

# MORDISCOS

Autores:

- José Ángel Ael Talavera

ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED  
CURSO 2020-2021

IES LEONARDO DA VINCI



# Contenido

1. Introducción.....	4
2. Objetivos.....	5
3. Componentes de la infraestructura.....	5
3.1. Diagrama lógico .....	6
3.2. Tecnologías empleadas .....	7
4. Organización y Gestion.....	8
4.1. Repositorio .....	8
4.2. Git.....	8
4.3. Tablero de Kanban .....	8
4.4. Máquina “Gestion” .....	8
4.5. Ansible .....	8
5. Desarrollo .....	9
5.1. Entorno de virtualización (Proxmox) .....	9
5.1.1 Instalación.....	10
5.1.2 Creación del cluster .....	14
5.1.3 Contenedores.....	17
5.2. Ansible .....	21
5.2.1 ¿Qué es Ansible?.....	21
5.2.2 ¿Cómo funciona Ansible? .....	21
5.2.3 ¿Cómo trabaja Ansible?.....	21
5.2.4 Roles .....	22
5.2.5 Seguridad con Ansible .....	25
5.3. Máquina “Gestion” .....	26
5.4. Capa de almacenamiento (Raid 1 y NFS).....	27
5.4.1 Creación del almacenamiento en raid 1 .....	27
5.4.2 Sistema de archivos en red (NFS).....	27
5.5. Sincronización horaria (NTP).....	30
5.5.1 Instalación.....	30
5.5.2 Configuración.....	31
5.6. Balanceo de carga (HAProxy).....	32
5.6.1 Instalación de HAProxy:.....	32
5.6.2 Configuración del servicio: .....	33
5.7. Acceso con alta disponibilidad (Heartbeat).....	36
5.7.1 Instalación y configuración .....	36



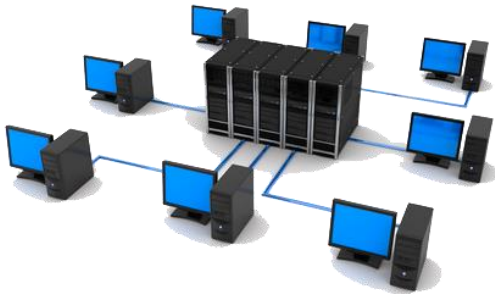
5.8. Firewall (Iptables).....	41
5.8.1 Script .....	41
5.8.2 Creación del servicio .....	42
5.9. Resolución de nombres (DNS).....	43
5.9.1 Instalación.....	43
5.9.2 Configuración.....	43
5.9.3 Compra del dominio mordiscos.ovh.....	45
5.9.4 Registros del dominio en el proveedor .....	45
5.9.5 Configuración de los servidores DNS del proveedor .....	46
5.10. Certificados .....	47
5.10.1 ¿Qué es certbot?.....	47
5.10.2 ¿Cómo funciona Webroot? .....	47
5.10.3 Instalación .....	48
5.10.4 Configuración.....	49
5.11. Bases de datos (MySQL) .....	50
5.11.1 Instalación .....	51
5.11.2 Configuración .....	51
5.11.3 Preparando la replicación.....	52
5.11.4 Iniciando la replicación.....	54
5.12. Backups (mysqldump) .....	55
5.12.1 Script.....	55
5.13. Contenedores y orquestadores (Docker y Docker Swarm) .....	56
5.13.1 Contenedores .....	56
5.13.2 Orquestadores.....	58
5.13.3 Servicios del cluster .....	59
5.13.4 Creación de nuestro primer servicio .....	60
6. Conclusiones .....	68
7. Bibliografía .....	69



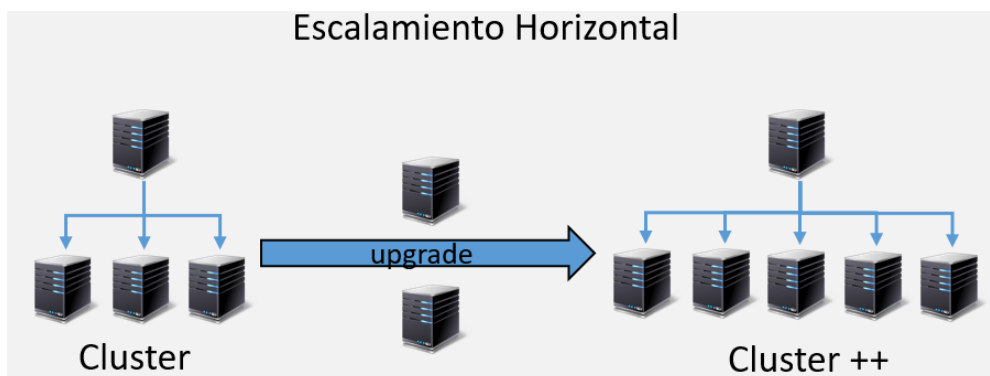
# 1. Introducción

Las empresas, a día de hoy, tienen que mantener una alta productividad y si algún sistema dejase de estar disponible, el rendimiento de ésta se vería afectada.

En este punto entran los términos de **alta disponibilidad y escalabilidad**, de tal forma que la alta disponibilidad será la encargada de garantizar la continuidad de servicios y la escalabilidad será la encargada de permitir una capacidad de crecimiento en magnitud a un sistema.



Cabe mencionar que hay dos tipos de escalabilidad: **escalabilidad vertical**, que es la que permite aumentar la potencia de un equipo, de tal forma que soporte mayor carga, y la llamada **escalabilidad horizontal**, que es la que permite aumentar el número de equipos para poder distribuir la carga.



Debido a estos dos términos, ha crecido la demanda de las infraestructuras de alto rendimiento.

No obstante, este tipo de infraestructuras conllevan el uso de unas aplicaciones complejas que, por regla general, **es necesario que estén instaladas en varias máquinas**.

Es por ello que son necesarias las **herramientas de orquestación**, permitiéndonos automatizar el despliegue, la gestión, el escalado, la interconexión y la disponibilidad.

En este proyecto se va a construir una infraestructura de este tipo, de tal forma que se pueda contemplar cómo funciona la alta disponibilidad y qué permite a la infraestructura que sea escalable de forma horizontal. Usaremos la herramienta de orquestación **“Ansible”** para el despliegue de toda la infraestructura. Cabe destacar, que a lo largo del documento explicaremos y desarrollaremos con detalle los pasos a seguir, de tal forma que se pueda replicar nuestra infraestructura **sin la necesidad de la utilización de Ansible**.

## 2. Objetivos

El principal objetivo del proyecto será tener una infraestructura en la que podamos parar el servicio de varias máquinas y que ésta siga sirviendo un servicio funcional, **aplicando así la alta disponibilidad**.

Otro de los objetivos fundamentales de la infraestructura es que sea capaz de poder implementar nuevas máquinas con las configuraciones apropiadas para lograr una escalabilidad horizontal.

## 3. Componentes de la infraestructura

Para lograr los objetivos mencionados con anterioridad, se ha de disponer de una serie de componentes en la infraestructura, tales como los siguientes:

Será necesario un **entorno de virtualización de servidores** y un equipo real en el que alojarlo. El entorno será el encargado del despliegue y Gestion de las máquinas virtuales y de los contenedores. En nuestro caso, vamos a utilizar **Proxmox Virtual Environment**, que nos permite crear un cluster entre dos equipos y una Gestion cómoda desde una interfaz web.

Dichas “máquinas virtuales” serán **contenedores**, permitiendo una mayor fluidez a los procesos y siendo menos propensos a la sobrecarga de los recursos del anfitrión.

El cluster de Proxmox estará dividido en **dos nodos**, uno de ellos será denominado como “**Rebel**” y el otro será denominado como “**Vulcan**”.

Rebel utilizará una máquina virtual llamada “**Gestion**”. Esta máquina, será la encargada de centralizar y gestionar servicios importantes, tales como los certificados de certbot, los registros DNS, **los despliegues con Ansible**, etc.

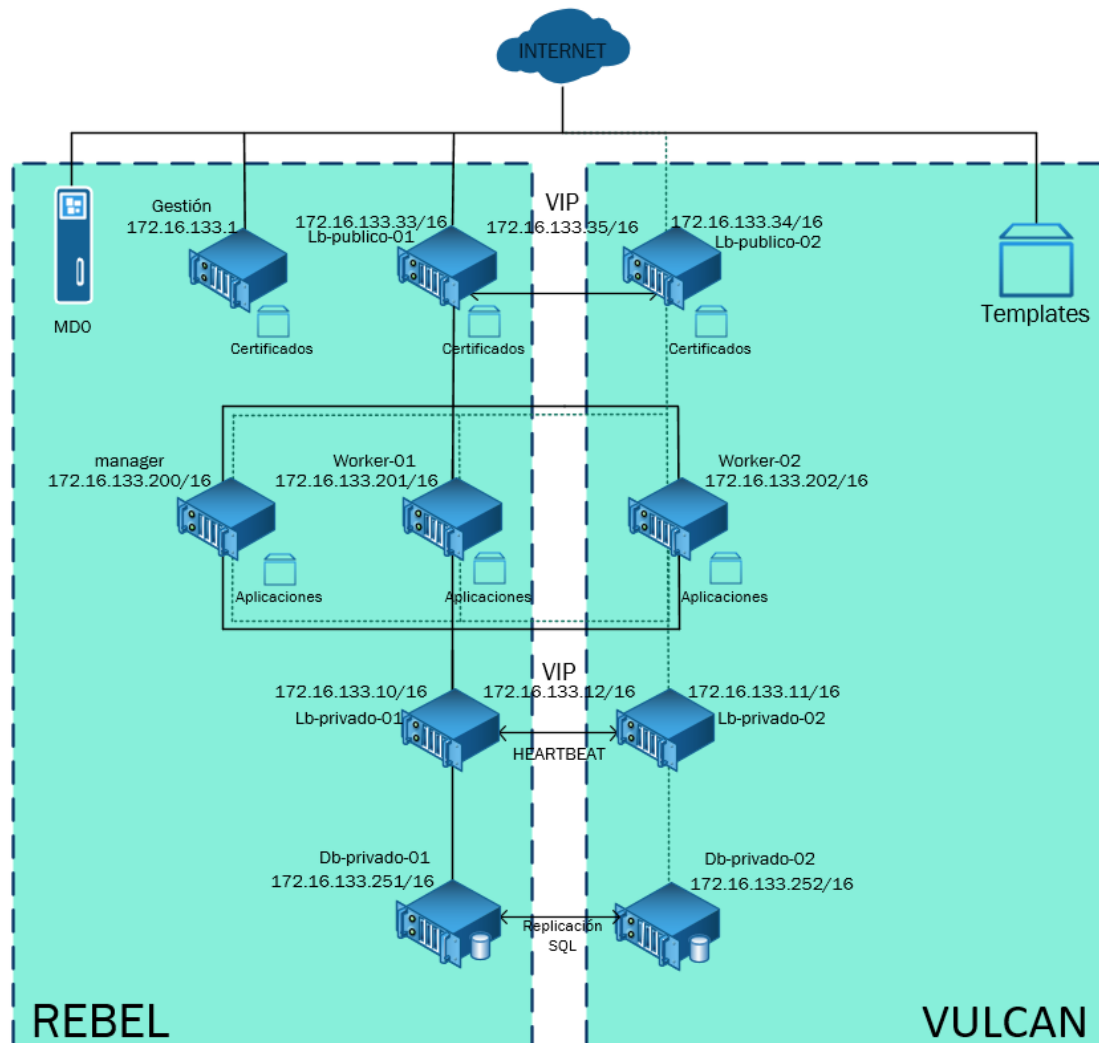
Para poder conseguir una alta disponibilidad, será necesario dividir en capas las distintas funciones de la infraestructura. Estas capas serán las siguientes:

- **Capa de aplicación:** será la encargada de servir los servicios reales que ofrecerá la infraestructura, tales como un servicio de ficheros en la nube (owncloud) o de VoIP (teamspeak).
- **Capa de almacenamiento:** en el caso de que las aplicaciones tengan que guardar ficheros de forma permanente, se utilizará esta capa.
- **Capa de datos:** será usada para guardar todos los datos.



### 3.1. Diagrama lógico

A continuación, se contempla el diagrama que muestra cómo está organizada la **infraestructura a nivel lógico**:



## 3.2. Tecnologías empleadas

Para poder llevar a cabo nuestra infraestructura, necesitaremos una serie de tecnologías que nos permitan realizar con éxito nuestro objetivo.

El primer nodo será “Rebel”, que dispondrá de 3 discos duros. Uno de ellos contendrá el sistema operativo y los restantes una **partición RAID 1** configurada por software.

El segundo nodo, “Vulcan”, dispondrá de un disco duro. También tendrá una **partición montada por red** para el uso de las imágenes ISO y las plantillas de los contenedores.

Nuestros contenedores dispondrán de un sistema operativo **Debian GNU/Linux**.

Dado que buscamos tener balanceo de carga y alta disponibilidad, usaremos la tecnología **HAProxy** que nos permitirá distribuir la carga de peticiones en varios servidores y, **Heartbeat**, que nos permitirá mantener una alta disponibilidad mediante el uso de una **VIP**, concepto que se explicará [más adelante](#).

Dispondremos de un cluster de **Docker swarm**, que será el encargado de servir los servicios de nuestra infraestructura. Para ello, se utilizará un cluster con 3 nodos, siendo el contenedor “**manager**” el nodo manager y los otros 2 nodos siendo los “**workers**”.

Para el almacenamiento de nuestras aplicaciones, se utilizará el servicio de **NFS** contra una partición del disco duro del nodo de “Rebel” que se encuentra en RAID1, mencionado con anterioridad.

Por último, para el almacenaje de los datos usaremos el sistema de Gestión de base de datos relacional, denominado MySQL, que ofrecerá una **alta disponibilidad** dado que se realizará una **replicación master-slave** a través de **log binario**, que será explicada [más adelante](#).



## 4. Organización y Gestion

Dado que es un proyecto en el que se deben realizar **múltiples tareas distintas**, es necesario contar con una serie de **herramientas y procesos para poder sistematizar el trabajo**.

### 4.1. Repositorio

Para poder organizar los recursos del proyecto, vamos a utilizar un **repositorio en la plataforma de Bitbucket de Atlassian**. Esto nos será útil también para mantener un control de versiones con Git.

### 4.2. Git

Para que nuestra manera de trabajar sea mucho más óptima, vamos a utilizar un **sistema de control de versiones**, en concreto, **Git**.

Esto nos va a permitir ver los cambios que ha realizado cualquiera en cada versión o incluso volver a una versión específica en caso de ser necesario.

Al tener el repositorio en Bitbucket (herramienta de Atlassian), **los mensajes de los commits se relacionan directamente con los IDs de las tareas** que tenemos en nuestro tablero de Kanban.

### 4.3. Tablero de Kanban

Algo imprescindible en la organización del proyecto es un **gestor de tareas**. Concretamente, Kanban nos permite ver las versiones (commits) en el repositorio, que hacen referencia en cada tarea.

Esto es útil para cuando se desea ver la evolución de cada tarea en concreto.

### 4.4. Máquina “Gestion”

Dentro de nuestra infraestructura, tenemos una máquina llamada “Gestion”, que es desde donde trabajamos en el proyecto.

Cada miembro del equipo tiene su propio usuario, pudiendo acceder a la máquina únicamente conectándose a la **VPN de la red**.

[Más adelante](#), se explicarán en detalle las configuraciones de esta máquina.

### 4.5. Ansible

Ansible es una herramienta que nos permite gestionar configuraciones, aprovisionamiento de recursos, **despliegue automático de aplicaciones**, etc.

En nuestro repositorio se encuentran los **distintos roles** con las distintas configuraciones para desplegar sobre la plataforma. Esta herramienta es especialmente útil para la automatización de las configuraciones.

[Más adelante](#), se explica en detalle todo lo relacionado con Ansible en el proyecto, incluida una pequeña explicación de la herramienta.





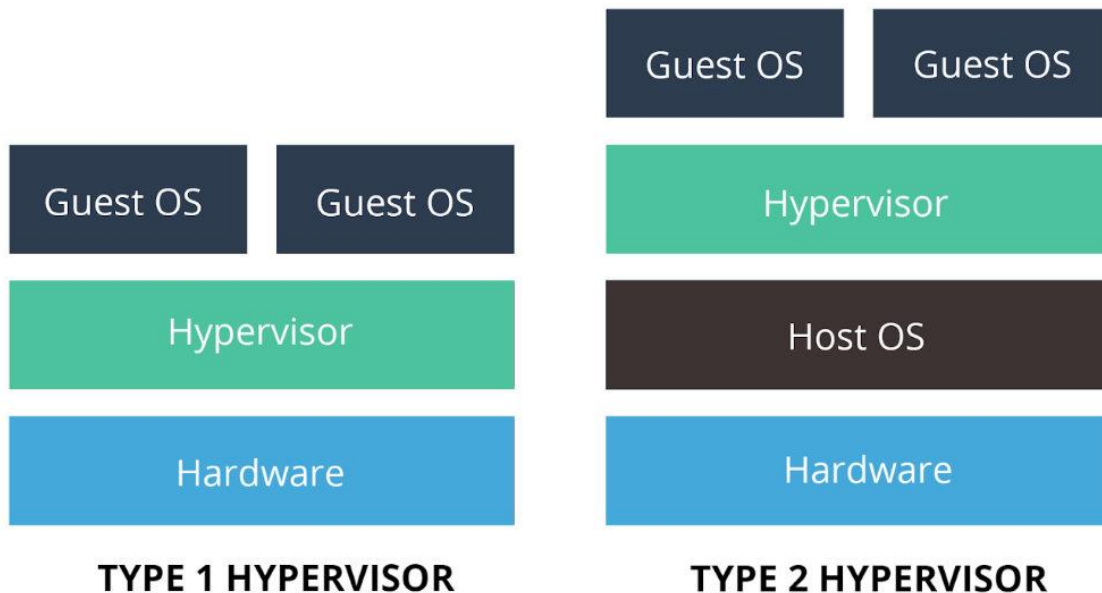
## 5. Desarrollo

### 5.1. Entorno de virtualización (Proxmox)

La virtualización consiste en una capa de software, llamada hipervisor, que, dentro de un sistema anfitrión, proporciona una capa de abstracción con el hardware. Ésta es utilizada para instalar un sistema operativo, consiguiendo que éste vea las mismas características de hardware que el sistema anfitrión.

Dentro de los hipervisores, encontramos 2 tipos:

- **Hipervisor tipo 1:** llamados hipervisores simples o **bare metal**, consisten en una capa de software instalada directamente sobre el servidor físico, sin ningún software ni sistema operativo en medio. Esto da un mayor rendimiento y una estabilidad mayor que los hipervisores de tipo 2.
- **Hipervisor tipo 2:** llamados hipervisores alojados, estos hipervisores se alojan en una capa de software por encima del sistema anfitrión.



