DEVELOPMENT OF A MULTI-TENANT CLOUD PLATFORM BASED ON OS CONTAINERS

A Degree Thesis
Submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by
Rafael Genés Durán

In partial fulfilment
of the requirements for the degree in
NETWORK ENGINEERING

Advisor: Jose Luis Muñoz Tapia

Barcelona, June 2017
Abstract

The target of this project is the development of a cloud platform called OpenHuaca that lets the user of the software the possibility of creating and managing some virtual containers in an easy and fast way. Moreover, it can be administrated with domains. It is thought to be implemented in small or medium stages; for example, in a teaching area can be used to control the computers of the university laboratories. OpenHuaca is ready to help with all the students’ necessities in a centralized way. So, every student can connect to all containers with their specific digital certificate.

The complexity of the project has made it necessary to create a working team formed by a supervisor and various TFG students from different universities in order to introduce new functionalities to the project quicker.
Resum

L'objectiu d'aquest projecte és el desenvolupament d'una plataforma cloud anomenada OpenHuaca que permet als usuaris la possibilitat de crear i gestionar diversos contenidors virtuals d'una forma ràpida i cómoda. A mes a mes, pot ser administrat per dominis. Esta pensat per petits o mitjans escenaris, per exemple, en l'àrea docent pot controlar els ordinadors dels laboratoris de l'universitat. OpenHuaca esta preparat per ajudar en totes les necessitats dels alumnes d'una forma centralitzada. Per lo tant, cada estudiant podrà conectarse als seus contenidors amb el seu propi certificat digital.

La complexitat del projecte ha fet necesari la creació d'un equip de treball format pel supervisor i alguns estudiants de TFG de diferentes universitats per crear noves funcionalitats al projecte.
Resumen

El objetivo de este Proyecto es el desarrollo de una plataforma cloud llamada OpenHuaca que da a los usuarios la posibilidad de crear y gestionar diversos contenedores virtuales de una forma rápida y sencilla. Además, puede ser administrada por dominios. Esta pensado para pequeños o medianos entornos, por ejemplo, en el área docente puede controlar los ordenadores de unos laboratorios de la Universidad. OpenHuaca está preparado para cubrir las necesidades de los alumnos de una forma centralizada. Por lo tanto, cada estudiante podrá conectarse a sus contenedores con su propio certificado digital.

La complejidad del proyecto ha hecho necesaria la creación de un equipo de trabajo formado por el supervisor y varios estudiantes de TFG de diferentes universidades para crear nuevas funcionalidades al proyecto.
Dedication:

First of all, my parents, who have given me support and have listened to my ideas and problems without understanding what I was talking about in many cases.

Secondly, Carla, who has never let me abandon my goals and has made me grow as a person.

At the end, Jose, who has transmitted me the passion for teaching and researching.
Acknowledgements

OpenHuaca has been divided in various TFG projects, so the supervisor and the team mates have helped in the development of the thesis:

Jose Luis Muñoz, who is a teacher from the ETSETB doctorate in Network, as the supervisor, has given the team advice during development. At the same time he was one of the beta testers and has done a functional review of all the documentation.

Gerard Bernado Cadenas, who is a student from Terrassa UPC, as a co-worker, has been another beta tester of the code.

Jorge Eduardo Buzzio Garcia, who is a student from Lima, Peru, as a co-worker, has been another beta tester of my code. In addition, he has done a functional review of the thesis with my supervisor.

Carla, who is a student from the ETSETB, as a co-worker, has done a writing review of the documentation.

Finally, the j3o networking team collaborated with the initial developments of the LXC extension.
**Revision history and approval record**

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12/06/2017</td>
<td>Document creation</td>
</tr>
<tr>
<td>1</td>
<td>19/06/2017</td>
<td>Document language revision</td>
</tr>
<tr>
<td>2</td>
<td>20/06/2017</td>
<td>Document content revision</td>
</tr>
<tr>
<td>3</td>
<td>30/06/2017</td>
<td>Delivery document</td>
</tr>
</tbody>
</table>

**DOCUMENT DISTRIBUTION LIST**

<table>
<thead>
<tr>
<th>Name</th>
<th>e-mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rafa Genés Durán</td>
<td><a href="mailto:rafa.gd.10@gmail.com">rafa.gd.10@gmail.com</a></td>
</tr>
<tr>
<td>Gerard Bernado Cadenas</td>
<td><a href="mailto:arktan4@gmail.com">arktan4@gmail.com</a></td>
</tr>
<tr>
<td>Carla Brugulat Rica</td>
<td><a href="mailto:cbrugulat@yahoo.es">cbrugulat@yahoo.es</a></td>
</tr>
<tr>
<td>Jorge Eduardo Buzzio Garcia [Perú]</td>
<td><a href="mailto:jbuzzio410@gmail.com">jbuzzio410@gmail.com</a></td>
</tr>
<tr>
<td>Jose Luis Muñoz Tapia</td>
<td>jose.muñ<a href="mailto:oz@entel.upc.edu">oz@entel.upc.edu</a></td>
</tr>
</tbody>
</table>

