



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Informática de Sistemas y Computadores

Servidor Web de alta disponibilidad con equilibrado de  
carga basado en un clúster de computadores

Trabajo Fin de Máster

Máster Universitario en Ingeniería de Computadores y Redes

AUTOR/A: Vasquez Villon, Vanessa

Tutor/a: López Rodríguez, Pedro Juan

CURSO ACADÉMICO: 2023/2024





UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA



Máster en Ingeniería de Computadores y Redes

# Servidor Web de alta disponibilidad con equilibrado de carga basado en un clúster de computadores

Trabajo Fin de Máster

**Autora:** *Vanessa Vásquez Villón*

**Tutor:** *Pedro López Rodríguez*

*Septiembre, 2024*

---

## AGRADECIMIENTOS

A Dios por la guía y fortaleza que me han permitido llegar hasta aquí.

A mi familia, por ser mi inspiración y apoyo constante.

A todas las personas que, de una forma u otra, han contribuido a mi desarrollo personal y profesional, gracias por su valiosa influencia.

---

## RESUMEN

En los últimos años el crecimiento en la demanda de los servicios de Internet ha provocado que la arquitectura de los servidores esté basada en un clúster de computadores. El clúster se compone de uno o varios nodos directores que se encargan de repartir el trabajo entre los nodos servidores que implementan los servicios ofertados. En este Trabajo Fin de Máster (TFM) se plantea instalar, configurar y verificar un servidor web PHP basado en un clúster de computadores, con sistema operativo Linux. El sistema incorporará tres nodos directores para garantizar un funcionamiento continuo, así como mecanismos para repartir la carga entre los servidores. Por otra parte, en el TFM también se explorarán soluciones escalables basadas en clúster para el almacenamiento de los datos. Con el objeto de facilitar la verificación de la solución adoptada, el servidor de Internet se configurará sobre un entorno virtualizado. Sobre el clúster configurado se realizarán pruebas de carga y rendimiento para evaluar la eficiencia y fiabilidad del sistema. También se realizarán ajustes en su configuración basados en los resultados obtenidos para optimizar el rendimiento y la disponibilidad.

**Palabras Clave:** Servicio de altas prestaciones; equilibrado de carga; alta disponibilidad; evaluación de prestaciones; clúster de computadores.

---

## ABSTRACT

In recent years, the growth in demand for Internet services has led to the architecture of servers being based on a computer cluster. The cluster is made up of one or more director nodes that are responsible for distributing the work among the server nodes that implement the services offered. This Master's Thesis (TFM) proposes to install, configure and verify a PHP web server based on a computer cluster, with a Linux operating system. The system will incorporate three director nodes to ensure continuous operation, as well as mechanisms to distribute the load among the servers. On the other hand, the TFM will also explore scalable cluster-based solutions for data storage. In order to facilitate the verification of the adopted solution, the Internet server will be configured on a virtualized environment. Load and performance tests will be performed on the configured cluster to evaluate the efficiency and reliability of the system. Adjustments will also be made to its configuration based on the results obtained to optimize performance and availability.

**Keywords:** High performance service; load balancing; high availability; performance evaluation; computer cluster.

---

## INDICE DE CONTENIDO

AGRADECIMIENTOS.....	iii
RESUMEN .....	iv
ABSTRACT .....	v
1 INTRODUCCIÓN .....	1
1.1 MOTIVACIÓN .....	1
1.2 OBJETIVO .....	2
1.3 OBJETIVOS ESPECIFICOS .....	2
1.4 METODOLOGÍA .....	2
1.5 ESTRUCTURA.....	3
2 CONCEPTOS BASICOS Y ESTADO DEL ARTE .....	5
2.1 VIRTUALIZACION .....	5
2.1.1 Importancia .....	6
2.1.2 Componentes .....	6
2.1.3 Tipos de Virtualización .....	7
2.1.4 Ventajas y Desventajas.....	8
2.1.5 Plataformas de Virtualización.....	9
2.2 SISTEMA OPERATIVO DEBIAN.....	9
2.2.1 Ventajas y Desventajas.....	10
2.3 ALTA DISPONIBILIDAD.....	11
2.3.1 Funcionamiento de la alta disponibilidad .....	11
2.3.2 Métricas de Alta Disponibilidad.....	12
2.3.3 Clústeres de Alta Disponibilidad.....	13
2.3.4 Configuraciones de Alta Disponibilidad.....	15
2.3.5 Implementación de un Clúster HA .....	16
2.3.6 Implementación de un Clúster con equilibrado de carga .....	20
2.3.7 Integración de HAProxy con Keepalived y Corosync+Pacemaker .....	23
2.4 SISTEMAS DE ALMACENAMIENTO .....	24
2.4.1 Soluciones de almacenamiento.....	25
2.5 CONTENEDORES.....	28
2.5.1 Plataformas para la implementación de contenedores .....	28

---

3	DISEÑO E IMPLEMENTACIÓN DEL SISTEMA.....	32
3.1	PROPUESTA.....	32
3.2	DESCRIPCIÓN DE LA PROPUESTA.....	33
3.3	IMPLEMENTACION DEL CLÚSTER.....	34
3.4	DESARROLLO DE LA IMPLEMENTACIÓN .....	37
3.4.1	Instalación del nodo servidor de almacenamiento NFS.....	38
3.4.2	Instalación del nodo Máster .....	43
3.4.3	Instalación de nodos Standby.....	63
3.4.4	Instalación de sistema de almacenamiento GlusterFS.....	67
3.4.5	Configuración de contenedores .....	71
4	PRUEBAS Y RESULTADOS.....	76
4.1	PRUEBAS DE FUNCIONALIDAD.....	77
4.1.1	Alta Disponibilidad con Corosync y Pacemaker .....	77
4.1.2	Alta Disponibilidad con Keepalived .....	84
4.1.3	Reparto de carga con HAProxy.....	88
4.1.4	Uso de sistema de almacenamiento con GlusterFS .....	100
4.1.5	Uso de sistema de almacenamiento con NFS .....	106
4.2	EVALUACIÓN DE PRESTACIONES .....	110
4.2.1	Resultados para página web estática .....	111
4.2.2	Resultados para página web dinámica .....	117
4.2.3	Resultados para página web estática con imagen .....	121
5	CONCLUSIONES Y RECOMENDACIONES.....	126
6	BIBLIOGRAFÍA.....	129

---



## INDICE DE FIGURAS

FIGURA 1 – TIPOS DE HIPERVISORES	7
FIGURA 2 - CLÚSTER DE ALTA DISPONIBILIDAD	14
FIGURA 3 - CONFIGURACIÓN DE CLÚSTER ACTIVO/ACTIVO	15
FIGURA 4 - CONFIGURACIÓN DE CLÚSTER ACTIVO/PASIVO	16
FIGURA 5 - GESTIÓN DE IPS CON KEEPALIVED	18
FIGURA 6 - PACEMAKER ARQUITECTURA DE ALTO NIVEL	20
FIGURA 7 - EJEMPLO DE DISEÑO DE UN CLÚSTER CON EQUILIBRIO DE CARGA	21
FIGURA 8 - ARQUITECTURA DE DOCKER	29
FIGURA 9 - DISEÑO LÓGICO DE CLÚSTER TFM	33
FIGURA 10 - VERSIÓN DE VIRTUALBOX	35
FIGURA 11 - DISEÑO LÓGICO DE CLÚSTER TFM	35
FIGURA 12 - DISEÑO FÍSICO DE CLÚSTER TFM	37
FIGURA 13 - DISEÑO DE CLÚSTER TFM	38
FIGURA 14 - CREACIÓN DE RAID	39
FIGURA 15 - FICHERO DE CREACIÓN DEL RAID	39
FIGURA 16 - DETALLES DEL RAID	40
FIGURA 17 - DIRECTORIO DEL RAID	40
FIGURA 18 - CONFIGURACIÓN NFS SERVER	41
FIGURA 19 - DIRECTORIO EXPORTADO POR NFS SERVER	41
FIGURA 20 - LISTADO DE DIRECTORIOS EXPORTADOS POR NFS SERVER	41
FIGURA 21 - FICHERO /ETC/FSTAB	42
FIGURA 22 - CONFIGURACIÓN LINK AGGREGATION	42
FIGURA 23 - TARJETA BOND0	43
FIGURA 24 - ARCHIVO DE CONFIGURACIÓN DE RED NODO MÁSTER	44
FIGURA 25 - ARCHIVO DE CONFIGURACIÓN DE DNS NODO MÁSTER	44
FIGURA 26 - TABLA DE RUTAS NODO MÁSTER	45
FIGURA 27 - ARCHIVO DE CONFIGURACIÓN DNSMASQ - PARTE 1	45
FIGURA 28 - ARCHIVO DE CONFIGURACIÓN DNSMASQ - PARTE 2	46
FIGURA 29 - ARCHIVO DE ASIGNACIÓN DE NOMBRES POR DHCP	46
FIGURA 30 - ARCHIVO DE RESERVAS DE DHCP	46
FIGURA 31 - SERVICIO DNSMASQ	47
FIGURA 32 - ARCHIVO DE CONFIGURACIÓN HA PROXY - PARTE 1	48
FIGURA 33 - ARCHIVO DE CONFIGURACIÓN HA PROXY - PARTE 2	49
FIGURA 34 - ARCHIVO HOSTS	51
FIGURA 35 - ARCHIVO DE CONFIGURACIÓN KEEPALIVED	52
FIGURA 36 - INSTALACIÓN DE PAQUETES PACEMAKER	53
FIGURA 37 - INSTALACIÓN DE PAQUETES DE ADMINISTRACIÓN PACEMAKER	53
FIGURA 38 - ARCHIVO DE CONFIGURACIÓN COROSYNC	54
FIGURA 39 - COPIAR ARCHIVO DE CONFIGURACIÓN COROSYNC	55
FIGURA 40 - CAMBIO DE CONTRASEÑA HA CLUSTER	55
FIGURA 41 - AUTORIZACIÓN NODOS DEL CLÚSTER	55
FIGURA 42 - INICIAR Y HABILITAR SERVICIO DE COROSYNC	56
FIGURA 43 - ESTADO DEL SERVICIO DE PACEMAKER	56
FIGURA 44 - HABILITAR SERVICIO DE PACEMAKER	56
FIGURA 45 - ESTADO DEL SERVICIO PCSD	57
FIGURA 46 - COMANDOS DE ADMINISTRACIÓN DE PACEMAKER	57
FIGURA 47 - COMANDOS DE ADMINISTRACIÓN DE PACEMAKER	58