Hay una gran variedad de entornos de virtualización dentro de los distintos tipos de hipervisores. En nuestro caso vamos a usar **Proxmox VE**; un hipervisor de tipo 1, con un panel de control web cómodo y sencillo.



### 5.1.1 Instalación

Deberemos usar la iso proporcionada por Proxmox

Instalaremos Proxmox VE

Proxmox VE 6.4 (iso release 1) - <https://www.proxmox.com/>



Welcome to Proxmox Virtual Environment

Install Proxmox VE

Install Proxmox VE (Debug mode)

Rescue Boot

Test memory (Legacy BIOS)

Deberemos leer y aceptar la licencia de uso de Proxmox



Proxmox VE Installer

#### END USER LICENSE AGREEMENT (EULA)

4. Intellectual Property Rights. The Programs and each components are owned by Proxmox and other licensors and are protected under copyright law and under other laws as applicable. The "Proxmox" trademark and the Proxmox company logo are registered trademarks of Proxmox in Austria and other countries. This EULA does not permit you to distribute the Programs or their components using Proxmox's trademarks, regardless of whether the copy has been modified. Title to the Programs and any component, or to any copy, modification, or merged portion shall remain with Proxmox and other licensors, subject to the applicable license.

5. Third Party Software. Proxmox may distribute third party software with the Programs. These third party programs are provided as a convenience to you, and are subject to their own license terms. If you do not agree to the applicable license terms for the third party software programs, then you may not install them.

6. Export Regulation. You warrant that you understand that the Programs and their components may be subject to export controls under the Austrian Export Administration Regulations.

7. Other terms. If any provision of this EULA is held to be unenforceable, the enforceability of the remaining provisions shall not be affected. Any claim, controversy or dispute arising under or relating to this EULA shall be governed by the laws of Austria (Europe), without regard to any conflict of laws provisions.

Copyright © 2013-2019 Proxmox Server Solutions GmbH. All rights reserved. "Proxmox" and the Proxmox logo are registered trademarks of Proxmox Server Solutions GmbH. "Linux" is a registered trademark of Linus Torvalds. All other trademarks are the property of their respective owners.

Abort

Previous

I agree



JOSE ÁNGEL AEL TALAVERA



Deberemos elegir dónde queremos instalar nuestro Proxmox. Se encargará de particionarnos el disco, instalar los paquetes requeridos, etc.

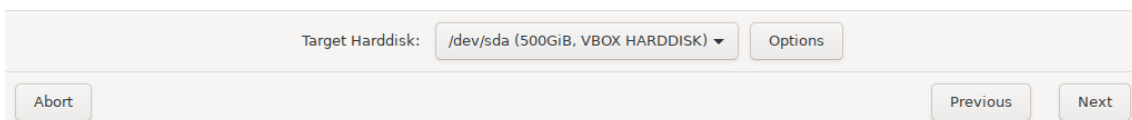


### Proxmox Virtual Environment (PVE)

**The Proxmox Installer** automatically partitions your hard disk. It installs all required packages and makes the system bootable from the hard disk. All existing partitions and data will be lost.

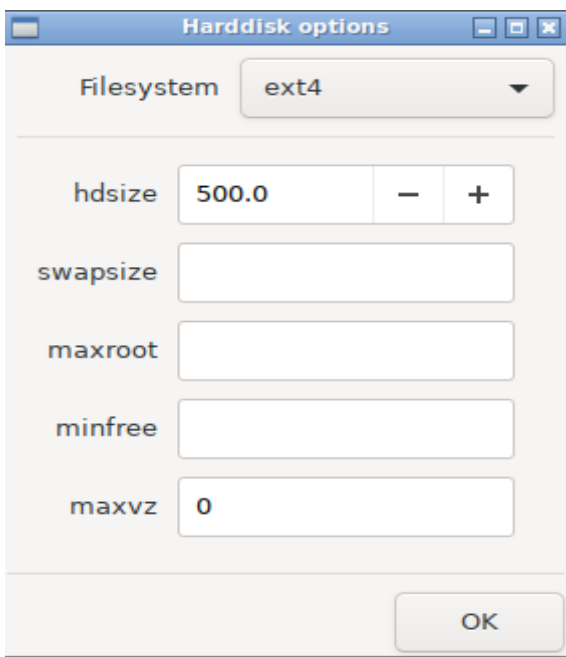
Press the Next button to continue the installation.

- **Please verify the installation target**  
The displayed hard disk will be used for the installation.  
Warning: All existing partitions and data will be lost.
- **Automatic hardware detection**  
The installer automatically configures your hardware.
- **Graphical user interface**  
Final configuration will be done on the graphical user interface, via a web browser.

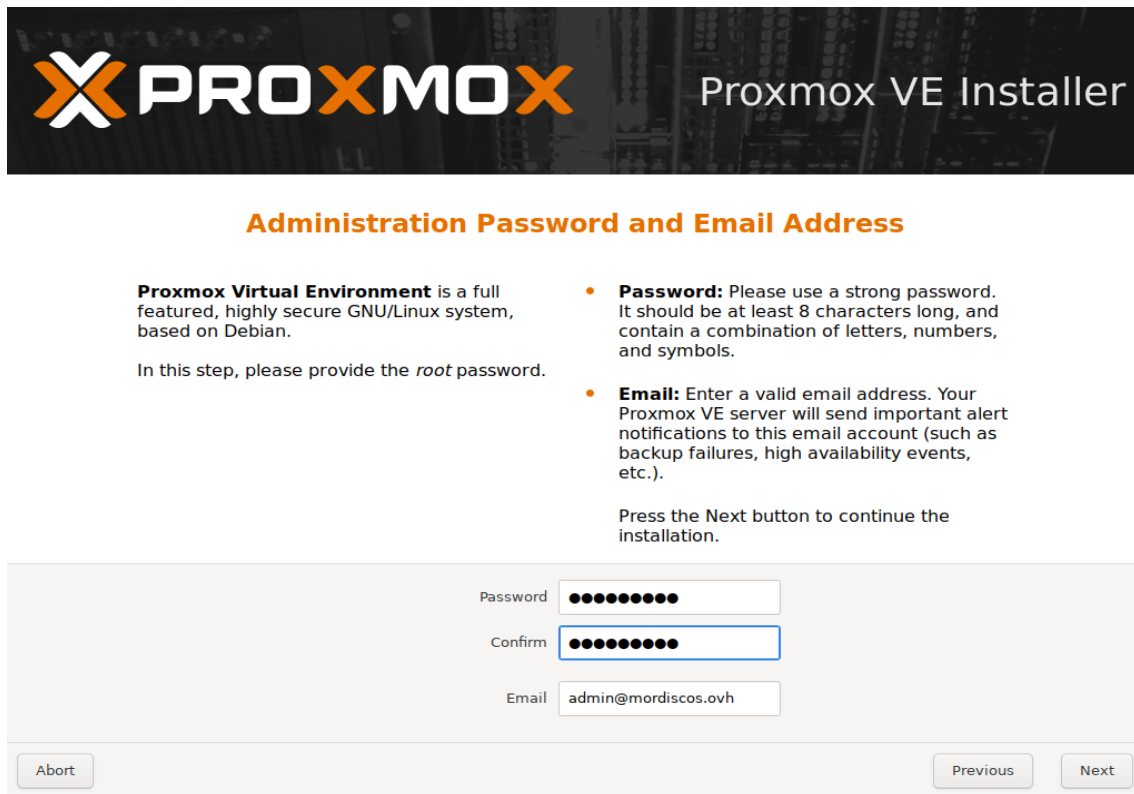


Para poder configurarlo de forma manual, podremos ir al menú de options e indicarle el tamaño del disco, el tamaño de swap, etc.

Deberemos indicar que el tamaño de maxvz sea 0.



Una vez configurado esto, en la siguiente ventana deberemos elegir una contraseña y email para la administración de Proxmox.



**Administration Password and Email Address**

**Proxmox Virtual Environment** is a full featured, highly secure GNU/Linux system, based on Debian.

In this step, please provide the *root* password.

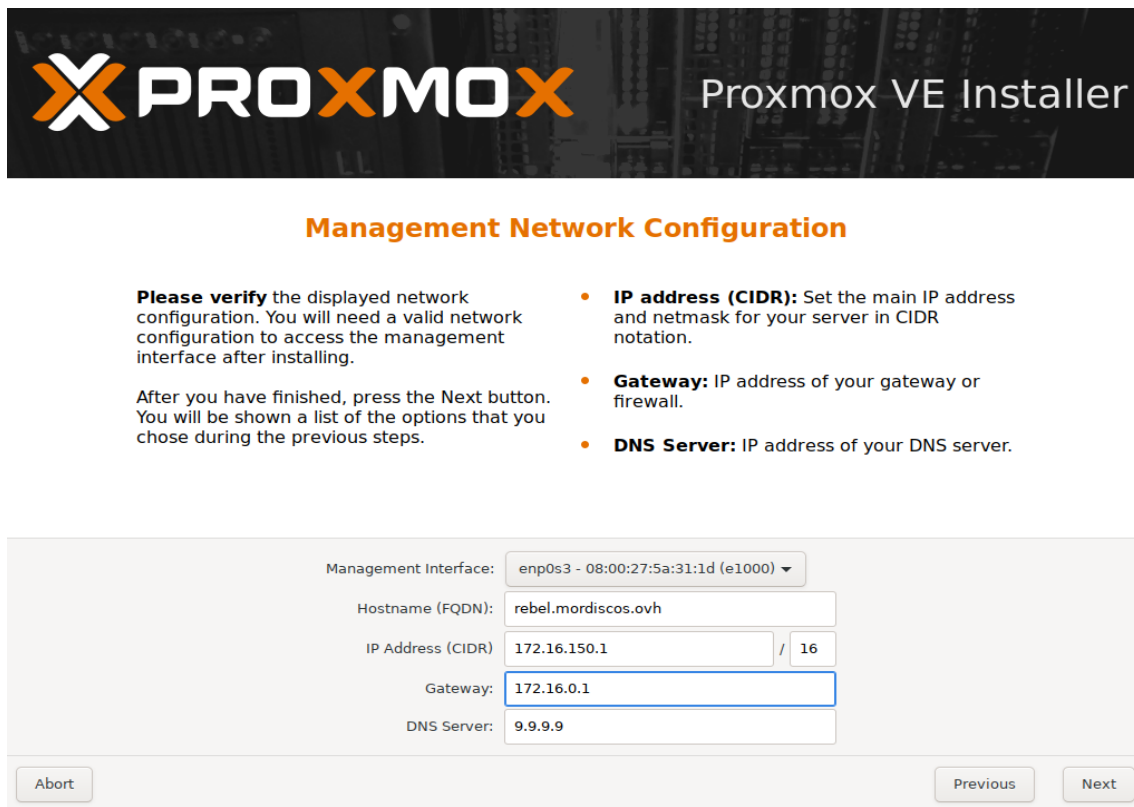
- Password:** Please use a strong password. It should be at least 8 characters long, and contain a combination of letters, numbers, and symbols.
- Email:** Enter a valid email address. Your Proxmox VE server will send important alert notifications to this email account (such as backup failures, high availability events, etc.).

Press the Next button to continue the installation.

Form fields:  
Password: [masked]  
Confirm: [masked]  
Email: admin@mordiscos.ovh

Buttons: Abort, Previous, Next

Ahora deberemos configurar la network a nuestro gusto. Esta es nuestra configuración.



**Management Network Configuration**

**Please verify** the displayed network configuration. You will need a valid network configuration to access the management interface after installing.

After you have finished, press the Next button. You will be shown a list of the options that you chose during the previous steps.

- IP address (CIDR):** Set the main IP address and netmask for your server in CIDR notation.
- Gateway:** IP address of your gateway or firewall.
- DNS Server:** IP address of your DNS server.

Form fields:  
Management Interface: enp0s3 - 08:00:27:5a:31:1d (e1000)  
Hostname (FQDN): rebel.mordiscos.ovh  
IP Address (CIDR): 172.16.150.1 / 16  
Gateway: 172.16.0.1  
DNS Server: 9.9.9.9

Buttons: Abort, Previous, Next



Aquí nos muestra un “resumen” de nuestra configuración, deberemos revisarla y comprobar que esta todo correcto.

Ya podremos comenzar la instalación.



### Summary

**Please confirm** the displayed information. Once you press the **Install** button, the installer will begin to partition your drive(s) and extract the required files.

Option	Value
Filesystem:	ext4
Disk(s):	/dev/sda
Country:	Spain
Timezone:	Europe/Madrid
Keymap:	es
Email:	admin@mordiscos.ovh
Management Interface:	enp0s3
Hostname:	rebel
IP CIDR:	172.16.150.1/16
Gateway:	172.16.0.1
DNS:	9.9.9.9

☒ Automatically reboot after successful installation

```

-----
Welcome to the Proxmox Virtual Environment. Please use your web browser to
configure this server - connect to:

  https://172.16.150.1:8006/

-----

Hint: Num Lock on

rebel login: root
Password:
Linux rebel 5.4.106-1-pve #1 SMP PVE 5.4.106-1 (Fri, 19 Mar 2021 11:08:47 +0100) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@rebel:~# _

```

Dado que queremos realizar un cluster necesitaremos otro Proxmox para poder realizar la unión de ambos.

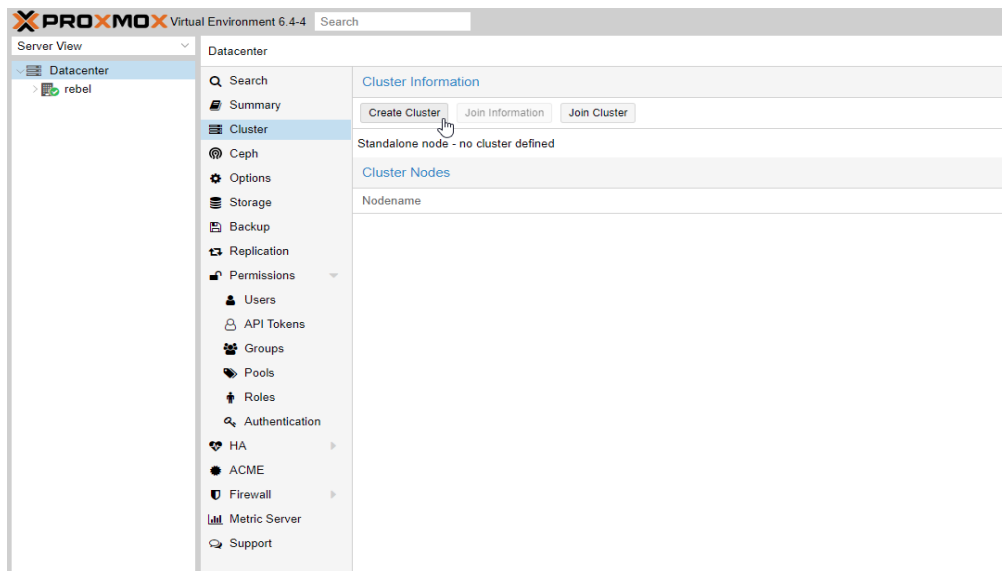
Para ello realizaremos la misma configuración anteriormente especificada cambiándole el nombre y asegurándonos que la dirección es distinta.

### 5.1.2 Creación del cluster

Para la centralización de los equipos se utiliza la tecnología llamada “cluster”, qué, de forma muy simple, nos permite agrupar los equipos para que colaboren entre ellos. De esta forma, conseguimos centralizar nuestros dos equipos, ahora llamados nodos.



Para esto, en cualquiera de nuestros nodos, deberemos ir a la sección de la creación de cluster.



Elegiremos el nombre del cluster y la ip:

Task viewer: Create Cluster

Output

Status

Stop

Corosync Cluster Engine Authentication key generator.  
Gathering 2048 bits for key from /dev/urandom.  
Writing corosync key to /etc/corosync/authkey.  
Writing corosync config to /etc/pve/corosync.conf  
Restart corosync and cluster filesystem  
TASK OK

Cluster Information

Create Cluster

Join Information

Join Cluster

Cluster Name: **mordiscos**

Config Version: **1**

Cluster Nodes

Nodename	ID ↑	Votes	Link 0
<b>rebel</b>	<b>1</b>	<b>1</b>	<b>172.16.150.1</b>

Cluster Join Information

Copy the Join Information here and use it on the node you want to add.

IP Address:

172.16.150.1

Fingerprint:

A2:CA:5C:4B:6B:BD:4A:82:84:7F:DD:34:7D:CC:D9:80:27:D1:C4:29:76:E5:22:6D:55:66:BE:6F:1F:1E:24:83

Join Information:

eyJpcEFkZHZlc3MiOiXNzluMTYuMTUwLjElClJmaWV5nZXJwcmludCI6IkEyOkNB0jVD0JRCojZCokJE0jRB  
OjgyOjg0OjdGOkREOjM0OjdEOkNDOKQ5OjgwOjl3OkQxOkM0OjI5Ojc2OkU1OjlyOjZE0jU1OjY2OkJFOjZG  
OjFGOjFFOjI0OjgzIiwicGVickxpbnRlbnRlbiAiOiXNzluMTYuMTUwLjElcSwicmluZS19hZGRyblpbjE3Mi4xNTA  
uMSldl.C.I0h3RIhSI6ev.lnhp.RlcmZhY2JLI0nsiMCi6ev.IsaW5rdnVtYmVudlinoMCiJ9fSwidmVyc2lybGlilil.C.iJh25

Copy Information

The screenshot shows the Proxmox VE 6.4-4 web interface. The left sidebar has a 'Server View' dropdown and a 'Datacenter' section with a 'vulcan' node. The 'Cluster' menu item is selected. The main content area is titled 'Cluster Information' and contains three buttons: 'Create Cluster', 'Join Information', and 'Join Cluster'. Below the buttons, it says 'Standalone node - no cluster defined' and 'Cluster Nodes'. At the bottom, there is a table with columns 'Nodename', 'ID ↑', and 'Votes'.





Una vez configurado esto podremos darle a Join “mordiscos”.

### Cluster Join

☒ Assisted join: Paste encoded cluster join information and enter password.

Information: `eyJpcEFkZHJlc3MiOiIxNzluMTYuMTUwLjEiLCJmaW5nZXJwcmVudCI6IkkEYOkNBOjVDOjRCojZCokJEOjRBOjgyOjg0OjJdGOKREOjM0OjdEOkNDOKQ5OjgwOjI3OkQxOkM0OjI5Ojc2OkU1OjlyOjZEOjU1OjY2OkJFOjZGOjFGOjFFOjI0OjgzIiwicGVlcXpbmtzljp7IjAiOiIxNzluMTYuMTUwLjEiSwicmluZ19hZGRyIjpbIjE3Mi4xNi4xNTAuMSJdLCJ0b3RlbiSl6evJpbmRlcmZhY2UiOnsiMCI6evJsaW5rbnVtYmVvIioiMCJ9fSwidmVvc2lvbil6IiilLCJib25maVWd`

Peer Address:  Password:

Fingerprint:

Cluster Network: Link: 0  peer's link address: 172.16.150.1

[Help](#) [Join 'mordiscos'](#)

Ya tendremos nuestro cluster de Proxmox con dos nodos finalizado.