Written by: Rafa Genés Durán        Reviewed and approved by:

<table>
<thead>
<tr>
<th>Date</th>
<th>Name</th>
<th>Position</th>
<th>Date</th>
<th>Name</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>13/06/2017</td>
<td>Rafa Genés</td>
<td>Project Author</td>
<td>30/06/2017</td>
<td>Jose Luis Muñoz Tapia</td>
<td>Project Supervisor</td>
</tr>
</tbody>
</table>
# Table of contents

Abstract ............................................................................................................................ 1  
Resum .............................................................................................................................. 2  
Resumen .......................................................................................................................... 3  
Acknowledgements .......................................................................................................... 5  
Revision history and approval record ................................................................................ 6  
Table of contents .............................................................................................................. 7  
List of Figures ................................................................................................................... 9  
List of Tables: ................................................................................................................... 10  
1.Introduction .................................................................................................................. 11  
  1.1.Statement of purpose ............................................................................................ 11  
  1.2.Requirements and specifications ........................................................................... 11  
  1.3.Methods and procedures ....................................................................................... 11  
  1.4.Work plan .............................................................................................................. 11  
  1.5.Incidences ............................................................................................................. 11  
2.State of the art of the technology used or applied in this thesis: ................................... 12  
  2.1.LXC ....................................................................................................................... 12  
  2.2.Docker .................................................................................................................. 12  
  2.3.Proxmox ................................................................................................................ 12  
3.Methodology / project development: ............................................................................ 13  
  3.1.Comunication ........................................................................................................ 13  
  3.2.Software delivery ................................................................................................... 13  
  3.3.Documentation delivery ......................................................................................... 13  
4.Results ........................................................................................................................ 14  
  4.1.Guide .................................................................................................................... 14  
  4.2.Tests ..................................................................................................................... 14  
5.Budget ......................................................................................................................... 15  
6.Environment Impact (Optional) .................................................................................... 16  
7.Conclusions and future development: .......................................................................... 17  
  7.1.Summary ............................................................................................................... 17  
  7.2.Conclusions .......................................................................................................... 17  
  7.3.To Do list ............................................................................................................... 17  
Bibliography: ................................................................................................................... 18  
Glossary ......................................................................................................................... 20
List of Figures

Each figure in the thesis must be listed in the “List of Figures” and each must be given a page number for its easy location.

Figure 1: Grantt Diagram 15
List of Tables:

Each table in the thesis must be listed in the “List of Tables” and each must be given a page number for its easy location.

Table 1: WP ref: #1 13
Table 2: WP ref: #2 14
Table 3: WP ref: #3 14
Table 4: WP ref: #4 14
Table 5: Milestones 15
Table 6: Summary 36
1. **Introduction**

This thesis is the origin of the open source project OpenHuaca. It is a cloud platform based on OS containers. At this moment, it has LXC containers that are a kind of virtual machines isolated from the kernel. It manages the container users, resources, permissions, and generates their own domains and certifications to make an access control of each container.

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

1.1. **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.

On the other hand, 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. 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.

At first, the main idea was to create an extended version of LXC, on the date of the critical design delivery; the project plan was updated and the objectives were extended. Finally, the project is a new platform and has the possibility to offer different containers, starting with LXC and with the idea of implementing KVM in a short term.
1.2. **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, for example, an Ubuntu 16.04 LTS.

In addition, the containers need an updated version of LXC, which can be found in the official repositories of Ubuntu.

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. It is possible to modify the resources dynamically using the command line.

1.3. **Methods and procedures**

Different methods have been followed for the software development, the documentation and the communication with the supervisor and other co-workers.

1.3.1. **Software**

The application is programmed in Bash, so initially in order to develop the project the code was written using **nano** that is a cli-text editor that is included in the Terminal. When the project was growing that programme was not efficient enough so, the rest of the code was written using the open source version of **sublime text editor**.

On the other hand, GiT was used to share the code with the tutor, the repository was created in a virtual machine in the j3o.upc.edu server and with the help of credentials the project could be modified in any computer remotely.

Thanks to that the team mates were able to access the code in a simple and fast way. In addition, in this machine was installed another container that let the possibility of installing OpenHuaca and test all its functionalities during development.
Finally, the j3o team of the networking department provided two softwares. The first "debrepos" is a script that allows the developer to generate a Debian package from a few directories and the second was an example script about how to create certification authorities and ssh certificates.