---

FIGURA 48 - COMANDOS DE ADMINISTRACIÓN DE PACEMAKER	58
FIGURA 49 - CREACIÓN DE RECURSOS EN EL CLÚSTER	59
FIGURA 50 - CREACIÓN DE RECURSO IP EN EL CLÚSTER	59
FIGURA 51 - VERIFICAR ESTADO DE LOS RECURSOS DEL CLÚSTER	59
FIGURA 52 - USO DE HERRAMIENTA CRM CONFIGURE	60
FIGURA 53 - VERIFICAR CONFIGURACIÓN DEL CLÚSTER HERRAMIENTA CRM	61
FIGURA 54 - VERIFICAR CONFIGURACIÓN DEL CLÚSTER HERRAMIENTA PCS	62
FIGURA 55 - VERIFICAR ESTADO DEL CLÚSTER HERRAMIENTA PCS	63
FIGURA 56 - MODIFICACIÓN HAPROXY NODO STANDBY1	65
FIGURA 57 - MODIFICACIÓN HAPROXY NODO STANDBY2	65
FIGURA 58 - MODIFICACIÓN KEEPALIVED NODO STANDBY1	66
FIGURA 59 - MODIFICACIÓN KEEPALIVED NODO STANDBY2	67
FIGURA 60 - FICHERO /ETC/FSTAB NODOS GLUSTER	68
FIGURA 61 - ESTADO DE SERVICIO GLUSTERD	68
FIGURA 62 - CREACIÓN Y ESTADO DE GRUPO GLUSTER	69
FIGURA 63 - CREACIÓN E INICIO DE VOLUMEN REPLICADO	69
FIGURA 64 - INFORMACIÓN DE LOS VOLÚMENES	70
FIGURA 65 - CONFIGURACIÓN DE GLUSTER-CLIENT	71
FIGURA 66 - MONTAJE DE VOLÚMENES GLUSTER	71
FIGURA 67 - CREACIÓN DE CONTENEDOR	72
FIGURA 68 - LISTADO DE CONTENEDORES CLUSTER2	73
FIGURA 69 - MAPEO DE PUERTOS CON HOST	73
FIGURA 70 - LISTADO DE CONTENEDORES CLUSTER3	74
FIGURA 71 - LISTADO DE CONTENEDORES CLUSTER4	74
FIGURA 72 - FLUJO DE PETICIONES WEB SERVER	75
FIGURA 73 - ESTADO DEL CLÚSTER HA	77
FIGURA 74 - RECURSOS DEL CLÚSTER EN NODO MÁSTER	78
FIGURA 75 - ASIGNACIÓN DE IPS VIRTUALES NODO MÁSTER	78
FIGURA 76 - REVISIÓN LOGS DE SERVICIO HAPROXY	79
FIGURA 77 - LOGS DE SERVICIO HAPROXY - NODO MÁSTER ACTIVO	79
FIGURA 78 - PRUEBA DE DESCONEXIÓN NODO MÁSTER - I	80
FIGURA 79 - PRUEBA DE DESCONEXIÓN NODO MÁSTER - II	80
FIGURA 80 - PRUEBA NODO MÁSTER OFFLINE	81
FIGURA 81 - PRUEBA NODO MÁSTER OFFLINE	81
FIGURA 82 - PRUEBA NODO MÁSTER OFFLINE	82
FIGURA 83 - PRUEBA NODO MÁSTER Y STANDBY1 OFFLINE	82
FIGURA 84 - PRUEBA NODO MÁSTER Y STANDBY1 OFFLINE	83
FIGURA 85 - PRUEBA NODO MÁSTER Y STANDBY1 OFFLINE	83
FIGURA 86 - COMANDO REVISIÓN DE LOGS KEEPALIVED	84
FIGURA 87 - KEEPALIVED NODO MAESTRO	84
FIGURA 88 - KEEPALIVED NODOS STAND-BY	84
FIGURA 89 - KEEPALIVED MANEJO DE IPV	85
FIGURA 90 - DESCONEXIÓN NODO MAESTRO	85
FIGURA 91 - KEEPALIVED TRANSFERENCIA DE ROL E IPVS - STANDBY1	86
FIGURA 92 - DESCONEXIÓN NODO STANDBY1	86
FIGURA 93 - KEEPALIVED TRANSFERENCIA DE ROL E IPVS - STANDBY2	87
FIGURA 94 - CONEXIÓN NODO MAESTRO	87
FIGURA 95 - KEEPALIVED - IPVS EN NODO MAESTRO	88
FIGURA 96 - FLUJO DE REQUERIMIENTOS Y RESPUESTA HA PROXY	89
FIGURA 97 - REDIRECCIÓN DE PUERTOS EN VIRTUALBOX	90
FIGURA 98 - PRUEBA INTERNA DE FUNCIONAMIENTO DE HAPROXY	92
FIGURA 99 - PRUEBA EXTERNA DE FUNCIONAMIENTO DE HAPROXY - MODO TCP	93
FIGURA 100 - PRUEBA EXTERNA DE FUNCIONAMIENTO DE HAPROXY - MODO HTTP	94

---

---

FIGURA 101 - ESTADÍSTICAS DE HAPROXY	95
FIGURA 102 - ESTADÍSTICAS DE HAPROXY CON SOBRECARGA	96
FIGURA 103 - PRUEBA DE CONTENEDORES DETENIDOS	97
FIGURA 104 - HAPROXY DETECCIÓN DE CONTENEDORES CAÍDOS	98
FIGURA 105 - SIMULACIÓN CAÍDA NODO CLUSTER3	98
FIGURA 106 - ESTADÍSTICAS DE HAPROXY DETECCIÓN DE CAÍDA DE NODOS	99
FIGURA 107 - ESTADÍSTICAS DE HAPROXY DETECCIÓN DE CAÍDA NODO MÁSTER	100
FIGURA 108 - VOLÚMENES GLUSTERS MONTADOS	101
FIGURA 109 - VOLUMEN REPLICADO GLUSTER1	102
FIGURA 110 - VOLUMEN REPLICADO GLUSTER2	102
FIGURA 111 - VOLUMEN REPLICADO GLUSTER3	102
FIGURA 112 - CONTENIDO DE DIRECTORIO REPLICADO DE GLUSTERFS	103
FIGURA 113 - VOLUMEN DISTRIBUIDO GLUSTER1	103
FIGURA 114 - VOLUMEN DISTRIBUIDO GLUSTER2	103
FIGURA 115 - VOLUMEN DISTRIBUIDO GLUSTER3	104
FIGURA 116 - CONTENIDO DE DIRECTORIO DISTRIBUIDO DE GLUSTER	104
FIGURA 117 - PRUEBA DE DESCONEXIÓN NODO GLUSTER1	104
FIGURA 118 - ACCESO A DIRECTORIO REPLICADO	105
FIGURA 119 - ESTADO DE NODOS GUSTERFS	105
FIGURA 120 - DIRECTORIO APACHE EN GUSTERFS	106
FIGURA 121 - DISPONIBILIDAD DE PÁGINA WEB CON GLUSTERFS	106
FIGURA 122 - CONTENIDO DIRECTORIO NFS	107
FIGURA 123 - PUNTO DE MONTAJE SISTEMA NFS	107
FIGURA 124 - PUNTO DE MONTAJE SISTEMA NFS	108
FIGURA 125 - FALLA DE DISCO EN RAID	108
FIGURA 126 - ACCESO A REPOSITORIO NFS CON FALLA EN RAID	109
FIGURA 127 - REMOVER Y AGREGAR DISPOSITIVO AL RAID	109
FIGURA 128 - DISPONIBILIDAD SERVIDOR NFS	110
FIGURA 129 - PÁGINA WEB ESTÁTICA	112
FIGURA 130 - MUESTRA DE PÁGINA WEB ESTÁTICA	112
FIGURA 131 - REPOSITORIO APACHE2 EN NFS	112
FIGURA 132 - EJECUCIÓN DE APACHE BENCHMARK	114
FIGURA 133 - REQUEST PER SECOND PÁGINA ESTÁTICA CON N=10000	116
FIGURA 134 - TIME PER REQUEST PÁGINA ESTÁTICA CON N=10000	117
FIGURA 135 - PAGINA WEB DINÁMICA	118
FIGURA 136 - MUESTRA DE PÁGINA WEB DINÁMICA	118
FIGURA 137 - EVALUACIÓN DE AB EN PÁGINAS WEB DINÁMICAS	119
FIGURA 138 - REQUEST PER SECOND PÁGINA DINÁMICA CON N=5000	120
FIGURA 139 - TIME PER REQUEST PÁGINA DINÁMICA CON N=5000	121
FIGURA 140 - MUESTRA DE PÁGINA WEB ESTÁTICA CON IMAGEN	122
FIGURA 141 - REPOSITORIO APACHE2 EN GLUSTERFS	122
FIGURA 142 - REQUEST PER SECOND PÁGINA ESTÁTICA CON IMAGEN N=10000	124
FIGURA 143 - TIME PER REQUEST PÁGINA ESTÁTICA CON IMAGEN N=10000	124

---

## INDICE DE TABLAS

TABLA 1 - CONFIGURACIÓN DE RED NODO NFS .....	38
TABLA 2 - CONFIGURACIÓN DE RED NODO MÁSTER .....	44
TABLA 3 - DIRECCIONES IPS DEL CLÚSTER.....	50
TABLA 4 - CONFIGURACIÓN DE RED NODO STANDBY1 .....	64
TABLA 5 - CONFIGURACIÓN DE RED NODO STANDBY2 .....	64
TABLA 6 - INFORMACIÓN DE LOS VOLÚMENES .....	70
TABLA 7 - MAPEO DE PUERTOS HOSTS Y CONTENEDORES .....	76
TABLA 8 - REDIRECCIÓN DE PUERTOS VIRTUALBOX .....	90
TABLA 9 - LISTADO DE DIRECCIONES IPS/PUERTOS DEL CLÚSTER.....	91
TABLA 10 - VOLÚMENES GLUSTERFS.....	100
TABLA 11 - RESULTADOS EVALUACIÓN AB PÁGINA ESTÁTICA.....	115
TABLA 12 - INTERVALOS PARA USO DE PÁGINA WEB DINÁMICA.....	119
TABLA 13 - RESULTADOS EVALUACIÓN AB PÁGINA DINÁMICA .....	120
TABLA 14 - RESULTADOS EVALUACIÓN PÁGINA ESTÁTICA CON IMAGEN .....	123

---

# 1 INTRODUCCIÓN

En la era digital actual, la disponibilidad continua de los servicios web se ha vuelto esencial para las organizaciones, pues prácticamente todas las facetas del negocio moderno dependen de las aplicaciones web. Desde el comercio electrónico hasta la banca en línea y los sistemas de gestión empresarial, estas aplicaciones deben estar disponibles para los usuarios en todo momento. La mínima interrupción en el servicio puede tener consecuencias significativas, como la pérdida de ingresos, el deterioro de la reputación de la empresa y la disminución de la confianza de los clientes.

Con el crecimiento exponencial en la demanda de servicios web, garantizar la alta disponibilidad se ha convertido en una prioridad tanto para los administradores de sistemas como para los desarrolladores de aplicaciones. La alta disponibilidad (HA) se refiere a la capacidad de un sistema para seguir funcionando a pesar de fallos en alguno de sus componentes. Esto se logra mediante la implementación de redundancia y mecanismos de conmutación por error, que aseguran que, si un componente falla, otro puede tomar su lugar sin interrumpir el servicio.

Por otro lado, el equilibrado de carga ha surgido como una técnica clave para gestionar de manera eficiente el tráfico de red entrante, distribuyéndolo entre varios servidores. El objetivo es evitar que un solo servidor se sature, lo que podría causar tiempos de respuesta lentos o, en el peor de los casos, la caída del sistema. Al distribuir el tráfico de forma inteligente entre varios servidores, se garantiza un uso óptimo de los recursos, mejorando la experiencia del usuario y aumentando la capacidad para manejar picos inesperados de tráfico.

En este contexto, mi trabajo de fin de máster se enfoca en desarrollar un sistema de servidor web que ofrezca alta disponibilidad con equilibrado de carga, utilizando un clúster de computadores. Este sistema podría aplicarse en una variedad de escenarios, desde pequeñas empresas que necesitan mejorar la resiliencia de sus servicios web, hasta grandes corporaciones que deben manejar altos volúmenes de tráfico con gran eficiencia.

## 1.1 MOTIVACIÓN

La motivación principal para llevar a cabo este trabajo de fin de máster es consolidar los conocimientos adquiridos durante el máster, aplicándolos de manera práctica en un sistema altamente demandado. Hoy en día, es crucial contar con sistemas de clúster que no solo garanticen la alta disponibilidad de un servidor web, sino que también optimicen el uso de los recursos a través de un equilibrado de carga eficiente.

A medida que las aplicaciones web continúan evolucionando, la demanda por soluciones capaces de escalar y manejar grandes volúmenes de tráfico de forma eficaz sigue creciendo. Además, con la creciente adopción de infraestructuras basadas en la nube y contenedores, se presenta una gran oportunidad para explorar nuevas arquitecturas de clúster que puedan satisfacer estas necesidades de manera más efectiva.

## 1.2 OBJETIVO

El objetivo principal de este trabajo es diseñar, implementar y evaluar un sistema de clúster de computadores que permita garantizar la alta disponibilidad y el equilibrado de carga para un servidor web. Con esto, se busca asegurar que el servicio esté siempre disponible, incluso si ocurren fallos en el hardware o el software, manteniendo un buen rendimiento y una respuesta rápida.

Para lograr este objetivo, se construirá un clúster que incorpore tecnologías que aseguren la alta disponibilidad, como la redundancia y la conmutación por error automática, junto con mecanismos de equilibrado de carga que repartan las solicitudes de manera eficiente entre varios servidores. Además, se explorarán diferentes tecnologías para gestionar el almacenamiento de datos, asegurando que estos sean accesibles y consistentes en todo momento.

Finalmente, se llevarán a cabo pruebas para medir qué tan bien funciona el sistema, evaluando su disponibilidad, el tiempo de respuesta bajo diferentes cargas, su capacidad para manejar grandes volúmenes de tráfico y cómo se comporta ante posibles fallos. Estos resultados permitirán verificar si el clúster es robusto y escalable en distintos escenarios.

## 1.3 OBJETIVOS ESPECIFICOS

- ▶ **Diseñar una arquitectura de clúster** que asegure la alta disponibilidad, implementando redundancia y mecanismos de conmutación por error para mantener el sistema siempre operativo.
- ▶ **Implementar y configurar un sistema de equilibrado de carga** que reparta el tráfico de manera eficiente entre los diferentes nodos del clúster, optimizando así el uso de los recursos y mejorando la respuesta del sistema.
- ▶ **Configurar un sistema de archivos distribuido y paralelo** que permita gestionar el almacenamiento de datos de manera eficiente dentro del clúster, garantizando que los datos estén siempre disponibles.
- ▶ **Evaluar el rendimiento del sistema** a través de pruebas que midan aspectos clave como la disponibilidad, la latencia y la capacidad de manejo de tráfico bajo distintas condiciones.