### PROXMOX Virtual Environment 6.4-4

Server View

Datacenter (mordiscos)

- rebel
- vulcan

Cluster Information

Create Cluster Join Information Join Cluster

Cluster Name: mordiscos Config Version: 2 Number of Nodes: 2

Cluster Nodes

Nodename	ID	Votes	Link 0
rebel	1	1	172.16.150.1
vulcan	2	1	172.16.150.2

Tasks Cluster log

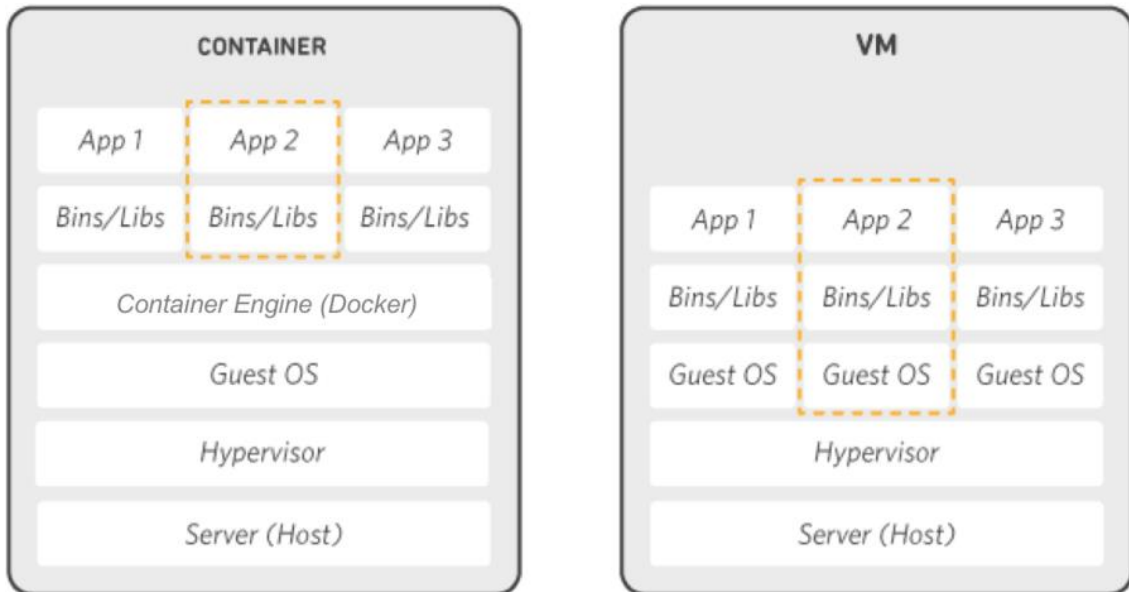
Start Time	End Time	Node	User name	Description	Status
Jun 09 17:57:45	Jun 09 17:57:50	vulcan	root@pam	Join Cluster	OK
Jun 09 17:51:23	Jun 09 17:51:25	rebel	root@pam	Create Cluster	OK
Jun 09 17:25:16	Jun 09 17:25:16	vulcan	root@pam	Start all VMs and Containers	OK





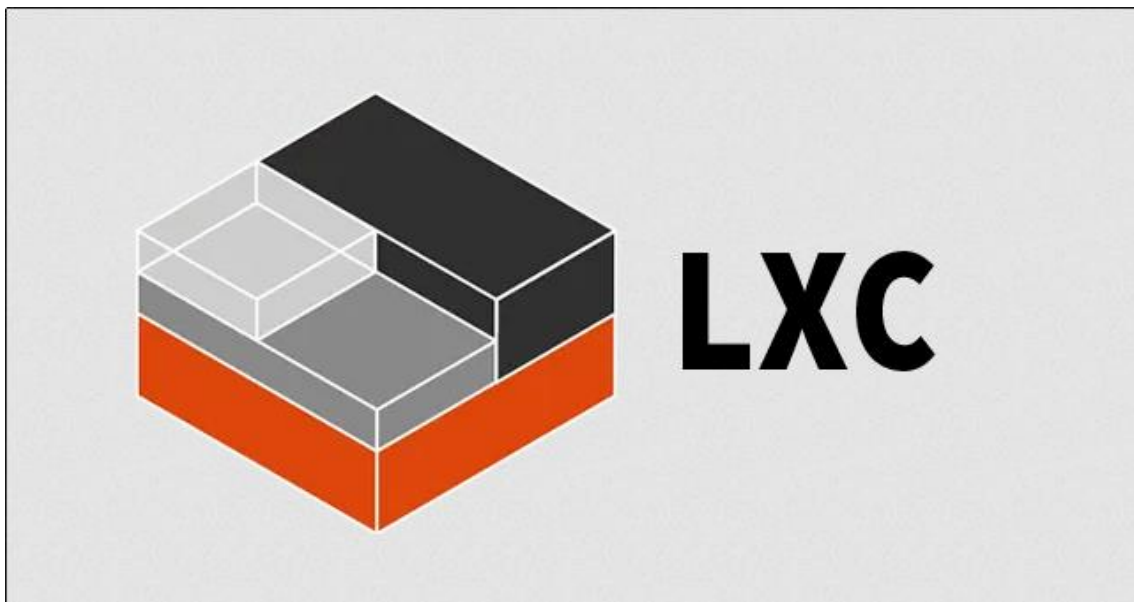
### 5.1.3 Contenedores

Los contenedores, a diferencia de las máquinas virtuales, permiten a las aplicaciones que se ejecuten e implementen como una instancia en el propio sistema operativo, sin llegar a ejecutar un sistema operativo en la capa de abstracción:



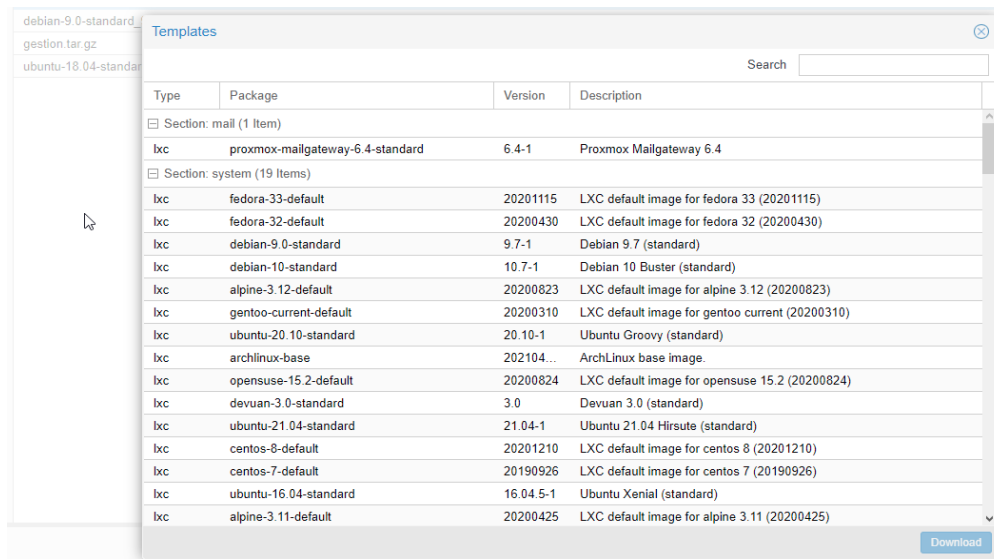
Los contenedores, por tanto, proporcionan una infraestructura ligera de empaquetado e implementación de aplicaciones o servicios.

En el caso de Proxmox, los contenedores son LXC (Linux Containers). Estos usan una tecnología de virtualización a nivel de sistema operativo para Linux, permitiendo que un servidor físico ejecute múltiples instancias de espacios aislados, conocidos como servidores privados virtuales o entorno virtuales.

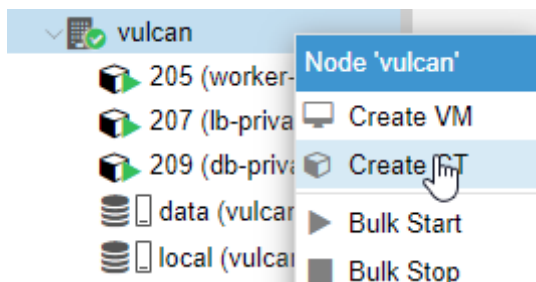


### 5.1.3.1. Creación de un contenedor

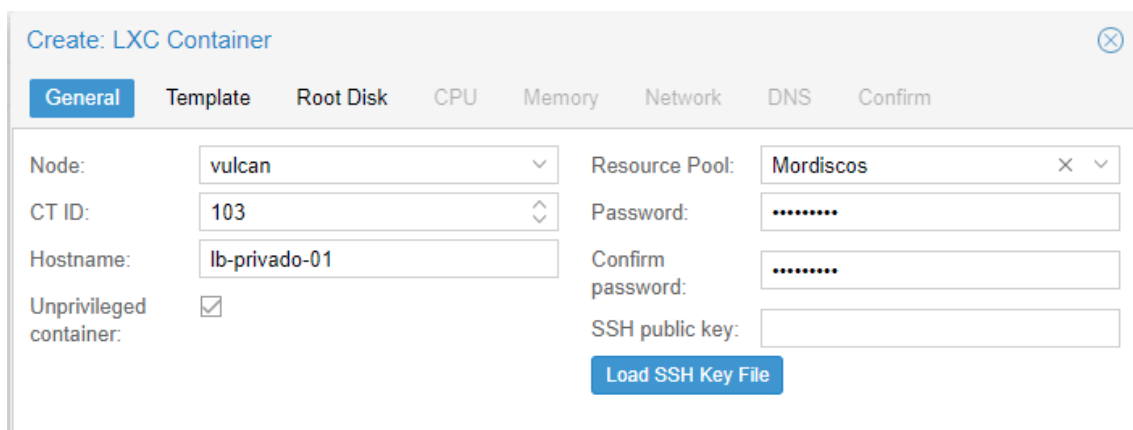
Para la creación de nuestros contenedores, usaremos templates. Nosotros usaremos una template con Debian 9 para las máquinas. No obstante, existen múltiples templates por defecto:



Ahora crearemos el container en uno de los dos nodos que disponemos, usaremos “Vulcan”:



En la sección General, deberemos elegir un nombre, una pool de recursos y una contraseña:



Create: LXC Container

General | Template | Root Disk | CPU | Memory | Network | DNS | Confirm

Node: vulcan

CT ID: 103

Hostname: lb-privado-01

Unprivileged container: ☒

Resource Pool: Mordiscos

Password: .....

Confirm password: .....

SSH public key:

Load SSH Key File

En template, tendremos que indicar donde esta almacenada nuestra template y cuál será la que usaremos para nuestro contenedor.



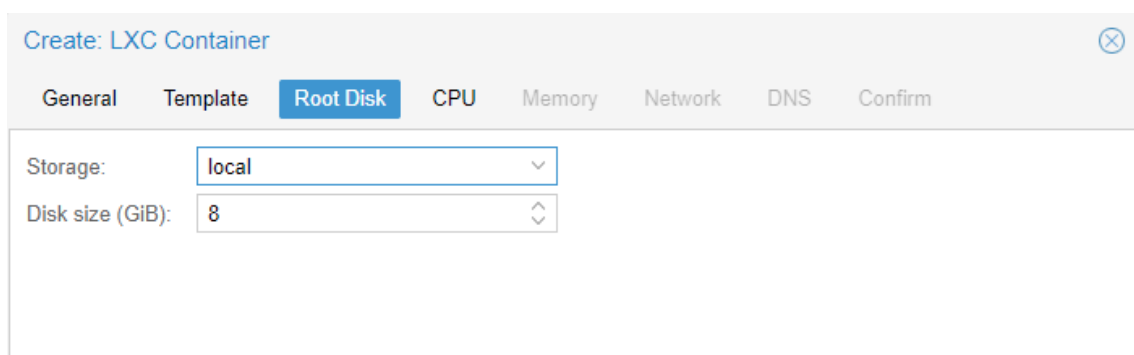
Create: LXC Container

General Template Root Disk CPU Memory Network DNS Confirm

Storage: data

Template: debian-10-standard\_10.7-1\_amc

Aquí indicaremos el root disk y que espacio tendrá.



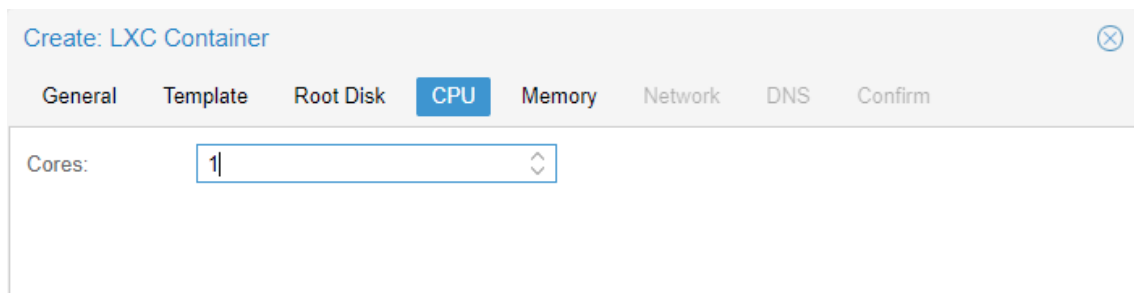
Create: LXC Container

General Template Root Disk CPU Memory Network DNS Confirm

Storage: local

Disk size (GiB): 8

El número de núcleos usados de la CPU.

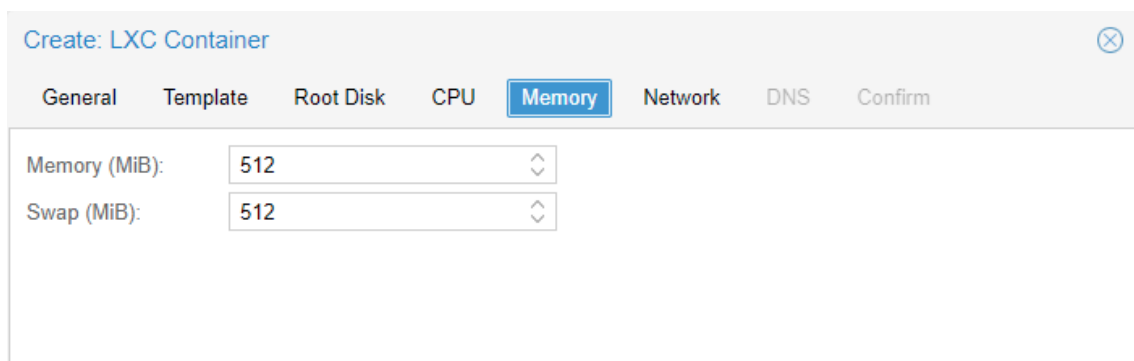


Create: LXC Container

General Template Root Disk CPU Memory Network DNS Confirm

Cores: 1

Especificaremos la memoria y la swap.



Create: LXC Container

General Template Root Disk CPU Memory Network DNS Confirm

Memory (MiB): 512

Swap (MiB): 512

En esta sección deberemos indicar la configuración de la red, el adaptador, el bridge, la ip, etc.



Por último, la configuración de los servidores DNS que nos ofrecerán servicio.

Una vez confirmado, en la siguiente tarea observaremos el proceso de creación del container.

Finalizado el proceso de la creación del container, ya podremos acceder a él.

**JOSE ÁNGEL AEL TALAYERA**



## 5.2. Ansible

### 5.2.1 ¿Qué es Ansible?

Como mencionamos con anterioridad, Ansible es una herramienta de orquestación. Nos permite gestionar servidores, configuraciones y aplicaciones de una forma sencilla, robusta y paralela de forma automatizada.

Con la automatización que ofrece Ansible, se pueden instalar sistemas, automatizar las tareas diarias, preparar la infraestructura, mejorar la seguridad, etc.

### 5.2.2 ¿Cómo funciona Ansible?

Ansible se basa en que, a través de un controlador, en nuestro caso la máquina “Gestion”, se conecta por SSH a los nodos y les inserta pequeños programas llamados módulos, los cuales permiten llevar a cabo tareas de automatización en la plataforma.

Para el correcto funcionamiento de Ansible, únicamente requiere que los servidores remotos dispongan de Python.

Ansible utiliza como lenguaje YAML, este sirve para describir las acciones a realizar y la configuración que se deben propagar a los diferentes nodos. Se utiliza este lenguaje dado que es mucho más descriptivo, sencillo de entender y limpio.

### 5.2.3 ¿Cómo trabaja Ansible?

#### 5.2.3.1. Inventario

Para que Ansible pueda saber a qué máquina tiene que aprovisionar, usaremos un inventario, en el que se encontrarán todos nuestros hosts listados en él. Estos hosts los podemos agrupar por grupos y patrones. Este archivo se denomina como hosts y viene por defecto una vez instalado Ansible.

#### 5.2.3.2. Tareas

La forma básica para interactuar con nuestros servidores es mediante tareas (tasks), que realizará Ansible a la hora del aprovisionamiento asegurándose de que el servidor quedará tal y como le hemos indicado en la tarea.

#### 5.2.3.3. Despliegue

Ansible se puede utilizar en modo Ad-Hoc o mediante un playbook. El modo Ad-Hoc nos permite ejecutar directamente comandos en una sola línea sobre nuestros hosts utilizando los módulos que ya tiene Ansible. Este modo se utiliza cuando queremos efectuar una acción simple sobre nuestros hosts, como hacer un reinicio, un ping, etc.



Para acciones más complejas, usaremos los playbooks. Estos están en formato YAML, y pueden estar en un solo fichero o siguiendo una estructura.

Aquí podremos ver un ejemplo de un playbook con una estructura simple en el mismo fichero.

```
1 ---
2
3 - hosts: webservers
4   vars:
5     http_port: 80
6     max_clients: 200
7     remote_user: root
8   tasks:
9     - name: ensure apache is at the latest version
10       yum: pkg=httpd
11         state=latest
12     - name: write the apache config file
13       template:
14         src=/srv/httpd.j2
15         dest=/etc/httpd.conf
16       notify:
17         - restart apache
18     - name: ensure apache is running service:
19       name=httpd state=started
```

Como hemos podido observar, tenemos toda la configuración en un mismo fichero. Este tipo de configuración está bien si lo utilizamos sólo para un único despliegue, o una configuración simple. Sin embargo, en escenarios más complejos, es mejor utilizar roles, donde podremos moldear más las configuraciones a nuestro gusto.

