OPENHUACA

A PLATFORM FOR EASILY BUILDING CLOUD INFRASTRUCTURE

A Master Thesis

Submitted to the Barcelona School of Informatics

Universitat Politècnica de Catalunya

(UPC) – BarcelonaTech

by

Rafael Genés Durán

In partial fulfilment

of the requirements for the MASTER in

INFORMATICS ENGINEERING

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

Director: JOSE LUIS MUÑOZ TAPIA

Department of Telematics, ETSETB, UPC

Advisor: JORDI TORRES VIÑALS

Department of Computer Architecture, FIB, UPC

Barcelona - January 2019

Abstract

The aim of this thesis is to define and implement an open source platform called OpenHuaca that is a private cloud platform based on LXC containers and KVM machines. Also, it includes advanced functionalities as dnsmasq, domains, NAT, SSH connections through certificates, bases, provisioning, and user management. OpenHuaca differs from its competitors because the user will be able to install it on any Debian Linux system. The objective of the platform is to optimize the environment building time and minimize the learning curve for Cloud technologies.

This platform is designed for teaching or research centers, where there are experts from other sectors who have the necessity to have a cloud environment where performing their tests and experiments, but they do not have the knowledge or practice to do so. Therefore, OpenHuaca let the users build small or medium environments, with any type of architecture, always looking for simplicity, efficiency and speed. So that platform can bring a Cloud to anyone, even without extensive knowledge about it.

Apart from the implementation of the project in Python, a publication has been prepared in LaTeX to publish the platform.

Resum

La tesis está basada en la definició i implementació d'una plataforma de codi lliure anomenada OpenHuaca, la qual és una plataforma cloud privada basada en contenidors LXC i màquines KVM. També inclou funcionalitats avançadas com dnsmasq, dominis, NAT, connexions SSH mitjançant certificats, bases, aprovisionament i gestió d'usuaris. OpenHuaca es diferencia dels seus competidors perquè l'usuari és capaç d'instal·lar-lo en qualsevol distribució Linux de tipus Debian. L'objectiu de la plataforma és optimizar el temps de creació d'entorns i minimizar la curva d'aprenentatge de les tecnologies Cloud.

La plataforma està pensada per centres docents o d'investigació, on hi ha experts en altres sectors que necessiten disposar d'un entorn cloud on realitzar les seves proves i experiments, però no disposen dels coneixements ni de la pràctica per fer-ho. Per tant, OpenHuaca permet construir petits o mitjans entorns, amb qualsevol tipus d'arquitectura, buscant sempre la simplicitat, eficiència i velocitat. De forma que podem apropar el mon cloud a qualsevol persona, inclús sense amplis coneixements del sector.

A més a més de la implementació del projecte en Python, s'ha preparat una publicació en LaTeX per donar a conèixer la plataforma.

Resumen

La tesis está basada en la definición e implementación de una plataforma de código libre llamada Openhuaca, la cual es una plataforma cloud privada basada en contenedores LXC y máquinas KVM. También incluye funcionalidades avanzadas como dnsmasq, dominios, NAT, conexiones SSH a través de certificados, bases, aprovisionamiento y gestión de usuarios. OpenHuaca se diferencia de sus competidores en que el usuario es capaz de instalarlo en cualquier distribución Linux de tipo Debian. El objetivo de la plataforma es optimizar el tiempo de creación de entornos y minimizar la curva de aprendizaje de las tecnologías Cloud.

La plataforma está pensada para centros docentes o de investigación, donde hay expertos en otros sectores que necesitan tener un entorno cloud donde realizar sus pruebas y experimentos, pero no disponen de los conocimientos ni de la práctica necesaria para ello. Por tanto, OpenHuaca permite montar pequeños o medianos entornos, con cualquier tipo de arquitectura, buscando siempre la simplicidad, eficiencia y velocidad. De forma que podamos acercar el mundo cloud a cualquier persona, incluso sin amplios conocimientos en el sector.

Además de la implementación del proyecto en Python, se ha preparado una publicación en LaTeX para dar a conocer la plataforma.

Acknowledgements

I would like to thank the help and collaboration of the teammates Jorge Buzzio and Martí Barri. Moreover thank to all the anonymous contributors with valuable comments and suggestions. Also, I would like to thanks both my director, Jose Luis Muñoz for their daily support and technical knowledge during the design process and my advisor, Jordi Torres, for their tips and lessons.

Revision history and approval record

Revision	Date	Purpose
0	15/12/2018	Document creation
1	28/12/2018	First revision
2	10/01/2019	Second revision
3	23/01/2019	Document Delivery

Document distribution list

Name	e-mail
Rafael Genés Durán	rafa.gd.10@gmail.com
Jose Luis Muñoz Tapia	jose.munoz@entel.upc.edu
Jordi Torres Viñals	torres@ac.upc.edu

Written by:		Reviewed and approved by:	
Date	15/12/2018	Date	23/01/2019
Name	Rafael Genés Durán	Name	Jose Luis Muñoz Tapia
Position	Project Author	Position	Project Supervisor

List of Figures

List of Tables

Table of contents

Introduction

This thesis is the continuation of the open source project OpenHuaca. It is a cloud platform based on OS containers. Last year a proof of concept was coded in shell as my TFG. At these moment it had LXC containers that are a kind of virtual machines isolated from the kernel. It could manage the container type, their users and the capacity to configure the access via SSH certificates.

Then, OpenHuaca contributors do some other features as testing to finally create the first release of the project. This thesis is based on these release, the creation of OpenHuaca, coded in Python 3, with the successful PoC of previous teammates and adding new designed features.

The document includes the objectives, the requirements, the implementation procedures and the incidences that the project suffered during the realization.

Statement of purpose

The purpose of this project, personally, is to learn about the existing methods of organization in order to develop an application properly and understand the necessities of a project and how to face them, either when making or revising code, communicating with the supervisor or working in parallel with other colleagues.

Also, the goal of OpenHuaca is to cover the need to have a cloud environment with virtual machines where a user does not need large configurations or environments but needs velocity. OpenHuaca has been designed for those small or medium stages where you need several containers and have few resources, so it has been created as an installable package so that in any Debian-type distribution can be installed and allow the user to create a very customizable cloud platform, both at the level of resources such as user management. Also, it is configured an base command to create virtual containers from a customized one.

Finally, to start learning how to write a WhitePaper for the PhD and to publish the platform, an article is written and attached to this thesis.

Requirements and specifications

This project has been designed to be compatible with the majority of equipment so the system requirements are minimal.

