
**Unibersitat Oberta de Catalunya**

Grado en Ingeniería Informática

# Configuración y puesta en marcha de un sistema de virtualización en alta disponibilidad vía Proxmox

*Proyecto Fin de Carrera*

Autor:
Mikel Arocena Errazquin

Tutores:
Iñigo Aldazabal Mensa, Joaquín López Sánchez-Montañes

15 de enero de 2023

Tabla 1: Resumen del proyecto

| | |
|---|---|
| **Título del trabajo** | Configuración y puesta de marcha de un sistema de virtualización en alta disponibilidad vía Proxmox |
| **Nombre del autor** | Mikel Arocena Errazquin |
| **Nombre del consultor/a** | Joaquín López Sánchez-Montañes<br>Iñigo Aldazabal Mensa |
| **Nombre del PRA** | Montse Serra Vizern |
| **Fecha de entrega (mm/aaaa)** | 01-2023 |
| **Titulación** | Grado en Ingeniería Informática |
| **Área del Trabajo Final** | Administración de Redes y Sistemas Operativos |
| **Idioma del trabajo** | Castellano |
| **Palabras clave** | Virtualización, Backup, Autogestión, Monitorización |

**Resumen del Trabajo (máximo 250 palabras)**: con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo

Haciendo uso de dos servidores físicos, se pretende montar una sistema de virtualización en alta disponibilidad y con almacenamiento sincronizado.

Para la alta disponibilidad, hacen falta como mínimo tres servidores, por tanto, además de los dos servidores mencionados anteriormente, el clúster dispondrá de un tercer nodo, correspondiente a una máquina virtual (MV) alojada en un servidor externo y cuya función va ser única y exclusivamente la de testigo, cumpliendo de esta forma la leyes del quórum.

Por otro lado, se llevará a cabo un proceso de migración de MVs desde una plataforma de virtualización en la nube a la nueva infraestructura. La configuración y mantenimiento de las diferentes MVs alojadas en el clúster se va a realizar a través de la herramienta de automatización TI.

Finalmente, se implementará un servicio de monitorización en el sistema con el fin de supervisar la infraestructura e identificar posibles problemas.

**Abstract (in English, 250 words or less)**:

Using two physical servers, it is intended to set up a virtualization system with high availability and synchronized storage.

For high availability, at least three servers are required, therefore, in addition to the two servers mentioned above, the cluster will have a third node, corresponding to a virtual machine (VM) hosted on an external server and whose function will be solely and exclusively that of a witness, thus fulfilling the laws of quorum.

On the other hand, a migration process of VMs from a virtualization platform in the cloud to the new infrastructure will be carried out. The configuration and maintenance of the different VMs hosted in the cluster will be carried out through the IT automation tool.

Finally, a monitoring service will be implemented in the system in order to supervise the infrastructure and identify possible problems.

# Índice general

# Capítulo 0

# INTRODUCCIÓN

## 0.1. Contexto y justificación del Trabajo

A través de la tecnología de virtualización, se pueden ejecutar múltiples sistemas en una sola máquina física, de esta forma se utilizan mejor los recursos y mejora la eficiencia. La virtualización también proporciona una gran flexibilidad para crear un entorno de prueba: se logra un grado de aislamiento, lo que mejora la seguridad ya que es un entorno de producción independiente. La agrupación de las máquinas virtuales, de aquí en adelante MVs, en diferentes servidores le brinda al sistema escalabilidad, tolerancia a fallos y más.

Gestionar cada una de las MVs alojadas en el sistema individualmente puede ser una tarea tediosa, es por eso que, en la actualidad, hay disponibles diferentes herramientas que permiten la gestión y administración del entorno de forma automática. Algunas de las ventajas que proporciona el uso de este tipo de aplicaciones son: mayor fiabilidad, reducción de costes y aumento de la productividad.

Además, a parte de los programas de gestión y automatización, es habitual disponer de un software especializado para la monitorización de los servidores, fundamental para obtener una visión general de los mismos y actuar más rápidamente en caso de errores.

## 0.2. Objetivos del trabajo

El trabajo se divide, principalmente, en tres bloques:

I Virtualización

- I Estudio de los diferentes entornos de virtualización más utilizados en la actualidad. Se escogerá el más adecuado para configurar un sistema de virtualización en alta disponibilidad y con almacenamiento sincronizado acorde a los requerimientos deseados y al hardware disponible.
- II Configuración y puesta en marcha del sistema de virtualización seleccionado. Creación de máquinas virtuales de prueba. Documentación de procedimientos habituales, pruebas de carga y resiliencia.
- III Análisis de diferentes métodos de realización de copias de seguridad específicas para el sistema de virtualización elegido. Implementación, realización de pruebas y redacción de procedimientos de recuperación en caso de incidentes.

II Gestión y automatización

- I Estudio de los principales sistemas de gestión de la configuración con el fin de administrar las máquinas virtuales alojadas en el sistema. Selección del más adecuado en función de los requerimientos definidos.
- II Migración de las máquinas virtuales existentes en la plataforma actual al entorno de virtualización seleccionado. Pruebas de carga en ambos sistemas sobre las máquinas virtuales reales y comparación del rendimiento.

III Monitorización

- I Estudio de los sistemas de monitorización más habituales e implementación del seleccionado. Ajuste del sistema, realización de pruebas de notificaciones y documentación de los procedimientos más habituales.

## 0.3. Enfoque y método seguido

Para este trabajo se ha optado por una metodología en cascada, o *waterfall*, donde las diferentes tareas y bloques se completarán de manera secuencial. Cada bloque consta de las siguientes etapas:

- Análisis
- Implementación
- Testeo
- Despliegue

Este tipo de metodología tiene una serie de aspectos positivos como su simplicidad y facilidad de aplicación, las diferentes fases quedan definidas de forma clara, de modo que gestionar cada una de estas se vuelve más sencillo.

## 0.4. Planificación del Trabajo

En este apartado se muestra el desglose general de los diferentes bloques definidos en el apartado 0.2, junto con una estimación preliminar del número de horas destinadas a las mismas.

Tabla 1: Horas estimadas para los diferentes bloques

| **BLOQUE I** | **BLOQUE II** | **BLOQUE III** | **HORAS** |
|---|---|---|---|
| Análisis | Análisis | Análisis | 60 |
| Instalación | | | 5 |
| Configuración | | | 110 |
| Testeo | | | 15 |
| | Instalación | | 5 |
| | Configuración | | 55 |
| | Testeo | | 10 |
| | | Instalación | 5 |
| | | Configuración | 30 |
| | | Testeo | 5 |
| | Horas totales | | 300 |

El Bloque I, Bloque II y Bloque III hacen referencia a las etapas de virtualización, autogestión, y monitorización, respectivamente. Estas pueden ser representadas mediante un diagrama de Gantt de la siguiente forma:

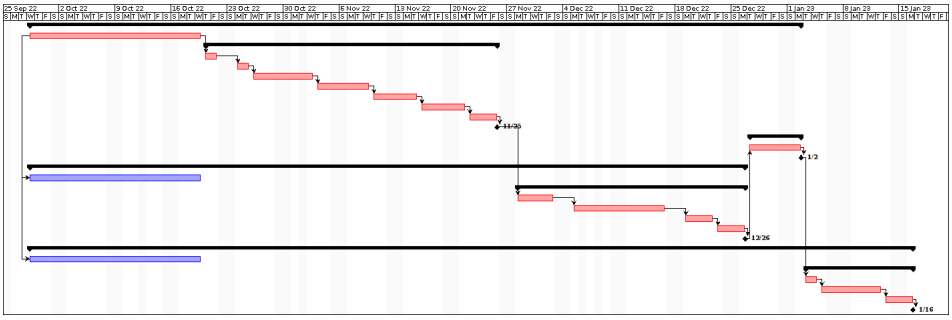

Figura 1: Diagrama de Gantt del proyecto

Tal y como se puede observar en la figura anterior, hay una tarea del Bloque I que se realiza finalizado el bloque II. Y es que en Proxmox, se configurarán tareas periódicas de *backups* una vez se hayan importado las correspondientes MVs.

## 0.5. Productos obtenidos

Finalizados los tres bloques mencionados en el apartado anterior, el producto resultante es un sistema de virtualización en alta disponibilidad formado por una máquina virtual y dos servidores físicos, con sincronización de datos entre estos dos últimos y, con copias de seguridad diarias de las máquinas virtuales alojadas en los mismos, las cuales se almacenarán en un servidor externo. El proceso de configuración y mantenimiento de las máquinas virtuales estará automatizado. El sistema contará con una herramienta de monitorización adicional para la detección temprana de errores.

## 0.6. Descripción de los otros capítulos de la memoria

Los capítulos 1, 2 y 3 están enfocados en la virtualización, autogéstión y monitorización de servidores, respectivamente. Cada uno de estos temas está dividido en: contexto histórico, concepto general, herramientas, beneficios e implementación.

Los capítulos finales hacen referencia a los resultados y las conclusiones, así como la bibliografía y posibles anexos que puedan ser de interés.

# Capítulo 1

# VIRTUALIZACIÓN

## 1.1. Historia

El concepto de virtualización fue desarrollado originalmente por IBM en la década de 1960 para particionar lógicamente hardware de las computadoras centrales en máquinas virtuales. Estas particiones permiten que las computadoras grandes ejecuten múltiples procesos y aplicaciones simultáneamente. En aquellos días, las computadoras centrales eran muy caras, por eso intentaron encontrar nuevas alternativas para usar los recursos de la manera más eficiente posible.

Durante las décadas de 1980 y 1990, los ordenadores de sobremesa y servidores x86 empezaron a ser más asequibles, de modo que la tecnología de virtualización pasó a un segundo plano y su uso disminuyó gradualmente. Las aplicaciones cliente-servidor y los sistemas operativos Windows y Linux hicieron que la computación sobre servidores fuera bastante más económica. Aun así, esto creó nuevos desafíos, como altos costes de mantenimiento y administración, altos costes de infraestructura y protección inadecuada contra desastres y fallos. Debido a estos retos, la tecnología de la virtualización empezó a retomarse de nuevo con la introducción de la virtualización para plataformas x86, con la que mejoró la eficiencia y disminuyeron los costes.

VMware fue pionera en desarrollar la virtualización para la arquitectura x86, en 1998, introdujo una solución de virtualización la cual transformó el sistema x86 en una infraestructura hardware compartida totalmente aislada. Fue en la década de los 2000, y gracias a los avances en la tecnología de virtualización, cuando empezaron a aparecer nuevas herramientas de código libre como Xen, VirtualBox, etc.

## 1.2. Concepto general

La virtualización es la abstracción de los recursos de una computadora mediante la creación de una capa intermedia, comúnmente conocida como hipervisor, entre el hardware de la máquina física (*host*) y la máquina virtual (*guest* o MV). Esta capa de software intermedio maneja, gestiona y arbitra los cuatro recursos principales de una computadora (CPU, Memoria, Dispositivos Periféricos y Conexiones de Red) para poder repartir dinámicamente dichos recursos entre todas las máquinas virtuales definidas en el *host*. Actualmente, se pueden distinguir dos tipos diferentes de hipervisor:

- Bare-metal (Tipo I): este tipo de hipervisores ponen en marcha las máquinas virtuales directamente sobre el hardware del sistema anfitrión.
- Hosted (Tipo II): este tipo de hipervisores se ejecutan sobre un sistema operativo convencional como una capa de software o una aplicación.

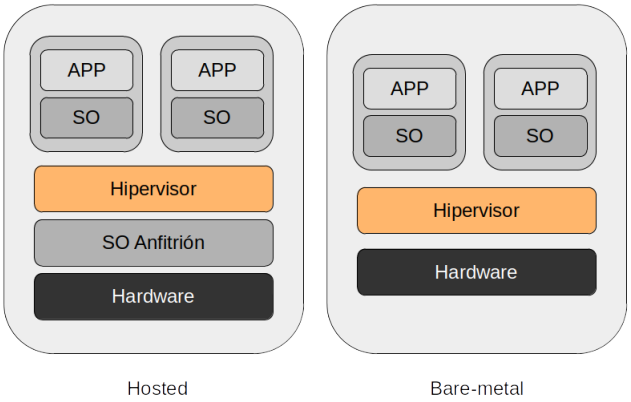


Figura 1.1: Hipervisor tipo I (bare-metal) VS tipo II (hosted)

Hoy en día el tipo de virtualización que más se emplea es la virtualización de hardware, y es en el que se va a basar el trabajo. Existen tres tipos:

**Virtualización completa.** En la virtualización completa, los sistemas operativos invitados se ejecutan en el SO anfitrión de forma aislada utilizando la ejecución directa y la traducción binaria. Uno de los principales inconvenientes de la virtualización completa es que las continuas traducciones entre los recursos físicos y virtuales, como la memoria y el procesador, pueden afectar al rendimiento del sistema.