## 1.4 METODOLOGÍA

Para desarrollar este trabajo de fin de máster, opté por una metodología "**hands-on**", es decir, un enfoque muy práctico y directo. La idea fue centrarme en el diseño, la implementación y las pruebas de un sistema de clúster que garantizara alta disponibilidad y equilibrado de carga. Este método me permitió no solo poner en práctica

los conocimientos teóricos que adquiriré durante el máster, sino también explorar y experimentar con tecnologías que tienen aplicaciones reales.

El proceso comenzó con el diseño del clúster, donde elegí las tecnologías más adecuadas basándome en criterios como la simplicidad, la disponibilidad de uso de recursos y la viabilidad del proyecto. Decidí utilizar la virtualización como herramienta principal para evaluar la funcionalidad del clúster, empleando VirtualBox para crear un entorno simulado que replicara lo más fielmente posible un sistema real. Escogí **Debian 12** como sistema operativo base debido a su licencia abierta y a su estabilidad en entornos de producción.

Con estas bases establecidas, me dediqué a crear y configurar las máquinas virtuales que formarían parte del clúster de alta disponibilidad. Cada máquina fue ajustada para cumplir con su función específica dentro del clúster, incluyendo la instalación y configuración de servicios clave. Después, realicé múltiples pruebas para asegurarme de que el sistema mantuviera su disponibilidad de manera continua.

El siguiente paso fue configurar el sistema de equilibrado de carga, implementando las tecnologías necesarias para repartir de manera eficiente las solicitudes entre los distintos nodos del clúster. Al igual que con la alta disponibilidad, se llevaron a cabo pruebas específicas para validar la efectividad del equilibrado de carga.

Paralelamente, diseñé y configuré los sistemas de almacenamiento compartido necesarios para el esquema de clúster. Comencé implementando un RAID para compartir recursos por NFS y gestionar el almacenamiento compartido, luego pasé a GlusterFS para asegurar una mayor flexibilidad y escalabilidad en la gestión de los datos.

Con todo el entorno ya configurado y funcionando, realicé pruebas de carga para evaluar el rendimiento general del sistema. Estas pruebas me proporcionaron datos esenciales para analizar la capacidad del clúster y llegar a conclusiones sobre su aplicabilidad en escenarios reales.

Definitivamente, este enfoque **"hands-on"** me permitió desarrollar un trabajo de fin de máster que no solo es sólido en teoría, sino que también es altamente práctico y aplicable en la vida real. Cada paso del proceso estuvo orientado a crear un sistema funcional, capaz de enfrentar las demandas actuales en términos de disponibilidad, rendimiento y escalabilidad.

## 1.5 ESTRUCTURA

Este documento se estructura en varios capítulos, cada uno de los cuales aborda aspectos clave del diseño, implementación y evaluación del sistema propuesto:

- ▶ **Capítulo 2: Conceptos Básicos y Estado del Arte.** En este capítulo se presentan los conceptos fundamentales relacionados con clústeres de computadores, alta disponibilidad y equilibrado de carga, además de revisar el estado actual de las tecnologías y prácticas en este campo.

- ▶ **Capítulo 3: Diseño e Implementación del Sistema.** Este capítulo se detalla la arquitectura del sistema propuesto, incluyendo con mayor detalle la selección de tecnologías, estrategias de alta disponibilidad y mecanismos de equilibrado de carga. Además, se describen los pasos necesarios para la configuración del entorno, la implementación del clúster y sus recursos, el despliegue de las aplicaciones, así como el diseño de las pruebas realizadas para validar el sistema.
- ▶ **Capítulo 4: Resultados y Análisis.** En este capítulo se presentan los resultados de las pruebas por cada una de las tecnologías aplicadas en el proceso de implementación y configuración del clúster, tanto para la parte de alta disponibilidad como para el sistema de equilibrado de carga, y se realiza un análisis del rendimiento del sistema.
- ▶ **Capítulo 5: Conclusiones y Trabajo Futuro.** Como parte final se presentan las conclusiones del trabajo, las limitaciones encontradas y se sugieren posibles opciones de implementación futura.
- ▶ **Capítulo 6: Referencias.** En este capítulo se listan todas las fuentes y referencias bibliográficas utilizadas a lo largo del trabajo.



## 2 CONCEPTOS BASICOS Y ESTADO DEL ARTE

En este capítulo revisaremos los conceptos fundamentales y el estado del arte de las tecnologías que se utilizarán en el diseño, implementación y configuración del clúster de alta disponibilidad con equilibrado de carga. La comprensión de estos conceptos es esencial para contextualizar las decisiones técnicas adoptadas a lo largo de este trabajo y para entender cómo cada componente contribuye al objetivo de garantizar un servicio web robusto, escalable y eficiente.

Iniciaremos revisando el tema de la virtualización, una tecnología clave que permite la creación de entornos simulados, como es el caso de VirtualBox, utilizado en este proyecto para replicar el entorno de producción. A continuación, mostraremos el sistema operativo Debian 12, elegido por su estabilidad y características que lo hacen ideal para la implementación de servidores en entornos críticos.

Posteriormente, abordaremos las tecnologías específicas de alta disponibilidad, comenzando con Keepalived y luego profundizando en el stack Corosync+Pacemaker+STONITH, que forman el núcleo de la gestión de recursos de alta disponibilidad en el clúster. Seguidamente, hablaremos sobre las herramientas utilizadas para el equilibrado de carga, en particular HAProxy, que permite una distribución eficiente del tráfico entre los nodos del clúster.

En cuanto al almacenamiento, exploraremos las tecnologías RAID, NFS y GlusterFS, fundamentales para asegurar la redundancia, disponibilidad y escalabilidad del almacenamiento en un entorno de clúster. Finalmente, cubriremos los aspectos de redes, incluyendo la técnica de link bonding, utilizada para optimizar el rendimiento y la disponibilidad de las conexiones de red en el clúster.

Este capítulo tiene como objetivo proporcionar una base teórica sólida y una comprensión clara de las tecnologías utilizadas en el ambiente, permitiendo así una mejor comprensión de su aplicación práctica en el contexto del clúster de alta disponibilidad y equilibrado de carga que se desarrollará en los capítulos siguientes.

### 2.1 VIRTUALIZACION

La virtualización es una tecnología que se puede usar para crear representaciones virtuales de servidores, almacenamiento, redes y otras máquinas físicas. El software virtual imita las funciones del hardware físico para ejecutar varias máquinas virtuales a la vez en una única máquina física. Las empresas recurren a la virtualización para utilizar sus recursos de hardware de manera eficiente y obtener retornos mayores de sus inversiones. <sup>(1)</sup> (AWS, 2024)

Esto significa que, en lugar de tener varios servidores físicos configurados, las organizaciones pueden consolidar sus recursos y ejecutar varias máquinas virtuales en un solo servidor, de esta manera tenemos la oportunidad de tener soluciones reales para optimizar el uso de nuestros servidores o computadoras, todo lo cual brinda la oportunidad de repotenciar los equipos ya existentes o disponer de nuevos equipos funcionando y operativos en el menor tiempo posible.

## 2.1.1 Importancia

Al utilizar la virtualización, es posible interactuar con cualquier recurso de *hardware* con mayor flexibilidad. Los servidores físicos consumen electricidad, ocupan espacio de almacenamiento y necesitan mantenimiento. Con frecuencia el acceso a estos está limitado por la proximidad física y el diseño de la red. La virtualización resuelve todas estas limitaciones al abstraer la funcionalidad del *hardware* físico en el *software*. Es posible administrar, mantener y utilizar la infraestructura de *hardware* como una aplicación en la web.

## 2.1.2 Componentes

Las máquinas virtuales y los hipervisores son dos conceptos importantes en la virtualización. <sup>(1)</sup> (AWS, 2024)

### Máquina virtual

Una *máquina virtual* es un equipo definido por software que se ejecuta en un equipo físico con un sistema operativo y recursos informáticos independientes. La computadora física se denomina *máquina host* y las máquinas virtuales son *máquinas invitadas* (*guest*). Se pueden ejecutar varias máquinas virtuales en una sola máquina física. Un hipervisor extrae las máquinas virtuales del hardware de la computadora.

### Hipervisor

El *hipervisor* es un componente de software que administra varias máquinas virtuales en una computadora. Garantiza que cada máquina virtual reciba los recursos asignados y no interfiera con el funcionamiento de otras máquinas virtuales. Existen dos tipos de hipervisores.

#### ► *Hipervisor de tipo 1*

Un hipervisor de tipo 1, o hipervisor bare metal, es un programa de hipervisor que, en vez de instalarse en el sistema operativo, se instala de forma directa en el hardware de la computadora. Por lo tanto, los **hipervisores de tipo 1 tienen un mejor rendimiento y se utilizan con frecuencia para las aplicaciones empresariales**. KVM utiliza el hipervisor de tipo 1 para alojar varias máquinas virtuales en el sistema operativo Linux.

#### ► Hipervisor de tipo 2

También conocido como hipervisor alojado, el hipervisor de tipo 2 está instalado en un sistema operativo. **Los hipervisores de tipo 2 son adecuados para la informática del usuario final.**

En la figura 1, se puede apreciar la arquitectura de cada tipo de hipervisor y adicional se muestran ciertas plataformas reconocidas por cada uno de ellos, entre comerciales y tipo opensource.

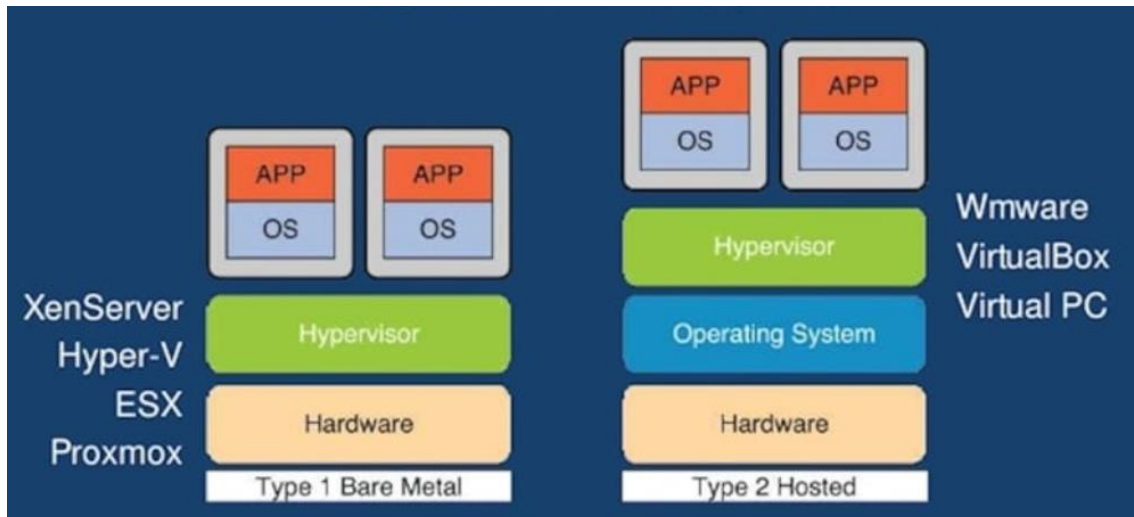


Figura 1 – Tipos de Hipervisores

Tomado de: “La Virtualización: sus tipos, ventajas e hipervisores”. Accedido el 5 de agosto de 2024. [En línea]. Disponible: <https://universodigital.org/virtualizacion/>

### 2.1.3 Tipos de Virtualización

La virtualización permite replicar las funciones que podemos encontrar en una infraestructura física, pero aprovechando todas las ventajas de un entorno virtualizado. No solo se trata de máquinas virtuales sino de todo un conjunto de recursos virtuales en el entorno virtual. <sup>(1)</sup> (AWS, 2024)

Existen distintos tipos de virtualización entre los que destacan:

- ▶ **Virtualización de servidores:** es un proceso que particiona un servidor físico en múltiples servidores virtuales. Es una forma eficaz y rentable de utilizar los recursos del servidor y de implementar los servicios de TI en una organización. Sin la virtualización de servidores, los servidores físicos únicamente aprovechan una pequeña cantidad de sus capacidades de procesamiento, lo que provoca que los dispositivos queden inactivos.
- ▶ **Virtualización del almacenamiento:** La virtualización del almacenamiento combina las funciones de los dispositivos de almacenamiento físico, como el almacenamiento conectado a la red (NAS) y la red de área de almacenamiento (SAN). Se puede agrupar el *hardware* de almacenamiento del centro de datos, aunque sea de diferentes proveedores o de diferentes tipos. La virtualización del almacenamiento utiliza todo el almacenamiento físico de datos y crea una gran unidad de almacenamiento virtual que se puede asignar y controlar mediante un *software* de administración.
- ▶ **Virtualización de red:** es un proceso que combina varios recursos de red (conmutadores, enrutadores, firewalls de múltiples ubicaciones geográficas) para centralizar las tareas administrativas. Los administradores pueden ajustar y controlar estos elementos virtualmente sin tocar los componentes físicos, lo que simplifica enormemente la administración de la red.