The first requirement is that the user needs a Debian-type distribution in order to install the facilitated package, my recommendation, an Ubuntu 18.04 LTS. In addition, the containers need an updated version of LXC, which can be found in the official repositories of Ubuntu. Also, OpenHuaca need the latest stable version of Python 3.

Finally, it should be noted that each container is located in the kernel next to the native operating system, so the user has to design the machine taking into account the resources that want to assign to the host and each container.

Methods and procedures

As my TFG of the Telematic Engineering Degree, I start this project, I create a *Proof of Concept* of OpenHuaca, coded in shell. This thesis is the continuation of these, it is the creation of the first release, coded in Python 3 with lots of new features.

Work Packages and Milestones

Project Design	WP #1
Design python command arch and features	Planned end date: 01/10/2018

Table 1. WP #1

Project Development	WP #2
Code subcommands and test it	Planned end date: 10/12/2018

Table 2. WP #2

WhitePaper	WP #3
Write and review article	Planned end date: 01/01/2019

Table 3. WP #3

Documentation	WP #4
Thesis documentation and presentation	Planned end date: 20/01/2019

Table 4. WP #4

WP#	Title	Milestone/Deliverable	Date
1	Design command arch.	Prepare general structure	2
1	Design subcommands	Prepare subcommand in different files	2
1	Test command	Test arch and subcommands	3
2	Feature: domains	Develop whole subcommand Domains	5
2	Feature: certification	Develop whole subcommand Certification	7
2	Feature: vm	Develop whole subcommand VM	11

2	Feature: bases	Develop whole subcommand Bases	15
3	WhitePaper	Write and review White Paper	18
4	Thesis	Write and review Thesis	19
4	Presentation	Prepare presentation	20

Table 5. Milestones

Gantt diagram

WP#	Title	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	Design command arch.																				
	Design subcommands																				
	Test command																				
2	Feature: domains																				
	Feature: certification																				
	Feature: vm																				
	Feature: bases																				
3	WhitePaper																				
4	Thesis																				
	Presentation																				

Table 6. Gantt diagram

State of the art

In this section it will be discussed the current state of the market, establish the main competitors that OpenHuaca have, the advantages and disadvantages of each analyzed platform and finally conclude if a new one is needed.

It will be divided into three parts, some tools are being chosen that cover the same market as OpenHuaca. That is, it will be selected Private Cloud platforms, which offer IaaS or PaaS, both virtual machines and containers. Then, the advantages and disadvantages of each of them will be analyzed and checked. Next, it will be performed a comparison to finally take conclusions where there may be gaps that do not cover any of the current tools.

The platforms selected are LXD, Docker, ProxMox, OpenStack, VMware and Citrix.

LXD

LXD is a private LXC container cloud platform created by Canonical. Offers a new design and improved user experience. This platform is built with a daemon that generates a REST API on a local socket to control the whole platform. It is designed taking security and scalability in mind. Also, it contains namespace isolation, resource control, basic storage and network management. It has an integration with OpenStack to ensure the availability of containers too.

Main advantages of LXD are:

- Containers of entire operative systems.
- API management.
- Network and storage management
- High availability integration

The main difference that has LXD from OpenHuaca is that provides a robust environment managed via API REST. Despite the fact that does not had the possibility to create machines with different kernel from the host. That means, you cannot create different Linux architectures or operative systems.

Docker

Docker is an open source software that automates the deployment of applications. Despite not being a private cloud platform, it allows the virtualization of processes through containers. Also it allows the management of operating systems, the isolation of namespaces, a resource control, an advanced storage and network system and it has integration with lots of products that add functionalities such as web interfaces, APIs or provisioning systems like ansible or puppet.
Main advantages of Docker are:

- Container of process.
- API management.
- Network and storage management
- Web interface

- Multiple integrations
- One of the biggest communities

The main difference from our tool is that Docker is a hard and robust software, thousands of people work to improve Docker, for that reason, it has a wide range of plugins and integrations. Moreover, Docker has been designed to deploy applications and execute containers with only a process. It has the option to virtualize an entire operative system but it is not its core function.

ProxMox

ProxMox is a private cloud platform that allows the virtualization of containers and KVM machines. It has advanced storage and network management. In addition, it has a web interface and an API to improve the management. It is also prepared to work in a Cluster, ensuring high availability. It has the possibility to do a live machine migration. Despite not having integrations, it has an api that makes it possible the tool can be managed by others, such as, for example, with provisioning softwares such as Ansible.

Main advantages of ProxMox are:

- Containers for entire operative systems.
- KVM for other Kernels.
- API management.
- Network and storage management
- Web interface
- High availability
- Live VM migration

All of those features provide the possibility to generate your own environments. It is a very robust tool, with a very polished software, an api and a web interface. In spite of all of that, it does not have integrations with the main provisioning or deployment tools. Another difference with respect to our platform are the bases, the management of domains or the management of users and their SSH keys.

OpenStack

OpenStack is a complex platform that allows you to manage all the hardware resources through a few components via web interface. It is prepared to support any type of technology, LXC, KVM, VMware, Citrix or Hyper-V. It has advanced computer managers, storage, network, among others. It is a very robust tool with integrations with a wide range of tools.

Main advantages of OpenStack are:

- Operating System Containers.
- KVM for other Kernels.
- API management.
- Network and storage management
- Web interface

- Multiple integrations
- High availability
- Complex system

The main differentiating element of OpenStack with respect to OpenHuaca is its complexity. To assemble a simple lxc container you need to have an entire infrastructure, with clusters, network and storage configurations, you have to install LXC, its dependencies and finally you can create it While the objective of our platform is based on the fact that you can do quick tests, that in a matter of seconds you have a small environment working.

VMware

Although VMware is not open source, it has a free license of its ESXi. It has virtualization of machines. It contains a web interface to manage resources. It allows to configure virtual networks. You can also access to NFS disks over network. The free version does not have high availability neither API.

Main advantages of VMware are:

- Virtualización of all kind of machines.
- Network management
- Local storage and NFS
- Web interface

Significant improvements are implemented using the payment tool, but this free version presents important limitations. In spite of this, it is a very good environment to create your machines for testing or production. Despite that, it does not provide provisioning tools or integrations that can slow tasks which OpenHuaca will try to speed up.

Citrix

Last tool to analyze, Citrix, developed by IBM, is a SaaS platform that allow machine virtualization. For that reason, your own environment can be set up to deploy applications. Allows lots of network and storage managements. It also provides integrations for high availability.