Este sería un ejemplo utilizando un rol:

```
- name: Configurando servidor NGINX
  include_role:
    name: nginx
```

## 5.2.4 Roles

A medida que vamos añadiendo funcionalidad y complejidad a nuestros playbooks, cada vez es mucho más difícil manejarlos en un solo fichero. Los roles nos permiten crear un playbook, con una mínima configuración y definir toda la complejidad de las acciones a bajo nivel.

Para una correcta utilización de los roles, será necesario crear una estructura de carpetas y subcarpetas donde incluiremos nuestra configuración. Para la creación de la estructura, tenemos dos opciones: de forma manual o utilizando Ansible Galaxy.

Ansible Galaxy es un sitio para la búsqueda, reutilización e intercambio de roles desarrollados por la comunidad.



Este sería un ejemplo de la estructura de un rol:

```
ruben@gestion:~/Mordiscos/ansible/roles/ntp$ tree
.
├── LICENSE
├── README.md
├── defaults
│   └── main.yml
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   └── main.yml
├── templates
│   ├── chrony.conf.j2
│   ├── clock.j2
│   └── ntp.conf.j2
└── vars
    ├── Archlinux.yml
    ├── Debian.yml
    ├── FreeBSD.yml
    ├── RedHat.yml
    └── Suse.yml
```

- **Defaults:** este directorio contendrá un fichero llamado main.yml en el cual se le indicará la información de las variables globales utilizadas por este rol.

```
---
ntp_enabled: true
ntp_timezone: Etc/UTC

# ntp_daemon: [various]
# ntp_package: ntp
# ntp_config_file: /etc/ntp.conf
# ntp_driftfile: [various]

ntp_manage_config: false
```

- **Files:** este directorio contendrá los archivos necesarios para realizar un despliegue. Los ficheros de este directorio no se podrán modificar y serán copiados así, a los hosts remotos, como, por ejemplo, el código fuente de una aplicación.



- **Templates:** estos son parecidos a los files, con la diferencia de que los templates sí admiten modificaciones. Podemos pasar variables de configuración al template para que éste lo aplique a los hosts. Ansible admite variables en los playbooks y templates utilizando Jinja2.

```

{{ ansible_managed | comment }}

# For more information about this file, see the man pages
# ntp.conf(5), ntp_acc(5), ntp_auth(5), ntp_clock(5), ntp_misc(5), ntp_mon(5).

driftfile {{ ntp_driftfile }}

# Enable this if you want statistics to be logged.
#statsdir /var/log/ntpstats/

statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

{% if ntp_tinker_panic is sameas true %}
# Always reset the clock, even if the new time is more than 1000s away
# from the current system time. Usefull for VMs that can be paused
# and much later resumed.
tinker panic 0
{% endif %}

```

- **Tasks:** las tareas se ejecutan en orden contra todos los hosts que cumplan el patrón definido. Las tasks pueden estar definidas en el fichero main.yml o en cualquier otro fichero .yml siempre y cuando se haga referencia a ellos desde el main.yml

```

--
- name: Include OS-specific variables.
  include_vars: "{{ ansible_os_family }}.yml"

- name: Set the ntp_driftfile variable.
  set_fact:
    ntp_driftfile: "{{ __ntp_driftfile }}"
  when: ntp_driftfile is not defined

- name: Set the ntp_package variable.
  set_fact:
    ntp_package: "{{ __ntp_package }}"
  when: ntp_package is not defined

- name: Set the ntp_config_file variable.
  set_fact:
    ntp_config_file: "{{ __ntp_config_file }}"
  when: ntp_config_file is not defined

- name: Set the ntp_daemon variable.
  set_fact:
    ntp_daemon: "{{ __ntp_daemon }}"
  when: ntp_daemon is not defined

```



- **Meta:** estos archivos describen las dependencias del rol.

```
---
dependencies: []
```

- **Vars:** tienen la misma función de defaults y se usa para almacenar variables. Estas variables tendrán más prioridad que las establecidas en default. También pueden ser llamadas desde las tasks, templates, etc.

```
---
ntp_daemon: ntp
ntp_timezone_package: tzdata
ntp_package: ntp
ntp_config_file: /etc/ntp.conf
ntp_driftfile: /var/lib/ntp/drift
ntp_cron_daemon: cron
~
~
```

- **Handlers:** son acciones que se dispararán al final de cada bloque de tareas en los playbooks, se utilizará la etiqueta “notify” para llamarlos. Sólo se activarán una vez, como, por ejemplo, reiniciar el servicio de Apache.

```
---
- name: restart ntp
  service:
    name: "{{ ntp_daemon }}"
    state: restarted
  when: ntp_enabled | bool
```

Aquí vemos como llama al handler en la task.

```
- name: Generate ntp configuration file.
  template:
    src: "{{ ntp_config_file | basename }}.j2"
    dest: "{{ ntp_config_file }}"
    mode: 0644
  notify: restart ntp
  when: ntp_manage_config | bool
```

### 5.2.5 Seguridad con Ansible

Para un mayor nivel de seguridad y poder proteger los datos de carácter sensible, utilizamos **Ansible-vault**, cifrando los ficheros o variables que contengan dicho tipo de datos.

```
password_mysql_replicationuser: !vault |
  $ANSIBLE_VAULT;1.1;AES256
  36333830623535303231376137663463653039323262323038633535623239653566353838343535
  3635636362353461633335666638383734356563653364360a6532636637383462396666135386538
```

Cabe destacar que toda nuestra infraestructura se realiza a través de despliegues con Ansible.



### 5.3. Máquina “Gestion”

Desde esta máquina, se realizan los despliegues de Ansible, se da servicio DNS y se gestionan los certificados de forma centralizada con certbot.

Para poder acceder a la máquina, es necesario que se acceda primero a la VPN con las credenciales del propio usuario.

El inventario de Ansible, con todos los hosts, está preparado para utilizar la configuración de conexión de un fichero concreto:

```
[all:vars]
ansible_ssh_common_args='-F ssh_global.config'
```

En este fichero, se establecen configuraciones para cada host, como, por ejemplo: qué IP es la que pertenece al host, a qué puerto conectarse, no verificar el host key, etc. Hay una opción concreta que establece una ruta para usar una clave RSA (común en todas las máquinas de la infraestructura):

```
Host db-privado-02
  User root
  Port 22
  HostName 172.16.133.252
  IdentityFile ~/.auth/id_rsa
  PasswordAuthentication no
  StrictHostKeyChecking no
```

El directorio al que apunta esta opción, se encuentra en todos los directorios home de los usuarios del proyecto:

```
jose@gestion:~$ ls -la | grep ".auth$"
drw-rw---- 2 root root 4096 May 4 19:11 .auth
jose@gestion:~$
```

Como hemos mencionado antes, para mayor seguridad, se utiliza Ansible-vault cuando se tiene que usar información de carácter sensible.

La clave para cifrar las contraseñas con vault, se encuentra en el directorio home del usuario de Ansible:

```
root@gestion:~# ls -l /home/.ansible/vault_password
-rw-rw---- 1 root root 43 May 6 20:21 /home/.ansible/vault_password
root@gestion:~#
```

Se puede observar en las capturas anteriores que el propietario de los directorios es “root” y sólo éste y su grupo tiene permisos sobre ellos.

Se ha otorgado privilegios de escalado, en los usuarios del proyecto, sobre los comandos específicos que tienen la necesidad de leer los ficheros que se encuentran en los directorios mencionados:

```
# User alias specification
User_Alias MORDISCOS = ruben, jose
# Cmd alias specification
Cmd_Alias ANSIBLE = /usr/bin/ansible-playbook, /usr/bin/ansible-vault

# User privilege specification
root          ALL=(ALL:ALL) ALL
MORDISCOS     ALL=(ALL) NOPASSWD: ANSIBLE
```

## 5.4. Capa de almacenamiento (Raid 1 y NFS)

Para poder mantener la centralización de los datos y su fiabilidad, vamos a utilizar 2 discos de 500 Gb para un volumen de datos en Raid 1. Esto es útil para guardar todos los datos persistentes que necesitemos redundados, tales como: logs, subida de ficheros desde las aplicaciones, repositorios, certificados, etc.

### 5.4.1 Creación del almacenamiento en raid 1

Para la creación de este, añadimos un nuevo volumen ZFS; le añadimos un nombre, el tipo de raid a utilizar y seleccionamos los discos:

Crear: ZFS

Nombre:  RAID Level:

Add Storage: ☒ Compresión:

ashift:

<input checked="" type="checkbox"/> Dispositivo	Modelo	Serial	Tamaño	Order
<input checked="" type="checkbox"/> /dev/sda	WDC_WD5000AAKX-08...	WD-WMC2E8726451	465.76 GiB	<input type="text" value="1"/>
<input checked="" type="checkbox"/> /dev/sdb	WDC_WD5000LPCX-24...	WD-WX51A251EN0F	465.76 GiB	<input type="text" value="2"/>

### 5.4.2 Sistema de archivos en red (NFS)

Hay veces que es necesario que un equipo tenga montada una partición que esté alojada en otro equipo para compartir una serie de directorios.

El protocolo NFS es un modelo cliente/servidor que permite esto, es decir, el cliente si está autorizado para ello, podrá “montar” dichos directorios del servidor en su propio sistema de archivos, pudiendo acceder a los archivos como si fueran locales.

Es por este motivo por el cual usaremos NFS para la Gestion centralizada de certificados, sesiones web, imágenes, etc.



### 5.4.2.1. Instalación

Primero deberemos actualizar los repositorios del sistema e instalar el servicio:

```
root@rebel:~# apt update
Hit:1 https://download.docker.com/linux/debian buster InRelease
Hit:2 http://security.debian.org buster/updates InRelease
Hit:3 http://ftp.es.debian.org/debian buster InRelease
Get:4 http://ftp.es.debian.org/debian buster-updates InRelease [51.9 kB]
Fetched 51.9 kB in 0s (132 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
8 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@rebel:~# apt install nfs-kernel-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
nfs-kernel-server is already the newest version (1:1.3.4-2.5+deb10u1).
0 upgraded, 0 newly installed, 0 to remove and 8 not upgraded.
```

### 5.4.2.2. Configuración

Como vamos a usar el directorio creado en el RAID1 (md0), no será necesario crearlo. En caso contrario se deberá crear el directorio.

Tendremos que darle los permisos adecuados. Como queremos que todos los clientes puedan acceder, eliminaremos los permisos restrictivos de la carpeta de exportación:

```
root@rebel:~# chown nobody:nogroup /md0/
root@rebel:~# ls -la | grep md0
drwxr-xr-x 11 nobody nogroup 11 May 23 11:23 md0
root@rebel:~#
```

Después de crear la carpeta y darle los permisos correctos, tendremos que editar el archivo /etc/exports y añadir las líneas indicando qué directorios queremos exportar, qué IP lo podrá montar y qué permisos tendrá:

```
# /etc/exports: the access control list for filesystems which may be exported
# to NFS clients. See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
#
/var/lib/vz/template/cache 172.16.150.2(rw,sync,no_subtree_check)
/md0/certificados 172.16.133.1(rw,sync,no_subtree_check) 172.16.133.32/30(ro,sync,no_subtree_check)
/md0/teamspeak 172.16.133.32/30(rw,sync,no_subtree_check) 172.16.145.145(rw,sync,no_subtree_check)
/md0/owncloud 172.16.133.32/30(rw,sync,no_subtree_check)
```



Existen una gran variedad de permisos, nosotros sólo usaremos los siguientes:

- **rw:** configura la exportación en modo lectura/escritura.
- **ro:** configura la exportación en modo lectura.
- **sync:** el servidor pone los datos en el disco de forma inmediata.
- **no\_subtree\_check:** permite que no se compruebe el camino hasta el directorio que se exporta, en el caso de que el usuario no tenga permisos sobre el directorio exportado.

Una vez configurados los directorios a compartir, deberemos actualizar la tabla de exportación y reiniciar el servicio nfs-kernel-server:

```
root@rebel:/# export -a
root@rebel:/# service nfs-kernel-server restart
```

Después de exportar y reiniciar el servicio deberemos ir a nuestro “cliente”, en este caso Vulcan, y montar la partición NFS en el directorio que queramos:

```
root@vulcan:~# mount 172.16.150.1:/md0/certificados /mnt/certificados/
```

Este montaje no es permanente, ya que al volver a iniciar el sistema este montaje habrá desaparecido.

Para hacerlo permanente deberemos añadirlo al /etc/fstab, de esta forma se montará de forma automática cada vez que se inicie el sistema:

```
# <file system> <mount point> <type> <options> <dump> <pass>
/dev/pve/root / ext4 errors=remount-ro 0 1
/dev/pve/swap none swap sw 0 0
proc /proc proc defaults 0 0
172.16.150.1:/var/lib/vz/template/cache /mnt/pve/data/template/cache nfs defaults 0 0
172.16.150.1:/md0/certificados /mnt/certificados/ nfs vers=4,rsize=8192,wsz=8192,timeo=14,intr 0 0
172.16.150.1:/var/lib/vz/template/cache /mnt/pve/data/template/cache/ nfs vers=4,rsize=8192,wsz=8192,timeo=14,intr 0 0
```



## 5.5. Sincronización horaria (NTP)

Tener la fecha y la hora mal configurada en nuestros equipos puede suponer un quebradero de cabeza, sobre todo a la hora de acceder a páginas webs.

Un ejemplo claro de este comportamiento podría ser el tener una fecha de sistema incorrecta, que llevase al sistema a invalidar un certificado por estar fuera de su plazo de vigencia, entre otras muchas cosas.

Por este motivo, para tener una buena sincronización entre las máquinas, es necesario que la hora y fecha sea síncrona en todos los equipos. Con esto evitaremos problemas como los mencionados anteriormente, ya sea de certificados, aplicaciones, etc.

Dentro de los múltiples servicios que ofrece el protocolo NTP, nosotros usaremos el propio servicio de NTP, que es el estándar de GNU/Linux. Este se encargará, a través de la red, de **sincronizar la hora y fechas de todos nuestros equipos**.



### 5.5.1 Instalación

Para realizar la configuración del servicio, primero deberemos instalarlo y habilitarlo:

```
root@gestion:~# apt install ntp
Reading package lists... Done
Building dependency tree
Reading state information... Done
ntp is already the newest version (1:4.2.8p12+dfsg-3ubuntu4.20.04.1).
0 upgraded, 0 newly installed, 0 to remove and 122 not upgraded.
```

```
root@gestion:~# systemctl enable ntp
Synchronizing state of ntp.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ntp
```

### 5.5.2 Configuración

Ahora deberemos establecer el servidor o servidores que queremos usar para la sincronización de nuestra zona horaria, y así tenerlo de forma síncrona todo el tiempo. Para ello, deberemos configurar esto en nuestro archivo de configuración que se encuentra en `/etc/NTP.conf`.

En este archivo encontraremos una gran cantidad de configuraciones, pero para ajustar la hora y fecha nos interesa, de momento, la sección de los servidores:

```
# Specify one or more NTP servers.

# Use servers from the NTP Pool Project. Approved by Ubuntu Technical Board
# on 2011-02-08 (LP: #104525). See http://www.pool.ntp.org/join.html for
# more information.
pool 0.ubuntu.pool.ntp.org iburst
pool 1.ubuntu.pool.ntp.org iburst
pool 2.ubuntu.pool.ntp.org iburst
pool 3.ubuntu.pool.ntp.org iburst
```

Como podemos observar, tenemos especificados una serie de “pools” que hacen referencia a una serie de servidores genéricos de sincronización horaria, pero podríamos establecer los específicos de nuestra región (por ejemplo, España). Para ello, solo tendríamos que modificarlos y encontrar los que hagan referencia a nuestra región.

Una vez configurados, reiniciaríamos el servicio para que se aplique la configuración y comprobaríamos que está la hora sincronizada en nuestros servidores:

```
root@gestion:~# sudo service ntp restart
root@gestion:~# date
Fri May 14 13:27:14 CEST 2021
root@gestion:~#
```

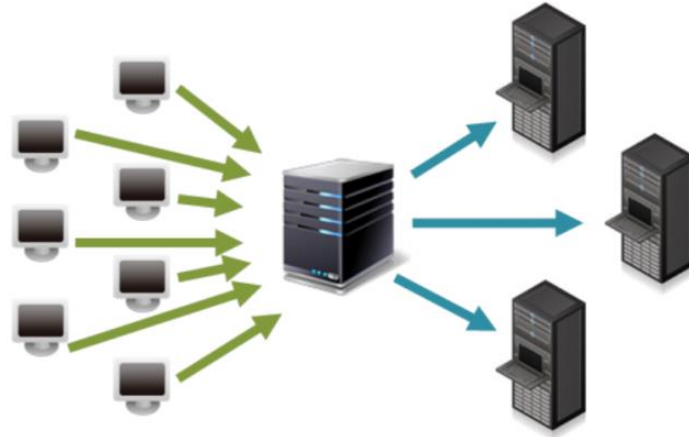




## 5.6. Balanceo de carga (HAProxy)

Hay veces en las que los servicios se congestionan y tardan en dar respuesta debido a las múltiples peticiones de clientes.

Una de las soluciones más comunes y eficientes a largo plazo es añadir servidores “espejo”, los cuales recibirán las peticiones a través de los balanceadores de carga, que irán repartiendo las peticiones con los diferentes algoritmos de reparto.



Hay múltiples servicios que pueden ofrecer balanceo de carga, nosotros usaremos el servicio HAProxy.

[Más adelante](#), se explicará cómo usar NGINX como balanceador de carga y proxy inverso.