**Paravirtualización.** La principal diferencia entre la virtualización completa y la paravirtualización es que cada sistema operativo del servidor es consciente de la presencia de los demás. Esto permite que toda una red trabaje conjuntamente para gestionar los recursos. La ventaja más significativa de la paravirtualización es que el hipervisor no necesita tantos recursos para funcionar porque los diferentes servidores virtuales son conscientes unos de otros y por lo tanto pueden compartir los recursos físicos de manera más eficiente.

**Virtualzación por S.O o semi-parcial** La forma más básica de virtualización de servidores es la virtualización a nivel de sistema operativo. Hay limitaciones cuando se utiliza este método, pero se puede implementar y mantener con menos recursos que la paravirtualización o la virtualización completa. Cuando se utiliza la virtualización a nivel de SO, no se necesita un hipervisor. La tarea de gestionar los recursos y separar las máquinas virtuales corre a cargo del sistema operativo del servidor físico. El inconveniente de esta solución es que cada máquina virtual tendrá que ejecutar el mismo sistema operativo, porque el SO está actuando como hipervisor.

## 1.3. Diferentes herramientas para la virtualización

El sector de la virtualización está repleto de productos de pago y de código abierto que ofrecen a los administradores de TI una gran variedad de opciones para desplegar sus entornos virtuales. A continuación se presentan los más populares a día de hoy.

### 1.3.1. Proxmox VE

Proxmox Virtual Environment, o Proxmox VE, es una completa plataforma de virtualización de servidores basada en la distribución Debian GNU/Linux. Proxmox VE es un sistema operativo libre y de código abierto y es conocido por su capacidad de gestionar tanto KVM como LXC en una única plataforma unificada. Al incorporar tanto KVM como LXC en su plataforma, Proxmox VE puede desplegar una amplia gama de casos de uso.

Según la documentación de Proxmox VE, la plataforma admite las cargas de trabajo de aplicaciones Linux y Windows más exigentes, al tiempo que ofrece rendimiento y alta disponibilidad (HA). Por ejemplo, los administradores pueden ampliar los recursos informáticos y de almacenamiento a medida que cambian sus necesidades, comenzando con un solo nodo y ampliando a un gran clúster para dar cabida a las crecientes cargas de trabajo.

La pila de clústeres está totalmente integrada en Proxmox VE y forma parte de la instalación por defecto. Los administradores pueden utilizar Proxmox VE tanto para la computación en la nube como para consolidar

los recursos de los servidores, todo ello soportando aplicaciones basadas en VMs y contenedores. Los administradores también pueden utilizar Proxmox VE para instalar, gestionar y supervisar una infraestructura hiperconvergente, lo que aumenta la versatilidad de los casos de uso del producto. Otra función que ofrece Proxmox VE son las migraciones en vivo, que permiten a los administradores trasladar las máquinas virtuales en ejecución de un nodo del clúster a otro sin interrupciones perceptibles.

Proxmox VE también incluye otras características que lo hacen adecuado para una variedad de usos. Por ejemplo, Proxmox VE puede trabajar con almacenamiento local o compartido, admitiendo configuraciones como el almacenamiento conectado directamente, el almacenamiento conectado a la red, las redes de área de almacenamiento y el almacenamiento distribuido Ceph. Proxmox VE utiliza puentes de red basados en software, que pueden manejar hasta 4.094 puentes por host. También permite el uso de LANs virtuales, y la unión y agregación de redes.

### 1.3.2. VMware ESXi

VMware ESXi, también llamado VMware ESXi Server, es un hipervisor *bare-metal* desarrollado por VMware para vSphere. ESXi es uno de los principales componentes de la suite de software de infraestructura de VMware.

ESXi es un hipervisor de tipo I, lo que significa que se ejecuta directamente en el hardware del sistema sin necesidad de un sistema operativo. ESXi está dirigido a empresas, pequeñas organizaciones y particulares. El VMkernel de ESXi interactúa directamente con los agentes de VMware y los módulos aprobados de terceros. Los administradores pueden configurar VMware ESXi mediante un cliente vSphere y actualizarlo con el complemento Lifecycle Manager. ESXi se instala directamente en un disco local de la máquina anfitriona. Cuando un VMkernel recibe una solicitud de recursos, el kernel envía la solicitud al hardware físico del host.

### 1.3.3. XCP-ng

Xen Cloud Platform es un producto de virtualización de código abierto que proporciona capacidades tanto de virtualización como de computación en la nube. Xen Cloud Platform incluye la gestión del ciclo de vida de las máquinas virtuales, grupos de recursos, seguimiento de eventos, compatibilidad con Open vSwitch, supervisión del rendimiento en tiempo real y Storage XenMotion.

Al profundizar en las características de XCP, la primera pieza que hay que mirar es el hipervisor del proyecto Xen, que es un hipervisor de tipo I en el núcleo de la pila de virtualización. El hipervisor contiene todas las características que los administradores esperarían, como la adición en caliente de CPU y RAM, la gestión de la memoria, la alta disponibilidad y

la tolerancia a fallos, aunque la HA y la tolerancia a fallos todavía están en fase experimental. El hipervisor también tiene la capacidad de soportar controladores de paravirtualización para aquellas aplicaciones que requieran soporte adicional.

Si se utiliza XenServer o el hipervisor de código abierto basado en este último, es decir, XCP-ng, hay una solución que puede facilitar mucho la administración y gestión de la plataforma. Xen Orchestra es una interfaz que proporciona todas las herramientas necesarias para gobernar la infraestructura de forma eficiente y sencilla. Además, Xen Orchestra incluye herramientas para realizar backup completo o replicación continua para asegurar la infraestructura.

## 1.4. Razones para utilizar la virtualización

La virtualización puede ser útil en el ámbito de servidores o de escritorios individuales. La virtualización de escritorio ofrece gestión centralizada de coste y eficiencia, y una mejor recuperación de desastres.

Cuando se utiliza en servidores, la virtualización puede beneficiar no solo en grandes redes, también en entornos con más de un servidor. La virtualización ofrece migración en vivo, alta disponibilidad, tolerancia de fallos y copias de seguridad programadas.

A continuación se presentan, más detalladamente, los beneficios de la virtualización.

**Menos recursos.** Utilizando la virtualización, ahorramos la necesidad de múltiples plataformas físicas. Esto significa que las máquinas consumirán menos electricidad y necesitarán menos refrigeración, por lo tanto el coste de energia se verá reducido. El coste inicial de comprar varias plataformas físicas, combinado con el consumo de recursos de las máquinas virtuales y la refrigeración, se reduce drásticamente utilizando la virtualización.

**Menos mantenimiento.** Siempre que se realice la planificación adecuada antes de migrar los sistemas físicos a los virtualizados, se necesitará menos tiempo para mantenerlos. Esto significa menos dinero invertido en trabajos de mantenimiento.

**Más tiempo de vida para el software instalado.** Puede que las versiones anteriores de los softwares no arranquen directamente en máquinas modernas, así que se pueden crear nuevos entornos con máquinas que sean compatibles con dichos softwares.

**Menos espacio.** La consolidación de servidores en menos máquinas significa que se requiere menos espacio físico para los sistemas informáticos.

**Mejora en los procesos de clonación y copia de sistemas.** Mayor facilidad para la creación de entornos de pruebas, que permiten poner en marcha nuevas aplicaciones sin tener ningún impacto en el entorno de producción, por lo que proporciona flexibilidad y agilidad en el proceso de pruebas.

**Mejora en la utilización del hardware.** Mediante la virtualización se aprovecha al máximo el uso del hardware de los sistemas en los que está en funcionamiento.

**Alta disponibilidad.** Instalando un clúster dedicado a servidores virtualizados, se obtiene un sistema tolerante a fallos y de alta disponibilidad.

## 1.5. Virtualización con Proxmox

Entre las diferentes herramientas descritas en el apartado 1.3, se ha optado por utilizar Proxmox VE por las siguientes razones:

- Nació en 2008, de modo que es un producto maduro
- Está basado en Linux y es *open-source*
- Soporta tanto MVs como contenedores
- Dispone de una alta gama de funcionalidades
- No necesita hardware específico para la alta disponibilidad
- Cumple con los estándares técnicos
- Dispone de una gran comunidad y soporte

### 1.5.1. Instalación

Se dispone de dos servidores idénticos con las siguientes caraterísticas:

Tabla 1.1: Carácteristicas del servidor

| | |
|---|---|
| **Procesador** | 2 procesadores AMD de 12 cores cada uno (Abu Dhabi 6344) |
| **Memoria RAM** | 48GB |
| **Unidades de disco duro para el sistema operativo** | 2x HDD de 260GB |
| **Unidades de disco dedicados a la virtualización** | 6x SSD de 512GB |
| **Compatibilidad con RAID** | RAID-0, RAID-1, RAID-5, RAID 6 |
| **Red** | Tarjeta de red 10Gb SFP+ para interconexión directa de los equipos para la replicación de datos |
| **Control Remoto** | Interfaz IPMI con Lan dedicada |

De manera adicional, se desplegará un MV para cumplir con las reglas del quórum, término que se explica más adelante, y poder habilitar de esta forma la alta disponibilidad o HA dentro del clúster. En el Anexo 6.1 se explica paso a paso como proceder con la instalación básica de Proxmox.

### 1.5.2. Configuración

#### Clúster

Un clúster es un grupo de múltiples ordenadores conectados mediante una red que trabajan coordinádamente. Los clúster permiten aumentar la escalabilidad, disponibilidad y fiabilidad.

Un clúster puede ser de gran interés en el ámbito empresarial, ya que pueden aprovechar las características mencionadas anteriormente para poder mantener sus equipos actualizados por un precio más económico que el que les supondría actualizar todos sus equipos informáticos, además, tendrían más capacidad de computación y disponibilidad. Un clúster está formado por diversos componentes de hardware y software.

**Nodo.** Cada una de las máquinas que componen el clúster. Pueden ser desde simples ordenadores personales a servidores dedicados conectados por una red. La regla general es que los nodos deben tener características similares como: arquitectura, componentes, sistema operativo, etc.

**Sistemas operativos.** Se utilizan sistemas operativos de tipo servidor con características de multiproceso y multiusuario, así como capacidad para abstracción de dispositivos y trabajo con interfaces IP virtuales.

**Middleware.** Es el software que actúa entre el sistema operativo y los servicios o aplicaciones finales. El middleware recibe los trabajos entrantes del clúster y los redistribuye de manera que el proceso se ejecute más rapido y el sistema no sufra sobrecargas en un servidor.

**Conexión de red.** Los nodos del clúster se pueden conectar con una simple red Fast Ethernet o utilizar tecnologías de red más avanzadas como Gigabit Ethernet, Infiniband, SCI, etc.

**Protocolos de comunicación.** Definen la intercomunicación entre los nodos del clúster.

**Sistema de almacenamiento.** El almacenamiento puede ir desde sistemas comunes de almacenamiento interno del servidor hasta redes de almacenamiento compartido como NAS o SAN.

**Servicios y aplicaciones.** Servicios y aplicaciones a ejecutar sobre el clúster.

En función de las características, se pueden diferenciar tres tipos diferentes de clúster:

- Clúster de alta disponibilidad (HA, High Availability): tiene como propósito principal proporcionar la máxima disponibilidad de los servicios que ofrece. Esto se consigue mediante software que monitoriza constantemente el clúster, detecta fallos y permite recuperarse frente a ellos.

- Clúster de alto rendimiento (HP, High Performance): se utiliza para ejecutar programas paralelizables que requieren una gran capacidad computacional. Se utilizan normalmente en la comunidad científica o industrias que tengan que resolver problemas complejos o simulaciones.
- Clúster de balanceo de carga (LB, Load Balancing): este tipo de clúster permite distribuir las peticiones de servicio entrantes hacia un conjunto de equipos que las procesa. Se utiliza principalmente para servicios de red sin estado, como un servidor web o un servidor de correo electrónico, con altas cargas de trabajo y de tráfico de red.

Dado que lo que se quiere lograr es que las MVs desplegadas en el clúster estén operativas la mayor parte del tiempo, para este proyecto se ha optado por un clúster de alta disponibilidad.

Para crear un clúster en Proxmox, en uno de los servidores, hay que dirigirse a *Datacenter* > *Cluster* y pulsar sobre *Create Cluster*. Se abrirá una ventana donde habrá que definir el nombre del clúster, de aquí en adelante VIRT, y su correspondiente red.

```
Cluster information
-------------------
Name:             VIRT
Transport:        knet
Secure auth:      on

Quorum information
------------------
Date:             Wed Dec 21 10:17:00 2022
Quorum provider:  corosync_votequorum
Nodes:            3
Quorate:          Yes

Votequorum information
----------------------
Expected votes:   3
Highest expected: 3
Total votes:      3
Quorum:           2
Flags:            Quorate

Membership information
----------------------
    Nodeid      Votes Name
0x00000001          1 158.227.180.235
0x00000002          1 158.227.181.71
0x00000003          1 158.227.181.207
```

Figura 1.2: Resumen del clúster VIRT

### Almacenamiento distribuido/sincronizado

Un sistema de almacenamiento distribuido es una infraestructura que puede repartir los datos entre varios servidores físicos y, a menudo, entre más de un centro de datos. Suele adoptar la forma de un clúster de unidades de almacenamiento, con un mecanismo de sincronización y coordinación de datos entre los nodos del clúster. Este sistema tiene las siguiente ventajas:

- Escalabilidad: la principal motivación para distribuir el almacenamiento es escalar horizontalmente, añadiendo más espacio de almacenamiento al clúster.
- Redundancia: los sistemas de almacenamiento distribuido pueden almacenar más de una copia de los mismos datos, con fines de alta disponibilidad, copia de seguridad y recuperación de desastres.
- Coste: el almacenamiento distribuido permite utilizar hardware básico y más barato para almacenar grandes volúmenes de datos a bajo coste.
- Rendimiento: el almacenamiento distribuido puede ofrecer un mayor rendimiento que un único servidor en algunos casos. Por ejemplo, puede almacenar datos más cerca de sus consumidores o permitir el acceso masivo en paralelo a archivos de gran tamaño.