Main advantages of Citrix are:

- Virtualization of all kind of machines.
- Network management
- Local storage or NFS
- Web interface

Once all the tools that we are going to compare have been defined, all the characteristics and features of each tool will be compared in the following table. All the advantages and disadvantages will be shown and it will check if there is any market niche that is not being covered by the proposed tools.

	LXD	Docker	ProxMox	Open Stack	VMware*	Citrix*
OS container	X	X	X	X	X	X
Process container		X				
KVM			X	X	X	X
Advanced Storage	X	X	X	X	X	X
Advanced Networks		X	X	X	X	X
Webapp interface			X	X	X	X
API	X	X	X	X		
Live VM migration			X			
Multiple integrations		X		X		
High availability	X		X	X		

Table 7. Comparative table

As it can be seen, VMware and Citrix are designed more for business environments, where important infrastructures are needed and, therefore, with the free licences you can do some tests so, end up buying licenses with a whole series of features and professional support. On the other hand, we have OpenStack, a very complex system that needs many resources and knowledge to set up the first and basic environment. It is a very powerful tool, but in our case we are looking for simplicity. Therefore, the mentioned platforms will be discarded.

LXD is a very useful tool for fast tests because it has containers. The problem is that many functionalities are missing, which allows you to build some dynamic scenarios, such as network configurations or the creation of machines with different kernels. The next option it would be to think about Docker, that we have considered it in spite of not being a cloud platform. But through its wide amount of plugins, completely free, we can get very sophisticated scenarios with a few simple steps. Docker seems to be the solution we were looking for, the only drawback we detect is that Docker focuses on a single process containers, not entire operating systems.

Although you can mount integral operating system environments, docker is not designed for it, therefore, we will leave this tool to deploy applications in processes and we will move to the last option, ProxMox, which provides Virtualization of machines through KVM, LXC containers, storage network management, API, web interface. It seems that it follows the same objectives and approaches that we want to propose in our platform. But it has a big difference, it is distributed as an image, so you need specific hardware to make it work, which slows down and makes testing difficult.

Background

OpenHuaca is a private cloud platform that integrates the possibility of creating both containers and virtual machines. Moreover, it offers the option to create complex network environments, called domains, with name and access control in an automated way. Also, the platform contains multiple integrations with Ansible, to supply all our machines created, and with Terraform, to offer a IaaC software (Infrastructure as a Code).

Another possibility offered by OpenHuaca is to create user own bases from machines that the user has created locally and share them in its HuacaHub portal, where the user will be able to find all kind of bases and HuacaFiles, which are compatible with Terraform.

OpenHuaca is an open source project that seeks to optimize the creation times of virtual environments so that a researcher or worker can set up small test environments quickly and be able to focus on his work. Below will be described the basic concepts to help on the well understanding of the platform:

Firstly, it is important to know the meaning of having a private cloud. Due to it implies that the user will need to have his own server with enough benefits enough to assemble all the environment planned. The user is also responsible for the maintenance of the mentioned server, so that he will have to take into account the resources of the machine to ensure the scalability and availability of the environment.

One of the main advantages for private cloud is that the user will have complete control over the platform. So, the user will be able to configure the hardware precisely, choose the operating system and ultimately, adapt the environment to his own design.

Secondly, both containers and virtual machines will be described. As it can be seen in the following image, the containers are machines that share the Host kernel, and therefore, are much faster and lighter. On the other hand, virtual machines have their own integrated kernel, so any operating system or architecture can work on them, although it is not the host one.

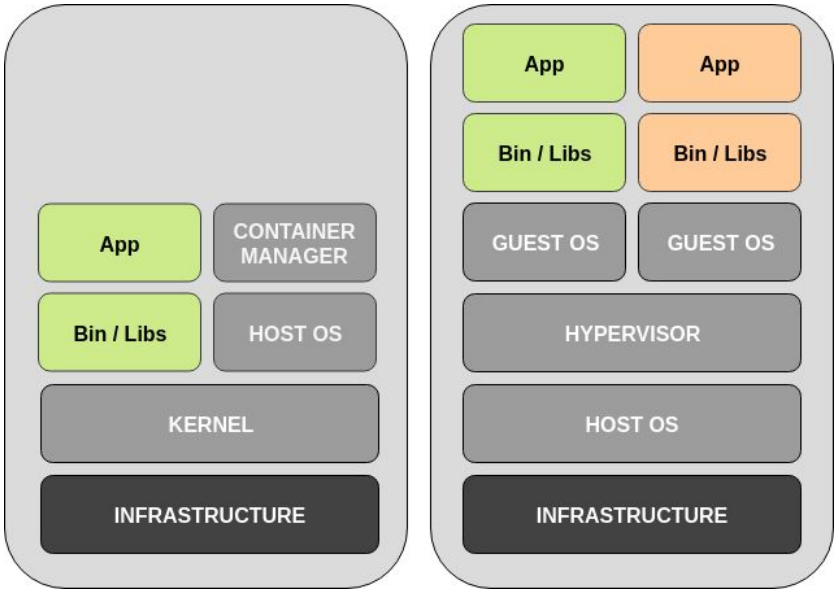

Figure 1. Container VS VM

Once the differences between containers and virtual machines are explained, it is much simple to understand that the most optimal way to set up environments is through containers in case the desired system is compatible with our host. But, if only containers are implemented it will be lost the possibility of creating a percentage of machines that have a different kernel.

In conclusion, the most optimal way is to implement containers whenever possible and KVM in the rest of cases.

Once the host operating system is assembled, the user will have to design the communication structure between the virtual machines and the host. The platform is designed to work through domains, these are Virtual Bridges that connect our machines directly to the host. Also, OpenHuaca contains an own dnsmasq to automate the DNS management of the machines and to be able to name them from the style machine1.domain1. The following image shows how to create both containers and virtual machines and group them in different domains according to the need of the environment too.