### 5.6.1 Instalación de HAProxy:

Lo primero que debemos hacer es instalar el servicio en los balanceadores:

```
root@lb-publico-01:~# apt install haproxy
Reading package lists... Done
Building dependency tree
Reading state information... Done
haproxy is already the newest version (1.7.5-2).
0 upgraded, 0 newly installed, 0 to remove and 33 not upgraded.
root@lb-publico-01:~#
```



## 5.6.2 Configuración del servicio:

Dentro de la configuración de HAProxy, ubicada en el fichero `/etc/HAProxy/HAProxy.cfg`, encontramos cuatro secciones:

- **Global settings:** configuraremos los valores deseados sobre el propio servicio como, por ejemplo, dónde estará ubicado el log, qué usuario ejecutará el servicio, etc.

```
global
log /dev/log local0
log /dev/log local1 notice
stats socket /run/haproxy/admin.sock
stats timeout 30s
user haproxy
group haproxy
daemon
# Default SSL material locations
ca-base /etc/ssl/certs
crt-base /etc/ssl/private
# Default ciphers to use on SSL-enabled listening sockets.
# For more information, see ciphers(1SSL). This list is from:
# https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
ssl-default-bind-ciphers kEECDH+aRSA+AES:kRSA+AES:+AES256:RC4-SHA:!kEDH:!LOW:!EXP:!MD5:!aNULL:!eNULL
ssl-default-bind-options no-ssl3
# Default ciphers to use on SSL-enabled listening sockets.
# For more information, see ciphers(1SSL). This list is from:
# https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
ssl-default-server-ciphers kEECDH+aRSA+AES:kRSA+AES:+AES256:RC4-SHA:!kEDH:!LOW:!EXP:!MD5:!aNULL:!eNULL
ssl-default-server-options no-ssl3
nbproc 1
```

- **Defaults:** estableceremos los parámetros por defecto que usarán las demás secciones como, por ejemplo, los timeouts de los servidores, el modo de conexión con el frontend o los HTML que devolverá dependiendo del código recibido:

```
defaults
log global
mode http
option httplog
option dontlognull
timeout connect 5000
timeout client 50000
timeout server 50000
errorfile 400 /etc/haproxy/errors/400.http
errorfile 403 /etc/haproxy/errors/403.http
errorfile 408 /etc/haproxy/errors/408.http
errorfile 500 /etc/haproxy/errors/500.http
errorfile 502 /etc/haproxy/errors/502.http
errorfile 503 /etc/haproxy/errors/503.http
errorfile 504 /etc/haproxy/errors/504.http
```

- **Frontends:** encontraremos los frontends declarados. Dentro de estos, podremos configurar una serie de ACLs para poder enviar las peticiones a distintos backends.
- **Backends:** se encontrarán los servidores a los que el balanceador redirigirá el tráfico, dependiendo del frontend.

A continuación, es muestran los frontends y backends de los balanceadores, tanto públicos como privados.



### 5.6.2.1. Balanceadores públicos

#### Frontends:

```
frontend http
description Tráfico http
bind 0.0.0.0:80
acl url_certbot path_beg "/.well-known"
redirect scheme https code 301 if ! url_certbot
use_backend certbot if url_certbot
frontend https
description Tráfico https
bind 0.0.0.0:443 ssl crt /mnt/certificados/mordiscos/mordiscos.ovh.pem
mode http
acl url_nagios path_beg "/nagios"
acl acl_owncloud hdr(host) -i owncloud.mordiscos.ovh
use_backend nagios if url_nagios
use_backend owncloud if acl_owncloud
default backend owncloud
```

#### Backends:

```
backend certbot
mode http
balance roundrobin
server gestion 172.16.133.33:80 check
backend nagios
mode http
balance roundrobin
server gestion 172.16.133.33:80 check
backend owncloud
mode http
balance roundrobin
server manager 172.16.133.200:8080 check
server worker-01 172.16.133.201:8080 check
server worker-02 172.16.133.202:8080 check
```

En nuestra configuración podemos distinguir 3 backends:

1. **Certbot:** este backend será utilizado cuando la ACL del frontend en cuestión, coincida con la condición puesta. En este caso, la condición es que llegue con “/.well-known” al frontend. Esto servirá para que Let’s Encrypt valide y firme nuestros certificados.
2. **Nagios:** este backend será utilizado cuando la ACL del frontend en cuestión, coincida con la condición puesta. En este caso la condición es que llegue con “/nagios” al frontend.
3. **Owncloud:** este backend será utilizado cuando las ACLs de los frontends no coincidan con la condición. No obstante, Owncloud está configurado para que solo permita conexiones cuando la cabecera “Host” tenga el nombre concreto de “owncloud.mordiscos.ovh”.



## 5.6.2.2. Balanceadores privados

### Frontends:

```
frontend mysql
description Tráfico mysql
bind 0.0.0.0:3306
mode tcp
default_backend mysqlservers
```

### Backends:

```
backend mysqlservers
mode tcp
balance first
server db-privado-01 172.16.133.251:3306 check
server db-privado-02 172.16.133.252:3306 check backup
```

En el caso de los balanceadores privados, solo contemplaremos el balanceo de carga entre los servidores de base de datos, realizando una redirección a nivel de capa de red en lugar de a nivel de capa de aplicación.

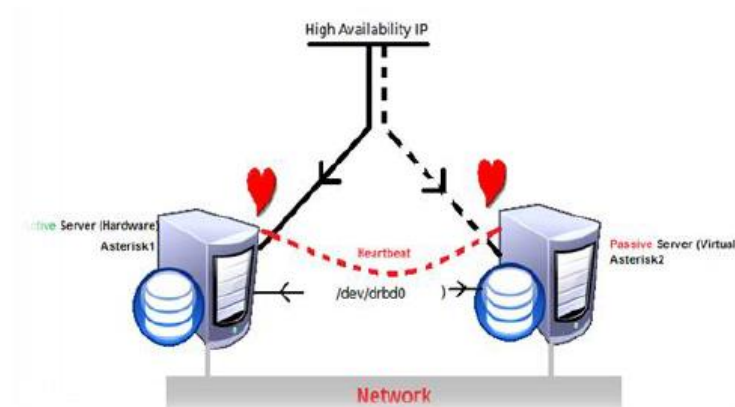


## 5.7. Acceso con alta disponibilidad (Heartbeat)

La caída de un servicio puede resultar fatal para una empresa, ya que dejar sin servicio a los clientes puede ocasionar un gran impacto económico, entre otras cosas.

Por este motivo, es necesario disponer de una alta disponibilidad que pueda evitar la pérdida de servicio, ya que es muy importante que, a la hora de la caída de un servidor, tengamos otro que responda por este a las peticiones. Para esto, dispondremos de una VIP (Virtual IP) que será la encargada de preguntar a nuestros servidores si “están vivos”, transfiriendo rápidamente la VIP a la máquina activa para que pueda seguir brindando servicios.

En nuestro caso un servicio llamado Heartbeat, será el responsable de esto.



### 5.7.1 Instalación y configuración

#### Servidor público 1 y servidor público 2:

Primero deberemos instalar el servicio de Heartbeat en los balanceadores:

```
root@lb-publico-01:~# apt install heartbeat
Reading package lists... Done
Building dependency tree
Reading state information... Done
heartbeat is already the newest version (1:3.0.6-5).
0 upgraded, 0 newly installed, 0 to remove and 32 not upgraded.
root@lb-publico-01:~#
```

Una vez instalado, podemos observar que tenemos tres ficheros de configuración:

- **ha.cf:** este es el archivo principal de configuración.
- **haresources:** este es el archivo de los recursos.
- **authkeys:** este es el archivo de autenticación.



Por último, vemos dónde se encuentran estos ficheros.

Una vez accedida a la ruta, podremos observar un README. Este contiene información más específica de cómo configurar nuestro servicio:

```
root@lb-publico-01:~# cd /usr/share/doc/heartbeat/
root@lb-publico-01:/usr/share/doc/heartbeat# ls
AUTHORS  README.cts.gz  authkeys      changelog.gz  ha.cf.gz
README  apphbd.cf      changelog.Debian.gz  copyright      haresources.gz
```

```
Install heartbeat, heartbeat-pils, and heartbeat-stonith on all three
machines. Set up the configuration on the cluster machines *and make
a copy of it on the test exerciser machine*. These are the necessary
files:
```

```
    /etc/ha.d/ha.cf
    /etc/ha.d/haresources
    /etc/ha.d/authkeys
```

También podemos ver que nuestros archivos de configuración están compresos en formato gz (excepto el fichero authkeys). El README nos indica que deben de estar en su correspondiente directorio, para ello deberemos copiarlos:

```
root@lb-publico-01:/usr/share/doc/heartbeat# cp /usr/share/doc/heartbeat/haresources.gz /etc/ha.d/
root@lb-publico-01:/usr/share/doc/heartbeat# cp /usr/share/doc/heartbeat/ha.cf.gz /etc/ha.d/
root@lb-publico-01:/usr/share/doc/heartbeat# cp /usr/share/doc/heartbeat/authkeys /etc/ha.d/
```

Una vez copiados, deberemos ir a la ruta donde los hemos copiado. Una vez en la ruta, será necesario descomprimir los archivos de ha.cf.gz y haresources.gz:

```
root@lb-publico-01:/etc/ha.d# gzip -d haresources.gz
```

```
root@lb-publico-01:/etc/ha.d# gzip -d ha.cf.gz
```

```
root@lb-publico-01:/etc/ha.d# ls
README.config  authkeys  ha.cf  ha.cf.gz  harc  haresources  haresources.gz  rc.d  resource.d  shellfuncs
```

Lo primero que haremos, será configurar el método y la clave con la que se van a autenticar nuestros servidores entre sí. Para ello, deberemos configurarlo en el authkeys:

```
auth 2
#1 crc
2 sha1 prueba
#3 md5 Hello!
```

Como podremos observar, tenemos distintos algoritmos hash. En nuestro caso elegiremos el número 2 (sha1).



Deberemos cambiar los permisos, de esta forma solo el usuario root podrá leer dicha información:

```
root@lb-publico-01:/etc/ha.d# chmod 600 /etc/ha.d/authkeys
root@lb-publico-01:/etc/ha.d# ls -la | grep authkeys
-rw----- 1 root root 43 May 18 18:12 authkeys
```

Ahora, se editará el archivo de configuración principal (ha.cf), en el cual se indicarán varias directivas necesarias para el correcto funcionamiento. En la página oficial se contemplan más directivas distintas. No obstante, nosotros solo usaremos las siguientes:

```
bcast eth1 eth0

keepalive 2

warntime 5

deadtime 10

udpport 694
auto_failback on

#Nodo principal
node lb-publico-01
#Nodo que entrara en accion si se cae el nodo principal
node lb-publico-02
```

**bcast:** en esta directiva se configuran en qué interfaces de red envía Heartbeat el tráfico de difusión UDP.

**keepalive:** esta directiva establece el intervalo entre los paquetes de Heartbeat.

**deadtime:** la usaremos para indicarle a Heartbeat la rapidez con la que decide que un nodo está muerto, con este valor deberemos tener cuidado ya que un valor muy bajo puede ocasionar que el sistema se declare muerto y, un valor muy alto, retrasaría demasiado el relevo tras el fallo de uno de los nodos.

**udpport:** especifica el puerto UDP por el cual Heartbeat realizará la comunicación.

**node:** esta directiva controla qué máquinas están en nuestro cluster.



A continuación, debemos configurar el haresources:

```
lb-publico-01 172.16.133.35/16/eth1
```

Esta VIP será la que utilizarán ambos nodos.

Por último, quedaría reiniciar el servicio y aplicar la configuración al servidor 2.

```
root@lb-publico-01:/etc/ha.d# service heartbeat restart
root@lb-publico-01:/etc/ha.d# service heartbeat status
* heartbeat.service - Heartbeat High Availability Cluster Communication and Membership
   Loaded: loaded (/lib/systemd/system/heartbeat.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-05-19 21:53:51 CEST; 2s ago
     Process: 31538 ExecStop=/usr/lib/heartbeat/heartbeat -k (code=exited, status=0/SUCCESS)
    Main PID: 31708 (heartbeat)
      Tasks: 6 (limit: 4915)
     CGroup: /system.slice/heartbeat.service
             |-31708 heartbeat: master control process
             |-31711 heartbeat: FIFO reader
             |-31712 heartbeat: write: bcast eth1
             |-31713 heartbeat: read: bcast eth1
             |-31714 heartbeat: write: bcast eth0
             `--31715 heartbeat: read: bcast eth0
```

### Servidor privado 1 y servidor privado 2:

Para la configuración de los servidores privados, lo primero que haremos nuevamente, será configurar el método y la clave con la que se van a autenticar nuestros servidores entre sí. Para ello, deberemos configurarlo en el authkeys y tendrá que tener la misma configuración que el servidor 2:

```
auth 2
#1 crc
2 sha1 prueba
#3 md5 Hello!
```

De nuevo, tenemos distintos algoritmos hash y elegiremos el número 2 (sha1).

Deberemos cambiar los permisos para que sólo el usuario root pueda leer dicha información:

```
root@lb-privado-01:/etc/ha.d# chmod 600 /etc/ha.d/authkeys
root@lb-privado-01:/etc/ha.d# ls -la | grep authkeys
-rw----- 1 root root 43 May 16 12:41 authkeys
root@lb-privado-01:/etc/ha.d#
```

En el caso de los balanceadores privados, ésta será la configuración que les aplicaremos:

```
bcast eth0
keepalive 2
warntime 5
deadtime 10
udpport 694
auto_failback on

#Nodo principal
node lb-privado-01
#Nodo que entrara en accion si se cae el nodo principal
node lb-privado-02
```



Debemos configurar por último el haresources con la configuración correspondiente a los balanceadores privados:

```
lb-privado-01 172.16.133.12/16/eth0
```

En este indicaremos el recurso con el que se trabajará. En nuestro caso, a diferencia de los servidores públicos, solo disponemos de una interfaz, por esto deberemos establecer solo una VIP, como vemos también le indicamos cuál será nuestro nodo principal.

Esta VIP será por la que se comuniquen ambos nodos.

Por último, quedaría reiniciar el servicio y comprobar que Heartbeat con la VIP funciona correctamente.

```
root@lb-privado-01:/etc/ha.d# service heartbeat restart
root@lb-privado-01:/etc/ha.d# service heartbeat status
* heartbeat.service - Heartbeat High Availability Cluster Communication and Membership
   Loaded: loaded (/lib/systemd/system/heartbeat.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2021-05-20 17:43:46 CEST; 3s ago
     Process: 30931 ExecStop=/usr/lib/heartbeat/heartbeat -k (code=exited, status=0/SUCCESS)
    Main PID: 31020 (heartbeat)
       Tasks: 4 (limit: 4915)
      CGroup: /system.slice/heartbeat.service
              |-31020 heartbeat: master control process
              |-31023 heartbeat: FIFO reader
              |-31024 heartbeat: write: bcast eth0
              `--31025 heartbeat: read: bcast eth0
```





## 5.8. Firewall (Iptables)

Dado que nuestros balanceadores públicos están expuestos a internet y necesitamos tener control sobre el tráfico entrante y saliente, vamos a usar un firewall (iptables) gestionado por un script.



Este script se genera a través de un rol Ansible, el cual genera el propio script a través de variables en una plantilla.

### 5.8.1 Script

```
# Políticas por defecto
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

# Open ports.
iptables -A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 443 -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 9000 -j ACCEPT

# Accept traffic from loopback interface (localhost).
iptables -A INPUT -i lo -j ACCEPT

# Habilitar ip_forward y MASQUERADE para la red interna de la infraestructura
sed 's/^#net.ipv4.ip_forward=1/net.ipv4.ip_forward=1/g' /etc/sysctl.conf
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE

# Forwarded ports.

# Accept icmp ping requests.
iptables -A INPUT -p icmp -j ACCEPT

# Allow NTP traffic for time synchronization.
iptables -A OUTPUT -p udp --dport 123 -j ACCEPT
iptables -A INPUT -p udp --sport 123 -j ACCEPT

# Additional custom rules.

# Allow established connections:
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
~
```

## 5.8.2 Creación del servicio

Este script lo moveremos a /etc/firewall.bash en la máquina Gestion:

```
root@gestion:~# ls -la /etc/firewall.bash
-rwxr--r-- 1 root root 1019 May  7 22:43 /etc/firewall.bash
```

Una vez movido, se procederá a la creación y configuración del servicio en systemd:

```
root@gestion:/etc/systemd/system# ls -la | grep "firewall.service"
-rw-r--r-- 1 root root 201 May  7 22:43 firewall.service
```

```
[Service]
Type=oneshot
ExecStart=/etc/firewall.bash
ExecStop=/sbin/iptables -F
RemainAfterExit=yes
```

Ahora habilitaremos e iniciaremos el servicio, comprobando que ha iniciado correctamente sin errores.

```
root@gestion:/home/ruben# systemctl enable firewall.service
Created symlink /etc/systemd/system/multi-user.target.wants/firewall.service → /etc/systemd/system/firewall.service.
root@gestion:/home/ruben# systemctl start firewall.service
root@gestion:/home/ruben# systemctl status firewall.service
● firewall.service - Firewall
   Loaded: loaded (/etc/systemd/system/firewall.service; enabled; vendor preset: enabled)
   Active: active (exited) since Wed 2021-05-12 20:49:03 CEST; 24h ago
     Main PID: 663 (code=exited, status=0/SUCCESS)
       Tasks: 0 (limit=9449)
        Memory: 0B
         CGroup: /system.slice/firewall.service

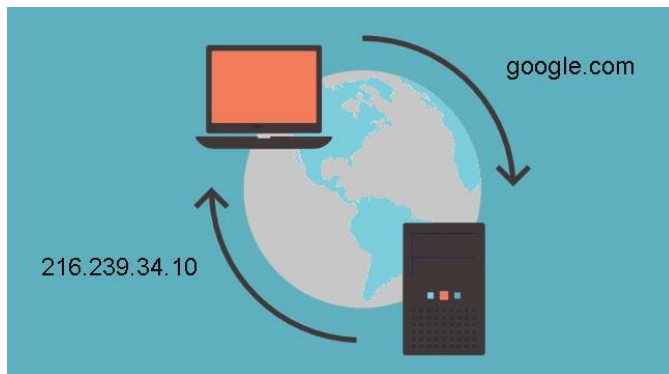
May 12 20:49:03 gestion systemd[1]: Starting Firewall...
May 12 20:49:03 gestion systemd[1]: Finished Firewall.
```



## 5.9. Resolución de nombres (DNS)

Cuando quieres acceder a una página web se necesita una IP, dado que los servidores funcionan con direcciones IP. Esto es un problema. Imagínate que, por ejemplo, tengas que aprenderte todas las direcciones IP de todas las páginas webs a las que entras al cabo del día. Como bien puedes suponer esto es imposible, es por este motivo que se utilizan nombres en lugar de IPs. A estos nombres se les denomina dominios.

Los dominios son traducidos a IPs mediante el protocolo DNS, que se encarga de la administración del espacio de nombre de dominio, teniendo como labor el resolver las peticiones de asignación de nombre.



Dentro de los múltiples servicios que ofrece el protocolo DNS, nosotros usaremos Bind9. Este es el servicio de DNS por excelencia de GNU/Linux.

### 5.9.1 Instalación

Para realizar la configuración del servicio, primero deberemos instalar el servicio Gestion:

```
root@gestion:/home/ruben# apt install bind9
Reading package lists... Done
Building dependency tree
Reading state information... Done
bind9 is already the newest version (1:9.16.1-0ubuntu2.8).
The following packages were automatically installed and are no longer required:
  linux-headers-5.4.0-71 linux-headers-5.4.0-71-generic linux-image-5.4.0-71-generic
  linux-modules-5.4.0-71-generic linux-modules-extra-5.4.0-71-generic
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 121 not upgraded.
```

### 5.9.2 Configuración

Ahora deberemos configurar nuestra zona maestra de mordiscos.ovh, en el fichero /etc/bind/named.conf.local, el cual se encarga de definir las zonas mediante una serie de parámetros.