- ▶ **Virtualización de datos:** las organizaciones modernas recopilan datos de varios orígenes y los almacenan en diferentes formatos. También pueden almacenar datos en diferentes lugares, como en una infraestructura en la nube y en un centro de datos local. La virtualización de datos crea una capa de *software* entre estos datos y las aplicaciones que los necesitan. Las herramientas de virtualización de datos procesan la solicitud de datos de una aplicación y devuelven los resultados en un formato adecuado.
- ▶ **Virtualización de aplicaciones:** extrae las funciones de las aplicaciones de modo que se ejecuten en sistemas operativos distintos de aquellos para los que fueron diseñadas. Por ejemplo, los usuarios pueden ejecutar una aplicación de Microsoft Windows en una máquina Linux sin cambiar la configuración de la máquina.
- ▶ **Virtualización de escritorios:** a través de este tipo de virtualización podemos definir diferentes sistemas operativos de escritorio en máquinas virtuales y los usuarios pueden acceder a estos equipos de forma remota. Esto permite a los administradores poder gestionar los escritorios de forma eficiente y segura, con lo que se ahorran dinero en *hardware* de escritorio.

### 2.1.4 Ventajas y Desventajas

La virtualización ofrece varias ventajas a los administradores y organizaciones en general, entre las principales se destacan: <sup>(2)</sup> (Barraza, 2023)

- ▶ **Consolidación de servidores:** permite consolidar varias máquinas virtuales en un único servidor físico, ahorrando espacio, energía y costos de refrigeración.
- ▶ **Eficiencia de costes:** Al ejecutar varias máquinas virtuales en un solo servidor, se reducen los gastos de hardware y mantenimiento.
- ▶ **Aislamiento:** Cada máquina virtual funciona de forma independiente, proporcionando un mejor aislamiento y seguridad entre aplicaciones y sistemas operativos.
- ▶ **Flexibilidad y escalabilidad:** Las máquinas virtuales son fáciles de crear, clonar y migrar, lo que facilita la escalabilidad.

Como desventajas de esta tecnología podemos citar:

- ▶ **Rendimiento:** A veces, la virtualización puede afectar el rendimiento debido a la sobrecarga del hipervisor.
- ▶ **Licencias:** Algunas soluciones de virtualización requieren licencias adicionales.
- ▶ **Complejidad:** La configuración y administración de entornos virtuales pueden ser complejas.

A pesar de las desventajas, la virtualización es una tecnología muy esparcida en la actualidad y más con su extensión hacia la computación en la nube, donde los grandes

centros de datos se basan en virtualización y ofrecen servicios sobre demanda y requerimientos de los usuarios y/o organizaciones.

### 2.1.5 Plataformas de Virtualización

Existen diferentes tipos de plataforma de virtualización, según los requerimientos de las organizaciones y el costo que estén dispuestos a asumir para su implementación o adquisición. Tanto en el ámbito comercial como en el esquema de código abierto, las principales que podemos destacar son:

- ▶ **Comerciales:** VMware vSphere, Microsoft Hyper-V, Citrix XenServer.
- ▶ **Open Source:** KVM (Kernel-based Virtual Machine), VirtualBox, Proxmox.

En ese sentido, dada su simplicidad, soporte multiplataforma y por ser de código abierto, VirtualBox ha sido la plataforma de virtualización escogida para el desarrollo de este proyecto de fin de máster. Sobre esta plataforma de virtualización se levantan todos los equipos virtuales que forman parte de la arquitectura del clúster.

VirtualBox es un potente producto de virtualización para arquitecturas x86 y AMD64 / Intel64 para uso empresarial y doméstico. VirtualBox no solo es un producto extremadamente rico en funciones y de alto rendimiento para clientes empresariales, sino que también es la única solución profesional que está disponible gratuitamente como Software de Código Abierto bajo los términos de GNU General Public License (GPL) versión 3. <sup>(3)</sup> (VirtualBox, 2024)

Actualmente, VirtualBox se ejecuta en hosts Windows, Linux, macOS y Solaris y admite una gran cantidad de sistemas operativos invitados, incluidos, entre otros, Windows (NT 4.0, 2000, XP, Server 2003, Vista, 7, 8, Windows 10 y Windows 11), DOS / Windows 3. x, Linux (2.4, 2.6, 3.x, 4.x, 5.x y 6.x), Solaris y OpenSolaris, OS / 2, OpenBSD, NetBSD y FreeBSD.

VirtualBox se está desarrollando activamente con lanzamientos frecuentes y tiene una lista cada vez mayor de funciones, sistemas operativos invitados compatibles y plataformas en las que se ejecuta.

## 2.2 SISTEMA OPERATIVO DEBIAN

Existen múltiples tipos de sistemas operativos que se pueden seleccionar para utilizar como software base para los equipos, sean físicos o virtuales. Los hay de tipo comercial o los tan conocidos de licencia OpenSource, con múltiples ventajas o desventajas según los requerimientos o exigencias del administrador, desarrollador u organización.

Para el entorno de trabajo de este proyecto se escogió como sistema operativo base para todas las máquinas que forman parte de la arquitectura del clúster, tanto para los equipos front-end como back-end, la distribución Linux DEBIAN en su última versión disponible a fecha Julio/2024, esto es Debian 12 (bookworm).

Debian es una distribución de Linux conocida por su estabilidad y seguridad, ideal para entornos de servidor y producción. Hay muchas razones por las que los usuarios eligen Debian como su sistema operativo – ya sea como usuario/a, para desarrollar, o incluso en entornos corporativos. La mayoría de los usuarios aprecian la estabilidad y los procesos de actualización de paquetes o la distribución entera sin complicaciones. Debian también se usa ampliamente en el sector de desarrollo de software y hardware porque funciona en numerosas arquitecturas y dispositivos, y ofrece un sistema de seguimiento de incidencias público y otras herramientas para el desarrollo. <sup>(4)</sup> (Razones para escoger Debian, 2024)

## 2.2.1 Ventajas y Desventajas

A continuación, mencionamos alguna ventajas y desventajas del uso de este sistema operativo:

### Ventajas:

1. **Distribución libre y gratuita**, tanto del Sistema Operativo como de las actualizaciones de este. Debian sigue siendo una elección purista con un compromiso total con el código abierto.
2. **Sistema operativo simple, estable y liviano**, su instalación y configuración son sencillas, lo que facilita su adopción incluso para usuarios menos experimentados. Además, prioriza la estabilidad sobre las últimas características, lo que la hace ideal para entornos críticos. Sus versiones se someten a rigurosas pruebas antes de su lanzamiento, lo que garantiza un funcionamiento confiable y predecible.

Adicional, Debian tiende a ser más liviano que su competidor más cercano Ubuntu, con requisitos mínimos que la hacen una distribución atractiva para sistemas antiguos o con recursos limitados.

3. **Seguro**, la comunidad de Debian mantiene un enfoque proactivo en la seguridad. Las actualizaciones y parches se aplican de manera constante para proteger el sistema contra vulnerabilidades.
4. **Amplia Selección de Paquetes**, Debian ofrece un vasto repositorio de paquetes pre-compilados y compilados estables, lo que permite a los administradores instalar software específico según sus necesidades.

### Desventajas:

1. **Versiones No Siempre Actualizadas:** Debido a su enfoque en la estabilidad, las versiones de software en Debian pueden no ser las más recientes. Esto puede afectar a los usuarios que requieren características más avanzadas.
2. **Documentación en Inglés:** Aunque existe documentación en varios idiomas, gran parte de los recursos están en inglés. Esto puede dificultar el acceso para algunos usuarios.
3. **Curva de Aprendizaje para Nuevos Usuarios:** A pesar de su simplicidad, algunos aspectos de Debian pueden requerir conocimientos técnicos más profundos.

En todo caso, la elección entre las diferentes distribuciones dependerá de la experiencia y preferencias personales de cada usuario. A nivel corporativo, la selección tiende a inclinarse por la que ofrezca soporte directo con el fabricante con tiempos de respuesta definidos según contrato.

## 2.3 ALTA DISPONIBILIDAD

La alta disponibilidad (HA, del inglés *High Availability*) se refiere al conjunto de protocolos de diseño cuya implementación va destinada a **eliminar los puntos únicos de fallo en sistemas TI para minimizar el impacto que pudiera tener una interrupción** en los sistemas, bases de datos y aplicaciones. En las organizaciones, todo esto se evidencia en la reducción del riesgo de afectar la productividad, la reputación y por tanto perder ingresos. La redundancia y la detección automática de fallos son aspectos clave para la HA.

Además de ello, la alta disponibilidad combina dos conceptos para determinar si un sistema cumple con su nivel de **rendimiento operativo**: el primero tiene que ver con la accesibilidad o la disponibilidad prácticamente permanentes de un servicio o servidor sin tiempo de inactividad (cumplimiento de los acuerdos de nivel del servicio (SLA) o con las expectativas establecidas entre el proveedor y el cliente); y el segundo se refiere a su funcionamiento según las expectativas razonables y durante un período establecido (que el sistema sea resistente, confiable y eficaz).

### 2.3.1 Funcionamiento de la alta disponibilidad

La infraestructura de alta disponibilidad depende de la detección y eliminación de los **puntos únicos de fallas que podrían aumentar el tiempo de inactividad** de los sistemas e impedir que las empresas alcancen sus objetivos de rendimiento. Los puntos únicos de fallas son aspectos de la infraestructura que podrían desconectar todo el sistema, y puede haber muchos de ellos en los sistemas complejos.

Las empresas también deben considerar los distintos tipos de fallas que se pueden producir en las infraestructuras de TI modernas y complejas, como las del hardware; del software (tanto en el sistema operativo como en las aplicaciones en ejecución); del servicio (como las que se producen por la falta de accesibilidad de la red y la latencia en los servicios de nube, o el deterioro del rendimiento), y las fallas externas, como por un corte de la energía eléctrica, terremotos, revueltas sociales, etc.

La primera medida que pueden tomar las empresas para lograr la alta disponibilidad es determinar los resultados específicos más importantes que esperan para sus servicios esenciales, los requisitos de cumplimiento normativo y de las cargas de trabajo, los parámetros de rendimiento, las aplicaciones fundamentales y las prioridades operativas.

### 2.3.2 Métricas de Alta Disponibilidad

En un sistema real, si falla uno de los componentes, es reparado o sustituido por un nuevo componente y esto se vuelve un ciclo entre componentes operativos, por sustituir o en proceso de reparación. Siendo así cada componente, durante su vida útil tendrá dos estados: funcionando o en reparación, todo lo cual se hace extensivo al sistema completo. Por lo tanto, podemos decir que el sistema tiene durante su vida, una media de tiempo para presentar fallas MTTF (Mean Time Between Failures) y un tiempo medio de reparación MTTR (Mean Time To Repair). Siendo así, el **tiempo de vida útil del sistema** es la suma de MTTFs en ciclos  $MTTF + MTTR$  ya vividos. <sup>(5)</sup> (Wikipedia, 2023)

De esta manera, se dice que la disponibilidad de un sistema es la relación entre la duración de la vida útil de este sistema y de su tiempo total de vida, lo cual puede ser representado por la fórmula:

$$\text{Disponibilidad} = \text{MTTF} / (\text{MTTF} + \text{MTTR})$$

Además de estas métricas, para evaluar la disponibilidad de un sistema, los equipos de TI utilizan otros indicadores comunes para determinar si la arquitectura de los entornos de alta disponibilidad cumple con sus objetivos. Es recomendable evaluarlos para establecer las expectativas de referencia en cuanto al rendimiento.

- ▶ **RTO u «Objetivo de tiempo de recuperación»:** define el tiempo máximo durante el cual es aceptable que un fallo afecte a la actividad normal. Las aplicaciones de misión crítica requieren un RTO muy bajo o incluso nulo ( $RTO=0$ ) de ser posible.
- ▶ **RPO u «Objetivo de punto de recuperación»:** define la cantidad máxima de datos que la empresa está dispuesta a perder durante un fallo.