1.3.2. Documentation

In the communication issues a clear procedure has been followed, the moodle of the university has been used to download the templates of the documents; All the documentation have been edited with free software, LibreOffice, and once delivered and reviewed by the tutor have been printed in pdf format to be uploaded.

In order to share all the documentation with the supervisor, it was created a SVN repository in a j3o machine, there were uploaded all the "doc" versions for the revision and after the "pdf" which were the ones that have been officially submitted for the evaluation of the work in the moodle.

Finally, some small PowerPoint or "txt" documents have been created to document pending tasks or doubts towards the supervisor or to some collaborator.

1.3.3. Communication

In matters of communication, it is necessary to divide the information into two big groups, face-to-face and remote communication.

In the face-to-face communication, in order to improve the quality of the communications when designing the functional scheme of the application, the supervisor provided me a workspace on the 1st floor of the C3 building on the north campus of the UPC. In this way, we could have small daily meetings both in the supervisor's office and in my work space. Group meetings were also scheduled together with the supervisor and several collaborators.
On the other hand, remote communications have focused on emails for solving small questions and concreting meetings. In addition, at the beginning the supervisor decided to use Skype as a VoIP calling tool. But finally, the web page meet.jit.si/OpenHuaca was a better option for these communications because it includes the possibility of conversing, chatting and sharing desks in a more efficient way.

Finally, in the j3o-enabled machine, a remote desktop was installed, VNC, so that several people could enter the desktop, allowing the team to have a remote meeting and everyone can view and edit in the same terminal.

1.4. **Work plan**

The project has been divided into 4 work packages; the first was the preparation of the environment and the two followings were about software development, the LXC expansion and the creation of OpenHuaca. Finally, the last work package includes all the documentation made during the course.

This section also contains the milestones list and the updated Gantt diagram of the project.

### 1.4.1. **Work Packages**

<table>
<thead>
<tr>
<th>Project: Cloud platform based on OS containers</th>
<th>WP ref: #1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major constituent: Hardware and environment</td>
<td></td>
</tr>
<tr>
<td>Short description:</td>
<td></td>
</tr>
<tr>
<td>Set up workspace. Install software, repositories, etc...</td>
<td>Planned start date: 16/02/2017  Planned end date: 05/03/2017</td>
</tr>
<tr>
<td>install Ubuntu, SVN, VNC, GIT and prepare j3o container.</td>
<td>Deliverables: -  Dates: -</td>
</tr>
</tbody>
</table>

Table 1: WP ref: #1
### Table 2: WP ref: #2

<table>
<thead>
<tr>
<th>Project: Cloud platform based on OS containers</th>
<th>WP ref: #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major constituent: Software</td>
<td></td>
</tr>
<tr>
<td>Short description:</td>
<td></td>
</tr>
<tr>
<td>Develop LXCE</td>
<td>Planned start date: 05/03/2017</td>
</tr>
<tr>
<td></td>
<td>Planned end date: 16/05/2017</td>
</tr>
<tr>
<td>Wrap LXC commands, add manuals and manage domains</td>
<td>Deliverables: -</td>
</tr>
</tbody>
</table>

### Table 3: WP ref: #3

<table>
<thead>
<tr>
<th>Project: Cloud platform based on OS containers</th>
<th>WP ref: #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major constituent: Software</td>
<td></td>
</tr>
<tr>
<td>Short description:</td>
<td></td>
</tr>
<tr>
<td>Update LXCe to OpenHuaca and add certificates</td>
<td>Planned start date: 16/05/2017</td>
</tr>
<tr>
<td></td>
<td>Planned end date: 30/06/2017</td>
</tr>
<tr>
<td>Update to OpenHuaca and manage user certificates</td>
<td>Deliverables: -</td>
</tr>
</tbody>
</table>

### Table 4: WP ref: #4

<table>
<thead>
<tr>
<th>Project: Cloud platform based on OS containers</th>
<th>WP ref: #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major constituent: Documentation</td>
<td></td>
</tr>
<tr>
<td>Short description:</td>
<td></td>
</tr>
<tr>
<td>Write documentation files</td>
<td>Planned start date: 16/02/2017</td>
</tr>
<tr>
<td></td>
<td>Planned end date: 10/07/2017</td>
</tr>
<tr>
<td>Project Proposal, Critical Review, Thesis and Presentation</td>
<td>Deliverables: -</td>
</tr>
</tbody>
</table>
1.4.2. Milestones