Nuestra zona será maestra y se corresponderá a mordiscos.ovh. También se especifica el fichero de zona donde se ubicarán los registros de recursos.

```
# Ansible managed
zone "mordiscos.ovh" IN {
    type master;
    file "/etc/bind/zones/mordiscos.ovh/db";
    allow-transfer {
        "none";
    };
    inline-signing yes;
}
```

Con esto, ya tendremos nuestra zona creada. Ahora deberemos configurar los registros. Para ello, deberemos ir a nuestro fichero en la ruta anteriormente especificada y configurarla:

```
Ansible managed

$TTL      3600
@         IN      SOA      mordiscos.ovh. admin.mordiscos.ovh. (
    1 ; Serial
    14400 ; Refresh
    86400 ; Retry
    2419200 ; Expire
    86400 ; Negative Cache TTL
)

@         IN NS      mordiscos.ovh.
dns       IN NS      mordiscos.ovh.
@         IN A       90.74.64.190
dns       IN A       90.74.64.190
ts3       IN CNAME   90.74.64.190
```

Como se puede observar, le indicamos nuestros registros, ya sean tipo ns, a, cname, etc.

Una vez configurado, deberemos validar que está todo correcto:

```
root@worker-01:/etc/bind# named-checkconf
root@worker-01:/etc/bind# named-checkzone mordiscos.ovh zones/mordiscos.ovh/db
zone mordiscos.ovh/IN: loaded serial 1
OK
```

Habilitaremos el servicio y observaremos si está iniciado correctamente:

```
root@worker-01:/# systemctl enable bind9
Synchronizing state of bind9.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable bind9
```

```
root@worker-01:/# systemctl status bind9
* bind9.service - BIND Domain Name Server
   Loaded: loaded (/lib/systemd/system/bind9.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2021-05-13 19:16:58 CEST; 3h 23min ago
     Docs: man:named(8)
    Main PID: 16744 (named)
      CPU: 37ms
     CGroup: /system.slice/bind9.service
            └─16744 /usr/sbin/named -f -u bind
```

Por último, comprobaremos que funciona haciendo una prueba de resolución DNS con nslookup:

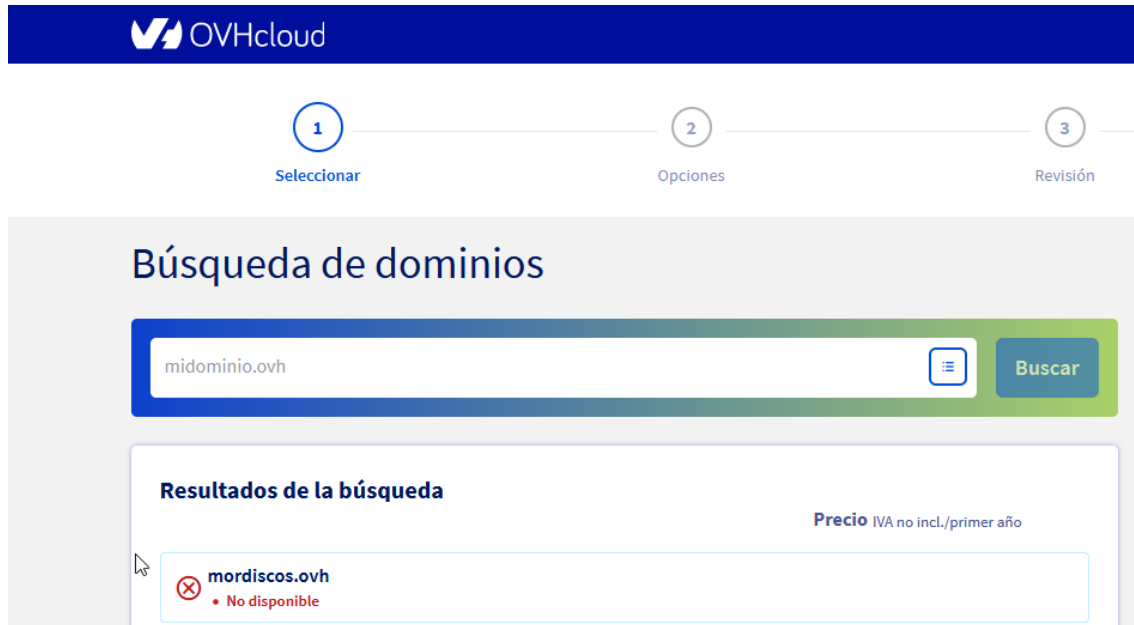
```
root@worker-01:/# nslookup
> server 127.0.0.1
Default server: 127.0.0.1
Address: 127.0.0.1#53
> set type=cname
> ts3.mordiscos.ovh
Server:      127.0.0.1
Address:     127.0.0.1#53

ts3.mordiscos.ovh      canonical name = 90.74.64.190.mordiscos.ovh.
```



### 5.9.3 Compra del dominio mordiscos.ovh

Lo primero será comprobar que el dominio está disponible y realizar la compra con los datos requeridos:



The screenshot shows the OVHcloud website header with the logo. Below it is a progress bar with three steps: 1. Seleccionar, 2. Opciones, and 3. Revisión. The main section is titled 'Búsqueda de dominios'. It features a search bar with the text 'midominio.ovh' and a 'Buscar' button. Below the search bar, under 'Resultados de la búsqueda', there is a single result for 'mordiscos.ovh' with a red 'X' icon and the text 'No disponible'. A price note 'Precio IVA no incl./primer año' is visible on the right.

Para poder gestionar desde nuestras máquinas el dominio, será necesario configurar los servidores DNS apuntando a nuestra IP pública.

### 5.9.4 Registros del dominio en el proveedor

Después de contratar el dominio, deberemos añadir un registro de tipo A apuntando a nuestra IP pública:



The screenshot shows a form titled 'Añadir un registro a la zona DNS' with the subtitle 'Paso 2 de 3'. A note states: '\* Los campos con asterisco son obligatorios.' The form has three main fields: 'Subdominio' with the value 'dns' and a domain suffix '.mordiscos.ovh.', 'TTL' with a dropdown menu set to 'Por defecto', and 'Destino \*' with the IP address '90.74.64.190'. Below the form, it says 'Va a generar el siguiente registro A:' followed by a box containing 'dns IN A 90.74.64.190'. At the bottom, there are three buttons: 'Cancelar', 'Anterior', and 'Siguiente'.



## 5.9.5 Configuración de los servidores DNS del proveedor

En el apartado “Servidores DNS” tendremos la siguiente ventana.

Para asegurarse de que la configuración funcionará correctamente, puede utilizar la [herramienta de verificación de DNS](#).

Servidor DNS	IP asociada	Estado	
dns107.ovh.net	-	Activo	
ns107.ovh.net	-	Activo	

Debemos configurar los servidores, eliminando los que haya y añadiendo el nuestro.

Los DNS se han cambiado correctamente. La propagación al resto de DNS podría tardar hasta 48 h.

Para asegurarse de que la configuración funcionará correctamente, puede utilizar la [herramienta de verificación de DNS](#).

Servidor DNS	IP asociada	Estado	
dns.mordiscos.ovh	90.74.64.190	En proceso	
dns107.ovh.net	-	Eliminando	
ns107.ovh.net	-	Eliminando	

Este cambio lo han de confirmar y suele tardar 1-2 días:

Servidor DNS	IP asociada	Estado	
dns.mordiscos.ovh	90.74.64.190	Activo	

A partir de aquí, los registros serán automatizados desde Ansible contra las máquinas que ofrezcan servicio DNS con bind9:

```
www      IN      CNAME   mordiscos.ovh

(jose@ DESKTOP-F2NU733) ~
$ dig www.mordiscos.ovh

; <<>> DiG 9.16.8-Debian <<>> www.mordiscos.ovh
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 1465
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:;; udp: 1232
;; QUESTION SECTION:
;www.mordiscos.ovh.      IN      A
;; ANSWER SECTION:
www.mordiscos.ovh.      43200  IN      CNAME   mordiscos.ovh.mordiscos.ovh.
;; AUTHORITY SECTION:
mordiscos.ovh.          3248   IN      SOA      mordiscos.ovh. admin.mordiscos.ovh. 1 604800 86400 2419200 86400
;; Query time: 20 msec
;; SERVER: 9.9.9.9#53(9.9.9.9)
;; WHEN: Wed Apr 28 19:01:16 CEST 2021
;; MSG SIZE rcvd: 116

(jose@ DESKTOP-F2NU733) ~
$
```

## 5.10. Certificados

Nuestras páginas webs son autenticadas mediante certificados, un certificado es el encargado de autenticar que la entidad es quien dice ser, que, a su vez, son administrador por una CA, siendo esta una entidad certificadora encargada de confirmar la identidad del solicitante mediante certificados.

Nuestros certificados serán expedidos mediante Let's Encrypt que es una CA que proporciona certificados de forma gratuita.

Para garantizar la seguridad usaremos SSL (secure socket layer) esto nos lo proporcionará certbot.

### 5.10.1 ¿Qué es certbot?

Certbot es un cliente automático que obtiene certificados SSL provistos por Let's Encrypt, está pensado para ser ejecutado directamente sobre un servidor web.

Utiliza una serie de comandos para solicitar acciones como obtención, renovación o revocación de certificados.

El cliente de certbot soporta dos tipos de plugins para obtener e instalar certificados:

- Autenticadores: encargados de validar que eres tú mismo el que controla el dominio para el que se está solicitando el certificado, obtiene un certificado para el dominio especificado y lo coloca en su respectivo directorio, cabe destacar que no instalara el certificado.
- Instaladores: encargados de instalar un certificado

Entre todos los plugins existentes, nosotros usaremos Webroot.

### 5.10.2 ¿Cómo funciona Webroot?

Webroot nos permite no detener el servicio web durante el proceso de emisión de certificados, este plugin funciona creando un archivo temporal para cada uno de los dominios solicitados llamado "dominio"/.well-known.

El servidor de validación de Let's Encrypt realiza peticiones HTTP para validar que los DNS de cada dominio solicitado resuelven en el servidor donde se ejecuta certbot.

Esta validación se realizará contra la máquina gestión en la cual tenemos centralizados los certificados, para ello, Let's Encrypt deberá pasar por los balanceadores públicos, es por esto que en nuestro haproxy deberemos aplicar una serie de reglas que permita realizar esto.

Cuando Let's Encrypt realiza esta petición a la ruta /.well-known para la validación de los certificados mediante el puerto 80, le tenemos indicado que redirija mediante a un backend de certbot en el cual apunta a la ip de gestión para que Let's Encrypt pueda realizar la validación.



### 5.10.3 Instalación

Primero deberemos agregar el repositorio de Certbot y actualizar las librerías sobre Gestion:

```
root@gestion:~# add-apt-repository ppa:certbot/certbot
```

```
root@gestion:~# apt update
Hit:1 http://archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://repo.mysql.com/apt/ubuntu bionic InRelease
Hit:4 http://es.archive.ubuntu.com/ubuntu focal InRelease
Get:5 http://es.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [9383 B]
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Packages [47.2 kB]
Get:7 http://es.archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
Get:8 http://es.archive.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:9 http://es.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [1032 kB]
Get:10 http://es.archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [13.5 kB]
```

Una vez hecho esto, podemos instalar certbot.

```
root@gestion:~# apt install certbot
Reading package lists... Done
Building dependency tree
Reading state information... Done
certbot is already the newest version (0.40.0-1ubuntu0.1).
0 upgraded, 0 newly installed, 0 to remove and 110 not upgraded.
```





### 5.10.4 Configuración

Deberemos configurar nuestro servidor, en este caso emplearemos HAProxy para indicarle hacia donde irán dirigidas las peticiones de Let's Encrypt mediante un frontend y un backend explicados anteriormente.

Ahora deberemos obtener el certificado SSL con Certbot. Usaremos el dominio owncloud.mordiscos.ovh, pero podremos indicarle todos los que necesitemos.

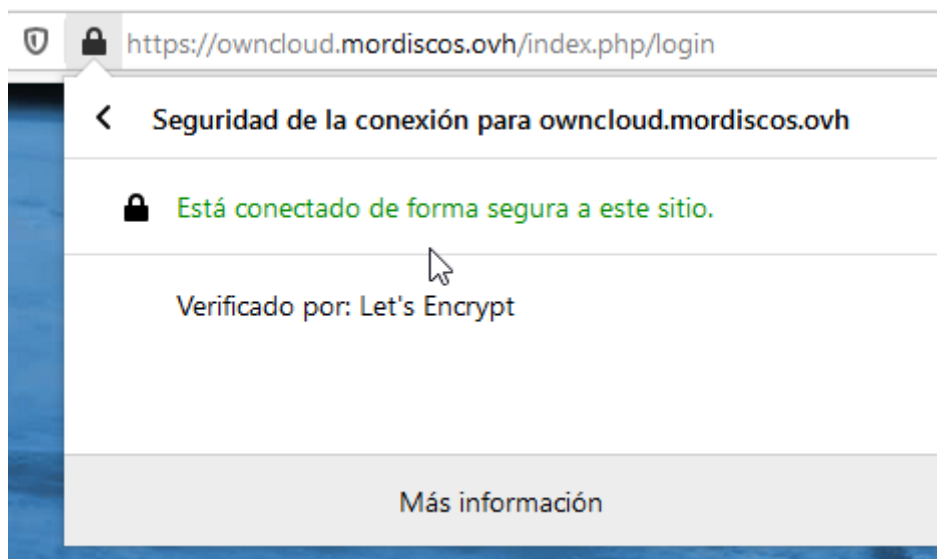
```
root@gestion:~# certbot certonly -d owncloud.mordiscos.ovh
Saving debug log to /var/log/letsencrypt/letsencrypt.log

How would you like to authenticate with the ACME CA?
-----
1: Spin up a temporary webserver (standalone)
2: Place files in webroot directory (webroot)
-----
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
Plugins selected: Authenticator webroot, Installer None
Obtaining a new certificate

IMPORTANT NOTES:
- Congratulations! Your certificate and chain have been saved at:
  /etc/letsencrypt/live/owncloud.mordiscos.ovh/fullchain.pem
  Your key file has been saved at:
  /etc/letsencrypt/live/owncloud.mordiscos.ovh/privkey.pem
  Your cert will expire on 2021-09-14. To obtain a new or tweaked
  version of this certificate in the future, simply run certbot
  again. To non-interactively renew *all* of your certificates, run
  "certbot renew"
- If you like Certbot, please consider supporting our work by:

  Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
  Donating to EFF: https://eff.org/donate-le
```

Para comprobar que el proceso ha funcionado deberemos acceder a nuestra web y comprobar que efectivamente el certificado esta validado y verificado por Let's Encrypt

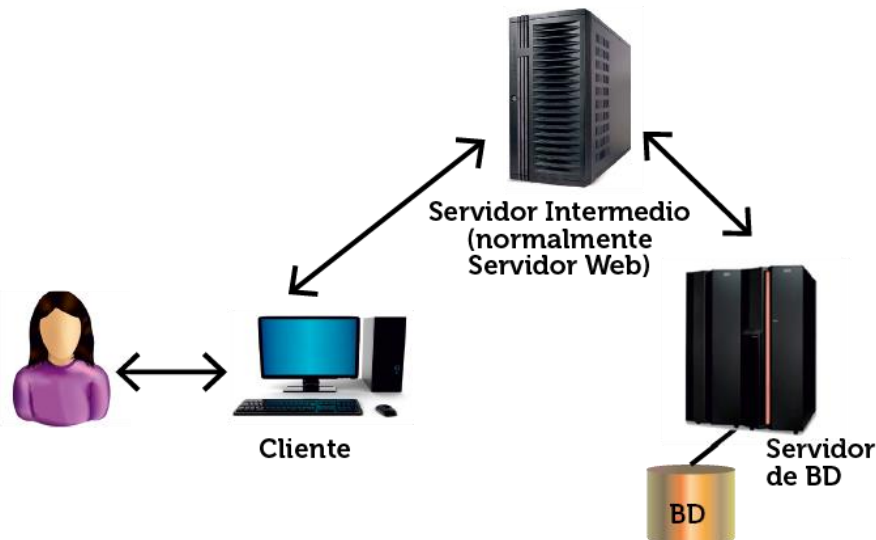


## 5.11. Bases de datos (MySQL)

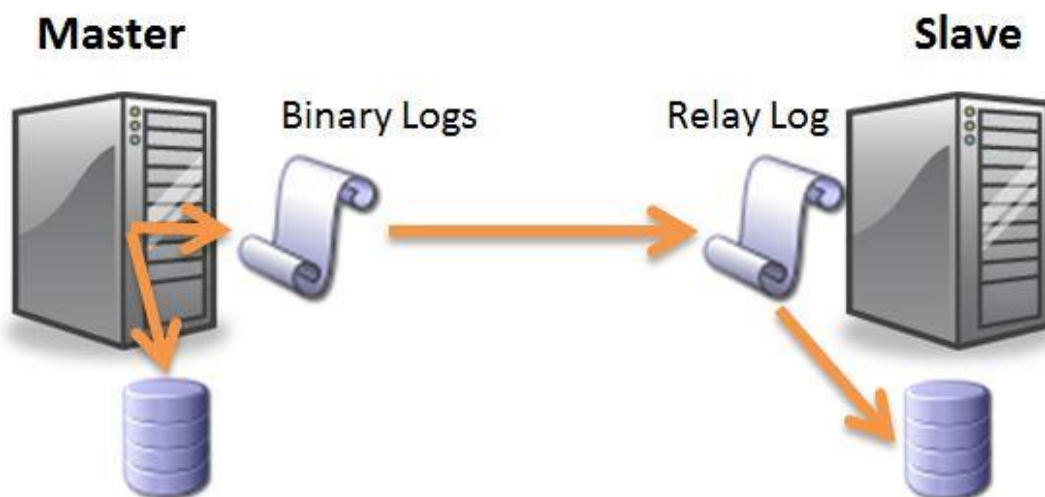
Dado que nuestras aplicaciones necesitan almacenar los datos de alguna manera, se usará una base de datos para su almacenaje.

Al disponer de una infraestructura dividida por capas, nuestra base de datos se localizará en la **capa de datos**.

La ventaja de usar una base de datos separada en una capa aparte, es que, ésta sólo permite conexiones entrantes mediante la capa intermedia. En este caso, los balanceadores.



Para suplir la alta disponibilidad, usaremos un modelo master-slave con dos servidores: el servidor01 que hará de master y el servidor02 que hará de slave. Esto quiere decir que toda la información llegada al servidor01 se replicará al servidor02.



Existen múltiples sistemas de Gestión de bases de datos. Nosotros usaremos MySQL.