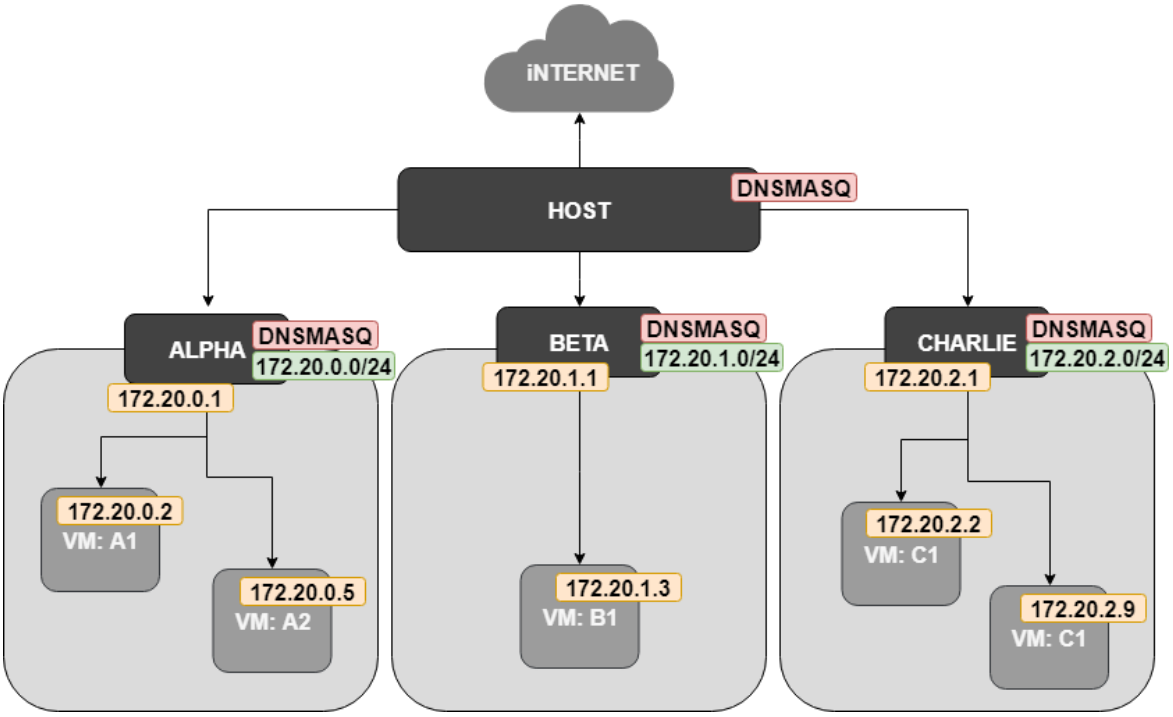

Figure 2. Domain Structure

The domains can also be used to generate a Certification Authority (CA) to direct connections to virtual machines, from the host to a machine or between various VMs. Even between different domains, thanks to dnsmasq, which will be explained later.

In addition, in order to optimize the time management of the tool, integration with Ansible has been implemented for tasks of provisioning, where the user will be able to assemble and configure environments from machines with a simple file. Also, remember that the platform includes the possibility of executing commands on a machine from the host, such as generating consoles using namespaces.

Lastly, OpenHuaca integrates an Infrastructure as a Code tool, IaaC, Terraform. This one offers the user the possibility of generating his own environment, domains, machines, network configurations, nat, etc .. all from a single file. Thanks to this tool, the platform includes its proper HuacaFiles, which will be detailed later, but in conclusion they are files with all the environment configuration, network, machines and their contents. So that the user will be able to download a file from the HuacaHub portal, execute it in his environment and in an agile and automated way get a complex system of machines.

Proposal

In this section it is explained in depth what OpenHuaca is, what benefits it brings and how to use it to work optimally. Initially, it will be described how to install the platform and how it is distributed. Secondly, an step by step description of all the functionalities, from principal ones, how to mount network environments, machines, configure accesses, etc... to more specific functionalities, such as NAT, management of resources or bases. Next, it will be explained the supply and infrastructures as code integrations. And finally, the web portal called HuacaHub , where the platform can distribute both the source code of the tool, the packages debian, all kinds of bases and the Huaca Files with all the information to set up environments through terraform will be described.

In order to manage the bases, the platform includes a small unique tool called the rebase, which will allow the user to update the bases of all his containers, so that he will be able to minimize the space of each container and all new ones will be perfectly updated. However, this will be extended below to it proper understanding.

Installation

To install OpenHuaca it is recommended to use an Ubuntu Server 18.04 LTS, despite being compatible with any Debian distribution. The .deb package can be found in the web portal. This will install all the packages needed, it will configure the services of the system and it will indicate where we have the configuration variables of the system.

For future advances, the installation of OpenHuaca in Custers is planned, with technologies such as etcd or integrations with OpenStack, but in this paper it is just considered the Standalone installation.

Machines

OpenHuaca uses two virtualization technologies, LXC for OS and KVM containers for virtual machines. As explained before, the Containers are much faster, but they have the limitation that they must be compatible with the host kernel. Therefore, we will prioritize the creation of containers as long as the kernel is compatible. Otherwise, we will install a KVM, heavier, slower but functional.

It should be noted that despite having explained how the user will achieve consensus between technologies LXC and KVM, for the administrator of the platform there will be no difference between some machines and others, if it is true that in the machine display the platform will have the differentiation of machines, but all the commands that will be introduced are exactly the same, all the technology differentiation management is applied automatically. The user should only indicate what he wants to do and the system adapts to both technologies in a dynamic way.

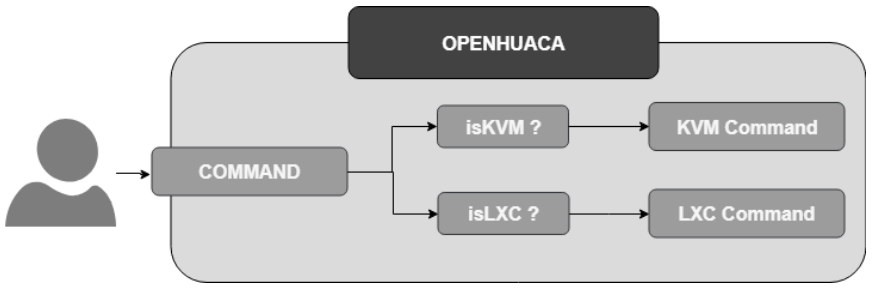

Figure 3. Command technology self-management& O.B.R & A.R & O.B.R & A.R & O.B.R & A.R \\

Resources

Like any IaaS platform, OpenHuaca includes the possibility to manage the resources of each machine, according to the speed of the processor, its memory or its storage capacity. The user will be able to also add or remove devices from the host, such as usb, memory cards or discs.

Network

One of the advantages of OpenHuaca is the high capacity of network configuration that can implement. The platform includes the possibility of creating domains that can separate machines in different network ranges to add versatility to environments. These domains are virtual bridges that are created on the host and interconnected through iptables. In addition, an own dnsmasq management is generated for each domain, in order to automate the DHCP and DNS management of each rank. In the following image it is exampled an environment with different domains.