<table>
<thead>
<tr>
<th>WP#</th>
<th>Short title</th>
<th>Milestone / deliverable</th>
<th>Date (week)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>Prepare workspace</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Feature: Initial files</td>
<td>Release 1.0</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Feature: LXC-cmd</td>
<td>LXCe-1.0</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Feature: manuals</td>
<td>LXCe-1.1</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>Feature: domains</td>
<td>LXCe-1.2</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>Feature: OpenHuaca</td>
<td>Huaca-1.0</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>Feature: certificates</td>
<td>Huaca-1.1</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>Doc: Project proposal</td>
<td>Deliver Proj. Prop.</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Doc: Critical Review</td>
<td>Deliver Crit. Review</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>Doc: Thesis</td>
<td>Deliver Thesis</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>Presentation</td>
<td>Deliver PPT</td>
<td>22</td>
</tr>
</tbody>
</table>

Table 5: Milestones

1.4.3. Grantt Diagram

![Grantt Diagram](image)

1.5. Plan changes and incidences

1.5.1. Plan changes

At first, the project was thought to create an extended version of LXC, LXCE, create a wrapper that means replicating the commands of LXC to be able to carry out actions before or after each command and finally creating several new commands. But, in the middle of the project, the main idea turns into a project with a broader objective that was creating an independent platform, OpenHuaca, which can have different containers from LXC or KVM. In the course of this work the LXC containers have been developed, but the KVM ones will be incorporated in a short-term.
1.5.2. Incidences

Mainly we have had two incidents during the process of realization. As it was explained before, the OpenHuaca project has been created, which was not initially thought, so the time expected at the beginning for every part of the project had to be changed completely because this new project, OpenHuaca, was more ambitious and needed to carry out other developments. So in the end the part of revoking and cleaning the certificates has remained as a task to be performed. In this moment, the command de dedicated to that las part only warns through the system of logs that the command is not implemented.

Finally, also comment that in the creation of the bash manuals, the fact of not having completed the command certification; it has not been possible to finish the implementation of the manual completely.
2. **State of the art of the technology used or applied in this thesis:**

In order to know the potential of use of this product in the market, it was necessary to make a research about the cloud platforms with the services that offer kernel-level virtualisation that exist nowadays. Therefore, services with Virtual Box are discarded because they create virtual machines on the host OS and not directly on the kernel of the computer.

So, the following paragraphs contain a review of the state of art of container virtualization, specifically with three open source programs, LXC, Docker and Proxmox.

2.1. **LXC**

LXC is a program without graphical interface where a user can carry out the container management, from creating diverse types of Linux distributions to assigning limited resources to each one. But unlike OpenHuaca, it does not have a user management, nor allow a user to manage the domains or any kind of modification of the network. It only generates a bridge on the host and binds all containers with VETH level 2 links. So, all the containers are generated in the same network and are connected to each other.

Also it should be noted that OpenHuaca prepares the containers to accept ssh connections and orchestrates the certification authorities and their certificates. In addition, the work of Gerard, one of the team mates, has been to create a web interface that let the user controlling and monitoring OpenHuaca. Finally, it must be said that OpenHuaca allow the user to create containers type LXC and type KVM, while LXC is limited to its technology.

LXC is a useful tool but it is too basic compared with that product, OpenHuaca. In short, it has not user, certificate or domain management. Nor does it have any kind of graphic or web interface. For these reasons, OpenHuaca can beat LXC in the market section.
2.2. **Docker**

Docker has also started from the LXC libraries, but has specialized in creating containers to run a single process, while OpenHuaca virtualizes an entire operating system. Docker manages all domains, interfaces and even has a large repository where you can find a large number of specific containers. The big difference is that these containers only execute a single process, so this makes a big difference to OpenHuaca.

Docker is designed to run from a host and create different services, so it performs container virtualization, but is not intended to manage an entire computer environment. However, it is a really useful tool; it is not focused on the same public as OpenHuaca, allowing the project to have a market that does not match it.

As a summary, Docker is a very complete and improved tool from LXC, but has focused on virtualizing processes and not the entire operating system creating differentiation from OpenHuaca.

2.3. **Proxmox**

The last one remarkable to talk about is Proxmox, is an open source container virtualization program, both the LXC type and KVM. It also has a great control of domains and has a very comfortable web interface where it is possible to monitor and manage the containers. This program is the most similar to OpenHuaca but has two major differences that can be considered large enough to think that OpenHuaca has a different audience from Proxmox.

The first difference is how to distribute and install the software. In Proxmox a user needs a cluster of machines where to install an ISO file, that means to carry out any test you need to invest as minimum 3 hardware equipment, in order to realize all the configuration to link them and finally to be able to create your first container. By contrast, OpenHuaca is distributed as a Debian installable package, so that in any Linux distribution the user will be able to create containers in a matter of seconds.

