., 2015d [on line]. Available on: https://linuxcontainers.org/lxc/news/.

Chilipirea C., Laurentiu G., Popescu M., Radoveneanu S., Cernov V., Dobre C. Linux Containers-LXD-News. Canonical Ltd., 2015e [on line]. Available on: https://linuxcontainers.org/lxd/news/.

Chilipirea C., Laurentiu G., Popescu M., Radoveneanu S., Cernov V., Dobre C. Roadmap, OpenVZ Virtuozzo Containers Wiki, 2015f [on line]. Available on: https://openvz.org/Roadmap.

Chilipirea C., Laurentiu G., Popescu M., Radoveneanu S., Cernov V., Dobre C. Comparison, OpenVZ Virtuozzo Containers

DOI: http://dx.doi.org/10.22201/fi.25940732e.2018.19n1.006

Wiki, 2015g [on line]. Available on: https://openvz.org/Comparison.

Chilipirea C., Laurentiu G., Popescu M., Radoveneanu S., Cernov V., Dobre C. A comparison of private cloud systems, 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2016a, pp. 139-143.

Chilipirea C., Laurentiu G., Popescu M., Radoveneanu S., Cernov V., Dobre C. OpenNebula 5.0 Deployment guide Release 5.0.2. OpenNebula Systems, 2016b [on line]. Available on: http://docs.opennebula.org/pdf/5.2/opennebula_5.2_deployment_guide.pdf.

Chilipirea C., Laurentiu G., Popescu M., Radoveneanu S., Cernov V., Dobre C. GitHub-OpenNebula/addon-lxc: Hypervisor Drivers for LXC, GitHub, Inc., 2016c [on line]. Available on: https://github.com/OpenNebula/addon-lxc.

De la Fé J.M., Vega S. LXCoNe, Installation & Configuration Guide, OpenNebula/addon-lxcone, GitHub, Inc., 2016 [on line]. Available on: https://github.com/OpenNebula/addon-lxcone.

Ectors M. LXD and Docker, Telruptive. Telruptive 2014a [on line]. Available on: http://telruptive.com/2014/11/11/lxd-and-docker/.

Ectors M. LXD and Docker-DZone Cloud. DZone, 2014b [on line]. Available on: https://dzone.com/articles/lxd-and-docker.

Ectors M. LXD and Docker-DZone, 2014c [on line]. Available on: https://dzone.com/articles/lxd-and-docker.

Ectors M. LXD: the next-generation container hypervisor for Linux | Cloud | Ubuntu, Canonical Ltd, 2016 [on line]. Available on: http://www.ubuntu.com/cloud/tools/lxd.

Graber S. LXC 1.0: Security features. Stéphane Graber's website, 2014a [on line]. Available on: https://stgraber.org/2014/01/01/lxc-1-0-security-features/

Graber S. Elastic Containers. ElasticHosts Blog, 2014b [on line]. Available on: https://www.elastichosts.com/blog/elastic-containers/.

Graber S. Large scale container management with LXD and OpenStack. Presented at the LinuxCon + CloudOpen + ContainerCon NA 2015, Sheraton Seattle, Seattle, WA, 2015a [on line]. Available on: http://events.linuxfoundation.jp/sites/events/files/slides/ContainerCon%202015-%20LXD%20%26%20OpenStack.pdf

Graber S. AWS, Amazon EC2 Container Service, Detalles del product. Amazon Web Services, Inc., 2015b [on line]. Available on: //aws.amazon.com/es/ecs/details/.

Graber S. Joyent TritonTM Elastic Container Service-Public Cloud-Joyent, Joyent, Inc., 2015c [on line]. Available on: https://www.joyent.com/public-cloud.

Graber S. LXD 2.0: Resource control. Stéphane Graber's website, 2016a [on line]. Available on: https://stgraber.org/2016/03/26/lxd-2-0-resource-control-412/

Graber S. Scalable Cloud Hosting on Linux Containers. Kyup., 2016b [on line]. Available on: https://kyup.com.

Graber S. Innovative Cloud Platform on Linux Container. Kyup., 2016c [on line]. Available: https://kyup.com/linux-containers.

Graber S. Joyent Public Cloud Pricing. Joyent, Inc., 2017a [on line]. Available on: https://www.joyent.com/pricing/cloud/compute.