Una limitación inherente a los sistemas de almacenamiento distribuido es que estos sistemas vienen definidos por el teorema de CAP. El teorema de establece que un sistema distribuido no puede mantener la consistencia, la disponibilidad y la tolerancia a la partición (capacidad de recuperarse ante un fallo de partición que contiene parte de los datos), se tiene que renunciar, al menos, a una de estas tres propiedades.

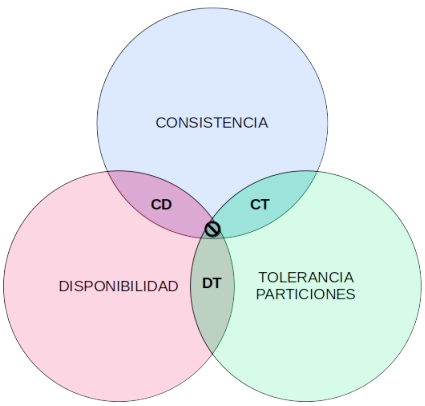


Figura 1.3: Teorema de CAP

Con la integración de Ceph, una plataforma de almacenamiento definido por software de código abierto, Proxmox VE tiene la capacidad de ejecutar y gestionar el almacenamiento directamente en los nodos del hipervisor. Ceph es un almacén de objetos distribuido y un sistema de archivos diseñado para ofrecer un rendimiento, una fiabilidad y una escalabilidad excelentes. Algunas ventajas de Ceph en Proxmox VE son:

- Fácil configuración y gestión mediante CLI y GUI
- *Thin provisioning*
- Compatibilidad con snapshots
- *Self-healing*
- Escalabilidad hasta los exabytes
- Configuración de grupos con diferentes carácteristicas de rendimiento y redundancia
- Replicación de datos
- Funciona con hardware básicos
- No necesita controladores RAID
- Código abierto

Para crear un clúster hiperconvergente con Proxmox y Ceph, se debe utilizar al menos tres servidores, preferiblemente idénticos, para la configuración. Además, es recomendable tener un ancho de banda de, al menos, 10Gb.

Puesto que el clúster VIRT no cumple con las especificaciones recomendadas para crear un sistema de almacenamiento distribuido vía Ceph, se ha optado por una solución alternativa que pasa por crear una partición ZFS idéntica en cada servidor, y configurar tareas de replicación periódicas entre los mismos. A esta práctica se le conoce como almacenamiento sincronizado.

ZFS es un sistema de archivos local y gestor de volúmenes lógicos creado por Sun Microsystems Inc. para dirigir y controlar la ubicación, el almacenamiento y la recuperación de datos en sistemas informáticos de clase empresarial. El sistema de archivos y gestor de volúmenes ZFS se caracteriza por la integridad de los datos, una gran escalabilidad y funciones de almacenamiento integradas como: replicación, deduplicación, compresión, *snapshots*, clones, protección de datos, etc.

Para crear un volúmen ZFS en Proxmox, basta con dirigirse a *Disks* > *ZFS* dentro del nodo y hacer clic sobre *Create: ZFS*. En este caso, teniendo en cuenta que se disponen de 6 discos SSD de 512GB de capacidad, se ha optado por una configuración en RAID-10 debido a su gran rendimiento en

cuanto a lectura y escritura se refiere. A cambio, hay que sacrificar el 50 % de la capacidad total.

Tal y como se puede ver a continuación, dos de los discos estarán configurados como *hot spares* para brindar seguridad al sistema.

```
pool: zfs-pool
 state: ONLINE
  scan: scrub repaired 0B in 00:00:08 with 0 errors on Sun Dec
      11 00:24:09 2022
config:

        NAME                              STATE  READ WRITE CKSUM
        zfs-pool                          ONLINE    0     0     0
          mirror-0                        ONLINE    0     0     0
            ata-Samsung_SSD_SSD_850       ONLINE    0     0     0
            ata-Samsung_SSD_SSD_850       ONLINE    0     0     0
          mirror-1                        ONLINE    0     0     0
            ata-Samsung_SSD_SSD_850       ONLINE    0     0     0
            ata-Samsung_SSD_SSD_850       ONLINE    0     0     0
        spares
          sde                             AVAIL
          sdf                             AVAIL
```

Figura 1.4: Configuración de la partición ZFS

La replicación de datos entre los dos servidores físicos se realizará a través de una red interna (10.10.10.0/24) previamente configurada en *System* > *Network*.

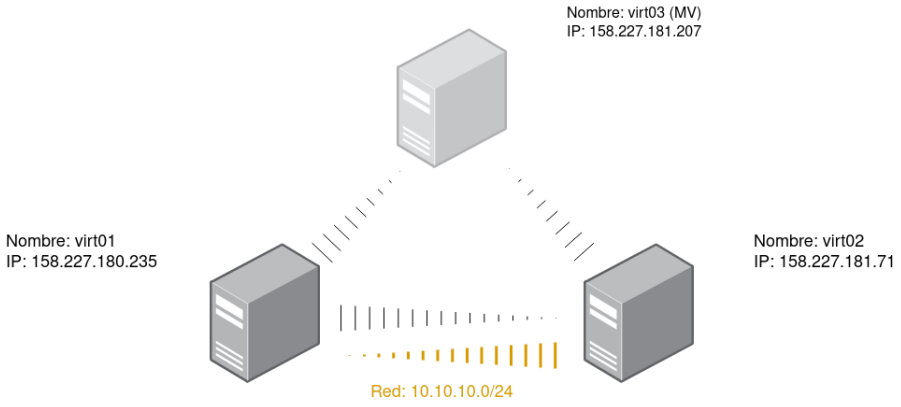


Figura 1.5: Subred para replicación de datos entre virt01 y virt02

En Proxmox, basta con dirigirse a Replication dentro del nodo y hacer clic sobre Add. Aparecerá un panel donde se deberá escoger la MV a replicar, el nodo destino y la peroicidad, entre otras cosas.

### Alta disponibilidad

Cuando se habla de alta disponibilidad (High Availability o HA), se hace referencia a un protocolo de diseño del sistema y su implementación asociada que asegura un determinado grado de continuidad operacional durante un periodo de medición dado. Disponibilidad se refiere a la capacidad de la comunidad de usuarios para acceder al sistema, utilizar sus servicios, lanzar nuevos trabajos, actualizar o alterar trabajos existentes o recoger los resultados de trabajos previos. Si un usuario no puede acceder al sistema, se dice que no está disponible. El término tiempo de inactividad, también conocido como *downtime*, se utiliza cuando el sistema no está disponible.

Se puede diferenciar entre tiempo de inactividad planificado (aquel que es imprescindible por actualizaciones del sistema, configuraciones y reinicios) y el tiempo de inactividad no planificado que surgen a causa de algún evento, tales como fallos en el hardware o anomalías ambientales. Ejemplos de eventos con tiempos de inactividad no planificados incluyen: fallos de potencia, fallos en los componentes de CPU o RAM, una caída por sobrecalentamiento, una ruptura lógica o física en las conexiones de red, rupturas de seguridad catastróficas o fallos en el sistema operativo, aplicaciones y middleware. Varias veces, se excluye el tiempo de inactividad planificado de los cálculos de disponibilidad, de modo que muchos sistemas pueden reclamar tener alta disponibilidad, lo cual da la ilusión de disponibilidad continua. Los sistemas que exhiben disponibilidad continua son caros, y tienen diseños cuidadosamente implementados que eliminan cualquier punto de fallo y permiten que el hardware, la red, el sistema operativo, middleware y actualización de aplicaciones, parches y reemplazos se hagan en caliente.

Por otro lado, la disponibilidad es usualmente expresada como un porcentaje del tiempo de funcionamiento en un año dado. Los valores comunes de disponibilidad se conocen como número de «nueves»:

$$disponibilidad = (t.disponible - t.inactivo)/t.disponible$$

```
t.disponible = 365 dias*24 horas/dia = 525600 minutos
t.inactivo = x

Para disponibilidad del 99%:
0.99 =  (525600 - x) / 525600
x = 5256 min
```

Siguiendo la fórmula, la tabla de alta disponibilidad quedaría de la siguiente manera:

Tabla 1.2: Niveles de alta disponibilidad

| **Disponibilidad** | **Interrupción Anual** | **Interrupción Semanal** |
|---|---|---|
| 98 % | 175.2 horas | 3.36 horas |
| 99 % | 87.6 horas | 1.68 horas |
| 99.9 % | 8.76 horas | 10.10 minutos |
| 99.99 % | 0.87 horas | 1.01 minutos |
| 99.999 % | 5.25 minutos | 6.06 segundos |

Se considera HA dinámica a todas las reconfiguraciones del clúster que garanticen la máxima disponibilidad de los servicios. Esta dinámica está orientada a los nodos integrantes del clúster y la forma en la cual el clúster responde. A continuación, se definen algunos conceptos importantes, y que se tendrán que tener en cuenta, en relación a la alta disponibilidad.

**Failover.** Es un término genérico que se usa cuando un nodo debe asumir la responsabilidad de otro nodo, importar sus recursos y levantar el servicio de datos. Se ha de entender que una situación de *failover* es una situación excepcional para cual la alta disponibilidad ha sido concebida, el fallo de un nodo. Si sólo queda un nodo en el clúster, tras los fallos de los demás, se estará ante un SPOF (Single Point Of Failure) hasta que el administrador del sistema verifique y restaure el clúster. También se ha de entender que el servicio de datos sigue levantado, que es el objetivo del la alta disponibilidad.

**Takeover.** Es un *failover* automático que se produce cuando un nodo nota un fallo en el servicio de datos. Para ello debe haber cierta monitorización con respecto al servicio de datos. El nodo que se declara fallido es forzado a ceder el servicio y recursos, o es simplemente eliminado.

**Switchover o Giveaway.** Es un *failover* manual, consiste en ceder los recursos de un servicio de datos y este mismo, a otro nodo del clúster, mientras se realizan ciertas tareas administrativas. A este procedimiento se le denomina *Node Outage*.

**Splitbrain.** Para la gestión de un clúster HA, es necesario un mecanismo de comunicación y verificación entre los nodos integrantes. Por este

mecanismo, cada nodo debe gestionar sus recursos correspondientes, según el estado del clúster, y a su vez cada nodo debe hacer chequeos o latidos (*beats*) a sus compañeros. Un *splitbrain* (división de cerebro) es un caso especial de failover, en el cual falla el mecanismo de comunicación y gestión de un clúster de dos nodos. Es una situación en la cual cada nodo cree que es el único nodo activo, y como no puede saber el estado de su nodo compañero, toma acciones en consecuencia, forzando un *takeover*.

Esta situación es peligrosa, ya que los dos nodos intentarán apropiarse de todos los recursos, incluyendo el servicio. El peligro aumenta sobre todo cuando se tienen recursos delicados, como recursos de almacenamiento, ya que cada nodo podría tomar y escribir por su cuenta, y quebrar la integridad de datos. Para evitar este problema, cada nodo debe actuar de una forma prudente, y utilizar los recursos compartidos como señal de que se está vivo. La forma de proceder de cada nodo es similar, ya que cada uno cree que su compañero ha desaparecido. Después de que un nodo aprecia este problema, tiene indicado que reserve un recurso llamado quórum. Un recurso quórum, es un recurso compartido, que se ha preestablecido en ambos nodos como tal. Este recurso es un recurso exclusivo, sólo un nodo del clúster puede reservarlo. Como este recurso sólo puede ser reservado por un nodo, el nodo que llegue tarde a la reserva del recurso, entiende que debe abandonar el clúster y ceder todos sus recursos. El quórum es utilizado como método de decisión. Otra forma de evitar esta situación, un poco más violenta, es que un nodo elimine a su compañero; el primero que apague a su compañero se queda con todos los recursos. Es un mecanismo muy brusco, pero muy eficaz.

**Fencing.** En los clústers HA existe una situación donde un nodo deja de funcionar correctamente pero todavía sigue levantado, accediendo a ciertos recursos y respondiendo peticiones. Para evitar que el nodo corrompa recursos o responda con peticiones, los clústers lo solucionan utilizando una técnica llamada *fencing*. La función principal del *fencing* es hacerle saber a dicho nodo que esta funcionando en mal estado, retirarle sus recursos asignados para que los atiendan otros nodos y dejarlo en un estado inactivo.