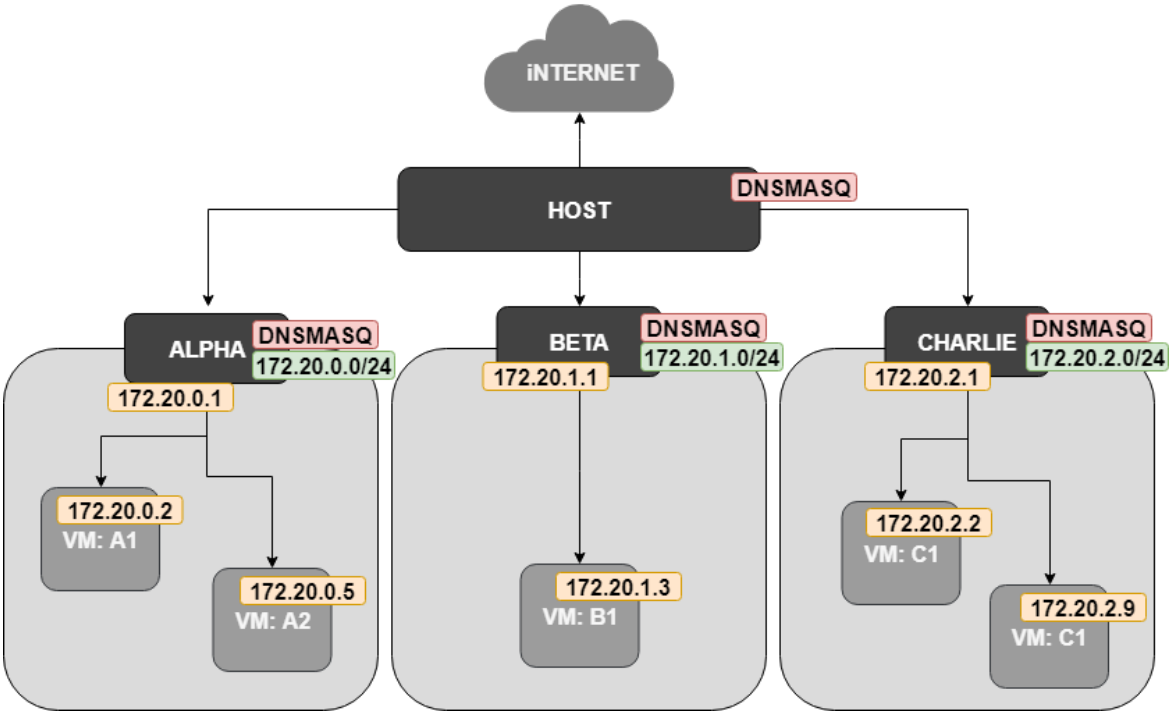

Figure 4. Structure with domains

As it is shown, they all share a range of class B IPs and each domain is assigned a complete class C. Although, this is only an example, but through the configuration files, different configurations can be generated. The dnsmasq has also been configured to respond to machines by *machine.domain*.

Finally, it should be noted that the machines do not differentiate between their technology it is the software that self-manages.

Certificates

At this point, the user will be able to connect from the host to any machine using the subcommand called console, where a tty of the machine desired is got if it has configured an ssh service in the virtual machine, the user will be able to access from any domain to any machine just knowing a username and his password. Something costly if the user wants to have a medium environment, with multiple machines and users.

For this reason, OpenHuaca has the ability to create and manage Certification Authorities, CA, to generate certificates, which will be stored on the host. These can be distributed among the machines or clients, so that it can facilitate the connection between machines.

OpenHuaca proposal is to add a CA for each domain, although you could create multiple too. In this way, access to domains can be limited, see the following photo:

Figure 5. Structure with domains and CAs

From these certification authorities, multiple certificates can be generated according to the domain and the user specifically. In the list below, it can be observed that the platform includes the option to generate different certificates, with different expiration dates.

Furthermore, it is possible to manually disable or revoke a certificate, this allows the user to have total control of access to his machines, without having to remember or set up passwords or third parties.

ID	STATUS	CA	USER	DUE DATE (w)
1	Active	alpha	huaca	52
2	Revoked	alpha	user1	2
3	Invalid	beta	user2	0

Table 8. Certificates

NAT

OpenHuaca not only has network options to improve access to the machines or their organization. It also has a natting service, which works through internal iptables of the host. In case a service is installed in one of the virtual machines, there is the possibility that the user wants to access from the internet or from another machine outside the host. Therefore, with OpenHuaca NAT manager, the desired port can be redirected to the created container service. In this way, see the following image, the services created from the host will be available, Allowing the user accessing from other equipment or making the necessary configurations in his network to be able to access from the internet.

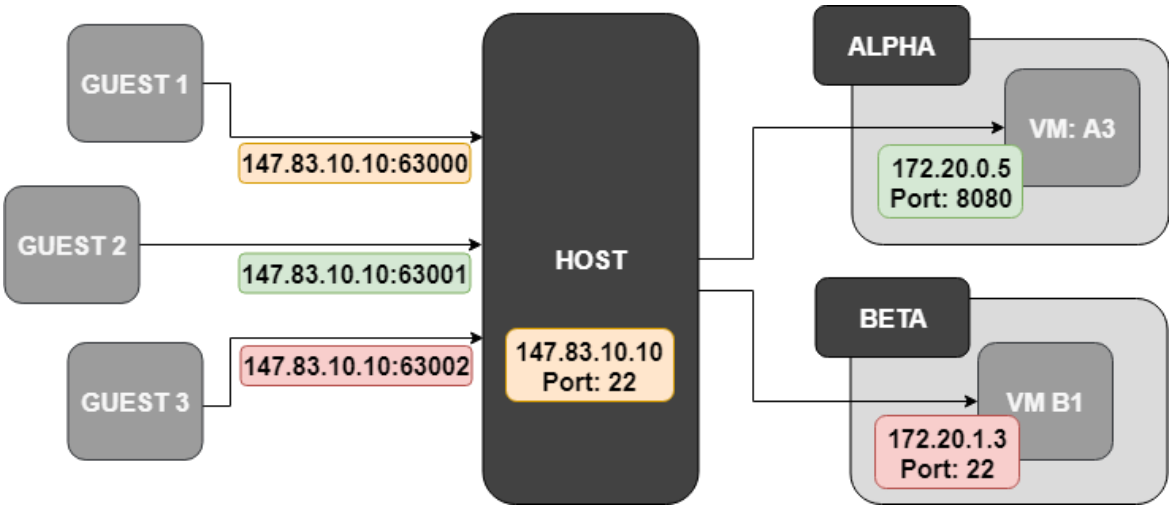

Figure 6. NAT

Bases

Initially, LXC and KVM had their own templates to create machines. But, they were not customizable, so, OpenHuaca platform is based on bases, which will be created from one of the machines, and will keep all the configuration and installed software decided previously on the machine configuration. Thanks to that, the user will be able to create a base with the whole environment of the machine and from this getting replicas. Automatically the system is able to erase problematic configurations, such as IPs or MACs and re-generate them in each creation of a new virtual machine. As it is shown in the following image, the platform let to create machines with personalized users, or with some service ready to work.