Durante el proceso de evaluación de una solución de alta disponibilidad es importante encontrar un equilibrio entre la tolerancia y el presupuesto, de esta manera para medir el tiempo de inactividad aceptable en un negocio es importante considerar los siguientes aspectos:



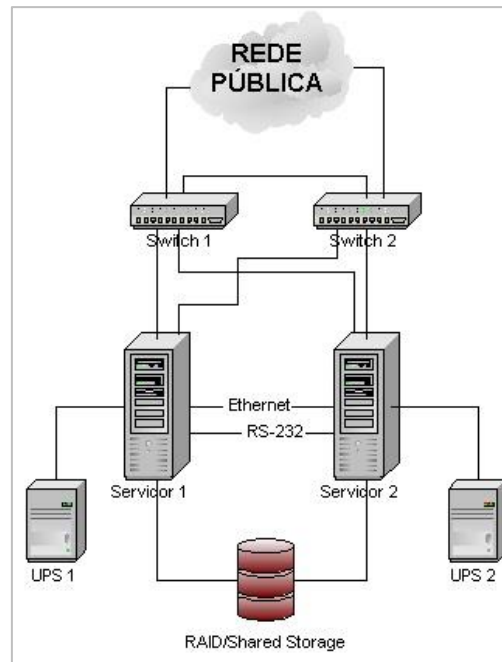
- Tiempo de inactividad por mantenimiento planificado o sin planificar.
- Tiempo de inactividad por fallos de software o hardware.
- Tiempo de actividad a nivel de sistema operativo y aplicaciones.
- El impacto del tiempo de inactividad en la experiencia de usuario.
- Pérdida de datos tolerable.
- Sistemas y aplicaciones críticos para el negocio.

Las aplicaciones de misión crítica requieren objetivos de RPO y RTO lo más bajos posible. Algunos de los sistemas críticos para el negocio son: CRMs, software de comercio electrónico, aplicaciones de gestión de la cadena de suministro, sistemas de inteligencia de negocio, bases de datos, etc.

### 2.3.3 Clústeres de Alta Disponibilidad

Un Clúster de Alta Disponibilidad consiste en un conjunto de dos o más servidores que funcionan de manera conjunta y se caracterizan por compartir un sistema de almacenamiento común, mientras se monitorizan mutuamente de forma continua. En caso de que se produzca un fallo en el hardware o en los servicios de alguno de los servidores que integran el clúster, el software de alta disponibilidad puede reiniciar automáticamente los servicios afectados en otro de los servidores del clúster. Una vez que el servidor que experimentó el fallo se recupera, los servicios pueden migrarse de nuevo a su ubicación original.

Esta capacidad de los clústeres para restablecer un servicio en cuestión de segundos, manteniendo la integridad de los datos, suele hacer que los usuarios no perciban que ha ocurrido un problema. En situaciones similares, un sistema sin clúster podría experimentar una interrupción del servicio durante horas. Los clústeres no solo son ventajosos ante caídas de servicio imprevistas, sino que también resultan útiles durante paradas programadas, como en casos de mantenimiento de hardware o actualizaciones de software.



**Figura 2 - Clúster de Alta Disponibilidad**

Tomado de: "Wikipedia: Clúster de alta disponibilidad". Accedido el 5 de agosto de 2024. [En línea]. Disponible:

[https://es.wikipedia.org/w/index.php?title=Cl%C3%BAsster\\_de\\_alta\\_disponibilidad&oldid=15541966](https://es.wikipedia.org/w/index.php?title=Cl%C3%BAsster_de_alta_disponibilidad&oldid=15541966)

5

En la Figura 2, se muestra un esquema de un clúster HA que considera alta disponibilidad con dos servidores y un repositorio común.

En la actualidad y dadas las altas demandas de servicios, las 24 horas del día y los 365 días del año, es casi un deber implementar clústeres HA en los servicios críticos de las organizaciones, más aún cuando se trata de infraestructuras de interés nacional. A continuación, se detallan algunas de las razones por la cuales es altamente recomendado implementar un clúster HA:

- Aumentar la disponibilidad
- Mejorar el rendimiento
- Escalabilidad
- Tolerancia a fallos
- Recuperación ante fallos en tiempo aceptable
- Reducir costes
- Consolidar servidores
- Consolidar el almacenamiento

Además de ofrecer estas características generales, también se pueden diseñar los clústeres de alta disponibilidad para tareas más específicas en función de las prioridades y las actividades de la infraestructura de TI.

### 2.3.4 Configuraciones de Alta Disponibilidad

Los clústeres de alta disponibilidad garantizan un rendimiento óptimo durante un fallo potencial, también conocido como “failover clusters”, y las configuraciones más comunes que se pueden encontrar en estos entornos son la configuración activo/activo y la configuración activo/pasivo.

- **Configuración Activo/Activo:** todos los servidores del clúster pueden ejecutar los mismos recursos simultáneamente. Es decir, los servidores poseen los mismos recursos y pueden acceder a estos independientemente de los otros servidores del clúster. Si un nodo del sistema falla y deja de estar disponible, sus recursos siguen estando accesibles a través de los otros servidores del clúster. La ventaja principal de esta configuración es que los servidores en el clúster son más eficientes ya que pueden trabajar todos a la vez. Sin embargo, cuando uno de los servidores deja de estar accesible, su carga de trabajo pasa a los nodos restantes, lo que produce una degradación del nivel global de servicio ofrecido a los usuarios.

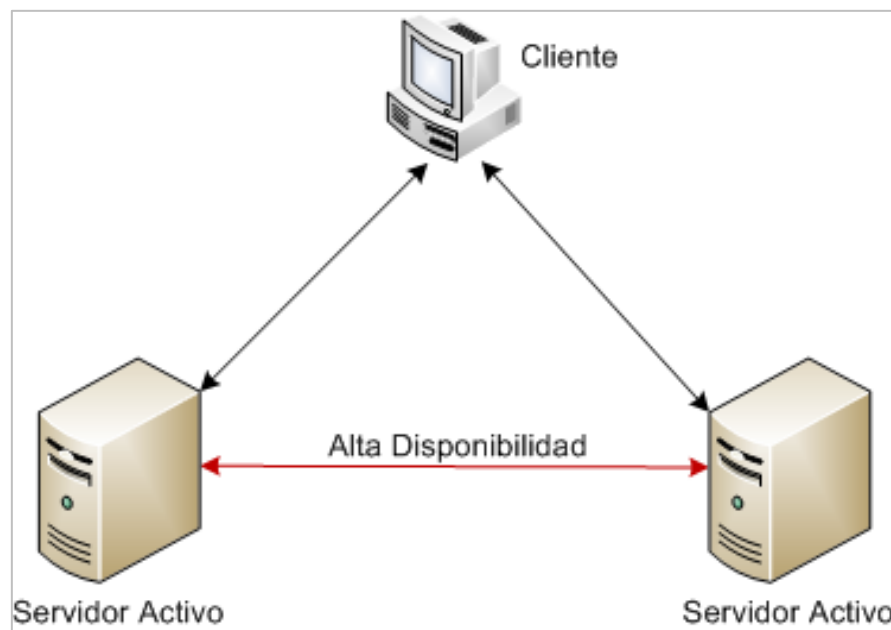


Figura 3 - Configuración de Clúster Activo/Activo

Tomado de: “ECURED – Clúster de Alta Disponibilidad”. Accedido el 5 de agosto de 2024. [En línea]. Disponible: <https://www.ecured.cu/C1%C3%BAster de Alta Disponibilidad>

En la figura 3 se muestra como ambos servidores están activos, proporcionando un mismo servicio a los diferentes usuarios. Los clientes acceden al servicio o

recursos de forma transparente y no tienen conocimiento de la existencia de varios servidores formando un clúster.

- **Configuración Activo/Pasivo:** en una configuración activo/pasivo, consiste en un servidor que posee los recursos del clúster y otros servidores que son capaces de acceder a esos recursos, pero no los activan hasta que el propietario de los recursos ya no esté disponible. Las ventajas de la configuración activo/pasivo son que no hay degradación de servicio y que los servicios solo se reinician cuando el servidor activo deja de responder. Sin embargo, una desventaja de esta configuración es que los servidores pasivos no proporcionan ningún tipo de recurso mientras están en espera, haciendo que la solución sea menos eficiente que el clúster de tipo activo/activo. Otra desventaja es que los sistemas tardan un tiempo en migrar los recursos ([failover](#)) al nodo en espera.

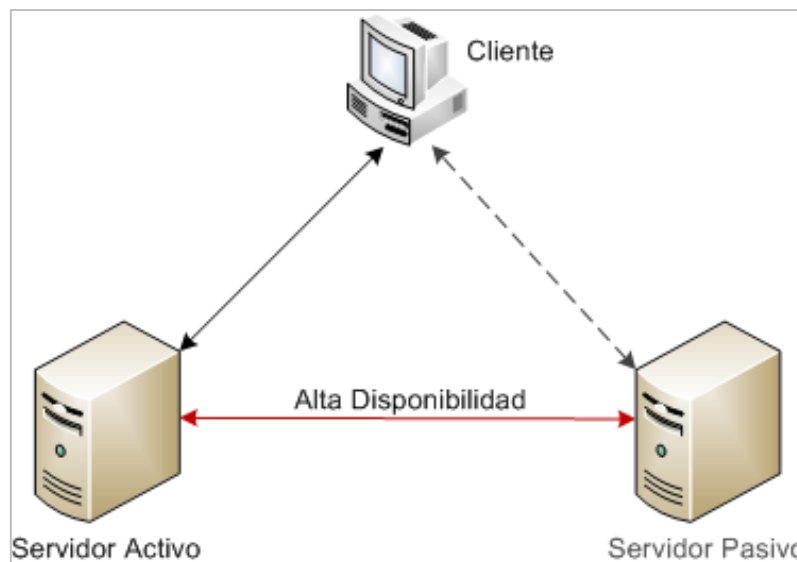


Figura 4 - Configuración de Clúster Activo/Pasivo

Tomado de: "ECURED – Clúster de Alta Disponibilidad". Accedido el 5 de agosto de 2024. [En línea]. Disponible: <https://www.ecured.cu/C1%C3%BAster de Alta Disponibilidad>

En la figura 4 se muestra un servidor activo, que es el que directamente atiende a las peticiones de los clientes. El servidor pasivo entrará en operaciones cuando el servidor activo deje de responder al monitoreo interno del clúster.

### 2.3.5 Implementación de un Clúster HA

La implementación de clústeres de alta disponibilidad es determinante en entornos donde la continuidad de los servicios es vital para el funcionamiento del negocio. Estos clústeres están diseñados para minimizar el tiempo de inactividad y asegurar que los

servicios sigan operativos incluso en caso de fallos de hardware, software, o problemas de red. Esta alta disponibilidad es especialmente importante en sectores como la salud, la banca, telecomunicaciones, comercio electrónico, y cualquier otro que dependa de la accesibilidad continua a aplicaciones y datos.

Para implementar clústeres de alta disponibilidad de manera efectiva, se requiere de software especializado que facilite la configuración, gestión y monitoreo de los clústeres, garantizando que cualquier interrupción se maneje de forma rápida y eficiente. Este software no solo permite la rápida recuperación de servicios, sino que también asegura la estabilidad y la integridad del sistema en todo momento.

## Soluciones de Software en Linux

Dado que el sistema operativo base que usaremos en nuestro entorno de pruebas, como parte de este trabajo de fin de máster, es Linux, desarrollamos la búsqueda de soluciones para implementar clústeres de alta disponibilidad sobre dicho sistema operativo.

En el ecosistema Linux, existen diversas soluciones de software que permiten implementar clústeres de alta disponibilidad, cada una con características específicas que se adaptan a diferentes necesidades y escenarios. Estas herramientas permiten crear entornos resilientes que pueden soportar caídas de hardware o software sin interrupciones significativas en los servicios. A continuación, se destacan algunas de las principales herramientas utilizadas en este ámbito, con un enfoque en **Keepalived** y el conjunto de herramientas formado por **Corosync**, **Pacemaker**, y **STONITH**.

### Keepalived

Keepalived es una herramienta ampliamente utilizada en entornos Linux para implementar soluciones de alta disponibilidad y balanceo de carga, especialmente en configuraciones de enrutamiento y servidores web. Su funcionamiento se basa en el protocolo VRRP (Virtual Router Redundancy Protocol), que permite crear sistemas altamente disponibles al proporcionar un mecanismo mediante el cual los servidores pueden respaldarse mutuamente al aplicar procesos de monitoreo entre todos los miembros del cluster. Esto significa que, si un servidor experimenta un fallo, otro puede asumir automáticamente su función, garantizando así la continuidad del servicio.

La función principal de Keepalived es monitorizar la disponibilidad de los servidores y redirigir el tráfico de red en caso de que uno de ellos falle, utilizando para ello el protocolo VRRP (Virtual Router Redundancy Protocol).<sup>(6)</sup> (Keepalived, 2024) Es ideal para configurar un clúster de alta disponibilidad en situaciones donde se requiere un failover rápido y eficiente de direcciones IP virtuales. Si el servidor maestro que gestiona la IP virtual falla, Keepalived transfiere esta IP a un servidor de respaldo, garantizando que el servicio permanezca accesible.