Para configurar la HA en VIRT, primero hay que definir un grupo en *Datacenter* > *HA* > *Groups* con los nodos participantes. A estos nodos se les puede asignar una prioridad de funcionamiento, el cual viene determinado por los recursos de cada servidor. A mayor número mayor es la prioridad. En el clúster VIRT, la prioridad de cada nodo es virt01:3, virt02:2 y virt03(MV):1. Como es lógico, la MV, la cuál dispone de los recursos mínimos para cumplir con el quórum, es la última en la cola de prioridades.

# Capítulo 2

# AUTOGESTIÓN DE SERVIDORES

## 2.1. Historia

En la década de 1950, el Departamento de Defensa de los Estados Unidos creó la gestión de la configuración como mecanismo para supervisar el complicado y potente hardware que tenía bajo su control: cosas como tanques, armamento, aviones, buques oceánicos, etc. Esto les permitía saber dónde estaban todos sus activos en cualquier momento. Al mismo tiempo, podían estar al tanto de cualquier cambio en su arsenal a lo largo del tiempo para garantizar que cada activo se mantuviera en condiciones óptimas de funcionamiento.

A medida que los sistemas de software se volvieron más complejos, la filosofía de la gestión de la configuración llegó al mundo del desarrollo. Como resultado, los equipos de desarrolladores son capaces de controlar dónde se encuentran todos sus activos virtuales y hacer un seguimiento de cada cambio realizado en cada uno de ellos.

Los principales equipos de DevOps de hoy en día utilizan la gestión de la configuración para asegurarse de que sus aplicaciones y sistemas funcionan tal y como se han diseñado y, por extensión, de que sus usuarios pueden utilizarlos tal y como se ha planificado desde un primer momento.

## 2.2. Concepto general

La gestión de la configuración es un proceso para mantener los sistemas informáticos, los servidores y el software en un estado deseado y coherente. Es una forma de asegurarse de que un sistema funciona como se espera que lo haga a medida que se realizan cambios a lo largo del tiempo.

La gestión de la configuración de los sistemas informáticos implica definir el estado deseado de un sistema y luego construir y mantener esos sistemas. Las configuraciones erróneas pueden dar lugar a un rendimiento deficiente, a incoherencias o a incumplimientos y afectar negativamente a las operaciones empresariales y a la seguridad. Cuando se realizan cambios no documentados en muchos sistemas y aplicaciones, se añade inestabilidad y tiempo de inactividad.

La identificación manual de los sistemas que requieren atención, la determinación de los pasos de corrección, la priorización de las acciones y la validación de la finalización son demasiado complicados de realizar en entornos grandes. Pero sin documentación, mantenimiento y un proceso de control de cambios, los administradores de sistemas y los desarrolladores de software podrían acabar sin saber qué hay en un servidor o qué software se ha actualizado.

Los sistemas de gestión de la configuración permiten definir de forma coherente los ajustes del sistema, así como construir y mantener esos sistemas de acuerdo con esos ajustes de referencia. La gestión de la configuración ayuda a los usuarios y a los administradores a saber dónde existen ciertos servicios y cuál es el estado actual de las aplicaciones.

## 2.3. Diferentes herramientas para la autogestión

### 2.3.1. Ansible

Ansible es una plataforma de gestión de la configuración de TI y de automatización de código abierto, proporcionada por Red Hat. Utiliza plantillas YAML legibles por humanos para que los usuarios puedan programar tareas repetitivas para que se produzcan automáticamente, sin necesidad de aprender un lenguaje avanzado.

Ansible sustituye el scripting *ad hoc* o la CM (*Configuration Management*) manual por un proceso automatizado y repetible. La herramienta envía el código de la aplicación, los programas y las instrucciones de configuración de la infraestructura de TI mediante módulos a los nodos gestionados, ya sean servidores físicos, máquinas virtuales (VM) o instancias en la nube. La herramienta también ofrece a los usuarios la opción de invertir su configuración a una arquitectura *pull*, en la que los nodos gestionados solicitan instrucciones a la herramienta, lo que suele hacerse para permitir el escalado.

Un usuario de Ansible configura las instrucciones como comandos o las empaqueta en jugadas reutilizables, ejecutadas en *playbooks*. Ansible realiza una función de orquestación, dando al usuario el control sobre el orden en que lleva a cabo los pasos automatizados.

Ansible no tiene agentes, lo que significa que no instala software en los nodos que gestiona. Esto elimina un posible punto de fallo y una vulnerabilidad de seguridad y, al mismo tiempo, ahorra recursos al sistema.

### 2.3.2. Chef

Chef es una plataforma de gestión de sistemas de código abierto y de automatización de infraestructuras en la nube.

Chef transforma la infraestructura en código para automatizar el despliegue y la gestión de servidores. Chef puede gestionar una variedad de tipos de nodos, incluyendo servidores, máquinas virtuales en la nube, dispositivos de red y contenedores. Gestiona Linux, Windows y otros sistemas. La herramienta está pensada para que los desarrolladores y los profesionales de operaciones de TI trabajen juntos para desplegar aplicaciones

Chef utiliza paquetes de código llamados recetas, compilados en *cookbooks*, para definir cómo configurar cada nodo. Una receta describe el estado en el que debe encontrarse un recurso en un momento dado. Chef compila las recetas dentro del *cookbook* junto con las dependencias y los archivos necesarios, como los atributos, las bibliotecas y los metadatos, para soportar una configuración particular.

Chef es una herramienta basada en agentes en la que el *chef-client* extrae la información de configuración del nodo gestionado desde el servidor Chef. El *chef-client* se instala en cada nodo para ejecutar la configuración real, y utiliza el lenguaje de programación Ruby.

### 2.3.3. Puppet

Puppet es una herramienta de gestión de sistemas de código abierto para centralizar y automatizar la gestión de la configuración. La gestión de la configuración es el registro detallado y la actualización de la información que describe el hardware y el software de una empresa.

Puppet tiene dos capas: un lenguaje de configuración para describir cómo deben ser los *hosts* y los servicios, y una capa de abstracción que permite al administrador implementar la configuración en una variedad de plataformas, incluyendo Unix, Linux, Windows y OS X. Los administradores pueden codificar la configuración de un servicio como una política, que Puppet supervisa y aplica.

### 2.3.4. SaltStack

SaltStack, también conocido como Salt, es una herramienta de gestión y orquestación de la configuración. Utiliza un repositorio central para aprovisionar nuevos servidores y otras infraestructuras de TI, realizar cambios en los existentes e instalar software en entornos de TI, incluyendo servidores físicos y virtuales, así como en la nube.

Salt se utiliza en organizaciones *DevOps* porque extrae el código de los desarrolladores y la información de configuración de un repositorio de código central, como GitHub o Subversion, y empuja ese contenido de forma remota a los servidores. Los usuarios de Salt pueden escribir sus propios *scripts* y programas, y pueden descargar configuraciones preconstruidas que otros usuarios han aportado a un repositorio público.

El componente principal de Salt, el motor de ejecución remota, crea una red de comunicaciones segura, bidireccional y de alta velocidad. Con un maestro en ejecución, un minion iniciado intenta generar *hashes* criptográficos y conectarse al maestro para formar la red. Después de usar la autenticación de clave pública, los minions pueden aceptar comandos de un maestro. Salt también puede ejecutarse en un modo de minion sin maestro.

Salt se diferencia de otras herramientas de gestión de configuración y automatización por su velocidad. Su diseño multihilo permite la ejecución de cientos o incluso miles de tareas simultáneas. Utiliza la mensajería ZeroMQ, que está desacoplada, lo que significa que no se requiere una conexión persistente.

Salt utiliza una configuración esclavo-maestro que permite la ejecución *push* y *pull*. El usuario puede empujar las actualizaciones y el nuevo código en masa, o establecer un horario para que los servidores comprueben el maestro de Salt para las actualizaciones y tirar de ellos en consecuencia. La arquitectura de gestión de la configuración de Salt es, por lo tanto, impulsada por eventos y auto-reparación, ya que el sistema puede enviar actualizaciones y responder a los problemas al mismo tiempo. Salt puede funcionar en modo basado en agentes o sin agentes.

## 2.4. Razones para utilizar herramientas de autogestión

Las empresas están adoptando cada vez más herramientas y procedimientos de gestión de la configuración debido a sus grandes beneficios, entre lo que se encuentran:

**Mayor tiempo de funcionamiento.** Dado que los equipos de software pueden supervisar fácilmente los cambios en los sistemas, tienen más visibilidad de la infraestructura subyacente que alimenta sus aplicaciones. Una mayor visibilidad significa que pueden detectar rápidamente cualquier problema o incluso evitar que se produzca, lo que reduce la probabilidad de que surjan interrupciones imprevistas.

**Agilidad.** Incluso con más visibilidad, el software es software, y algo se romperá en algún momento. Gracias a la gestión de la configuración, cuando surgen problemas, el personal encargado puede responde rápidamente ante ellos. Aumenta el tiempo medio de resolución ($MTTR$) y garantiza que los sistemas se pongan en línea y se reconfiguren adecuadamente lo antes posible.

**Experiencia de usuario.** Un mayor tiempo de actividad y la mejora de las métricas $MTTR$ se traducen en sistemas que funcionan para sus usuarios tal y como fueron diseñados, casi siempre. Como resultado, los empleados hacen más cosas y la experiencia de los clientes mejora. Esto significa que las empresas disfrutan de una mayor productividad del personal. Una vez más, en caso de que algo no funcione correctamente, los equipos de *DevOps* pueden actuar rápidamente para volver a poner los sistemas en línea más rápidamente.

**Rentabilidad.** La gestión de la configuración ofrece a las organizaciones un mayor conocimiento de sus sistemas. Esto les permite prestar atención a todos los detalles, incluso a los aparentemente menores. Al supervisar constantemente la salud de sus sistemas, puede evitar errores costosos. Hay que arreglar los problemas menores antes de que se conviertan en problemas graves. Además, la gestión de la configuración facilita la prevención de las violaciones de la seguridad, junto con las devastadoras multas y los clientes no demasiado satisfechos que pueden conllevar. Por último, la gestión de la configuración también le permite asegurarse de que no se realiza un trabajo duplicado. Esto le proporcionará una eficiencia adicional. Si lo suma todo, la gestión de la configuración aumenta la rentabilidad de forma sustancial.

**Escalabilidad.** Las aplicaciones modernas deben ser capaces de escalar para acomodarse a los períodos de alto tráfico. La gestión de la configuración puede ayudar aquí porque permite a los equipos saber, con certeza, que sus sistemas están funcionando a un nivel óptimo. Gracias a la automatización, los equipos pueden escalar rápidamente mediante el aprovisionamiento de nuevos servidores correctamente configurados en tan sólo unos minutos.

## 2.5. Autogestión con Ansible

Entre las diferentes herramientas descritas en el apartado 2.3, se ha optado por utilizar Ansible por las siguientes razones:

- Simplicidad
- Ausencia de agentes
- Baja curva de aprendizaje
- Versatilidad
- Comunidad

Tal y como se ha comentado en apartado 2.3, Ansible es una plataforma de gestión de la configuración de TI y de automatización de código abierto. Gracias a esta herramienta, la configuración y mantenimiento de las MVs que se alojen en el clúster VIRT va a ser una tarea rápida y sencilla.

Antes de nada, hay que realizar un proceso de migración de MVs a VIRT, las cuáles están alojadas en una platafroma de virtualización en la nube, y cuyo sistema operativo es CentOS-6. El EOL (*end-of-life*) de esta distribución de Linux dió lugar el 30 de noviembre de 2020, lo que significa que no va a haber más actualizacoines ni parches a partir de esa fecha en adelante. Esto puede derivar en graves problemas de seguridad, de modo que la migración ha de realizarse a otra versión de CentOS o, alternativamente, a otra distribución.

En este caso, la migración va a ser de CentOS-6 a Rocky-8. Desafortunadamente, no hay una ruta de migración directa entre estas dos distribuciones de Linux, es por eso que hay que escoger entre uno de los siguientes métodos para dicha tarea:

- Actualizar de CentOS-6 a CentOS-7, de este a CentOS-8, y de este último migrar a Rocky 8
- Construir una nueva MV sobre Rocky-8 y migrar los datos

El primer método puede llevar un tiempo considerable, además, es bastante probable que surgan contratiempos adicionales debido a los cambios de paquetes entre las versiones principales. Dicho esto, se crearán MV nuevas sobre Rocky-8 y se migrarán única y exclusivamente los datos necesarios para el correcto funcionamiento del sistema.Para agilizar la creación de futuras MVs en Rocky-8, en el clúster de Proxmox, se creará una plantilla de esta distribución, tal y como se explica en el Anexo 6.2.