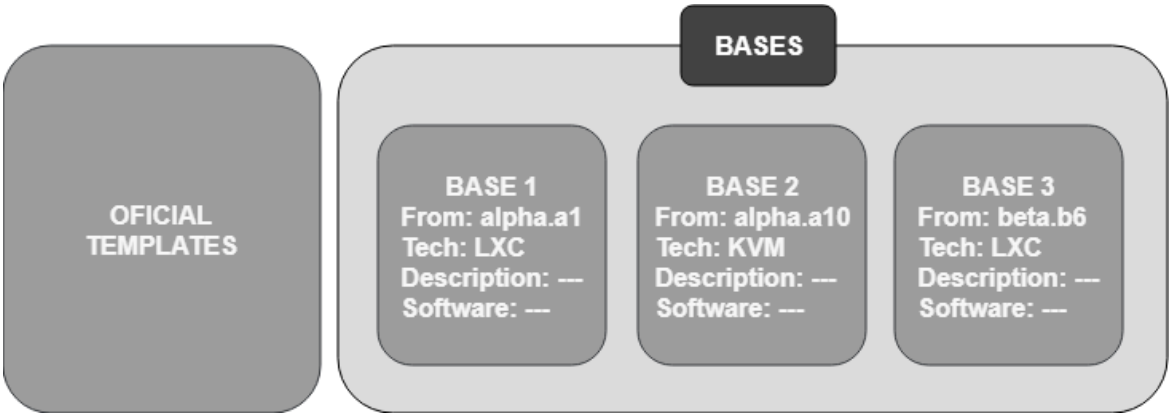

Figure 7. Bases

Rebase

A problem detected when creating bases, is that after a while, the machines have been updated and new software has been installed causing a descordination between machines.

In order to understand that better, this section will be focused in the following example: a laboratory of a network subject in the university has initially created 10 machines from the same template. The students have been working on them for 6 months, and finally, all the machines are left updated and a large percentage of the class has installed a specific software, for example nmap, software to analyze ports and network configurations. At this point, if for the next semester, the user administrator wants to install nmap in all the machines by default he will have to install it individually in all the machines, so the software will be replicated in all of them. Also, if he wants to create some more machines, the user administrator of the university will have to update it with the new software nmap so that it contains the same as the rest machines.

For this reason, the platform includes a tool called the rebase that allows the user to identify all the differences, both at the software level and at the configurations, between the virtual machines and their base. This analysis is carried out automatically and generates a file with all the tasks that must be done in the base to match all the machines. The procedure when executing is as follows: first, it will make a copy of the base, then the necessary tasks will be managed to create the machines with the same

configuration they had initially. Once everything in the new prepared environment is ready, the platform will request to stop the old machines and turn on the new ones. Finally, when the configurations of the new machines are replicated the old environment could be deleted.

One of the main advantages is that the software is not replicated in all the machines, but is installed in the base itself, therefore, it reduces the difference between machines. The process that has been followed is detailed graphically in the following image.

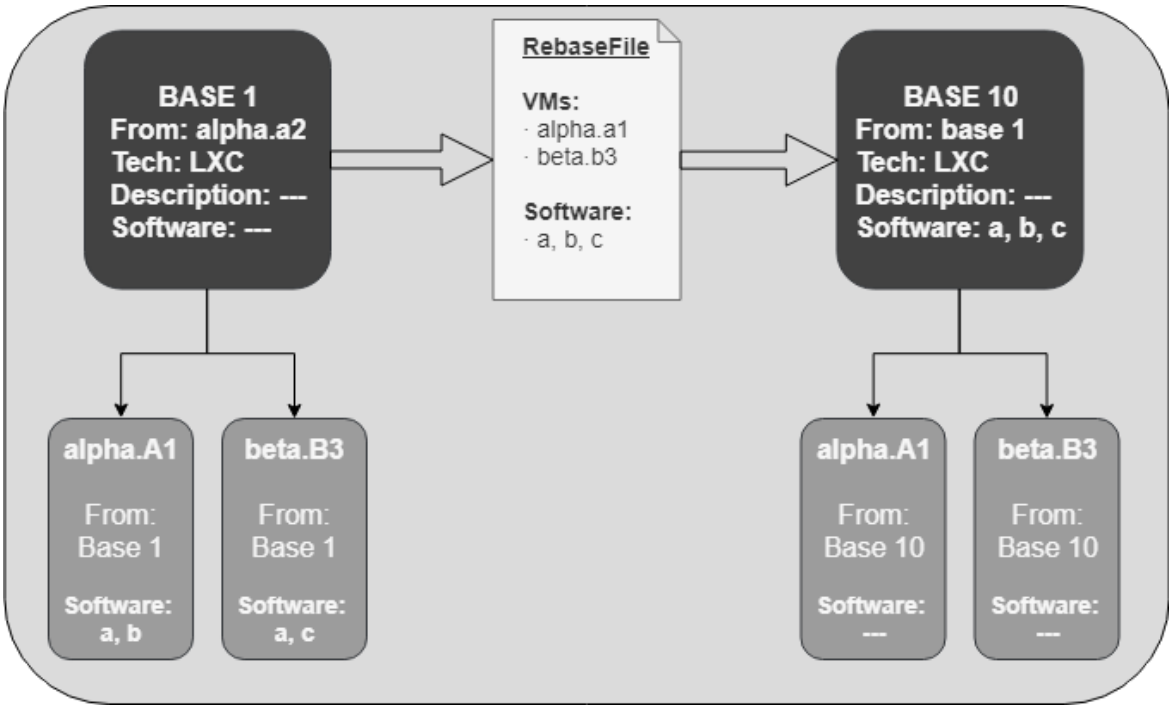

Figure 8. Rebase

Provisioning

OpenHuaca natively brings diverse provisioning tools, it can execute commands remotely from the host and also it has the ability to manage users of the machines or generate consoles to be able to make the necessary checks. Also, of course, of the meditating SSH connection certificates.