Y es que VRRP opera con el concepto de direcciones IP virtuales, donde varios servidores comparten la administración de una IP, pero solo uno de ellos, conocido como

el nodo "Master", controla esa IP en un momento dado. Keepalived, al gestionar estas IP virtuales, asegura que si el nodo Máster falla, otro nodo en el clúster asuma rápidamente el control, manteniendo el servicio en funcionamiento.

Es importante destacar que Keepalived se centra en la gestión de la red y la conmutación por error (failover) entre servidores, sin intervenir directamente en la activación o desactivación de servicios específicos en los nodos. Por este motivo, es especialmente adecuado para configuraciones donde se requiere redundancia en enrutadores, cortafuegos y proxies, pero no es idóneo para arreglos de discos o sistemas de archivos, donde se necesitaría una gestión más específica de los servicios subyacentes.

### Características:

- ▶ Facilita la creación de configuraciones de alta disponibilidad sin puntos únicos de falla.
- ▶ Se suele combinar con IPVS o HAProxy para balancear la carga entre servidores activos.
- ▶ Muy útil en configuraciones donde se necesita un failover automático y rápido de direcciones IP,

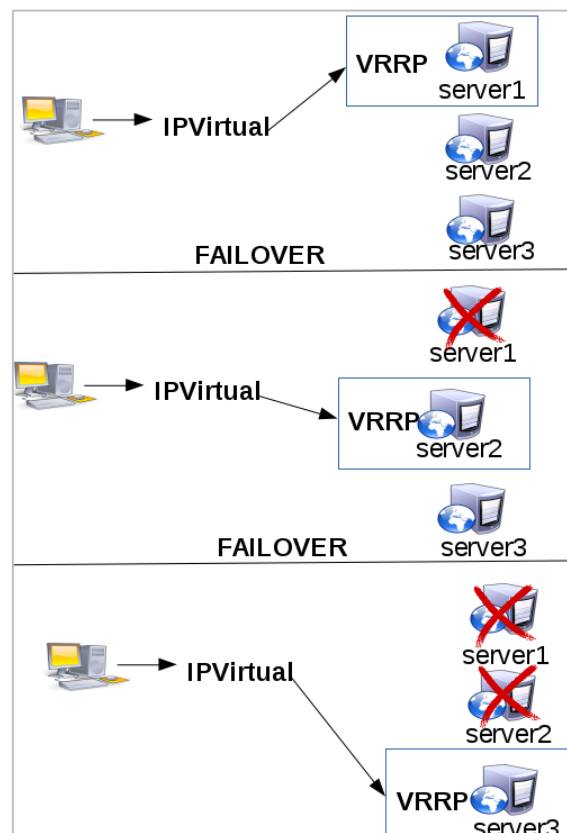


Figura 5 - Gestión de IPs con Keepalived

Tomado de: "GITHUB – Failover con Keepalived". Accedido el 5 de agosto de 2024. [En línea].  
Disponible: <https://github.com/cgomeznt/Keepalived/blob/master/guia/failover.rst>

## Corosync, Pacemaker y STONITH

El stack compuesto por **Corosync**, **Pacemaker**, y **STONITH** es una de las soluciones más completas y flexibles para la implementación de clústeres de alta disponibilidad en Linux.

- ▶ **Corosync:** Es la capa de **comunicación y monitoreo del clúster**. Corosync se encarga de mantener la coherencia del clúster, distribuyendo información sobre el estado de los nodos y gestionando el quórum, lo que asegura que las decisiones se toman de manera consistente en todo el clúster.
- ▶ **Pacemaker:** Funciona como el corazón de la **gestión de recursos en un clúster**. Pacemaker toma decisiones sobre la asignación de recursos, como bases de datos, servidores web, sistemas de archivos o direcciones IPs, asegurando que estos recursos se ejecuten en los nodos adecuados y se reubiquen en caso de fallos. Utiliza la información proporcionada por Corosync, Pacemaker también gestiona el reinicio de servicios y la migración de recursos entre nodos para mantener la disponibilidad.

Los siguientes componentes, además de Corosync, forman la arquitectura de Pacemaker: <sup>(7)</sup> (RedHat, 2024)

- ▶ **Base de información de clusters (CIB)**

El demonio de información de Pacemaker, que utiliza XML internamente para distribuir y sincronizar la configuración actual y la información de estado del Coordinador Designado (DC)

- ▶ **Demonio de gestión de recursos del clúster (CRMd)**

Las acciones de los recursos del cluster Pacemaker se enrutan a través de este demonio. Los recursos gestionados por CRMd pueden ser consultados por los sistemas cliente, movidos, instanciados y modificados cuando sea necesario.

Cada nodo del clúster también incluye un demonio gestor de recursos local (LRMd) que actúa como interfaz entre CRMd y los recursos. LRMd pasa comandos de CRMd a los agentes, como el arranque y la parada y la transmisión de información de estado.

- ▶ **STONITH (Shoot The Other Node In The Head):** es la implementación de cercado (fencing) de Pacemaker, utilizado para aislar nodos fallidos en un clúster para evitar que causen problemas (por ejemplo, escribiendo datos corruptos). Si un nodo falla o se comporta de manera anómala, STONITH se asegura de que el nodo problemático se reinicie o se apague, evitando así la corrupción de datos y asegurando la integridad del clúster. STONITH está configurado en el CIB y puede ser monitorizado como un recurso de cluster normal.

Un claro caso de uso de Stonith es un clúster de base de datos, si un nodo deja de responder, pero sigue encendido, STONITH puede forzar un reinicio de ese nodo para evitar que corrompa la base de datos.

Todos estos componentes forman un conjunto para crear clústeres de alta disponibilidad, manejando la comunicación, la gestión de recursos y la protección contra fallos. Por este motivo, es utilizado en una variedad de aplicaciones críticas, incluyendo bases de datos, sistemas de archivos compartidos, y aplicaciones empresariales que requieren alta disponibilidad y tolerancia a fallos.

**Características del conjunto Corosync + Pacemaker + STONITH:**

- ▶ Configuración robusta para alta disponibilidad.
- ▶ Asegura que los servicios críticos sigan funcionando incluso si uno o más nodos fallan.
- ▶ Permite la gestión automatizada de recursos, incluyendo el failover y la prevención de errores.

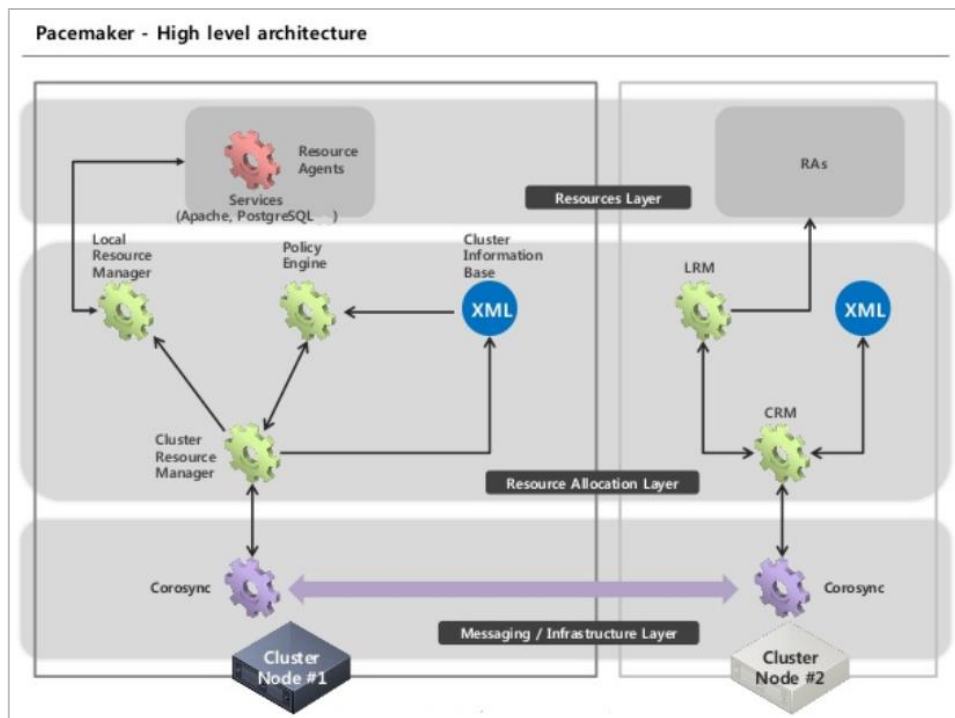


Figura 6 - Pacemaker Arquitectura de Alto Nivel

Tomado de: “JWB System – HA Cluster con Pacemaker: Cluster Componens”. Accedido el 5 de agosto de 2024. [En línea]. Disponible: <https://jwb-systems.com/high-availability-cluster-with-pacemaker-part-5-cluster-componens/>

**2.3.6 Implementación de un Clúster con equilibrado de carga**

El equilibrio de carga es un componente fundamental en la arquitectura de clústeres, especialmente en sistemas donde la disponibilidad y el rendimiento son críticos. Este



esquema se encarga de distribuir de manera eficiente las solicitudes de los usuarios entre varios servidores, evitando que un solo servidor se vea sobrecargado y asegurando que los recursos del sistema se utilicen de manera óptima. En la actualidad, el equilibrio de carga es esencial para mantener la continuidad del servicio y garantizar una experiencia de usuario fluida, especialmente en entornos web con alta demanda.

Este tipo de cluster está compuesto por uno o más ordenadores (llamados nodos) que actúan como front-end del clúster y se ocupan de repartir las peticiones de servicio que reciba el clúster a otros ordenadores que forman su back-end. En la figura 7, se puede apreciar un modelo de arquitectura de clusters con equilibrio de carga, donde tenemos dos nodos front-end que reparten la carga entre 4 nodos back-end.

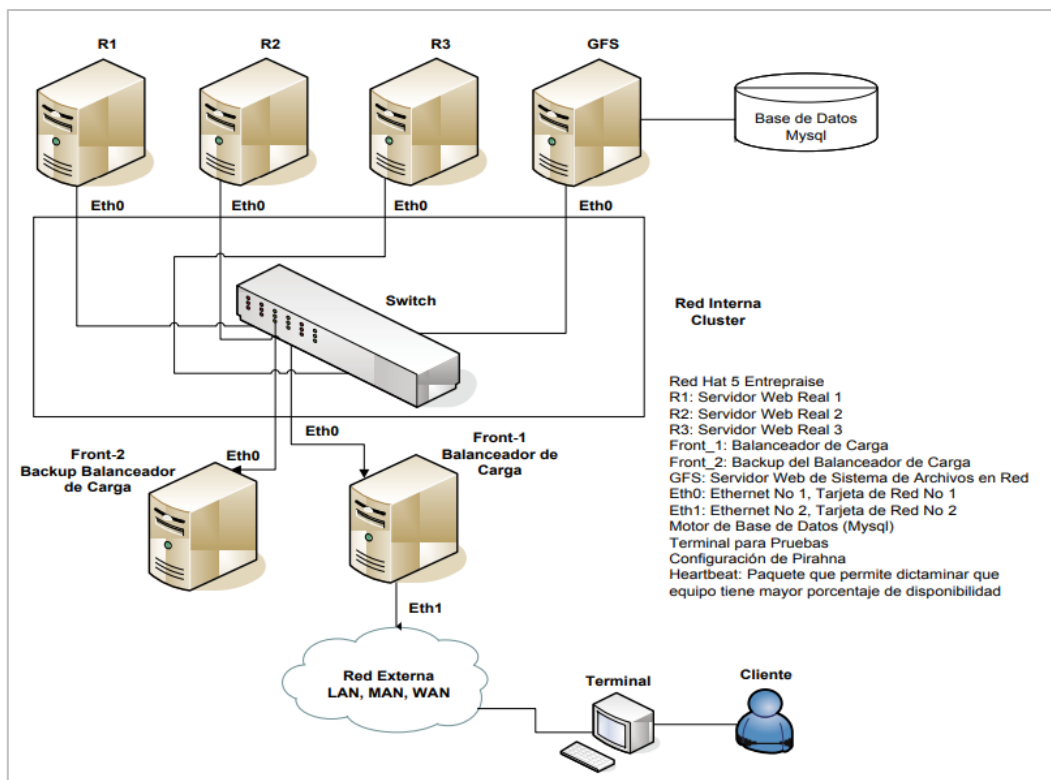


Figura 7 - Ejemplo de diseño de un clúster con equilibrio de carga

Tomado de: “Clúster de balanceo de carga y alta disponibilidad para servicios web y mail”.

Accedido el 6 de agosto de 2024. [En línea]. Disponible:

<https://dialnet.unirioja.es/descarga/articulo/4364562.pdf>

En nuestro caso haremos uso de esta tecnología para distribuir la carga entre varios servidores back-end que implementan contenedores, cada uno con un servidor web.