The other big difference is that Proxmox is intended for large environments, as it is said before; the installation needs at least a cluster of 3 nodes. It is true that gain reliability in your machines, but for small environments in general a user does not have 3 or 5 hardware equipment, so OpenHuaca offers the user the possibility of in a single system install the package and set up their own virtual environment.
In short, Proxmox is the most similar tool to OpenHuaca, but there are two key points that differentiate the product of this project and make it go to one public or another. Proxmox is distributed in ISO files, so the user have to install their own operating system while OpenHuaca distributes Debian packages, DEB, that allows to be installed in any Linux distribution. The other difference is that Proxmox is designed for big stages so a user needs an initial investment in hardware much greater than using OpenHuaca.
3. **Methodology / project development:**

In this section there is a detailed explanation about the methodology followed when communicating with the tutor and team mates. Also, it includes a description about the steps taken to develop the code and how where shared with the supervisor. At the end, there is the procedure of creating the work documentation.

3.1. **Communication**

As it was stated in the introduction, the communications can be divided between presence and remote.

In the beginning of the project realization the supervisor hold a face-to-face meeting each week to discuss the progress and the improvements. At first, it was very comfortable, especially when it was necessary to configure the environment, but later, during the development of the software, a space in the C3 building was adapted for the project needs, so that in case of doubt the tutor could help quicker.

Thanks that the face-to-face communication based in weekly meetings was substituted by remote communication through electronic mail and sporadic consultations to the supervisor. In addition, the complete team had some meetings during the process of development through remote conversations.

On the other hand, if the tutor could not meet in person, the meetings were remotely. Thanks to a KVM machine enabled in the server j3o, which will go into detail in the software part, the group could make audio meetings by skype and share the desktop of the virtual machine, so apart from speaking this tool let the team work on the same screen

Finally, when the conversation involved Jorge, from Peru, was needed to use jit.si because the j3o machine has not got enough speed to work properly. Jit.si is a web service that allowed the team to share audios, videos and even sharing desktop in an acceptable video quality.
3.2. **Software**

At the beginning of the work the supervisor decided to use GIT as a method of sharing the software, so Jose, the tutor, gave me a KVN virtual machine on the j3o server of the networking department. In the machine a certificate was generated in order to get access and configure a repository GIT with the code. So, from my personal computer locally in the code and go uploading updates "commits".

Also, the tutor had access to the repository, so he could download the code and check the updates and improvements daily. This development environment let the supervisor to add new co-workers easily in order to contribute in the development of the code.

Once defined the methods with which the team shared the code, the following paragraph includes a brief explanation about how was the process of developing the code.

Initially, the code was edited directly from the terminal, using a text editor without graphical interface, called Nano, which is included in the official Ubuntu repository. This tool was useful while the code was not more than a couple of scripts, because it let a user edit a single document, so when the code was growing it was not comfortable at all. For this reason, the text editor was changed from Nano to Gedit that comes integrated in Ubuntu. This interface allowed working with different documents in parallel, but it did not seem to be quite comfortable enough, because it had some lacked functionalities, mainly when looking for text in multiple documents at the same time. Finally, the code was developed with a freeware editor called Sublime Text Editor, free software from a company, which allowed editing multiple documents and replace texts in multiple documents.

Once the development of the version was done, the next step was uploading it to GIT and later, from the KVM machine, using the software "deprepos-j3o" creating an installable Debian package from the repository with the newly updated code. After that, in another virtual machine it was necessary to install de software and make test about all the functionalities.
3.3. Documentation

In the documentation part has been followed a simple way, although maintaining the philosophy of free software, so all the documentation has been written with LibreOffice. Once all the documentation was done, it was sent to the supervisor for validation. Finally, when the documents were approved by the tutor a pdf version was print and uploaded in the UPC Moodle service.

In order to share all the documentation with the supervisor, he created a SVN repository in the machine j3o, so all the files were saved there in its editable version and in PDF.

Finally, the next documentation tasks that have to be done is the realization of a paper, which will be done in Latex language and will be shared in the svn repository.
4. Results

The final result of this TFG project is the OpenHuaca-1.1 version. In this version it has been possible to create an LXC wrapper, so it has exactly the same functionalities of LXC, but realized with the Huaca commands. Also, some extra commands have been created to manage the generation of domains and their monitoring. Finally, we added other commands in order to administrate the users of each container and a last command to create certification authorities and control their certificates.

At the end of this document there are attached the installable package of OpenHuaca, the source files and the Developer’s Guide, so there is the entire source code of the version 1.1. The guide for developers of OpenHuaca, it is designed in order to facilitate the joining of new co-workers to the project or to make it easier the understanding of the functionalities by a user.

I decided to create it because it is a project that will last a long time where more people will work, in this way, I decided to create the guide to know the tool and facilitate the collaboration of new students.

In this section, there is a brief summary about the commands and the basic configuration of an OpenHuaca environment. All the detailed information of each command and a guide to create any type of environment can be found in the documentation attached to this project. In addition, there is also a technical explanation of how each functionality has been performed.