En está ocasión, y teniendo en cuenta que la mayoría de las MVs que están alojadas en el servidor actual son LAMPs (Linux, Apache, MySQL, PHP/Perl/Python), el uso de Ansible irá enfocado a la configuración y mantenimiento de estos servicios en las nuevas MVs Rocky-8 alojadas en VIRT. Además, las MVs pueden disponer de funcionalidades extra como, por ejemplo, un servicio de administración de activos TI, más comúnmente conocido como GLPI. De modo que, por un lado, se necesitan los servicios básicos que conforman un servidor LAMP (los cuales se desplegarán en cada una de las MVs) y, por otro lado, servicios espécificos que solamente afectarán a alguna de estas, como el servicio GLPI mencionado anteriormente.

Para comenzar, hay que asegurarse de tener instalado Ansible en el sistema principal. En caso de que no lo esté, basta con lanzar el siguiente comando en la terminal:

```
sudo apt install ansible
```

El siguiente paso es crear el sistema de ficheros que Ansible tomará como referencia para autogestionar las MVs alojadas en VIRT, el cual viene definido en el siguiente esquema:

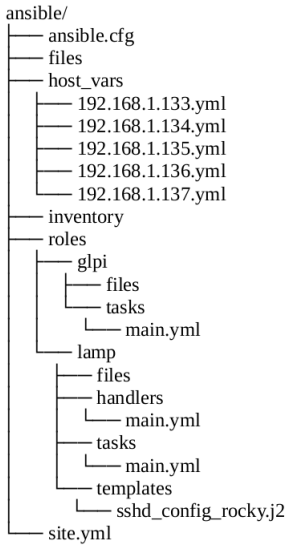


Figura 2.1: Estructura de ficheros para Ansible

El archivo *ansible.cfg* rige el comportamiento de todas las interacciones realizadas por el nodo de control:

```
[defaults]
inventory = inventory
private_key_file = ~/.ssh/id_ed25519
remote_user = root
```

El ficher *inventory* define los *hosts* y grupos de *hosts* sobre los que operan los comandos, módulos y tareas de un *playbook*:

```
[lamp_servers]
192.168.1.133
192.168.1.134
192.168.1.135
192.168.1.136
192.168.1.137

[glpi]
192.168.1.133
```

El *playbook* principal está alojado en *site.yml*:

```
---

- hosts: all
  become: true
  pre_tasks:
    - name: install updates
      tags: always
      dnf:
        update_only: yes
        update_cache: yes


- hosts: lamp_servers
  become: true
  roles:
    - lamp

- hosts: glpi
  become: true
  roles:
    - glpi
```

Las tareas especificas para cada rol van en */ansible/roles/rolename/tasks/main.yml*. En este caso, las tareas correspondientes a los servidores LAMP estarán definidas en */ansible/roles/lamp/tasks/main.yml*:

```
- name: create mikel user
  tags: always
  user:
    name: mikel
    groups: root

- name: add ssh key for mikel
  tags: always
  authorized_key:
    user: mikel
    key: "ssh-ed25519
        AAAAC3NzaC1lZDI1NTE5AAAAIDQFgoErWxs5hK0OiGShlEHsb+
        aqaX6cr3izGZbcuB/I mikel.arocena@ehu.eus"

- name: generate sshd_config file from template
  tags: ssh
  template:
    src: "{{ ssh_template_file }}"
    dest: /etc/ssh/sshd_config
    owner: root
    group: root
    mode: 0644
  notify: restart_sshd

- name: install apache2 package
  tags: httpd
  dnf:
    name:
      - httpd
      - php
    state: latest

- name: start httpd
  tags: httpd
  service:
    name: httpd
    state: started
    enabled: yes

- name: open firewall for http/https
  tags: httpd
  firewalld:
    service: http
    permanent: yes
    state: enabled
  notify: reload_firewalld

- name: install mariadb package
  tags: mariadb
  dnf:
```

```
    name:
      - mariadb-server
      - mariadb
    state: latest
```

Tal y como se observa en el fichero */ansible/roles/lamp/tasks/main.yml*, Ansible carga en las nuevas máquinas un fichero a partir de un *template*, los cuales deben ir alojados en */ansible/roles/rolename/templates/template_name.j2*. En este caso, el fichero hace referencia a un archivo de configuración SSH en el que se ha habilitado el acceso a un determinado usuario:

```
...

#ListenAddress ::

AllowUsers {{ ssh_users }}

HostKey /etc/ssh/ssh_host_rsa_key

...
```

Los handlers son acciones que se activan despúes de una determinada acción y van alojados en */ansible/roles/lamp/handlers/main.yml*:

```
- name: reload_firewalld
  systemd:
    name: firewalld
    state: reloaded
```

Tanto *ssh_template_file* como *ssh_users* son variables que van definidas en */ansible/host_vars/*, y tiene la forma siguiente:

```
ssh_users: mikel
ssh_template_file: sshd_config_rocky.j2
```

En cuanto al servicio GLPI, los paquetes necesarios para su correcto funcionamiento están definidos en */ansible/roles/glpi/tasks/main.yml*:

```
- name: install GLPI related packages
  tags: glpi
  dnf:
    name:
      - php-mysqlnd
      - php-gd
      - php-intl
      - php-ldap
      - php-xmlrpc
      - php-opcache
    state: latest
```

Por úlimo, ya con todos los ficheros preparados, hay que lanzar el siguiente comando en la terminal para configurar las nuevas MVs en cuestión de segundos:

```
ansible-playbook site.yml
```

# Capítulo 3

# MONITORIZACIÓN

## 3.1. Historia

Al principio, la monitorización informática era una forma de comprobar la disponibilidad de los servidores y la conectividad de los dispositivos. Pronto pasó a utilizarse en las áreas de hardware y sistemas operativos.

Este tipo de monitorización ofrece un análisis individualizado del estado de los dispositivos y compara los datos obtenidos con parámetros preestablecidos. Si los valores no están dentro de estos parámetros, la herramienta genera una alerta.

Sin embargo, la supervisión centrada en los dispositivos tiene varios inconvenientes. Entre ellos, la falta de contexto y visión global o la generación de un gran volumen de datos que no siempre son útiles. Aun así, muchas empresas siguen utilizando este tipo de monitorización en la actualidad.

Más tarde, se produjo un cambio hacia la especialización en la monitorización, con diferentes departamentos o equipos que gestionan diferentes herramientas específicas. Estas herramientas proporcionan más detalles sobre aspectos concretos, como la infraestructura o las aplicaciones.

Complementariamente, empezaron a aparecer herramientas que concentran toda la información relevante y generan "alertas complejas". Con ellas, el término telemetría ganó protagonismo. La telemetría engloba los conceptos de eventos, métricas, logs y trazas, y se define como "todo el conjunto de datos relevantes para la monitorización, independientemente de la herramienta que los facilite y de su formato".

Pero este tipo de monitorización tampoco está exento de inconvenientes. Para empezar, la especialización y la gestión descentralizada no facilitan la resolución de problemas. Además, aunque las herramientas especializadas generen una cantidad importante de información, eso no significa que sea más fácil de entender e interpretar, y puede dar lugar a problemas de almacenamiento. Además, en este nivel de especialización sigue faltando una visión global de todos los aspectos vigilados.

## 3.2. Concepto General

Los tipos básicos de monitorización de TI incluyen la monitorización de la disponibilidad, la monitorización del rendimiento de la web, la gestión de las aplicaciones web y la gestión del rendimiento de las aplicaciones, la gestión de las API, la monitorización del usuario real, la monitorización de la seguridad y la monitorización de la actividad empresarial. Estas herramientas de supervisión de la infraestructura de TI pueden dividirse en tres categorías generales o tipos de dispositivos de red en función de su uso:

**Observación.** Son los tipos más básicos de herramientas de monitorización de TI, utilizadas para observar el hardware, el software o los servicios, e informar sobre su eficacia operativa. La mayoría de las herramientas de supervisión de la disponibilidad, incluidas las herramientas de supervisión y gestión de la infraestructura, las herramientas de supervisión del rendimiento de las aplicaciones y las herramientas de supervisión del rendimiento de la web entran en esta categoría.

**Análisis.** Este tipo de herramienta de supervisión de TI se encarga de tomar los datos de observación y analizarlos más a fondo. Estos datos pueden analizarse para determinar dónde se originan los problemas o, lo que es más importante, para determinar por qué se producen esos problemas. Las herramientas de análisis avanzadas, como los sistemas AIOps, se encargan de prever dónde es probable que surjan los problemas basándose en tendencias y patrones históricos.

**Compromiso.** Como último nivel de las herramientas de monitorización de TI, las herramientas de compromiso están diseñadas para actuar sobre la información creada por las herramientas de análisis y de observación. Esto puede adoptar una forma simple, en el caso de los tickets de servicio o las alertas que se entregan de forma inteligente al analista o gestor de negocio apropiado, o más comúnmente, se utiliza para poner en marcha servicios adicionales, reiniciar hardware o software problemático, o ejecutar copias de seguridad.

## 3.3. Diferentes herramientas para la monitorización

Para decidirse por una herramienta de monitorización, hay que determinar qué información o métricas necesitan seguimiento y por qué. Las razones incluyen la supervisión de los datos, el refuerzo del rendimiento de las aplicaciones, el seguimiento de los problemas de salud del sistema y la planificación a largo plazo. A continuación, se describen las herramientas de monitorización más empleadas en la actualidad.

### 3.3.1. Nagios

Nagios es un sistema de supervisión de infraestructuras informáticas de código abierto. Se diseñó originalmente para ejecutarse en un sistema operativo Linux, pero ahora puede ejecutar variantes de Unix y sistemas operativos Windows.

Nagios permite a los administradores de TI detectar los problemas antes de que se conviertan en un problema. Comprueba los recursos de la aplicación, la red y el servidor, y envía notificaciones si los sistemas alcanzan niveles críticos. Esto garantiza que los administradores de TI puedan abordar cualquier problema antes de que se les vaya de las manos. El sistema también puede ejecutar configuraciones sin agentes y basadas en agentes.

Dos herramientas de software populares de Nagios incluyen Nagios Core y Nagios XI. Nagios Core es gratuito y es bueno para las empresas de menor tamaño. Nagios XI es mejor para las empresas más grandes, ya que incluye características adicionales, tales como gráficos detallados, informes y planificación de la capacidad.

### 3.3.2. Check-mk

Checkmk es un sistema integral de monitorización de TI que permite a los administradores de sistemas, a los gestores de TI y a los equipos de *DevOps* identificar problemas en toda su infraestructura de TI (servidores, aplicaciones, redes, almacenamiento, bases de datos) y actuar rápidamente para resolverlos. Entre sus principales características, se pueden encontrar:

- Monitorización de salud, métricas, registros y eventos
- Compatible con servidores, redes, aplicaciones y más
- Descubrimiento automático de redes y servicios
- Potentes opciones de búsqueda y filtrado
- Inventario de hardware y software

### 3.3.3. Zabbix

Zabbix es una herramienta de monitorización de infraestructuras que brilla por la flexibilidad del sistema de monitorización. La herramienta cubre una amplia variedad de componentes de TI, como MVs, servidores, servicios en la nube y redes. Proporciona métricas para la red, la carga de la CPU y el consumo de espacio en disco.

Zabbix es una buena opción para las empresas que buscan una herramienta personalizable, ya que ofrece plantillas de paneles automatizadas y personalizadas. También proporciona una API que permite a los administradores crear nuevas aplicaciones, automatizar tareas e integrarse con software

de terceros. Esto proporciona una mejor extensibilidad y acceso a las características y datos de monitorización de Zabbix.

Los administradores de TI pueden utilizar Zabbix para una supervisión basada en agentes y sin agentes. Puede ejecutarse en la nube o en las instalaciones, pero no tiene una aplicación SaaS alojada. Zabbix es de código abierto y de descarga gratuita.

## 3.4. Razones para utilizar la monitorización

Actualmente existen infinidad de herramientas de monitorización de infraestruras TI, cada una con sus pros y sus contras, sin embargo, todas ellas comparten las siguientes características:

**Identificación y alerta rápida de problemas.** Una de las ventajas de la monitorización de servidores en tiempo real es su capacidad para identificar problemas en los servidores y alertar a los administradores antes de que los posibles problemas afecten a los usuarios. Para las empresas que requieren una fiabilidad inquebrantable, estas alertas en tiempo real permiten a los administradores mantener los servidores operativos el 100 % del tiempo, sin ningún fallo de funcionalidad.

**Mapas de dependencia.** Otra ventaja destacada de la supervisión de servidores en tiempo real es su funcionalidad de mapas de dependencia. De este modo se pueden identificar y resolver los problemas antes de que alcancen una importancia crítica y amenacen potencialmente la seguridad y fiabilidad del sistema. Dependiendo del formato de la infraestructura de servidores que se esté supervisando, esto puede ayudar a identificar los cuellos de botella del sistema, mejorar la fiabilidad y eliminar cualquier proceso o componente superfluo.