Graber S. Joyent Triton Compute. Joyent, Inc., 2017b [on line]. Available on: https://www.joyent.com/triton/compute.

Graber S. Pricing ElasticHosts Linux, Windows VPS Hosting. Elastichosts, 2017c [on line]. Available on: https://www.elastichosts.com/pricing/.

Morabito R., Kjällman.J., Komu M. Hypervisors vs. lightweight virtualization: A performance comparison, on: 2015 IEEE International Conference on Cloud Engineering, 2015, pp. 386-393.

Petazzoni J. Anatomy of a container: namespaces, cgroups, and some filesystem magic. Presented at the LinuxCon + CloudOpen + ContainerCon NA 2015, Sheraton Seattle, Seattle, WA, 2015 [on line]. Available on: http://events.linuxfoundation.jp/sites/events/files/slides/Anatomy%20of%20a%20container.pdf

__. Linux Containers-LXC-Security. Canonical Ltd., 2017 [on line]. Available on: https://linuxcontainers.org/lxc/security/.

Scott. A quick introduction to LXD, Scott's Weblog, 2015a [on line]. Available on: http://blog.scottlowe.org/2015/05/06/quick-intro-lxd/.

__. Linux Containers-LXD-Introduction, Canonical Ltd, 2015b [on line]. Available on: https://linuxcontainers.org/lxd/introduction/.

Steven J. Vaughan-Nichols. Ubuntu LXD: Not a Docker replacement, a Docker enhancement. ZDNet, 2014 [on line]. Available on: http://www.zdnet.com/article/ubuntu-lxd-not-a-docker-replacement-a-docker-enhancement/.

Wallner R. Linux Containers: Parallels, LXC, OpenVZ, Docker and More. Au Courant Ttechnology [on line], 2015. Available on: http://aucouranton.com/2014/06/13/linux-containers-parallels-lxc-openvz-docker-and-more/.

Wallner R. LXC, 2014 [on line]. Available on: https://help.ubuntu.com/lts/serverguide/lxc.html.

DOI: http://dx.doi.org/10.22201/fi.25940732e.2018.19n1.006

Suggested citation:

Chicago style citation

García-Perellada, Lilia Rosa, Sergio Vega-Gutiérrez, José Manuel De la Fé-Herrero, Yalina Rodríguez-De Armas, Alain Abel Garófalo-Hernández. Driver LXC development for OpenNebula. *Ingeniería Investigación y Tecnología*, XIX, 01 (2018): 63-76.

ISO 690 citation style

García-Perellada L.R., Vega-Gutiérrez S., De la Fé-Herrero J.M., Rodríguez-De Armas Y., Garófalo-Hernández A.A. Driver LXC development for OpenNebula. *Ingeniería Investigación y Tecnología*, volume XIX (issue 1), January-March 2018: 63-76.

About the authors

Lilia Rosa García-Perellada. Engineer in Telecommunications and Electronics from the José Antonio Echeverría Higher Polytechnic Institute (CUJAE), La Habana, Cuba. She holds a M.S. from the CUJAE University too. She is an assistant professor in the Telecommunication and Telematics Department of the CUJAE University.

Sergio Vega-Gutiérrez. Engineer in Telecommunications and Electronics from the José Antonio. Echeverría Higher Polytechnic Institute (CUJAE), La Habana, Cuba. He is currently working in Telecommunication and Telematics Department of the CUJAE University.

José Manuel de la Fé-Herrero. Engineer in Telecommunications and Electronics from the José Antonio Echeverría Higher Polytechnic Institute (CUJAE), La Habana, Cuba. He is currently working in Telecommunication and Telematics Department of the CUJAE University.

Yalina Rodríguez-De Armas. Engineer in Telecommunications and Electronics from the José Antonio Echeverría Higher Polytechnic Institute (CUJAE), La Habana, Cuba. She is an instructor adjunct professor in the Telecommunication and Telematics Department of the CUJAE University, and works in the IT Services Department of the CUJAE University.

Alain Abel Garófalo-Hernández. Engineer in Telecommunications and Electronics from the José Antonio Echeverría Higher Polytechnic Institute (CUJAE), La Habana, Cuba. He holds a M.S. and a Ph.D. from the CUJAE University too. He is an assistant adjunct professor in the Telecommunication and Telematics Department of the same university.