4.1. OpenHuaca Commands

In this section there is a brief description about every command and it is specified the parameters needed by all of them. Finally, there is included at least one typical example of the functionality. Nowadays, OpenHuaca functionality focuses on CLI, so all the interactions between the user and the product are implemented with commands.

After this small introduction there is the complete list of commands in alphabetical order:
• **huaca-attach**
  Let the user interact with the container wanted. It has the option of connecting to any active container with the help of the namespaces. Also, it can execute a command in a remote way that means not connecting to the container directly. That tool is very useful in order to scripting; the script will allow the user the possibility of starting a container and executing a command as privileged user into the container.

• **huaca-autostart**
  LXC has the functionality to activate containers while the OS host is starting. This command allows the user to manage the state of a determinate container and also the order of the execution can be modified by priorities.

• **huaca-backup**
  This command has been made by the j3o team of the network department. It makes backups of the filesystems. It can be configured in order to make the backups from some containers or from all of them. Also, the user can decide if the backups have to be done periodically, in a fixed date or in the same moment.

• **huaca-bases**
  This command has been made by the j3o team of the network department. It lists the templates created. It means that, the user can create a basic container and from it making copies.

• **huaca-build**
  This command complements the one over, huaca-bases, let the user list the templates and also create new containers from the existent bases. It is useful to create containers with software or with an environment preconfigured.
• **huaca-certification**
  Let the user manage the authorities of certification and the certificates. Once the certification authority is created, it is configured in all the containers of the domain. On the other hand, when a certificate is created, a public and a private key are generated too. These three files are kept in a directory from the host in order to be distributed easily.

• **huaca-cgroup**
  This command has been wrapped from LXC. It restricts the resources offered to every container. For example, the user can define a memory limit or a bound for the CPU available in each machine.

• **huaca-checkconfig**
  This command has been wrapped from LXC. It shows the whole information related with the environment configuration, it is inherited from LXC so it displays the information corresponding to this program.

• **huaca-checkpoint**
  This command has been wrapped from LXC. It allows serializing containers on the disk so that they do not consume resources and let the possibility of returning to the original state in case of necessity. Moreover, it let the user to control the state of the containers when are serialized, that means stopping or restarting them on time of recovering.

• **huaca-config**
  This command has been wrapped from LXC. It let the user to modify the predefined variables such as the directory where the containers are stored, the memory assigned by default or the place where the certifications are kept.
• **huaca-console**
  It allows the user to login in a container. It changes the established terminal or another for the tty of the container specified. It is so useful in order to access to the user own machines with the help of the namespaces.

• **huaca-copy** It creates a copy of a container. It is useful in order to create a template, a base, and from it, coping various containers. Very used in the creation of the command build and backup.

• **huaca-create**
  This command has been wrapped from LXC. It creates new containers, it has several options like defining the domain where have to be added or giving it a name. Apart from that the user can configure the type of the container. Nowadays, the program dispose of the official LXC templates repository, but in following updates of the project the user will be able to add KVM containers too. Finally, the filesystem needed can be also defined.

• **huaca-destroy**
  It destroys one or various containers from a domain. It is important to have the container stopped, if not the command is not going to work.

• **huaca-device**
  It adds a device to a container like a USB, a HD or a disc reader. That let the user to append hardware to the filesystem wanted.

• **huaca-domain**
  This command has been created in order to manage the domains. It lets the user to create, delete, list, start, stop or restart domains. Apart from that, OpenHuaca offers to the users a service that starts the domains at the initialization of the OS host.
• **huaca-execute**
  It combines the functionality of the commands create and attach. It let the user to test if a container exist, if not it creates the new container and executes the commands.

• **huaca-freeze**
  It “freezes” a container; it means that it blocks the cpu to all the processes of that container. In this way, we can leave the container in a state that does not consume resources and restore it when we think that it is convenient.

• **huaca-info**
  It is a help command. It gives to the user all kinds of relevant information about a container. For example, the name, the state of the machine, the cpu and the memory in use, the Tx and Rx consumed, etc.

• **huaca-ls**
  It is another help command, allows the user to view in a friendly way all the containers created, being able to filter them by domains, states, network data, etc. It is very useful to know relevant information of several containers at the same time.

• **huaca-monitor**
  This command has been wrapped from LXC. It let the user to monitor the state of a container. So, the user will be able to create a register of states of every machine.

• **huaca-nat**
  This command has been made by the j3o team of the network department. It manages the nat between the containers and the host. So, the user can redistribute the traffic from the host to one of the containers. For example, if arrives a packet to the host requesting a web page in the Port 80, the user can have a register in the nat
in order to send it to the container “web” and that will be the manager of returning the content to the webpage.

- **huaca-rebase**
  This command has been created by Jorge, a Peru project mate. It was developed by the need to update a container that was created from a database and has been updated. This command generates a new container from the new updated database but preserving the data and software installed in the desired container.