**Vista centralizada.** Para los administradores de servidores, realizar un seguimiento de cada componente funcional del servidor puede suponer un reto inmenso. Para las grandes empresas con una amplia infraestructura de servidores, esta tarea puede ser categóricamente abrumadora para los administradores de servidores, causando a menudo descuidos potencialmente catastróficos. Una de las ventajas de la monitorización de servidores en tiempo real es su capacidad para ayudar a los administradores a realizar un seguimiento de todos los componentes del servidor desde un único punto de vista.

## 3.5. Monitorización con Check-mk

Entre las diferentes herramientas descritas en el apartado 3.3, se ha optado por utilizar Check-mk por las siguientes razones:

- Gran colección de plug-ins
- Escalado
- Informes detallados
- Alto grado de personalización
- Alertas inteligentes

En Check-mk, lo habitual es disponer de un servidor maestro, cuya finalidad va a ser comunicarse con el resto de equipos, denominados esclavos, los cuales tienen un agente integrado en el sistema. En el Anexo 6.4 se explica paso a paso como proceder con la instalación básica del servidor maestro.

Con el servidor principal configurado, y teniendo en cuenta que lo que se quiere monitorizar son servidores Proxmox (distribución de GNU/Linux basada en Debian), basta con instalar al agente correspondiente en cada *host* esclavo tal y como se explica el en Anexo 6.5. Un agente recopila los datos relevantes para la monitorización, este puede ser un pequeño programa instalado en el host, un agente SNMP que se ejecuta independientemente de Checkmk, un agente especial que obtiene la información a través de una API proporcionada por el sistema de destino, o una comprobación activa que consulta los servicios basados en la red. La siguiente imagen muestra las distintas formas en que Checkmk puede acceder a los sistemas a monitorizar:

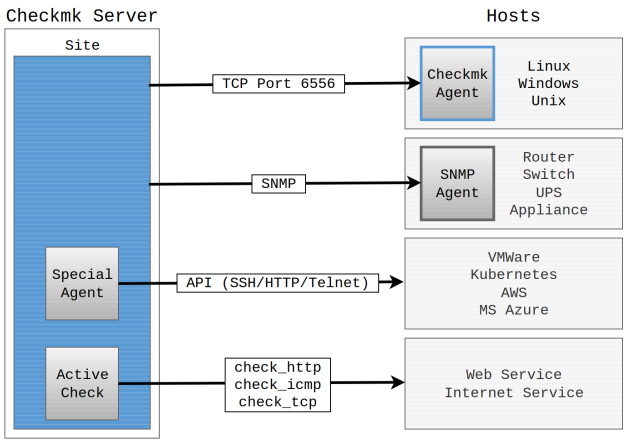


Figura 3.1: Protocolos de comunicación en checkmk entre servidor y host

En este caso, los agentes serán pasivos y escucharán en el puerto TCP 6556. Sólo al recibir una consulta del servidor Checkmk se activarán estos agentes y responderán con los datos requeridos. Estos datos son representados en modo de servicios en la GUI (*Graphical User Interface*) del servidor. Un servicio es un objeto lógico que resume uno o más aspectos de un host. Por ejemplo, tamaño, utilización y tendencias de los sistemas de archivos, utilización de la CPU, temperaturas, antigüedad y número de programas en ejecución, puertos, sensores, etc. En un momento dado, cada servicio de la monitorización tiene uno de los estados OK, WARN, CRIT, UNKNOWN o PEND, siempre está asignado exactamente a un host y opcionalmente contiene una o más métricas.

Para visualizar los servicios de un determinado host, previamente hay que añadirlo el entorno de configuración de Checkmk. En la GUI del servidor hay que dirigirse a *Setup* > *Hosts* > *Main directory* y hacer clic sobre *Add host*. Una vez dentro del panel de configuración del host, habrá que rellenar el campo *hostname* y clicar sobre *Save & go to service configuration*. En esta sección aparecerán todos los servicios disponibles para el servidor, en este caso, los servidores que forman parte del clúster VIRT disponen de un total de 35, tal y como se puede observar en la siguiente figura:

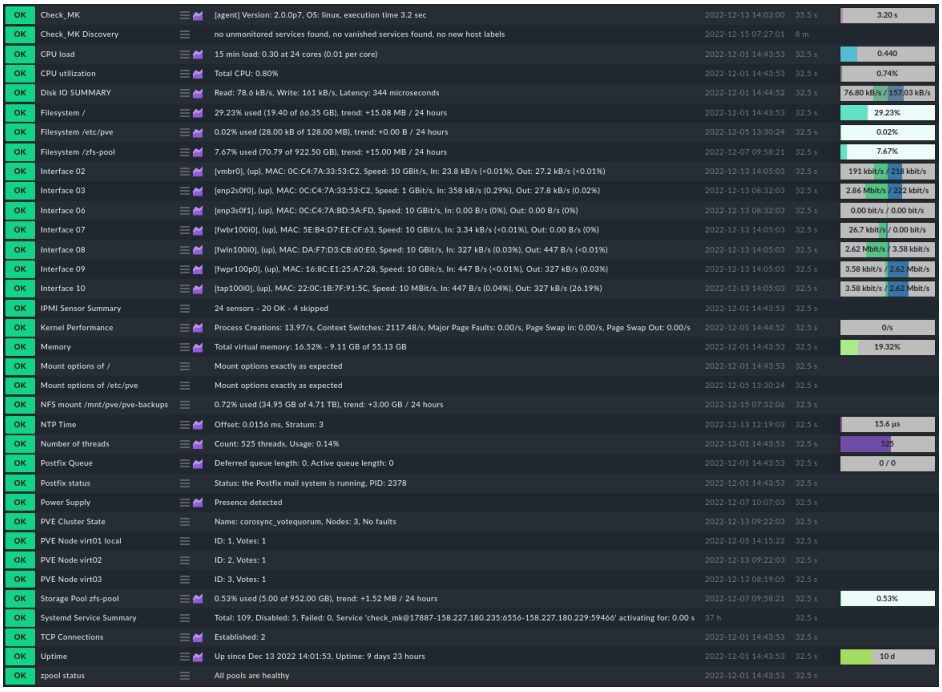


Figura 3.2: Servicios de virt01 y virt02 monitorizables en checkmk

Checkmk guarda todos los cambios un entorno de configuración temporal, sólo activando los cambios pendientes se transferirán a la monitorización.

# Capítulo 4

# Resultados

Tras haber migrado las MVs desde la plataforma antigua a la nueva de Proxmox, se quiere comparar cual es el rendimiento de cada una de estas, tanto desde el lado del sistema como desde el lado del servicio que presta.

En primer lugar, y teniendo en cuenta que se ha intentado reproducir las nuevas MVs lo más fielmente posible a sus predecesoras, a través del comando *sysbench*, se harán tests de CPU, memoria y operaciones I/O de la siguiente manera:

- sysbench –test=cpu run

```
# SISTEMA ACTUAL
sysbench 1.0.17 (using system LuaJIT 2.0.4)

Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   786.20

General statistics:
    total time:                          10.0003s
    total number of events:              7865

Latency (ms):
         min:                                    1.11
         avg:                                    1.27
         max:                                    2.84
         95th percentile:                        1.50
         sum:                                 9969.65

Threads fairness:
    events (avg/stddev):           7865.0000/0.00
    execution time (avg/stddev):   9.9697/0.00
```

```
# SISTEMA NUEVO
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Prime numbers limit: 10000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:  1316.55

General statistics:
    total time:                          10.0020s
    total number of events:              13172

Latency (ms):
         min:                                    0.74
         avg:                                    0.76
         max:                                    5.31
         95th percentile:                        0.80
         sum:                                 9990.64

Threads fairness:
    events (avg/stddev):           13172.0000/0.00
    execution time (avg/stddev):   9.9906/0.00
```

- sysbench –test=memory run

```
# SISTEMA ACTUAL
sysbench 1.0.17 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Running memory speed test with the following options:
  block size: 1KiB
  total size: 102400MiB
  operation: write
  scope: global

Initializing worker threads...

Threads started!

Total operations: 2147315 (214667.40 per second)

2096.99 MiB transferred (209.64 MiB/sec)


General statistics:
    total time:                          10.0002s
    total number of events:              2147315

Latency (ms):
         min:                                    0.00
         avg:                                    0.00
         max:                                    0.08
         95th percentile:                        0.00
         sum:                                 3177.50

Threads fairness:
    events (avg/stddev):           2147315.0000/0.00
    execution time (avg/stddev):   3.1775/0.00
```

```
# SISTEMA NUEVO
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Running memory speed test with the following options:
  block size: 1KiB
  total size: 102400MiB
  operation: write
  scope: global

Initializing worker threads...

Threads started!

Total operations: 25246405 (2523489.55 per second)

24654.69 MiB transferred (2464.35 MiB/sec)


General statistics:
    total time:                          10.0002s
    total number of events:              25246405

Latency (ms):
         min:                                    0.00
         avg:                                    0.00
         max:                                    1.17
         95th percentile:                        0.00
         sum:                                 4024.66

Threads fairness:
    events (avg/stddev):           25246405.0000/0.00
    execution time (avg/stddev):   4.0247/0.00
```

- sysbench –test=fileio –file-test-mode=seqwr run

```
# SISTEMA ACTUAL
sysbench 1.0.17 (using system LuaJIT 2.0.4)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Extra file open flags: (none)
128 files, 16MiB each
2GiB total file size
Block size 16KiB
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing sequential write (creation) test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      0.00
    writes/s:                     5874.51
    fsyncs/s:                     7529.56

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               91.79

General statistics:
    total time:                          10.0055s
    total number of events:              134038

Latency (ms):
         min:                                    0.01
         avg:                                    0.07
         max:                                   52.21
         95th percentile:                        0.06
         sum:                                 9455.38

Threads fairness:
    events (avg/stddev):           134038.0000/0.00
    execution time (avg/stddev):   9.4554/0.00
```

```
# SISTEMA NUEVO
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Extra file open flags: (none)
128 files, 16MiB each
2GiB total file size
Block size 16KiB
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing sequential write (creation) test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      0.00
    writes/s:                     2996.72
    fsyncs/s:                     3838.79

Throughput:
    read, MiB/s:                  0.00
    written, MiB/s:               46.82

General statistics:
    total time:                          10.0380s
    total number of events:              68530

Latency (ms):
         min:                                    0.01
         avg:                                    0.15
         max:                                   43.12
         95th percentile:                        0.07
         sum:                                 9987.67

Threads fairness:
    events (avg/stddev):           68530.0000/0.00
    execution time (avg/stddev):   9.9877/0.00
```

Desde el lado de la aplicación, se va a tomar como referencia la MV que aloja la web institucional. De forma externa, se realizarán trabajos de carga a través del siguiente comando:

```
ab -n 1000 -c 100 website_url
```

Para un número de peticiones de 1000 con 10 conexiones concurrentes:

- Las peticiones por sergundo resueltas en el servidor actual ascienden a 129.53

```
# SISTEMA ACTUAL
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.
    zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.
    org/

Benchmarking cfm.ehu.es (be patient)

Server Software:        nginx/1.10.2
Server Hostname:        cfm.ehu.es
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-RSA-AES256-GCM-SHA384
    ,2048,256
Server Temp Key:        ECDH P-256 256 bits
TLS Server Name:        cfm.ehu.es

Document Path:          /
Document Length:        22342 bytes

Concurrency Level:      100
Time taken for tests:   7.720 seconds
Complete requests:      1000
Failed requests:        0
Keep-Alive requests:    0
Total transferred:      22986000 bytes
HTML transferred:       22342000 bytes
Requests per second:    129.53 [#/sec] (mean)
Time per request:       772.017 [ms] (mean)
Time per request:       7.720 [ms] (mean, across all concurrent
    requests)
Transfer rate:          2907.61 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        9  605 131.3    642     732
Processing:    67  122  58.2    103     401
Waiting:       60  110  58.8     92     393
Total:         77  727 131.4    746    1035

Percentage of the requests served within a certain time (ms)
  50%    746
  66%    755
  75%    765
  80%    774
  90%    823
  95%    879
  98%    974
  99%   1005
 100%   1035 (longest request)
```

- Las peticiones por sergundo resueltas en el nuevo servidor ascienden a 633.83

```
# SISTEMA NUEVO
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.
    zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.
    org/

Benchmarking services-01-test.cfm.ehu.es (be patient)

Server Software:        Apache/2.4.37
Server Hostname:        services-01-test.cfm.ehu.es
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-RSA-AES256-GCM-SHA384
    ,2048,256
Server Temp Key:        X25519 253 bits
TLS Server Name:        services-01-test.cfm.ehu.es

Document Path:          /
Document Length:        21606 bytes

Concurrency Level:      100
Time taken for tests:   1.578 seconds
Complete requests:      1000
Failed requests:        0
Keep-Alive requests:    0
Total transferred:      22274000 bytes
HTML transferred:       21606000 bytes
Requests per second:    633.83 [#/sec] (mean)
Time per request:       157.771 [ms] (mean)
Time per request:       1.578 [ms] (mean, across all concurrent
    requests)
Transfer rate:          13787.02 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        4   21  23.5     12     134
Processing:    22  131  21.0    137     207
Waiting:       16  131  21.0    136     206
Total:         28  152  27.4    150     279

Percentage of the requests served within a certain time (ms)
  50%    150
  66%    155
  75%    159
  80%    162
  90%    173
  95%    200
  98%    238
  99%    258
 100%    279 (longest request)
```