La importancia del equilibrio de carga radica en su capacidad para aumentar la escalabilidad y resiliencia de los sistemas. Al distribuir las cargas de trabajo, se puede manejar un mayor volumen de tráfico sin comprometer el rendimiento, lo que es crucial en aplicaciones de misión crítica. Además, en caso de que uno de los servidores falle,

el tráfico se puede redirigir automáticamente a otros servidores en el clúster, minimizando el impacto en el usuario final.

De esta manera podemos indicar que las características más destacadas de este tipo de clúster son:

- ▶ Escalabilidad. Se puede ampliar su capacidad fácilmente añadiendo más ordenadores al clúster.
- ▶ Robustez. Ante la caída de alguno de los ordenadores del clúster, el servicio se puede ver mermado; pero mientras haya ordenadores en funcionamiento estos seguirán dando el servicio. (10) (Sinisterra, 2012)

Sin embargo, aunque el equilibrio de carga ofrece numerosas ventajas, también presenta desafíos. Uno de los principales inconvenientes es la complejidad de su configuración y mantenimiento, ya que requiere una cuidadosa planificación y monitoreo continuo. Además, si no se implementa correctamente, puede introducir puntos de fallo únicos en el sistema, como en el caso de un equilibrador de carga centralizado que no cuenta con redundancia.

En el ecosistema Linux, existen diversas soluciones de software para implementar clústeres con equilibrio de carga. Entre las más utilizadas se encuentran **HAProxy**, **IPVS** (IP Virtual Server), **NGINX**, y **LVS** (Linux Virtual Server).

El enfoque durante este trabajo es el uso de HAProxy para brindar al clúster de alta disponibilidad una capa adicional con el reparto de carga entre los nodos de back-end disponibles para ofrecer el servidor web a los usuarios finales.

### HA Proxy (High Availability Proxy)

HAProxy es un proxy inverso open source, muy rápido y confiable que ofrece alta **disponibilidad, equilibrio de carga y proxy para aplicaciones basadas en TCP y HTTP**. Es particularmente adecuado para sitios web de muy alto tráfico y se utiliza en una parte significativa de los más visitados del mundo. A lo largo de los años, se ha convertido en el equilibrador de carga de código abierto standard, por este motivo se lo encuentra por defecto en la mayoría de las distribuciones principales de Linux y, a menudo, se implementa de forma predeterminada en plataformas en la nube. Dado que no se anuncia a sí mismo, solo se conoce que se usa cuando los administradores lo denuncian. (9) (HAProxy, 2024)

HA Proxy es capaz de distribuir el tráfico entrante entre varios servidores de back-end, asegurando que las solicitudes se manejen de manera eficiente y que los recursos se utilicen de manera óptima. HAProxy es conocido también por su rendimiento, flexibilidad y capacidad para manejar grandes volúmenes de tráfico.

La implementación de HAProxy en un entorno Linux implica la instalación del software en un servidor dedicado que actuará como el repartidor de carga. HAProxy permite la configuración de reglas específicas para determinar cómo se distribuye el tráfico, basándose en diversos criterios como el balanceo de carga por rondas, la menor cantidad de conexiones, o el menor tiempo de respuesta.

El archivo de configuración de HAProxy es altamente personalizable y permite definir grupos de servidores back-end, reglas de enrutamiento y políticas de failover. Una vez

configurado, HAProxy puede manejar múltiples tipos de tráfico, incluyendo HTTP, TCP, y UDP, lo que lo convierte en una solución versátil para diferentes tipos de aplicaciones.

#### Características:

- Repartidor de carga con diferentes algoritmos (round-robin, least connections, etc.).
- Capacidad de detección de fallos en servidores back-end.
- Puede manejar SSL/TLS.
- Muy flexible y altamente configurable.

#### Ventajas:

- ▶ **Alto rendimiento:** HAProxy es capaz de manejar miles de conexiones simultáneas con baja latencia.
- ▶ **Flexibilidad:** Su archivo de configuración permite un alto grado de personalización, adaptándose a diferentes escenarios y requisitos.
- ▶ **Soporte para múltiples protocolos:** Es compatible con HTTP, TCP y SSL, entre otros.
- ▶ **Alta disponibilidad:** Puede integrarse con herramientas como Keepalived para crear un entorno de alta disponibilidad.

#### Desventajas:

- ▶ **Complejidad:** La configuración de HAProxy puede ser compleja, especialmente para los administradores menos experimentados.
- ▶ **Falta de interfaz gráfica:** HAProxy no incluye una interfaz gráfica nativa, lo que puede hacer su gestión más difícil para algunos usuarios.
- ▶ **Costos de mantenimiento:** Requiere un monitoreo y ajuste continuo para asegurar un rendimiento óptimo.

### 2.3.7 Integración de HAProxy con Keepalived y Corosync+Pacemaker

HAProxy puede integrarse con **Keepalived** para proporcionar una solución de alta disponibilidad completa. Keepalived se utiliza para gestionar la conmutación por error (failover) entre múltiples instancias de HAProxy, asegurando que, si un nodo falla, otro pueda asumir su rol sin interrupción del servicio.

Además, en configuraciones más complejas, **Corosync** y **Pacemaker** pueden emplearse para gestionar el clúster en su totalidad. Pacemaker, junto con Corosync,

proporciona un marco de alta disponibilidad que puede coordinar la ejecución de HAProxy, asegurando que todos los componentes del clúster estén sincronizados y operando correctamente. **STONITH** se utiliza en estas configuraciones para evitar condiciones de "split-brain", asegurando que solo un nodo controle el recurso en cualquier momento.

## 2.4 SISTEMAS DE ALMACENAMIENTO

En cualquier arquitectura de clúster, el sistema de almacenamiento es un componente crucial que no solo almacena los datos, sino que también garantiza su disponibilidad y protección contra fallos. En entornos de alta disponibilidad, el sistema de almacenamiento debe ser capaz de soportar el acceso concurrente desde múltiples nodos, replicar o distribuir datos para evitar la pérdida de información y mantener la coherencia y la integridad de los datos a lo largo del tiempo.

Los sistemas de almacenamiento en clúster pueden ser clasificados principalmente en dos tipos: almacenamiento distribuido y almacenamiento replicado.

- ▶ **Almacenamiento Distribuido:** Este tipo de almacenamiento distribuye los datos entre varios nodos del clúster, lo que mejora la disponibilidad y la escalabilidad. Los sistemas distribuidos permiten que diferentes fragmentos de un archivo se almacenen en distintos servidores, optimizando el uso de los recursos y permitiendo un acceso más rápido y paralelo a los datos. Sin embargo, la complejidad en la gestión de los datos y la necesidad de mantener la coherencia entre los nodos son desafíos considerables.

Entre las ventajas de este tipo de almacenamiento podemos citar:

- **Escalabilidad:** Permite añadir más nodos para aumentar la capacidad de almacenamiento y mejorar el rendimiento.
- **Optimización de recursos:** Los datos se almacenan en diferentes nodos, lo que permite el acceso paralelo y mejora el tiempo de respuesta.
- **Disponibilidad:** Los datos están repartidos entre múltiples nodos, lo que reduce la posibilidad de pérdida total.

Y como desventajas:

- **Complejidad:** La gestión y configuración son más complicadas, especialmente en cuanto a la coherencia y sincronización de datos.
- **Latencia:** En algunos casos, puede haber retrasos en el acceso a los datos si no están localizados en el nodo que los solicita.

- ▶ **Almacenamiento Replicado:** En este esquema, los datos se copian en múltiples nodos del clúster. La replicación asegura que, si uno de los nodos falla,

los datos todavía están disponibles en otro nodo. Esto proporciona una alta resiliencia ante fallos, pero puede ser costoso en términos de espacio de almacenamiento y puede introducir latencia si los datos deben sincronizarse entre ubicaciones geográficas dispersas.

De esta manera resaltamos como principales ventajas:

- **Redundancia:** Los datos están duplicados, lo que protege contra la pérdida de datos si un nodo falla.
- **Alta disponibilidad:** Los datos siempre están disponibles en otro nodo, lo que es vital en entornos críticos.
- **Facilidad de recuperación:** En caso de fallo, la recuperación es más rápida, ya que los datos no necesitan ser reconstruidos desde cero.

Y como desventajas:

- **Costos:** Duplicar los datos en varios nodos consume mucho más espacio de almacenamiento.
- **Sincronización:** Puede haber desafíos y retrasos al sincronizar grandes volúmenes de datos entre nodos.

## 2.4.1 Soluciones de almacenamiento

Varios sistemas o esquemas de almacenamiento se utilizan comúnmente en clústeres para garantizar la alta disponibilidad y la eficiencia. A continuación, se describen algunos de los más destacados:

- ▶ Con respecto al almacenamiento distribuido, **GlusterFS** y **Ceph** son dos tecnologías clave que complementan la alta disponibilidad en clústeres Linux.

- **GlusterFS:** es un sistema de archivos distribuido que permite almacenar gran cantidad de datos distribuidos entre clústeres de servidores con una disponibilidad muy alta. <sup>(8)</sup> (Morvan, 2022)

GlusterFS es sencillo de implementar y gestionar, lo que lo hace ideal para entornos donde se requiere almacenamiento escalable y altamente disponible.

Se compone de una parte del servidor que se instalará en todos los nodos de los clústeres del servidor. Los clientes pueden acceder a los datos mediante el comando **glusterfs** o **mount**.

GlusterFS puede trabajar en dos modos diferentes:

- **Modo replicado:** Cada nodo del clúster tiene todos los datos.
- **modo distribuido:** no hay redundancia de datos. Si un nodo de almacenamiento falla, se perderán los datos almacenados en ese nodo.

Los dos modos pueden utilizarse conjuntamente para proporcionar un sistema de archivos replicado, así como un sistema de archivos distribuido, siempre y cuando se disponga del número correcto de servidores.

Los datos se almacenan dentro de los bloques. Un bloque es la unidad básica de almacenamiento en GlusterFS, representada por un directorio de exportación en un servidor dentro del grupo de almacenamiento de confianza.

- **Ceph:** Ofrece un sistema de almacenamiento distribuido que proporciona almacenamiento en bloque, de objetos, y de archivos en un solo clúster unificado. Ceph es conocido por su escalabilidad y su capacidad de recuperación automática, siendo utilizado en grandes despliegues en la nube y en centros de datos.
- ▶ Otro tipo de almacenamiento ampliamente utilizado son los **NAS (Network Attached Storage)**, un sistema de almacenamiento conectado a una red que permite a múltiples usuarios y dispositivos acceder a los datos de manera centralizada. A diferencia del almacenamiento directo (DAS - Direct Attached Storage), que está vinculado a un servidor específico, un NAS está disponible en la red para que cualquier dispositivo autorizado pueda acceder a los archivos como si estuvieran en un disco local. Los sistemas NAS son esencialmente servidores de archivos especializados, optimizados para el acceso rápido y eficiente a datos compartidos. Esto es crucial en aplicaciones donde es necesario compartir datos entre diferentes nodos de manera eficiente y sin replicar los datos en cada servidor individual.

La implementación de un NAS suele ser sencilla y puede variar desde configuraciones caseras hasta soluciones empresariales. Los pasos básicos incluyen:

- **Hardware:** Selección de un dispositivo NAS, que puede ser un servidor dedicado o una unidad de almacenamiento comercial diseñada específicamente para actuar como NAS.
- **Conectividad:** Conexión del dispositivo NAS a la red, generalmente a través de un puerto Ethernet. Para mejorar el rendimiento, algunos sistemas NAS permiten la agregación de enlaces de red o "link aggregation". Precisamente esta será la aplicación dentro del ambiente de clúster que configuraremos como parte de este trabajo de fin de máster.
- **Configuración del Sistema:** Configuración del sistema operativo del NAS, que puede incluir la creación de usuarios, la configuración de permisos de acceso, y la definición de los volúmenes y carpetas compartidas.
- **Compartición de Archivos:** Configuración de protocolos de red como SMB/CIFS, NFS o AFP para compartir archivos con dispositivos en la red. En nuestro caso particular, utilizaremos NFS.

- **Acceso Remoto:** Configuración de acceso remoto a través de servicios como VPN o servicios en la nube para que los usuarios puedan acceder a los datos desde cualquier lugar.

Algunas ventajas del uso de NAS se detallan a continuación:

- **Facilidad de Uso:** Los NAS suelen tener interfaces de gestión amigables, lo que facilita la configuración y el mantenimiento, incluso para usuarios con conocimientos técnicos limitados.
- **Acceso Centralizado:** Permite a múltiples usuarios acceder a los mismos archivos desde diferentes ubicaciones, lo que facilita la colaboración y la gestión de datos.
- **Escalabilidad:** Es fácil añadir más espacio de almacenamiento simplemente conectando más discos al NAS o expandiendo la capacidad con unidades externas.
- **Reducción de Costos:** Centralizar el almacenamiento reduce la necesidad de almacenamiento redundante en cada dispositivo individual, lo que puede ser más costoso.