### 5.11.1 Instalación

Lo primero será instalar el servicio en los servidores bases de datos y comprobar si está correctamente iniciado:

```
root@db-privado-02:~# apt install mysql-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
mysql-server is already the newest version (5.7.34-1debian9).
0 upgraded, 0 newly installed, 0 to remove and 88 not upgraded.
root@db-privado-02:~# systemctl status mysql.service
* mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2021-05-24 15:04:21 CEST; 1h 30min ago
     Process: 14701 ExecStart=/usr/sbin/mysqld --daemonize --pid-file=/var/run/mysqld/mysqld.pid (code=exited, status=0/SUCCESS)
     Process: 14666 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
    Main PID: 14703 (mysqld)
      Tasks: 30 (limit: 4915)
     CGroup: /system.slice/mysql.service
            └─14703 /usr/sbin/mysqld --daemonize --pid-file=/var/run/mysqld/mysqld.pid

May 24 15:04:21 db-privado-02 systemd[1]: Starting MySQL Community Server...
May 24 15:04:21 db-privado-02 systemd[1]: Started MySQL Community Server.
```

### 5.11.2 Configuración

Lo primero que deberemos hacer, será realizar algunas actualizaciones de seguridad en nuestra nueva instalación:

```
root@db-privado-01:~# mysql_secure_installation
```

Primero, nos preguntará sobre un complemento de validación de contraseña. Esto servirá para hacer más o menos estricta la política de contraseñas. Habilitar esto depende de la seguridad que deseemos.

Lo siguiente será si queremos cambiar la contraseña de root.

Se le preguntará si desea eliminar el usuario anónimo de MySQL, deshabilitar el inicio de sesión de root remoto, eliminar la base de datos de prueba y volver a cargar las tablas de privilegios para garantizar que los cambios anteriores se apliquen correctamente.

Una vez realizada esta configuración de seguridad, ya podemos probar nuestro MySQL.

```
root@db-privado-01:~# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 327
Server version: 5.7.34-log MySQL Community Server (GPL)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```



### 5.11.3 Preparando la replicación

Como hemos comentado antes, vamos a realizar una configuración la cual nos permitirá disponer de una alta disponibilidad.

Para ello, deberemos habilitar la replicación del servidor master al servidor slave. Todo esto se realizará mediante un log binario que explicaremos más adelante.

#### Servidor-01:

El servidor slave se autenticará frente al servidor master como un usuario normal.

Deberemos crear un usuario dentro de MySQL en el servidor-01 que sólo tendrá permiso de acceso desde la máquina slave y solo podrá realizar replicación.

```
mysql> CREATE USER 'replicationuser'@'172.16.133.252' IDENTIFIED BY [REDACTED];
```

```
mysql> GRANT REPLICATION SLAVE ON *.* TO 'replicationuser'@'172.16.133.252' IDENTIFIED BY [REDACTED];
```

Una vez creado, deberemos agregar en el fichero de configuración por defecto del servidor MySQL un identificador único para este, dicho fichero se encontrará en /etc/MySQL, el fichero a modificar será my.cnf.

Comprobaremos que en el fichero de configuración encontraremos distintos parámetros como: configuración de la memoria, parámetros sobre innnoDB (motor de almacenamiento de MySQL), configuración de logs, etc.

```
# Memory settings.
key_buffer_size = 256M
max_allowed_packet = 64M
table_open_cache = 256
sort_buffer_size = 1M
read_buffer_size = 1M
read_rnd_buffer_size = 4M
mysam_sort_buffer_size = 64M
```

Añadiremos lo siguiente indicando el ID de nuestro servidor, como este es el servidor-01 estableceremos el ID 1:

```
# Replication
server-id = 1
```

Como comentamos anteriormente, realizaremos la réplica con el log binario, para ello deberemos indicarlo también en el fichero de configuración.

Al especificar el parámetro de log\_bin estamos activando el log binario. Le daremos un nombre:

```
log_bin = mysql-bin
```

Este log binario sirve principalmente para realizar la replicación, pero también sirve para guardar un registro de las consultas que afectan a los datos de la base de datos, ya que todas las consultas SELECT (lectura) no se guardan aquí.

También destacar que activando el log binario puedes reconstruir los datos de una base de datos en un 99% desde el último backup realizado. Por lo tanto, es fundamental para mantener sana la integridad de la base de datos y sus datos.

**Servidor-02:**

Deberemos agregar en el fichero de configuración por defecto del servidor MySQL un identificador único para este, distinto en este caso al master, dicho fichero se encontrará en `/etc/MySQL`, el fichero a modificar será `my.cnf`.

Añadiremos nuestro identificador único. En este caso, como es el servidor-02, estableceremos el ID 2.

```
# Replication
server-id = 2
```

No será necesario especificar el parámetro de `log_bin` dado que usaremos el del master con la replicación.



### 5.11.4 Iniciando la replicación

Deberemos entrar a la consola de MySQL y detener el slave.

```
mysql> stop slave;
Query OK, 0 rows affected (0.00 sec)
```

Ahora, en el servidor master, miramos la información de estado del proceso y anotamos los valores *File* y *Position*. (servidor-01)

```
mysql> show master status;
+-----+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+-----+-----+-----+-----+-----+
| mysql-bin.000002 | 793648   |              |                  |                  |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Debemos bloquear las tablas para evitar que nos cambie la posición.

```
mysql> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0.02 sec)
```

En el servidor slave aplicaremos la configuración de la replicación.

```
change master to master_host='ip_servidor_maestro', master_user='usuario_servidor_maestro',
master_password='password_servidor_maestro', master_log_file='file_servidor_maestro',
master_log_pos='position_servidor_maestro';
```

Arrancamos el slave:

```
mysql> start slave;
Query OK, 0 rows affected (0.01 sec)
```

Desbloqueamos las tablas:

```
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

Ahora podremos revisar la configuración del slave:

```
Master_User: replicateuser
Master_Port: 3306
Connect_Retry: 60
Master_Log_File: mysql-bin.000002
Read_Master_Log_Pos: 797290
Relay_Log_File: db-privado-02-relay-bin.000001
Relay_Log_Pos: 4
Relay_Master_Log_File: mysql-bin.000002
Slave_IO_Running: Connecting
Slave_SQL_Running: Yes
Replicate_Do_DB:
Replicate_Ignore_DB:
```



## 5.12. Backups (mysqldump)

Los datos almacenados en las bases de datos son, por lo general, de gran importancia para el buen funcionamiento de la estructura interna o externa de muchas empresas, además de alojar datos sensibles, como cuentas, información personal, etc.

Debido a su gran importancia, proteger los sistemas de bases de datos deber ser una prioridad y debe contar con las medidas de seguridad necesarias. Es por ello que hacer copias de seguridad de forma regular previene y garantiza la protección de datos e información a largo plazo, permitiéndonos realizar una recuperación en cualquier momento en caso de pérdida de datos.

Nosotros mediante un script generado a través de un rol de ansible, usaremos la herramienta mysqldump, que es el sistema de copias de seguridad integrado de MySQL.

### 5.12.1 Script

El script realizará un backup completo de todas nuestras bases de datos, permitiendo generar respaldos de ellas.

Funciona mediante una serie de comandos y filtros, el backup será transferido a un fichero el cuál se comprimirá. Consta de función de rotado eliminando los que superen más de 7 días de antigüedad, este backup se realizará de forma diaria, ya que el script es ejecutado por un crontab todos los días.

Mediante una tarea de ansible nos permitirá también, extraer y restaurar el backup de la base de datos en caso de que sea necesario.

```
#!/bin/bash

# Declaramos la variable DATE que contendra el destino donde se almacenará el backup
DEST=/backups/mysql/
# Declaramos la variable FILENAME que contendra el formato de fecha
FILENAME="date +%d-%m-%Y_%H:%M".tar.gz
# Declaramos la variable HOSTNAME que contendra el hostname del que se hara el backup.
HOSTNAME="lb-privado"
# Declaramos la variable USER que contendra el usuario para realizar la conexion.
USER="backupuser"
# Declaramos la variable PASS que contendra la contraseña para realizar la conexion.
PASS=$(cat /root/.mysqlpass)
# Declaramos la variable DATABASES, que contendra el nombre de todas las bases de datos de las que haremos copia de seguridad.
DATABASES=$(mysql -h $HOSTNAME -u $USER -p$PASS -e "SHOW DATABASES;" | tr -d " " | grep -v "Database\|information_schema\|performance_schema\|borrar\|pruebas")

# Comprueba si el destino es un directorio, si no es asi, lo crea.
[ ! -d $DEST ] && mkdir -p $DEST

# Comprueba si databases es un directorio, si no es asi, lo crea.
if [ ! -d databases ]
then
    mkdir databases
fi

# Realiza un dump de todas las bases de datos indicadas en la variable DATABASES y mueve el contenido a un fichero sql.
for db in $DATABASES
do
    mysqldump --single-transaction --routines --quick -h $HOSTNAME -u $USER -p$PASS -B $db > databases/$db.sql
done

# Movemos el contenido al fichero
cat databases/*.sql > all_databases.sql

# Creación del archivo comprimido
tar cvfz $DEST/$FILENAME all_databases.sql

# Realiza un rotado del backup
find $DEST -mtime +7 | xargs rm -fr
```



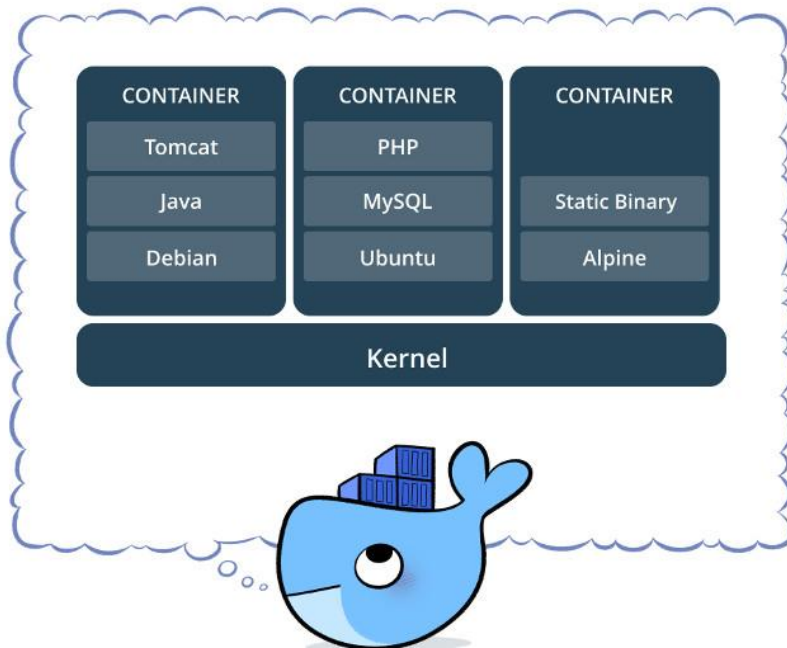


## 5.13. Contenedores y orquestadores (Docker y Docker Swarm)

Hay un problema con las aplicaciones de hoy en día, y es que, si la aplicación se mueve de un entorno a otro, surgen problemas. Estos problemas suelen deberse a las distintas configuraciones en las bibliotecas de la aplicación. Debido a esto, surgen los contenedores, solucionando estos problemas.

### 5.13.1 Contenedores

Para gestionar nuestros contenedores, vamos a utilizar la herramienta diseñada, tanto para desarrolladores como para administradores de sistemas: “**Docker**”.





### 5.13.1.1. Instalación de Docker en las máquinas

Para la instalación, lo primero que deberemos hacer es instalar los requisitos previos de Docker en la máquina que hará de manager:

```
root@manager:~# apt-get update && apt install -y apt-transport-https ca-certificates curl gnupg lsb-release
Hit:1 http://security.debian.org stretch/updates InRelease
Ign:2 http://ftp.debian.org/debian stretch InRelease
Hit:3 http://ftp.debian.org/debian stretch-updates InRelease
Hit:4 http://ftp.debian.org/debian stretch Release
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
gnupg is already the newest version (2.1.18-8~deb9u4).
lsb-release is already the newest version (9.20161125).
apt-transport-https is already the newest version (1.4.11).
ca-certificates is already the newest version (20200601~deb9u2).
curl is already the newest version (7.52.1-5+deb9u14).
The following packages were automatically installed and are no longer required:
  libglib2.0-0 libglib2.0-data shared-mime-info xdg-user-dirs
Use 'apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 105 not upgraded.
root@manager:~#
```

Una vez instalados los requisitos, debemos importar la public key de Docker en nuestras bibliotecas:

```
root@manager:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
root@manager:~# echo \
> "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
> xenial stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
root@manager:~#
```

Una vez hecho esto, actualizaremos las bibliotecas e instalaremos los paquetes de Docker, terminando así la instalación:

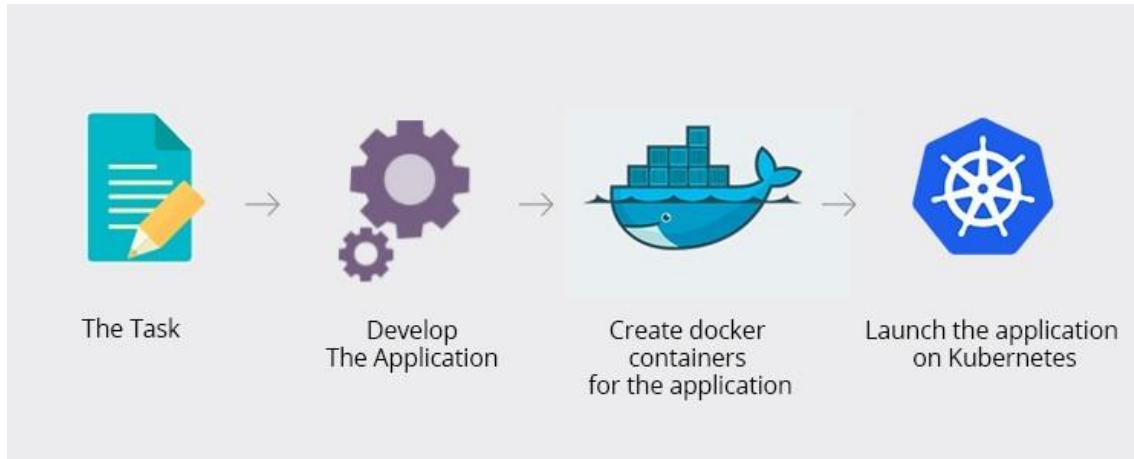
```
root@manager:~# apt update && apt install -y docker-ce docker-ce-cli containerd.io
Ign:1 http://ftp.debian.org/debian stretch InRelease
Hit:2 http://security.debian.org stretch/updates InRelease
Hit:3 http://ftp.debian.org/debian stretch-updates InRelease
Hit:4 http://ftp.debian.org/debian stretch Release
Hit:5 https://download.docker.com/linux/debian stretch InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
99 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree
Reading state information... Done
containerd.io is already the newest version (1.4.3-1).
docker-ce-cli is already the newest version (5:19.03.15~3-0~debian-stretch).
docker-ce is already the newest version (5:19.03.15~3-0~debian-stretch).
0 upgraded, 0 newly installed, 0 to remove and 99 not upgraded.
```



### 5.13.2 Orquestadores

Se puede hacer muy tedioso el hecho de llevar manualmente un sistema de contenedores, teniendo que realizar actividades como: el montaje de máquinas, inicializar contenedores, reemplazar contenedores fallidos, enlazar contenedores unos con otros, etc.

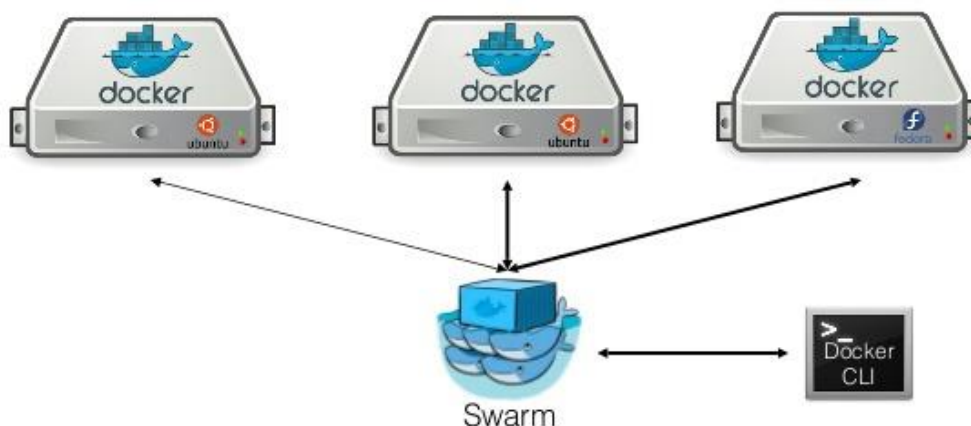
Para esto tenemos los orquestadores, encargados de aprovechar al máximo los contenedores, realizando tareas y procesos de forma automatizada.



Hay muchos sistemas de orquestación, entre ellos Kubernetes. No obstante, Kubernetes es un sistema de orquestación complejo que no necesitamos en nuestra infraestructura actual. Por ello, hemos optado por elegir “Docker Swarm”.

Docker Swarm es una herramienta integrada en Docker que nos permite agrupar una serie de hosts con Docker para crear un cluster y gestionarlos de forma centralizada, imitando la orquestación de contenedores.

#### With Docker Swarm



### 5.13.2.1. Creación del cluster de Docker swarm

Para la creación de nuestro cluster, utilizaremos la máquina “manager” con el rol de manager dentro de éste. Para la instalación, debemos indicar en qué IP escucha y qué IP aceptará las uniones:

```
root@manager:~# docker swarm init \
> --listen-addr 172.16.133.200:2377 \
> --advertise-addr 172.16.133.200
Swarm initialized: current node (ym3fczk7gw18vywa1mzmvpvj) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-33k2eg1kajplgiagdm7v9y3w2ygrwd1ha0r2bbjat50vn3ks55-3n6j48a13ahteztd5h3adhkm5 172.16.133.200:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

root@manager:~#
```

Una vez inicializado el cluster, debemos unir las demás máquinas con el token que nos ha devuelto la consola a la hora de inicializarlo.

```
root@worker-01:~# docker swarm join --token SWMTKN-1-33k2eg1kajplgiagdm7v9y3w2ygrwd1ha0r2bbjat50vn3ks55-3n6j48a13ahteztd5h3adhkm5 172.16.133.200:2377
This node joined a swarm as a worker.
root@worker-01:~#

root@worker-02:~# docker swarm join --token SWMTKN-1-33k2eg1kajplgiagdm7v9y3w2ygrwd1ha0r2bbjat50vn3ks55-3n6j48a13ahteztd5h3adhkm5 172.16.133.200:2377
This node joined a swarm as a worker.
root@worker-02:~#
```

Dentro del nodo manager podremos observar las distintas máquinas que componen el cluster, así como el rol que tienen asignados y el estado en el que se encuentran.