# Capítulo 5

# Conclusión

En base a los resultados obtenidos en el apartado anterior, se puede concluir que, en las MVs alojadas en la nueva plataforma:

- El rendimiento de la CPU se ha visto incrementado en un 67.4 %
- La memoria soporta 10.8 veces más operaciones por segundo
- La velocidad de escritura se ha visto reducida a la mitad [ -49 % ]

En relación a los test de carga de la página web institucional, en Proxmox VE el tiempo de respuesta se ve reducido en, prácticamnete, 5 veces, tal y como se puede observar en la siguiente figura:

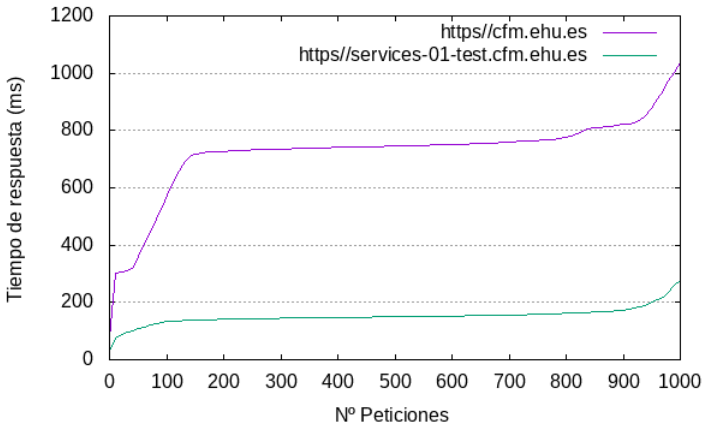


Figura 5.1: Tiempo de respuesta de la web institucional: 1. Línea morada. Servidor actual; 2. Línea verde. Servidor Proxmox

A todo esto hay que añadir que el nuevo sistema dispone de alta disponibilidad y que, adémas, es un sistema que se gestiona de forma local y no depende de terceros, como el caso de la plataforma actual, la cual se aloja en la nube. Ante cualquier imprevisto, la respuesta va a ser más rápida.

# Capítulo 6

# Anexos

## 6.1. Instalación de Proxmox VE 7.3

Para que Proxmox funcione de manerá óptima, el sistema debe cumplir, al menos, los siguiente requisitos:

- Intel EMT64 o AMD64 con *flag* CPU Intel VT/AMD-V
- Memoria: Mínimo 2 GB para el sistema operativo y los servicios Proxmox VE, además de la memoria designada para los invitados. Para Ceph y ZFS, se requiere memoria adicional; aproximadamente 1 GB de memoria por cada TB de almacenamiento utilizado.
- Almacenamiento rápido y redundante, los mejores resultados se consiguen con unidades SSD.
- Almacenamiento del SO: Utilice un RAID de hardware con caché de escritura protegida por batería ("BBU") o no RAID con ZFS (SSD opcional para ZIL).
- Almacenamiento VM:
  - Para el almacenamiento local, utilice un RAID de hardware con caché de escritura respaldada por batería (BBU) o no RAID para ZFS y Ceph. Ni ZFS ni Ceph son compatibles con un controlador RAID por hardware.
  - Es posible el almacenamiento compartido y distribuido.
- NICs (Multi-)Gbit redundantes, con NICs adicionales dependiendo de la tecnología de almacenamiento preferida y configuración del clúster.
- Para el paso PCI(e) la CPU necesita soportar el flag VT-d/AMD-d.

### 6.1.1. Preparar el medio de instalación

Descargarse la imagen ISO de: `https://www.proxmox.com/en/downloads/category/iso-images-pve`. El medio de instalación de Proxmox VE es una imagen ISO híbrida. Funciona de dos maneras:

- Un archivo de imagen ISO listo para grabar en un CD o DVD.
- Un archivo de imagen de sector en bruto (IMG) listo para copiar en una unidad flash USB (memoria USB).

El uso de una unidad flash USB para instalar Proxmox VE es la forma recomendada porque es la opción más rápida.En sistemas operativos tipo Unix, se puede utilizar el comando dd para copiar la imagen ISO en la unidad flash USB. Primero se busca el nombre de dispositivo correcto de la unidad flash USB.

```
# dd bs=1M conv=fdatasync if=./proxmox-ve_*.iso of=/dev/XYZ
```

Hay que sustituir */dev/XYZ* por el nombre correcto del dispositivo y adaptar la ruta del nombre del archivo de entrada (if).

También se puede usar software de terceros, Etcher es una de las herramientas más populares y es compatible con Linux, macOS y Windows.

### 6.1.2. Usando el instalador Proxmox VE

El instalador incluye lo siguiente:

- Sistema operativo completo (Debian Linux, 64 bits)
- El instalador de Proxmox VE, que particiona el disco o discos locales con ext4, XFS, BTRFS o ZFS e instala el sistema operativo
- Kernel Linux Proxmox VE compatible con KVM y LXC
- Conjunto completo de herramientas para administrar máquinas virtuales, contenedores, el sistema host, clústeres y todos los recursos necesarios
- Interfaz de gestión basada en web

Una vez insertado el medio de instalación se configura la BIOS del sistema para que arranque desde el mismo.


Figura 6.1: Menú inicial de instalación en Proxmox VE

Después de elegir la entrada correcta (p. ej. Arrancar desde USB) se mostrará el menú de Proxmox VE y se podrá seleccionar una de las siguientes opciones:

**Install Proxmox VE.** Comienza instalación estándar.

**Advanced Options: Install Proxmox VE (Debug mode).** Inicia la instalación en modo de depuración. Se abrirá una consola en varios pasos de la instalación. Esto ayuda a a depurar la situación si algo va mal.

**Advanced Options: Rescue Boot.** Con esta opción puedes arrancar una instalación existente. Busca en todos los discos duros conectados. Si encuentra una instalación existente, arranca directamente en ese disco utilizando el kernel Linux de la ISO. Esto puede ser útil si hay problemas con el bloque de arranque (grub) o la BIOS es incapaz de leer el bloque de arranque desde el disco.

**Opciones avanzadas: Memoria de prueba.** Ejecuta memtest86+. Esto es útil para comprobar si la memoria está funcional y libre de errores.

Después de seleccionar Instalar Proxmox VE y aceptar el EULA, aparecerá el mensaje para seleccionar el/los disco(s) duro(s) de destino. El botón Opciones abre el diálogo para seleccionar el sistema de archivos de destino. El sistema de archivos por defecto es ext4. El Gestor de Volúmenes Lógicos

(LVM) se utiliza cuando se selecciona *ext4* o *xfs*. También se pueden configurar opciones adicionales para restringir el espacio LVM. Proxmox VE puede instalarse en ZFS. Como ZFS ofrece varios niveles de RAID por software, esta es una opción para sistemas que no tienen un controlador RAID de hardware. Los discos de destino deben seleccionarse en el cuadro de diálogo Opciones Opciones. Se pueden cambiar más ajustes específicos de ZFS en Opciones avanzadas.


Figura 6.2: Menú de selección de disco en Proxmox VE

La siguiente página solicita opciones básicas de configuración como la ubicación, la zona horaria y la distribución del teclado. La ubicación ubicación se utiliza para seleccionar un servidor de descarga cercano para acelerar las actualizaciones. El instalador suele autodetectar estas opciones. Sólo es necesario cambiarlos en el caso poco frecuente de que falle la detección automática o de que deba utilizarse una distribución de teclado diferente.

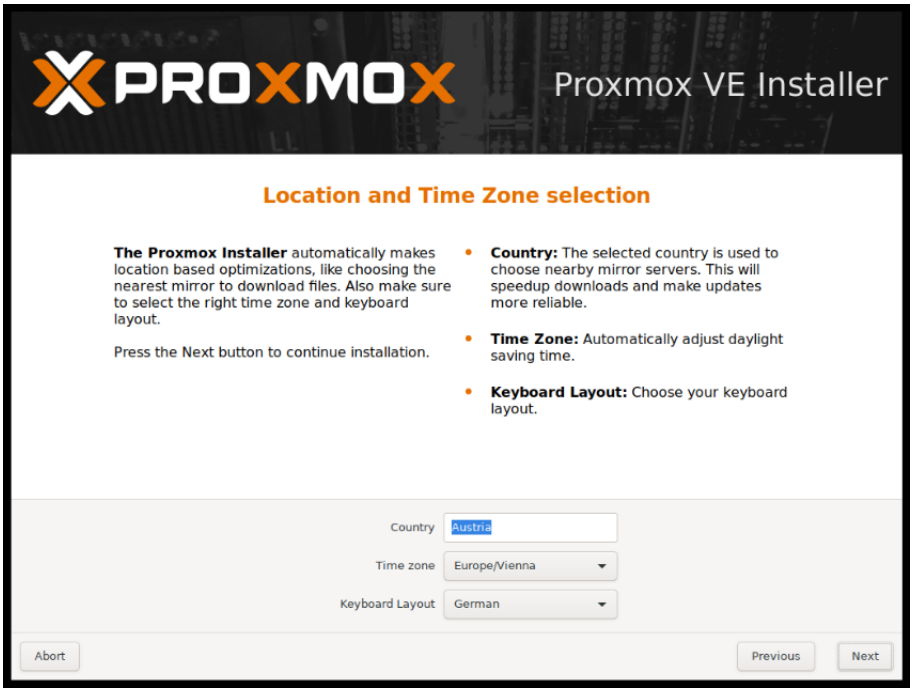


Figura 6.3: Selección de ubicación y zona horaria en Proxmox VE

A continuación se debe especificar la contraseña del superusuario (root) y una dirección de correo electrónico. La contraseña debe constar de al menos 5 caracteres. Es muy recomendable utilizar una contraseña fuerte. Algunas directrices son:

- Utilizar una longitud mínima de contraseña de 12 a 14 caracteres.
- Incluir caracteres alfabéticos en minúsculas y mayúsculas, números y símbolos.
- Evitar la repetición de caracteres, patrones de teclado, palabras comunes de diccionario, secuencias de letras o números, nombres de usuario, nombres de familiares o mascotas, nombres románticos, etc. nombres de usuario, nombres de familiares o mascotas, vínculos sentimentales (actuales o pasados) e información biográfica (por ejemplo números de DNI, nombres de antepasados o fechas).

La dirección de correo electrónico se utiliza para enviar notificaciones al administrador del sistema. Por ejemplo:

- Información sobre actualizaciones de paquetes disponibles.
- Mensajes de error de trabajos CRON periódicos.

El último paso es la configuración de la red. Tenga en cuenta que durante la instalación puede utilizar una dirección IPv4 o IPv6, pero no ambas. Para configurar un nodo de doble pila, añada direcciones IP adicionales después de la instalación.


Figura 6.4: Configuración de red en Proxmox VE

El siguiente paso muestra un resumen de las opciones seleccionadas anteriormente. Para aceptar, hay que pulsar sobre *Install*. La instalación comienza a formatear los discos y a copiar los paquetes en el destino. Una vez finalizado el proceso, se retira el medio de instalación y se reinicia el sistema.

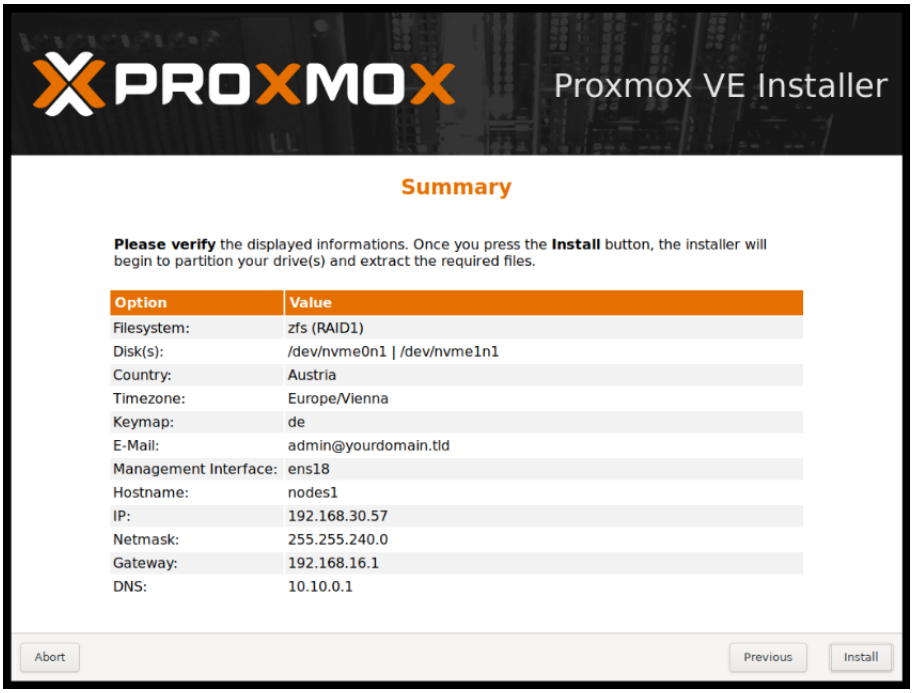


Figura 6.5: Resumen de instalación en Proxmox VE

La configuración posterior se realiza a través de la interfaz web de Proxmox alojada en puerto 8006 (https://direccion_ip:8006).