- **huaca-snapshot**
  This command has been wrapped from LXC. It manages the snapshots created, the user can copy, restore or delete old captures.

- **huaca-sshconf**
  This command has been made by the j3o team of the network department. It is a complementary command of the huaca-nat that configures the connections of the containers created.
  This command is deprecated, so it will be eliminated in following updates of the project.

- **huaca-start**
  This command has been wrapped from LXC. It let the user to create a container from a LXC template. It generates the containers in a directory specified in the configuration of the system, and it separates them in domains. Also, it configures the containers to accept the corresponding CAs.

- **huaca-stop**
  This command has been wrapped from LXC. It function is the contrary of huaca-start, so it stops the containers selected. This command does not delete the containers, it just stop them.
• **huaca-top**
  This command allows the user to list all the containers according to the consumption they have in real time. It can also filter them to show just a single domain or several.

• **huaca-unfreeze**
  It function is the contrary of huaca-freeze, so it is responsible for reassigning cpu times to the desired container by returning it to the exact state where it was previously.

• **huaca-unshare**
  This command has been wrapped from LXC. Nowadays it is deprecated, so it will be eliminated in following updates of the project.

• **huaca-user**
  It manages the creation, the deletion and the list of the users from a container. It is possible to add a password and to establish permissions of administration in the container.

• **huaca-userexec**
  This command has been wrapped from LXC. Nowadays it is deprecated, so it will be eliminated in following updates of the project.

• **huaca-wait**
  It controls the state of a specific container and it does not execute any command before the container has arrived to the established state.
4.2. **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
dpkg -i 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.

```
huaca-create --version
```

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

```
huaca-domain create 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.

```
huaca-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.

```
Huaca-domain start alpha
Huaca-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.
Once the domain is created, the user can generate the container, in this example it is used a LXC template called Ubuntu 16.04, with the following command just indicating the name and the domain:

```
Huaca-create –t Ubuntu –d alpha –n a1
```

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:

```
Huaca-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.

```
Huaca-domain list
Huaca-ls –f
```

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

```
Huaca-attach –d alpha –n a1
```

In order to enable access to the users of the environment created, it is necessary to generate a list of users. By default the system has created the user "Ubuntu". So, a pair of new users is incorporated to the environment with the commands below, one without administrator permissions and the other with.

```
Huaca-user create –d alpha –n a1 –u huaca –p huaca123
Huaca-user create –d alpha –n a1 –u superhuaca –p superhuaca123 –s
```

The parameter list it can be used to see if the generation of the users have been done correctly:

```
Huaca-user list –d alpha –n a1
```
The next step is creating a certificate per user with the parameter –w the user can specify the period of validation, expressed in weeks.

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huaca-certificate cert create –d alpha –u huaca –w 1</td>
</tr>
<tr>
<td>Huaca-certificate cert create –d alpha –u superhuaca –w 52</td>
</tr>
</tbody>
</table>

Against the user can validate the task done before with the parameter list:

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huaca-certificate cert list –d alpha</td>
</tr>
</tbody>
</table>

Once the environment is completely configured, the last step is giving the certificates to the users in order to let them connect to the virtual machines. All certificates are stored in the "/etc/huaca/certification/cert/" directory. It is very important not to delete them from this directory in order to keep a control of the issued certificates and to be able to send them again in case the user loses his proper.

To confirm that the user has access, the best way is to test the ssh connection.

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ssh <a href="mailto:huaca@a1.alpha">huaca@a1.alpha</a> –i /etc/huaca/certification/cert/alpha-huaca*</td>
</tr>
</tbody>
</table>

Observe, the superhuaca user is in the sudo group and therefore has administration permissions, while not huaca.

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

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huaca-user delete –d alpha –n a1 –u huaca</td>
</tr>
<tr>
<td>Huaca-user delete –d alpha –n a1 –u superhuaca</td>
</tr>
<tr>
<td>Huaca-stop –d alpha –n a1</td>
</tr>
<tr>
<td>Huaca-destroy –d alpha –n a1</td>
</tr>
<tr>
<td>Huaca-certification ca delete –d alpha</td>
</tr>
<tr>
<td>Huaca-domain stop alpha</td>
</tr>
<tr>
<td>Huaca-domain destroy alpha</td>
</tr>
</tbody>
</table>

After that the container has been deleted together with the CAs and the domain. The best option to prove that is using the two commands below:

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huaca-ls –f</td>
</tr>
<tr>
<td>Huaca-domain status</td>
</tr>
</tbody>
</table>
5. **Budget**

That project 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 networking department.

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 lent all the resources to create the container on the server j3o.

**Hardware cost:** i5 computer with 8g of RAM and 1TB of HDD → €600

At the software level, freeware has been used, so we have not had to invest money in licenses. We have not had any costs in this section.