```
root@manager:~# docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
ym3fczk7gw18vywa1mzmvpvj *	manager	Ready	Active	Leader	19.03.15
4ok5p79hiwhwf8sgoc44ur13p	worker-01	Ready	Active		19.03.15
c72jg1y6jgdzj8kn261ce79js	worker-02	Ready	Active		19.03.15

```
root@manager:~#
```

### 5.13.3 Servicios del cluster

Como hemos dicho antes, un orquestador nos permite ejecutar tareas y actividades de manera automatizada. Aquí es donde entran los servicios, que permiten designar un conjunto de tareas individuales que procesarán los contenedores independientes en el cluster.

Dentro de los servicios, podremos distinguir 2 tipos:

- Servicios replicados: replican los contenedores en un número definido, permitiendo escalar el servicio cambiando el número de réplicas.
- Servicios globales: cada nodo del cluster contendrá un contenedor del servicio. Si se añade un nuevo nodo al cluster, se le añadirá automáticamente el contenedor del servicio.



## 5.13.4 Creación de nuestro primer servicio

### 5.13.4.1. Preparando la imagen de los contenedores

Para crear nuestro primer servicio, debemos preparar la imagen del contenedor.

La imagen contendrá un servidor de Teamspeak (VoIP) que será configurado por environments en la creación del contenedor.

Hemos preparado la configuración del servicio para que obtenga los datos contra la base de datos interna de la infraestructura, separando así la parte de aplicación con la parte de datos.

La base de datos la hemos creado y preparado con anterioridad para la ejecución del servidor.

```
root@manager:~/teamspeak# cat start.sh
cd /teamspeak
touch /teamspeak/.ts3server_license_accepted

# declaracin de variables
sed -i "s/password=/password=`cat $TS3SERVER_DB_PASSWORD_FILE`/g" ts3db_mariadb.ini
sed -i "s/host=/host=$TS3SERVER_DB_HOST/g" ts3db_mariadb.ini
sed -i "s/port=/port=$TS3SERVER_DB_PORT/g" ts3db_mariadb.ini
sed -i "s/username=/username=$TS3SERVER_DB_USER/g" ts3db_mariadb.ini
sed -i "s/database=/database=$TS3SERVER_DB_NAME/g" ts3db_mariadb.ini

./ts3server_startscript.sh start inifile=/teamspeak/ts3server.ini
while :
do
    sleep 200
done
root@manager:~/teamspeak#
```

Cabe destacar que, concretamente, la variable de la password será un secret de Docker. Los secrets de Docker proporcionan a los contenedores los valores que necesitan, transmitiéndoselos de forma cifrada.

Dicho esto, crearemos nuestro secret para el servicio:

```
root@manager:~/teamspeak# printf %s $(cat /dev/urandom | tr -dc 'a-z0-9' | fold -n 32 | tr -d '\n') | docker secret create TS3SERVER_DB_PASSWORD -"
```

Después de crear el secret, que usaremos más adelante, lanzaremos una imagen base sobre la que configurar nuestro servidor. En nuestro caso, utilizaremos una imagen que viene preparada con todos los requisitos para lanzar un servidor de teamspeak:

```
root@manager:~# docker run -t -d --name teamspeak luzifer/teamspeak3 /bin/bash
Unable to find image 'luzifer/teamspeak3:latest' locally
latest: Pulling from luzifer/teamspeak3
bfde2ec33fbc: Pull complete
2cc4cf22e325: Pull complete
1b6ef701dd00: Pull complete
Digest: sha256:fd5a5c7e54aa587db88e94b380b792af612c60fd3cb18fbfe4dc0d8983fd921f
Status: Downloaded newer image for luzifer/teamspeak3:latest
d2f134669b277fe1ca3849df49386246a5ab52bab31f348956a26a0079f81680
root@manager:~#
```

Una vez creado el contenedor, debemos pasarle los de la aplicación para el servidor de teamspeak:



```
root@manager:~# docker cp /root/teamspeak/ teamspeak:/
root@manager:~# docker exec teamspeak ls /teamspeak
CHANGELOG
LICENSE
core
crashdumps
doc
files
libmariadb.so.2
libts3_ssh.so
libts3db_mariadb.so
libts3db_postgresql.so
libts3db_sqlite3.so
logs
query_ip_allowlist.txt
query_ip_denylist.txt
redis
serverquerydocs
sql
ssh_host_rsa_key
start.sh
ts3db_mariadb.ini
```

Ya con el contenedor preparado, lo que haremos será crear una imagen a partir del contenedor:

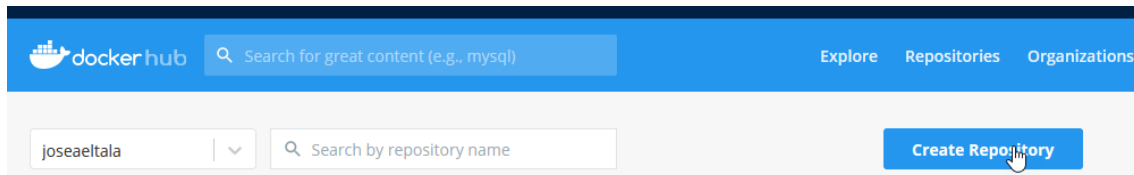
```
root@manager:~# docker commit teamspeak
sha256:4a71aa4055e17d673147069940737103b210ca667410d5a1c2633572ce9b6975
root@manager:~# docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<none>	<none>	4a71aa4055e1	4 seconds ago	265MB

Y renombraremos la imagen por nuestro nombre de usuario indicándole un repositorio para poder subir la imagen a Docker hub:

```
root@manager:~# docker tag 4a71aa4055e1 joseaeltala/ts3
root@manager:~#
```

En Docker hub (una vez registrados y logueados) en nuestro perfil, en el apartado de <repositorios>, encontraremos el botón para crear nuestro repositorio:



Crearemos nuestro repositorio, configurando las opciones de este a nuestro gusto y dándole un nombre:

## Create Repository

joseaeltala


ts3

Description

## Visibility

Using 0 of 1 private repositories. [Get more](#)

☒ **Public**   
Appears in Docker Hub search results

☐ **Private**   
Only visible to you

## Build Settings (optional)

Autobuild triggers a new build with every **git push** to your source code repository. [Learn More](#).

### Please re-link a GitHub or Bitbucket account



We've updated how Docker Hub connects to GitHub and Bitbucket. You'll need to re-link a GitHub or Bitbucket account to create new automated builds. [Learn More](#)

  
Disconnected

  
Disconnected

Cancel

Create

Create & Build

Una vez preparado nuestro repositorio en Docker hub, debemos loguearnos en la terminal con Docker login:

```
root@manager:~# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: joseaeltala
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
root@manager:~#
```

Y tras esto podremos subir nuestra imagen:


```
root@manager:~# docker push joseaeltala/ts3
The push refers to repository [docker.io/joseaeltala/ts3]
a0fc0d4a0d86: Pushed
c5de7d1d6de9: Mounted from joseaeltala/teamspeak
934a981ad6c0: Mounted from joseaeltala/teamspeak
2f4ee6a2e1b5: Mounted from joseaeltala/teamspeak
latest: digest: sha256:31f96abb658ef12c96bb4d5b4ceb8218db0a39e5b7b374c25a22bfe9cea1025f size: 1160
root@manager:~#
```



JOSE ÁNGEL AEL TALAVERA



Ahora podremos crear contenedores a partir de la imagen que hemos creado y configurado a nuestro gusto:



joseaeltala/ts3:latest

DIGEST: sha256:31f96abb658ef12c96bb4d5b4ceb8218db0a39e5b7b374c25a22bfe9cea1025f

OS/ARCH	COMPRESSED SIZE	LAST PUSHED
linux/amd64	76.39 MB	a few seconds ago by joseaeltala

Image Layers Vulnerabilities

IMAGE LAYERS ⓘ

1	ADD file ... in /	43.28 MB	<p>Command</p> <pre>ADD file:d9e4f6f4fc33703b766642a5642cabb2b01675bb55cf6050f2e91507577ff570 in /</pre>
2	CMD ["bash"]	0 B	
3	LABEL maintainer=Knut Ahlers <knut@ahlers.me>	0 B	
4	ENV TEAMSPEAK_VERSION=3.13.5 TEAMSPEAK_SHA256=dad497f...	0 B	
5	/bin/bash -exo pipefail -c #(nop)	0 B	
6	/bin/bash -exo pipefail -c apt-get	13.11 MB	
7	/bin/bash -exo pipefail -c #(nop)	721 B	
8	/bin/bash -exo pipefail -c #(nop)	0 B	
9	/bin/bash -exo pipefail -c #(nop)	0 B	
10	/bin/bash -exo pipefail -c #(nop)	0 B	
11	/bin/bash	20 MB	





### 5.13.4.2. Creación del servicio

Lo que debemos hacer ahora es crear nuestro servicio con las configuraciones que deseemos, en nuestro caso hemos preparado un “Docker compose”, lo que nos permitirá crear el servicio de una forma más cómoda a través de un fichero YAML. Estos ficheros son stacks, es decir, un grupo de servicios y dependencias interconectados que se ejecutan de forma conjunta.

En nuestro fichero podremos encontrar, principalmente, las environment para la configuración de nuestra imagen junto con el secret creado anteriormente.

Otro apartado importante es el montaje de volúmenes. En nuestro caso, como se puede observar en la siguiente imagen, montaremos un volumen en la ruta del contenedor “/teamspeak/files” para que todos los archivos que se suban en el servidor, sean guardados en el volumen montado por NFS de las máquinas, que apunta al RAID 1 de Rebel. De esta forma conseguimos separar la capa de aplicación con la capa de almacenamiento:

```
version: '3.2'
services:
  teamspeak:
    image: joseaeltala/ts3
    deploy:
      mode: replicated
      replicas: 1
    command: "/bin/bash /teamspeak/start.sh"
    volumes:
      - /mnt/teamspeak/files:/teamspeak/files
    ports:
      - published: 9987
        target: 9987
        protocol: udp
        mode: host
      - published: 10011
        target: 10011
        protocol: tcp
        mode: host
      - published: 30033
        target: 30033
        protocol: tcp
        mode: host
    secrets:
      - source: TS3SERVER_DB_PASSWORD
        target: /run/secrets/ts3user_db_password
        uid: '0'
        gid: '0'
        mode: 292

    environment:
      - TS3SERVER_DB_PLUGIN=ts3db_mariadb
      - TS3SERVER_DB_HOST=172.16.133.12
      - TS3SERVER_DB_PORT="3306"
      - TS3SERVER_DB_USER=ts3user
      - TS3SERVER_DB_PASSWORD_FILE=/run/secrets/ts3user_db_password
      - TS3SERVER_DB_NAME=teamspeak
      - TS3SERVER_LICENSE=accept
secrets:
  TS3SERVER_DB_PASSWORD:
    external: true
```



Una vez creado el fichero con nuestras configuraciones para el servicio, debemos de lanzar el stack:

```
root@manager:~# docker stack deploy -c teamspeak_service-stack.yml teamspeak
Creating service teamspeak_teamspeak
root@manager:~#
```

Una vez lanzado, podremos ver que el servicio está en ejecución en el modo que le hemos indicado:

```
root@manager:~# docker service ls
ID                NAME                MODE                REPLICAS            IMAGE                PORTS
agk60j8i6uhr     teamspeak_teamspeak replicated           1/1                 joseaeltala/ts3:latest *:10011->10011/tcp, *:30033->30033/tcp
root@manager:~#
```

Podremos obtener más información haciendo un `<Docker service ps>` al servicio concreto. De esta forma podremos saber datos como: si tiene algún error, en qué nodo se está ejecutando, en qué estado se encuentra, etc.

```
root@manager:~# docker service ps teamspeak_teamspeak
ID                NAME                IMAGE                NODE                DESIRED STATE       CURRENT STATE       ERROR                PORTS
woyoeqivvs4b     teamspeak_teamspeak.1 joseaeltala/ts3:latest worker-02            Running              Running 3 minutes ago
root@manager:~#
```



### 5.13.4.3. Accediendo al servicio desde fuera

Este servicio, concretamente, será ofrecido por un proxy inverso de Nginx, en lugar de usar HAProxy. Esto es debido a que teamspeak usa UDP en el protocolo de transporte, y HAProxy no soporta UDP.

Para realizar la configuración de Nginx deberemos habilitar el módulo de stream, para poder usar las directivas correspondientes.

Deberemos añadir la línea `load_module` al principio del archivo de configuración del servicio de Nginx `/etc/nginx/nginx.conf`

```
load_module /usr/lib/nginx/modules/ngx_stream_module.so;
user www-data;
worker_processes {{ ansible_processor_count }};
pid /run/nginx.pid;
```

Una vez habilitado el módulo podremos usar nuestra directiva `stream` para poder añadir frontends y backends en nuestra configuración.

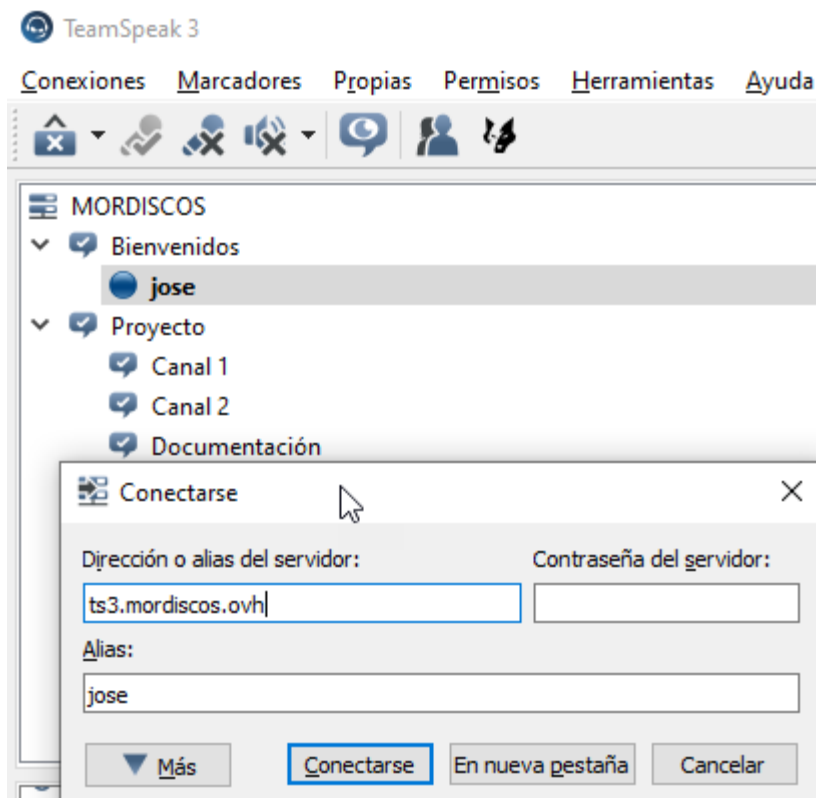
Debido a que Nginx solo lo vamos a usar para puertos UDP y el servicio de teamspeak usa diferentes puertos, tanto UDP como TCP, tendremos toda la configuración centralizada en Nginx.

```
stream {
    server {
        listen 9987 udp;
        listen [::]:9987 udp;
        proxy_pass ts3_stream_backend;
        proxy_timeout 60s;
        proxy_responses 0;
    }
    upstream ts3_stream_backend {
        server 172.16.133.200:9987;
        server 172.16.133.201:9987;
        server 172.16.133.202:9987;
    }
    #TS server file port
    server {
        listen 30033;
        proxy_pass ts3_stream_files;
    }
    upstream ts3_stream_files {
        server 172.16.133.200:30033;
        server 172.16.133.201:30033;
        server 172.16.133.202:30033;
    }
    server {
        listen 10011;
        proxy_pass ts3_stream_server;
    }
    upstream ts3_stream_server {
        server 172.16.133.200:10011;
        server 172.16.133.201:10011;
        server 172.16.133.202:10011;
    }
}
```

Dentro de la directiva stream, añadiremos la directiva server que será la encargada de configurar nuestro proxy inverso, mediante una serie de directivas como, por ejemplo, indicarle en qué puertos queremos que escuche, protocolo, etc.

Le indicaremos un proxy\_pass a un backend, en el cual le tenemos indicado los tres servidores para que detecte que servidor se encuentra activo, ya que solo puede haber uno activo a la vez, estos servidores hacen referencia a los nodos del cluster de Docker.

Una vez configurados los balanceadores, podremos acceder a nuestro servicio con el cliente de VOIP teamspeak3:



## 6. Conclusiones

Este proyecto nos ha aportado conocimiento sobre muchas tecnologías nuevas, las cuales han hecho posible el despliegue de una infraestructura que podría estar en producción en una empresa.

El uso de una infraestructura de este tipo (con alta disponibilidad) supone un mayor coste económico. No obstante, la caída de un sistema sin una alta disponibilidad, puede suponer grandes pérdidas económicas. En conclusión, la empresa u organización debe sopesar la necesidad de ofrecer sus servicios frente al coste que le supondría implementar y mantener la infraestructura.

Esta infraestructura mejoraría en un entorno de cloud computing, permitiendo tener todas las instancias en zonas geográficas distintas, eliminando así puntos únicos de fallo, como pueden ser: caídas de red, cortes de luz, etc. También mejoraría la escalabilidad aplicando medidas como el autoscaling, añadiendo y eliminando instancias para el balanceo de carga, de forma automática según la necesidad en el momento.

En general, podemos decir que ha sido todo un reto el enfrentarnos a estas nuevas tecnologías.

Nos agrada compartir nuestros conocimientos adquiridos durante el proyecto y esperamos que sirva de aporte para la iniciación en este tipo de infraestructuras.



## 7. Bibliografía

- Página oficial de Ansible.  
<https://www.ansible.com>
- Página oficial de Ansible Galaxy.  
<https://galaxy.ansible.com>
- Página de la documentación oficial de Proxmox.  
<https://pve.proxmox.com/pve-docs/>
- Página de la documentación oficial de Docker.  
<https://docs.docker.com>
- Página de la documentación oficial de Heartbeat.  
[http://www.linux-ha.org/wiki/Main\\_Page](http://www.linux-ha.org/wiki/Main_Page)
- Página de la documentación oficial de Nginx.  
<https://nginx.org/en/docs/>
- Página de la documentación oficial de HAProxy.  
<http://www.haproxy.org>
- Página de la documentación oficial de NTP.  
<http://www.ntp.org/documentation.html>
- Página de la documentación oficial de Certbot.  
<https://certbot.eff.org/docs/>
- Página de la documentación oficial de MySQL.  
<https://dev.mysql.com/doc/>
- Página de la documentación oficial de Bind9.  
<https://wiki.debian.org/Bind9>
- Página de documentación sobre NFS.  
[https://wiki.linux-nfs.org/wiki/index.php/Nfsv4\\_configuration](https://wiki.linux-nfs.org/wiki/index.php/Nfsv4_configuration)
- Sin olvidarnos del mejor amigo de los informáticos.  
<https://www.google.com>