In addition to provisioning tools, monitoring tools have been prepared, both resources and processes of each machine. Everything from the host, without the need to connect to any machine.

As the platform already had basic monitoring tools and the proposal was to minimize the impact of OpenHuaca on their machines, the final decision was not to install any agent in them, therefore, the integration with Ansible has been prepared, which combined with our connection SSH and its certificates, generates a perfect blend for the automation of all types of processes.

Later, it is explained what HuacaHub is, but for the moment, a simple description is that playbook templates have been created and distributed through the platform portal, so that the users will have both official templates tested and contributions from the community.

Infrastructure as a Code

OpenHuaca also has a IaaC service, Infrastructure as a Code, through integration with Terraform. In this way, the user will be able to create a HuacaFile, with all the configurations of a test environment, teacher, etc.. and simply executing it in the platform software generating everything of the desired environment. Likewise, with the Ansible playbooks, the Huaca Files will also be distributed in our HuacaHub.

HuacaHub

Finally, HuacaHub is a web platform that intends to create a community, a dynamic and interactive area where the platform bases, playbooks and Huaca Files will be distributed. Also, the users will be able to upload their environments, to share them and other users can evaluate the rest. There are also forums and user guides to facilitate the learning curve of our tool.

Results

The final result of this Thesis is the first release of OpenHuaca. In this version it has been possible to create the LXC functionalities, but not the KVM. Moreover, there are extra subcommands to manage domains, certificates and bases. A first approach to provisioning are implemented with an attach subcommand to execute processes in a container, monitor it, the possibility of administrate users of each container and the ability to create certification authorities and control their certificates.

Attached to this document there are the installable package of OpenHuaca, the and WhitePaper and the source files, so there is the entire source code.

In this section, there is a brief summary about the commands and the basic configuration of an OpenHuaca environment.

Command design

1. openhuaca
   - 1.1. [args]:
     - 1.1.1. -i, --ini-file — Custom config file.
     - 1.1.2. -v, --version — Display current version
     - 1.1.3. --help — Show this message and exit.
   - 1.2. bases — Manage bases/templates
     - 1.2.1. copy — Copy a base to a new one
     - 1.2.2. create — Create a base form a VM
     - 1.2.3. destroy — Destroy a base
     - 1.2.4. list — List current bases
     - 1.2.5. rebase — Apply rebase algorithm to a base and their VMs
     - 1.2.6. rename — Rename a base
   - 1.3. certification
     - 1.3.1. ca — Manage Certification Authorities
       - 1.3.1.1. create — Create CA associated to a Domain
       - 1.3.1.2. destroy — Destroy CA
       - 1.3.1.3. list — List current CAs
     - 1.3.2. cert — Manage Certificates
       - 1.3.2.1. create — Create a certificate from {CA, user, expiration time}
       - 1.3.2.2. list — List current certificates
       - 1.3.2.3. purge — Remove revoked certificates from the system
       - 1.3.2.4. revoke — Revoke the permission of an active certificate
   - 1.4. domain — Manage networks
     - 1.4.1. autostart — Manage whose domains start in boot time
     - 1.4.2. create — Create new domain.
     - 1.4.3. destroy — Destroy domain
     - 1.4.4. monitor — Monitor current domains
     - 1.4.5. restart — Restart specific domain or all.
     - 1.4.6. start — Start a domain
     - 1.4.7. status — Get status of all domains and their main variables
     - 1.4.8. stop — Stops a domain or all.
   - 1.5. vm

1.5.1.	attach	Execute process in a VM
1.5.2.	autostart	Starts VM in boot time
1.5.3.	console	Get a remote console of the specified VM
1.5.4.	copy	Copy a VM with a new name
1.5.5.	create	Create a VM from the oficial templates or a base
1.5.6.	destroy	Destroy a VM
1.5.7.	info	Information about a VM
1.5.8.	list	List of VMs and common variables
1.5.9.	nat	Manage NAT from VMs with the host
1.5.10.	resources	Limit resources of each VM
1.5.11.	start	Starts a VM
1.5.12.	stop	Stops a VM
1.5.13.	top	List usage processes in a VM
1.5.14.	user	Manage users of a VM
1.6.	wait	Freeze process until a STATE is accomplished.

Basic environment

This first environment is a basic simulation; so the following information describes the creation of a domain and a unique container. Then, it will be added some users, one with root privileges. Finally, the certificates will be created to allow access to the users. Once checked the correct working of the environment it will be detailed the way to destroy it and leave the host installation clear.

First of all it is necessary to install the Debian package. It can be done with different installations; all of them are into the official Ubuntu repositories like *dpkg* or *gdebi*.

```
gdebi -n huaca*.deb
```

In order to know if the installation has been done correctly, the user can check it executing any command with the parameter version that shows you the current version of LXC and OpenHuaca.

```
openhuaca --version
```

Once checked, the next step is creating the first domain, it will be called alpha.

```
openhuaca domain create -d alpha
```

The output of the command shows the network assigned, for example in this case: 172.20.1.0/24. The parameter status let the user to consult the states of the domains.

```
openhuaca domain status
```

After executing the command below, the user can observe a list with the whole domains generated, in this case just one, and can observe that is stopped and it does not have any certificate authorization, CA. This means that the user is not able to make certificates to this domain.

First of all, the user has to start the domain. This action allows the creation of containers and CAs. A domain with an active CA can generate certificates to let the user access in the containers. In this case, the name of the domain, alpha, it is set as the CA password.

```
openhuaca domain start -d alpha
openhuaca certification ca create -d alpha
```

If the parameter status is executed, it is observed that the domain has the previous assigned network, the CA is active and the state is RUNNING.

```
openhuaca domain status
```

Once the domain is created, the user can generate the container, in this example it is used a LXC template called Ubuntu 18.04, with the following command just indicating the name and the domain:

```
openhuaca vm create -d alpha -n a1 -t
```

The user must have to wait until the template is completely downloaded, after that the base is kept in the cache and creates the container from it. This LXC template is stored in the computer, so if the user wants to create a second container with it, the time of processing will be considerably reduced. In order to start to work with the container, it has to be started with the following command:

```
openhuaca vm start -d alpha -n a1
```

After that, the dnsmasq has to assign a direction to the container, this process can take a time, and finally configures the container in the background.

```
openhuaca vm list
```