**Software cost:** open source projects → €0

Different people have contributed to the work, so the following salaries have been allocated.

**Personal salaries:**

- **Jose Luis Muñoz,** project manager, charges 15€ / hour and has dedicated half a day during the 4 months of the duration of the project.
  
  Salary: 15€ / hour x 4h x 5days x 22 weeks = 6600 €, adding contributions and tax of approximately 12%, it has cost the project a gross salary of 7,500€

- **Gerard and Jorge,** junior QA tester, charge 10€ / hour and have dedicated 10h each to review the code and the guide.
  
  Salary: 10€ / hour x 10h x 2 people = 200€ adding taxes and added a cost to the project of approximately 230 €.

- **Carla Brugulat,** translation and revision of documents, charges 12€ / hour and has dedicated 10h so you get a salary of 120€ adding tax approximately we spent 130 €.
- Finally, as a junior developer, I charge 10€ / hour and I have dedicated full time for 4 months.

Salary: € 10 / hour x 8h x 5days x 22 weeks = 8,800€ with a 12% tax we get 10,000 €.

The total cost of the project has needed an investment of **18,460€**.
6. **Environment Impact**

OpenHuaca project does not have too much affect to the environment because it is based on developing software.

The fact of creating a virtual environment requires less hardware, or rather less equipment but more powerful ones, which reduces the consumption of materials. By contrast, having more powerful equipment, the levels of energy consumed grow, so OpenHuaca is engaged to find the correct balance between these two concepts explained before.
7. Conclusions and future development:

7.1. Summary

<table>
<thead>
<tr>
<th>Description</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create domain</td>
<td>huaca-domain  create domain</td>
</tr>
<tr>
<td>Destroy domain</td>
<td>Huaca-domain delete domain</td>
</tr>
<tr>
<td>List domains</td>
<td>Huaca-domain status</td>
</tr>
<tr>
<td>Create CA</td>
<td>Huaca-certification ca create –d domain</td>
</tr>
<tr>
<td>Delete CA</td>
<td>Huaca-certification ca delete –d domain</td>
</tr>
<tr>
<td>Create container</td>
<td>Huaca-create –t template –d domain –n name</td>
</tr>
<tr>
<td>Delete container</td>
<td>Huaca-destroy –d domain –n name</td>
</tr>
<tr>
<td>Start container</td>
<td>Huaca-start –d domain –n name</td>
</tr>
<tr>
<td>Stop container</td>
<td>Huaca-stop –d domain –n name</td>
</tr>
<tr>
<td>List containers</td>
<td>Huaca-ls –f</td>
</tr>
<tr>
<td>Create user</td>
<td>Huaca-user create –d domain –n name –u user –p pass</td>
</tr>
<tr>
<td>Create super user</td>
<td>Huaca-user create –d domain –n name –u user –p pass -s</td>
</tr>
<tr>
<td>Delete user</td>
<td>Huaca-user delete –d domain –n name –u user</td>
</tr>
<tr>
<td>List users</td>
<td>Huaca-user list –d domain –n name</td>
</tr>
<tr>
<td>Create cert</td>
<td>Huaca-certification cert create –d domain –u user –w 52</td>
</tr>
<tr>
<td>List cert</td>
<td>Huaca-certification cert list</td>
</tr>
<tr>
<td>Attach to a container</td>
<td>Huaca-attach –d domain –n name</td>
</tr>
</tbody>
</table>

Table 6: Summary
7.2. Conclusions

At the beginning, this work was thought to create an extended version of LXC, but the supervisor proposed to broaden the initial objective and create a bigger project. In this way has been born OpenHuaca, which has all the development in bash, a very useful language to create prototypes or initial ideas but later was considered necessary to make a change of language for future developments. Nowadays, there are two parts that are going to be added: revoke and purge certificates; everything else has been perfectly functional.

On the other hand, personally, I am very happy of the work done during this TFG, since it has taught me to work in a didactic way and I have learned a lot. In addition, speaking with my supervisor I have considered staying at the University and dedicate myself to the research, because it seems to me a very interesting job to be able to do large projects like OpenHuaca.

7.3. Work in Progress

In this part of the document the updates and future developments of the project are described and planned.

- **In Progress:**
  
  Gerard ➔ Implementing the web interface.

  Jorge ➔ Implementing Terraform plugin

  Rafa ➔ Purging and revoking the functionalities of huaca-certification.

- **Tasks to do:**
  
  Implement KVM.

  Create some bases.

  Update the manuals.

  Rewrite project commands in Python.
It should be noted that this first version of the document, in order to be presented in Rafa’s TFG thesis, only includes the work done by him. The next versions of the document will include the work done by all project partners.
**Bibliography:**