## 6.2. Creación de template para Rocky Linux 8 con cloud init en Proxmox VE

El primer paso es preparar la MV. Simplemente hay que instalar los paquetes de Cloud-Init dentro de la MV que se quiera preparar. En sistemas basados en RHEL esto es tan simple como:

```
dnf install cloud-init
```

Muchas distribuciones ya proporcionan imágenes Cloud-Init listas para usar (proporcionadas como archivos .qcow2), por lo que, como alternativa, puede simplemente descargar e importar dichas imágenes. Para el siguiente ejemplo, utilizaremos la imagen de nube proporcionada por Rocky en `https://download.rockylinux.org/pub/rocky/8/images/x86_64/`

```
# download the image
wget https://cloud-images.ubuntu.com/bionic/current/bionic-
    server-cloudimg-amd64.img

# create a new VM with VirtIO SCSI controller
qm create 9000 --memory 2048 --net0 virtio,bridge=vmbr0 --
    scsihw virtio-scsi-pci

# import the downloaded disk to the local-lvm storage,
    attaching it as a SCSI drive
qm set 9000 --scsi0 local-lvm:0,import-from=/path/to/bionic-
    server-cloudimg-amd64.img
```

El siguiente paso es configurar una unidad de CD-ROM, que se utilizará para pasar los datos de Cloud-Init a la MV.

```
qm set 9000 --ide2 local-lvm:cloudinit
```

Para poder arrancar directamente desde la imagen Cloud-Init, hay que establecer el parámetro de arranque a order=scsi0 para restringir a BIOS a arrancar sólo desde este disco. Esto acelerará el arranque, ya que MV BIOS se salta la comprobación de un CD-ROM de arranque.

```
qm set 9000 --boot order=scsi0
```

Para muchas imágenes Cloud-Init, es necesario configurar una consola serie y utilizarla como pantalla. Sin embargo, si la configuración no funciona para una imagen determinada, vuelva a la pantalla predeterminada.

```
qm set 9000 --serial0 socket --vga serial0
```

En un último paso, es útil convertir la VM en una plantilla. A partir de esta plantilla se pueden crear rápidamente clones vinculados. El despliegue desde plantillas VM es mucho más rápido que crear un clon completo (copia).

```
qm template 9000
```

## 6.3. Copia de seguridad vía NFS

En esta sección se explicará como vincular un volumen de datos alojado en un servidor TrueNAS con Proxmox, con el objetivo de programar copias de serguridad diarias de las MVs alojadas en este último, proporcionando de esta forma una capa adicional de seguridad al sistema.

La comunicación entre TrueNAS y Proxmox se va a realziar a través del protocolo NFS. *Network File System*, o NFS, es un protocolo de nivel de aplicación, según el Modelo OSI. Es utilizado para sistemas de archivos distribuido en un entorno de red de computadoras de área local. Posibilita que distintos sistemas conectados a una misma red accedan a ficheros remotos como si se tratara de locales.

En TrueNAS, para crear un nuevo recurso compartido, primero hay que asegurarse de que existe un conjunto de datos apto para dicha tarea. Dicho esto, para crear un recurso compartido, basta con dirigirse a *Sharing* > *Unix Shares (NFS)* y clicar sobre *ADD*. Después, en el explorador de archivos, hay que seleccionar el conjunto de datos en cuestión y pulsar *SUBMIT*. En caso de que el servicio NFS no este habilitado, aparecerá un aviso tal y como se muesra en la siguiente figura:

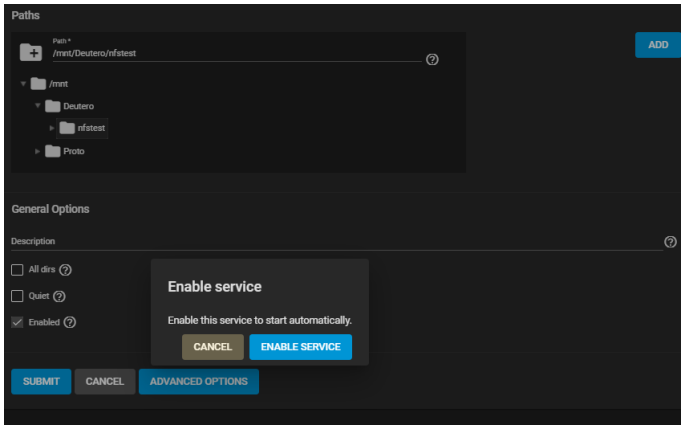


Figura 6.6: Recurso compartido NFS en TrueNAS

Finalmente, en Promox, para añadir el volumen NFS, hay que dirigirse a *Datacenter* > *Storage* y clicar sobre *Add: NFS*. Se abrirá un menú donde habrá de especificar el servidor y el volumen de datos a compartir, asi como su funcionalidad. Si todo ha dio bien, debería verse un nuevo volumen en el clúster. Será este volumen el encargado de almacenar los *backups* de las MVs alojadas en el servidor (o clúster). Para dicho propósito, tan solo hay que dirigirse a *Datacenter* > *Backup* y crear una tarea de *backup*. En el menú de configuración habrá que seleccionar las MVs a las que se quiere realizar una copia de seguridad, así como el volumen destino y la frecuencia, entre otras cosas. Lo habitual es hacer un tarea de backup diaria.

## 6.4. Instalación del servidor Checkmk en Linux

Una vez descargado el paquete de software en el equipo, hay que copiar el archivo en el sistema Linux de destino. Se puede hacer, por ejemplo, con el programa WinSCP o, si a través de SSH, con la herramienta de línea de comandos scp.

```
root@linux# apt install openssh-server
root@linux# scp check-mk-raw-2.1.0p1_0.focal_amd64.deb
    root@mymonitoring.mydomain.org:
```

Todos los paquetes se firman utilizando GnuPG. Mediante el uso de esta firma, por un lado se puede verificar si el paquete es realmente de Checkmk, y por otro lado se puede verificar que el paquete está completo.

Dependiendo de los paquetes opcionales que se hayan instalado durante la configuración de la distribución, puede que se necesite instalar dpkg-sig incluyendo todas sus dependencias para que la verificación se realice correctamente. Para ello, basta con introducir el siguiente comando:

```
root@linux# apt install dpkg-sig
```

Para que estos paquetes firmados puedan instalarse de la forma habitual, se debe importar la clave pública para que la firma sea de confianza. En primer lugar, hay que cargar la clave directamente desde el sitio web:

```
root@linux# wget https://download.checkmk.com/checkmk/Check_MK-
    pubkey.gpg
```

A continuación, hay que añadir la clave a la lista de firmas de confianza. En Debian y Ubuntu se requiere el siguiente comando:

```
root@linux# gpg --import Check_MK-pubkey.gpg
```

Una vez añadida la clave, se puede verificar el paquete Checkmk mediante el siguiente comando:

```
root@linux# dpkg-sig --verify check-mk-raw-2.1.0p1_0.
    focal_amd64.deb
```

Después se puede instalar el paquete Checkmk usando el siguiente comando. Hay que asegurarse de pasar la ruta de archivo completa al archivo DEB después de apt install:

```
root@linux# apt install /tmp/check-mk-raw-2.1.0p1_0.focal_amd64
    .deb
```

Después de instalar correctamente Checkmk y todas sus dependencias, se tendrá acceso al comando omd. Este comando crea y gestiona sitios de monitorización. Para verificar que la instalación ha sido exitosa, se puede comprobar la versión mediante:

```
root@linux# omd version
```

## 6.5. Instalación del agente Checkmk en Linux

El agente Checkmk está formado por el Agent Script y el Agent Controller, que se comunica con el Agent Receiver en el servidor Checkmk.

**Agent Script.** Es responsable de la recopilación de los datos de monitorización y llama a los comandos del sistema existentes para la recopilación de datos en secuencia. Para obtener dicha información, el agente también requiere privilegios de root, por lo que debe ejecutarse bajo root.

**Agent Controller.** Es el componente dentro del agente que se encarga de transportar los datos recogidos por el script del agente. El controlador se ejecuta bajo el usuario cmk-agent, que tiene privilegios limitados, por ejemplo, no tiene shell de inicio de sesión, y se utiliza únicamente para la transferencia de datos. El usuario cmk-agent se crea durante la instalación del paquete del agente. El controlador de agentes se inicia como un daemon de systemd y se acopla a él como un servicio. El controlador escucha en el puerto TCP 6556 las conexiones entrantes del sitio Checkmk y consulta el script del agente a través de un socket Unix (de una unidad systemd).

El agente Linux se instala a través del paquete RPM o DEB. La extensión depende de la distribución de Linux en la que desee instalar el paquete.

Tabla 6.1: Agentes checkmk para distribuciones Linux

| **Paquete** | **Extensión** | **Compatibilidad** |
|---|---|---|
| RPM | .rpm | RHEL, SLES, Fedora, openSuse |
| DEB | .deb | Debian, Ubuntu |

Antes de la instalación hay que obtener el paquete y llevarlo al cliente (por ejemplo con scp o WinSCP) donde se ejecutará el agente. En la Edición CRE Checkmk Raw se pueden encontrar los paquetes Linux del agente a través de Setup > Agents > Linux.

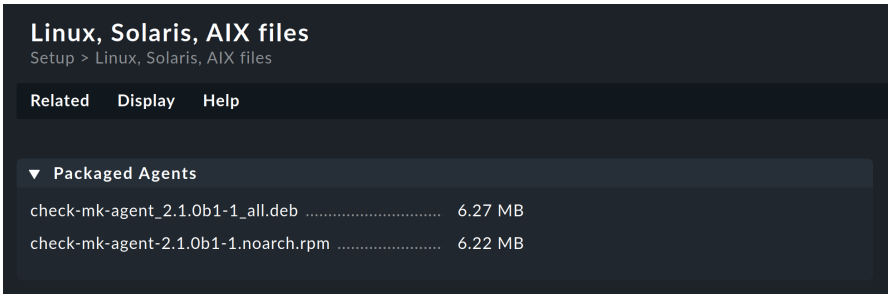


Figura 6.7: Agentes checkmk para Linux

Todo lo que necesita se encuentra en la primera casilla denominada Packaged Agents, es decir, los paquetes RPM y DEB listos para instalar el agente Linux con su configuración predeterminada.

A veces, descargar a una máquina y luego copiar a la máquina de destino utilizando scp o WinSCP puede ser muy engorroso. También se puede descargar el paquete desde el servidor Checkmk directamente al sistema de destino mediante HTTP. La forma más sencilla de hacerlo es con wget:

```
root@linux# wget http://mycmkserver/mysite/check_mk/agents/
    check-mk-agent-2.1.0b1-1_all.deb
```

Una vez obtenido el paquete RPM o DEB y, si es necesario, lo haya copiado al cliente que va a supervisar mediante *scp*, WinSCP u otros medios, la instalación se realiza con un único comando.

El paquete RPM se instala bajo root con el comando rpm -U:

```
root@linux# rpm -U check-mk-agent-2.1.0b1-1.noarch.rpm
```

La instalación del paquete DEB se realiza bajo root con el comando dpkg -i:

```
root@linux# dpkg -i check-mk-agent_2.1.0b1-1_all.deb
```

Ya con el agente instalado, ahora tocaría añadir el nuevo equipo al entorno de configuración Checkmk.

# Bibliografía


[1] Oracle Corp. (2012) *Introduction to Virtualization*

[2] Yahya Al-Dhuraibi, Fawaz Paraiso, Nabil Djarallah, Philippe Merle (2017) *Elasticity in Cloud Computing: State of the Art and Research Challenges*

[3] Ben Lutkevich (2021) High Availability (HA)

[4] Ceph.io (2016) *Welcome to Ceph*

[5] Java.net (2013) *Solaris ZFS Wiki*

[6] Proxmox Server Solutions GmbH (2022) *Proxmox VE Administration Guide*

[7] Truenas Enterprise (2022) *TrueNAS: Open Storage*

[8] Red Hat (2022) *Ansible Documentation*

[9] Chef Software DevOps Automation Solutions (2022) *Chef Platform Overview*

[10] Puppet Infrastructure & IT Automation at Scale (2021) *Welcome to puppet documentation*

[11] Salt Infra (2023) *Salt Project*

[12] Nagios (2018) *Nagios Core 4.x*

[13] Tribe29 GmbH (2022) *The official Checkmk User Guide*

[14] Zabbix (2023) *Zabbix Manual*

[15] Alexey Kopytov (2009) *SysBench manual*

[16] Apache (2022) *Apache HTTP server benchmarking tool*