The administrator has access to any of the machines as root with the help of the command below that uses namespaces:

```
openhuaca vm console -d alpha -n a1
```

Also, It could send commands with the attach feature.

```
openhuaca vm attach -d alpha -n a1 -c COMMAND
```

Then, if Ubuntu Bionic AMD64 is selected, it needs to be installed and configured SSH service. It can be done with the previous explained subcommands.

```
openhuaca vm console -d alpha -n a1
~#: apt install openssh-server    # Install ssh packages
~#: passwd                       # Set root password
~#: vi /etc/ssh/sshd_config      # Enable root ssh login
~#: systemctl restart sshd       # Restart service
```

At this point, a SSH connection could be made only from the host. To enable this connection from any source direction that can see the host, a NAT configuration is needed. Openhuaca provides NAT management. It needs the domain and a specific virtual machine, and select the external port (host port) and a internal one (vm port).

```
openhuaca vm nat add -d alpha -n a1 -e 8080 -i 22
```

It can be checked that the nat is configured properly with the following command.

```
openhuaca vm nat list
ID  DOMAIN  NAME  IP             EXT_PORT  INT_PORT  PROTOCOL
0   alpha   a1    172.20.0.198   8080      22        tcp
```

Finally, in order to destroy the current environment the administrator has to execute the following commands.

```
openhuaca vm nat delete -i 0
openhuaca vm stop -d alpha -n a1
openhuaca vm destroy -d alpha -n a1
openhuaca domain stop -d alpha
openhuaca domain destroy -d alpha
```

Budget

I am going to consider the cost of this project in case of it is realized in a business or as a research public project.

It has not required any kind of prototype or hardware in order to be developed and all the software used and implemented has been done with open source programs. Only a team has been needed to carry out the software development and a container of the j3o machine of the telematic department, ETSETB.

For this reason, in the following economic study it is just considered that a computer has been purchased to carry out the project development and the fact of being a research project the university has rent all the resources to create the container on the server j3o.

Hardware cost: Computer with i5 cpu, 8GB of RAM and 128GB of M2 disk = **600€**

As I said, no software fee is needed due to open source programs.

Different people have contributed to the work, so the following salaries have been allocated during the project (20 weeks).

- Jose Luis Muñoz. Project director. 50€/h. 10h/week
  - 10.000€ + 15% in taxes. **11.500€**

- Jordi Torres. Project advisor. 50€/h. 1h/week
  - 1.000€ + 15% in taxes. **1.150€**

- Marti . Junior developer. 20€/h. 40h/week
  - 16.000€ + 15% in taxes. **18.400€**

- Rafael Genés . Senior developer. 30€/h. 40h/week
  - 24.000€ + 15% in taxes. **27.600€**

The total cost of the project has needed an investment of **59.250€**

Conclusions and future development

As a summary, a study of the current private cloud tools has been made and we have detected that there is no clear tool to solve the proposed needs. Those proposed needs are looking for a simple, configurable and useful tool that allows us to set up small test environments or even deploy applications without having to waste an important part of the project time assembling and configuring more than improving the project or investigating new ways.

Therefore, upon discovering that there is not a perfect platform, the decision about doing this project was taken. It is decided to distribute it as an installable to optimize its installation time and the versatility that can work on any computer with a linux debian distribution. We also add containers to optimize the time of installation and execution of environments whenever possible, despite this, it will be also enabled the virtualization of machines through KVM to ensure that we can setup any environment. Access management has also been added through SSH credentials, domains, NAT or user management. Everything, as we have said, with the clear idea of reducing the time you spend assembling an environment and minimizing its learning curve of the platform.

This has been our initial proposal for OpenHuaca, but some improvements in its functionalities are necessary to obtain a robust tool. The main tools that would consolidate OpenHuaca as a platform to match its major competitors would be the KVM to unlimit the kernel compatibility, the creation of an API for its management, create a webapp to publish our platform, create a community and share auto-installable extensions, so as to allow integrations with other provisioning services, IaaC, or even we can implement improvements in the management of our environments, adding an optional graphic interface in a webapp.

Bibliography

[1] Rafael Genés Durán. “Development of a Multi-Tenant Cloud Platform Based on OS Containers”. Degree Thesis. Barcelona: Escola Tècnica d’Enginyeria de Telecomunicació de Barcelona, Universitat Politècnica de Catalunya, June 2017.

[2] Daniel Campos Gómez. “Integration of KVM in the Openhuaca Cloud Platform”. Degree Thesis. Barcelona: Escola Tècnica d’Enginyeria de Telecomunicació de Barcelona, Universitat Politècnica de Catalunya, June 2018.

[3] M. K. L. Singh, W. A. T. Quijano, and G. Koneru. Evaluation of network performance - type1 open source virtual-ization platforms. In 2015 International Conference on Computer Communication and Informatics (ICCCI), pages 1–5, Jan 2015.

[4] J. Ma, H. Kim, and Y. Kim. The virtualization and performance comparison with lxc-lxd in arm64bit server. In 2016 6th International Conference on IT Convergence and Security (ICITCS), pages 1–4, Sep. 2016.

[5] W. Blair, A. Olmsted, and P. Anderson. Docker vs. kvm: Apache spark application performance and ease of use. In 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), pages 199–201, Dec 2017.

[6] F. Gomez-Folgar, A. Garcia-Loureiro, T. F. Pena, and R. Valin. Performance of the cloudstack kvm pod primary storage under nfs version 3. In 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications, pages 845–846, July 2012.

[7] S. S. Sahasrabudhe and S. S. Sonawani. Comparing openstack and vmware. In 2014 International Conference on Advances in Electronics Computers and Communications, pages 1–4, Oct 2014.

[8] Click Documentation (7.x). url: https://click.palletsprojects.com/en/7.x/ (visited on 15/01/2019).

[9] HowTo/dnsmasq - Debian Wiki. url: https://wiki.debian.org/HowTo/dnsmasq (visited on 16/01/2019).

[10] IptablesHowTo - Community Help Wiki. url: https://help.ubuntu.com/community/IptablesHowTo (visited on 16/01/2018).

[11] Kernel Virtual Machine. url: https://www.linux-kvm.org/page/Main_Page (visited on 10/29/2018).

[12] Linux Containers - LXC - Introduction. url: https://linuxcontainers.org/lxc/ (visited on 10/29/2018).

[13] Man page of DNSMASQ. url: https://wiki.debian.org/HowTo/dnsmasq (visited on 10/21/2018).

[14] TigerVNC. url: https://tigervnc.org (visited on 09/21/2018).