Por otra parte, algunas desventajas son:

- **Punto Único de Falla:** Si el NAS falla, todos los usuarios pierden acceso a los datos hasta que se restaure el sistema. Para mitigar esto, es necesario implementar redundancia y backups regulares.
- **Rendimiento:** Aunque los NAS son adecuados para la mayoría de las tareas, pueden no ser tan rápidos como los sistemas de almacenamiento directamente conectados (DAS), especialmente en entornos con alta demanda de I/O.
- **Seguridad:** Dado que está conectado a la red, un NAS es susceptible a ataques de red si no se protege adecuadamente. Es esencial configurar medidas de seguridad como firewalls, cifrado y acceso limitado.
- **Costo Inicial:** Aunque el NAS puede reducir costos a largo plazo, el costo inicial de una solución NAS con suficiente capacidad y redundancia puede ser significativo.

Para este trabajo de fin de máster, nos centramos en el uso de GlusterFS como un recurso de almacenamiento disponible para todos los equipos que forman parte del clúster, tanto los equipos front-end como los back-end. Además, se contará con un servidor que compartirá otro recurso de almacenamiento a través del servicio NFS, donde el recurso se basa en discos asociados a un RAID.

## 2.5 CONTENEDORES

En los últimos años, el uso de contenedores ha revolucionado la forma en que se despliegan y gestionan aplicaciones en entornos de clústeres. Un contenedor es una unidad estándar de software que empaqueta el código y todas sus dependencias, permitiendo que la aplicación se ejecute de manera rápida y confiable en cualquier entorno. Esta tecnología ha ganado popularidad en la creación de clústeres de alta disponibilidad y en la distribución eficiente de cargas debido a su flexibilidad, portabilidad y eficiencia en la utilización de recursos.

Y es que los contenedores se han convertido en un componente esencial en arquitecturas de alta disponibilidad debido a su capacidad para aislar aplicaciones y facilitar su despliegue rápido y escalado. En un entorno de clúster, los contenedores permiten que las aplicaciones se ejecuten de manera distribuida en varios nodos del clúster, garantizando que, si un nodo falla, los contenedores afectados puedan ser reiniciados automáticamente en otros nodos. Esto asegura que las aplicaciones continúen funcionando sin interrupciones, incluso en caso de fallos en el hardware o software subyacente.

Además, los contenedores simplifican el reparto de carga, ya que las aplicaciones pueden ser escaladas horizontalmente (añadiendo más instancias de contenedores) y de forma dinámica en respuesta a la demanda. Esto permite que los recursos se utilicen de manera óptima y que el clúster pueda manejar picos de tráfico de manera eficiente.

### 2.5.1 Plataformas para la implementación de contenedores

La adopción de contenedores en clústeres ha sido facilitada por un conjunto de tecnologías y software que permiten no solo la creación y gestión de contenedores, sino también su orquestación en entornos distribuidos. A continuación, se detallan las principales herramientas y arquitecturas utilizadas en este ámbito.

#### ► Docker: La Plataforma de Contenedores Estándar

**Docker** es la plataforma de contenedores más popular y ampliamente utilizada. Introducida en 2013, Docker permite a los desarrolladores empaquetar aplicaciones y sus dependencias en contenedores ligeros que pueden ejecutarse en cualquier entorno que soporte Docker. Su popularidad radica en la simplicidad con la que permite construir, distribuir y ejecutar aplicaciones de manera consistente en diferentes entornos. <sup>(11)</sup> (Docker, 2024)

#### Características principales:

- **Portabilidad:** Los contenedores Docker pueden ejecutarse en cualquier sistema que tenga instalado Docker, ya sea en servidores locales, en la nube o en dispositivos edge.



- **Eficiencia:** Los contenedores Docker comparten el kernel del sistema operativo, lo que reduce significativamente el uso de recursos en comparación con las máquinas virtuales.
- **Facilidad de uso:** Docker proporciona un conjunto de herramientas CLI (Interfaz de Línea de Comandos) y GUI (Interfaz Gráfica de Usuario) que facilitan la creación y gestión de contenedores.

Docker utiliza una arquitectura cliente-servidor. El cliente Docker se comunica con el demonio Docker, que se encarga de la tarea pesada de crear, ejecutar y distribuir los contenedores Docker. El cliente y el demonio Docker pueden ejecutarse en el mismo sistema, o puede conectar un cliente Docker a un demonio Docker remoto. El cliente y el demonio Docker se comunican mediante una API REST, a través de sockets UNIX o una interfaz de red. Otro cliente Docker es Docker Compose, que le permite trabajar con aplicaciones que constan de un conjunto de contenedores. <sup>(11)</sup> (Docker, 2024)

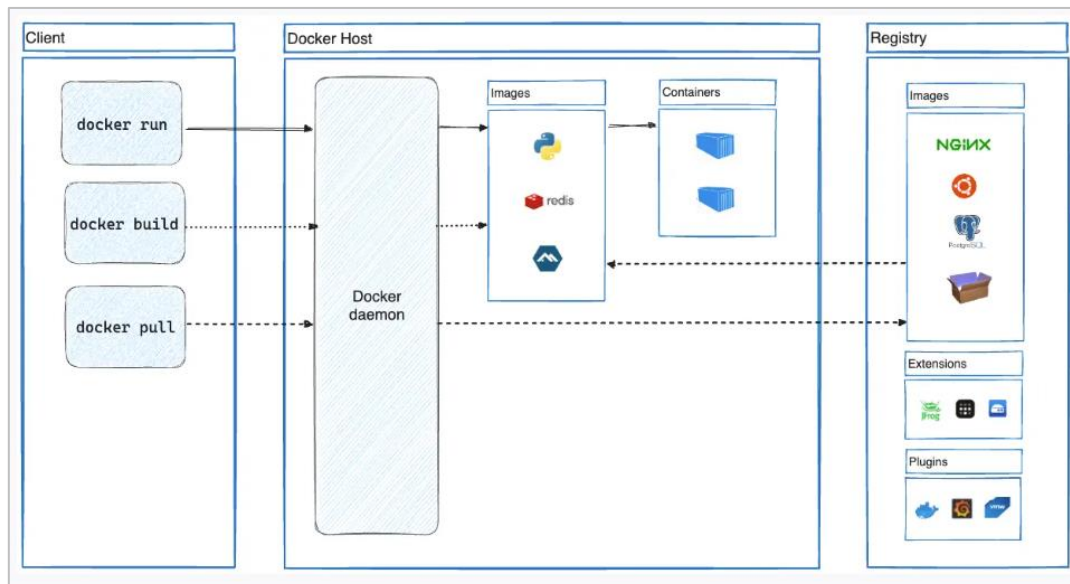


Figura 8 - Arquitectura de Docker

Tomado de: "Docker docs". Accedido el 6 de agosto de 2024. [En línea]. Disponible: <https://docs.docker.com/get-started/docker-overview/>

Para este trabajo de fin de máster, se usará Docker como plataforma para la creación y gestión de contenedores con servicio Apache+PHP. Todo esto nos permitirá escalar y brindar reparto de carga eficiente dentro de nuestro clúster de alta disponibilidad.

### ► Kubernetes: Orquestación de Contenedores a Gran Escala

**Kubernetes** es la solución de orquestación de contenedores más utilizada en la actualidad. Desarrollado originalmente por Google, Kubernetes permite automatizar el despliegue, la gestión, el escalado y las operaciones de contenedores en clústeres.

### Arquitectura de Kubernetes:

- **Master Node:** Es el cerebro de Kubernetes y se encarga de gestionar el clúster. Incluye componentes como el API Server, el Scheduler y el Controller Manager.
- **Worker Nodes:** Son los nodos que ejecutan los contenedores. Cada nodo contiene un Kubelet (para comunicarse con el Master) y un Runtime de contenedores (Docker o containerd).
- **Pods:** Son las unidades mínimas de despliegue en Kubernetes, que pueden contener uno o más contenedores.
- **Services:** Abstracciones que permiten exponer los Pods a la red y balancear la carga entre ellos.
- **Persistent Volumes:** Kubernetes maneja el almacenamiento persistente a través de estos volúmenes, lo que permite a los contenedores acceder a datos persistentes incluso si son reiniciados o migrados entre nodos.

### Ventajas:

- **Automatización avanzada:** Kubernetes facilita la automatización de tareas complejas como el escalado automático, la implementación continua y la recuperación ante fallos.
- **Flexibilidad:** Soporta múltiples arquitecturas de red, almacenamiento y control de acceso, lo que lo hace adaptable a diversos entornos empresariales.
- **Escalabilidad:** Puede gestionar desde clústeres pequeños hasta despliegues de cientos o miles de nodos.

### Desventajas:

- **Complejidad:** La configuración y gestión de Kubernetes pueden ser desafiantes, especialmente para equipos con poca experiencia en orquestación de contenedores.
- **Requiere recursos:** A diferencia de soluciones más simples, Kubernetes puede ser exigente en términos de recursos de hardware y conocimientos técnicos.

### ► OpenShift: Kubernetes Empresarial

**OpenShift** es una plataforma de orquestación de contenedores basada en Kubernetes, desarrollada por Red Hat. Ofrece una experiencia similar a Kubernetes, pero con características adicionales que facilitan su uso en entornos empresariales.

### Características adicionales:

- **Soporte empresarial:** OpenShift incluye soporte de Red Hat, lo que lo convierte en una opción atractiva para empresas que requieren estabilidad y asistencia técnica.

- **Security Enhanced Linux (SELinux):** OpenShift integra SELinux para mejorar la seguridad de los contenedores.
- **Despliegue simplificado:** Incluye herramientas para automatizar el ciclo de vida completo de las aplicaciones, desde el desarrollo hasta la producción.

## 3 DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

Para dar inicio al Capítulo 3, en el cual se detallará la implementación práctica de todas las tecnologías descritas en el capítulo anterior, es fundamental recalcar la importancia de este enfoque en el ámbito de los clústeres de alta disponibilidad y balanceo de carga. Después de haber revisado y comprendido los conceptos teóricos y las herramientas involucradas en este tipo de infraestructuras, este capítulo se centrará en la creación y configuración de un entorno práctico que refleja los desafíos y soluciones reales que se encuentran en la industria.

El objetivo de este capítulo es mostrar paso a paso cómo se ha construido un clúster de alta disponibilidad utilizando máquinas virtuales, detallando la instalación y configuración de cada uno de los servicios críticos. Desde la elección del sistema operativo base hasta la implementación de tecnologías como Keepalived, Corosync, Pacemaker, HAProxy, y GlusterFS, se mostrará cómo estos componentes trabajan en conjunto para garantizar la resiliencia, el equilibrio de carga, y la disponibilidad continua de los servicios.

Este entorno no solo es un ejercicio académico, sino que está diseñado con la intención de replicar las condiciones de un sistema que podría ser implementado en un entorno de producción. La importancia de este enfoque radica en su aplicabilidad directa a escenarios reales, donde la interrupción de servicios puede tener consecuencias significativas para las organizaciones. Además, se pretende demostrar que, con el conocimiento adecuado y las herramientas correctas, es posible construir soluciones robustas que aseguren la continuidad del negocio en situaciones críticas.

En las siguientes secciones, se presentará cada paso de la instalación, desde la creación de las máquinas virtuales hasta la configuración de los servicios que conforman el clúster. Se documentarán las configuraciones específicas, los retos encontrados y las soluciones aplicadas, proporcionando un recurso valioso tanto para aquellos que desean replicar este ambiente, como para quienes buscan comprender en profundidad la implementación práctica de un clúster de alta disponibilidad.

### 3.1 PROPUESTA

El proyecto consiste en configurar un servidor web basado en clúster con alta disponibilidad y reparto de carga, que aceptará peticiones HTTP y las distribuirá entre los nodos servidores del clúster, tendremos nodos front-end que recibirán los requerimientos y los distribuirán hacia los nodos back-end según la disponibilidad y carga del momento, para ello implementaremos contenedores con Apache+PHP. Además, dentro del clúster se tendrán dos repositorios comunes, uno para almacenamiento de información variada (GlusterFS) y otro para uso del motor web (NAS con RAID en sus discos), este último con link aggregation configurado en sus interfaces de red.

Las máquinas virtuales que componen el clúster se ejecutarán sobre un ordenador anfitrión (mi equipo personal con sistema operativo Windows 11), sin más requisitos que

tener instalada la aplicación VirtualBox y **suficiente espacio de almacenamiento** para alojar las imágenes de las máquinas virtuales.

### 3.2 DESCRIPCIÓN DE LA PROPUESTA

Se instalará un clúster con 10 nodos virtuales como se muestra en la Figura 9.

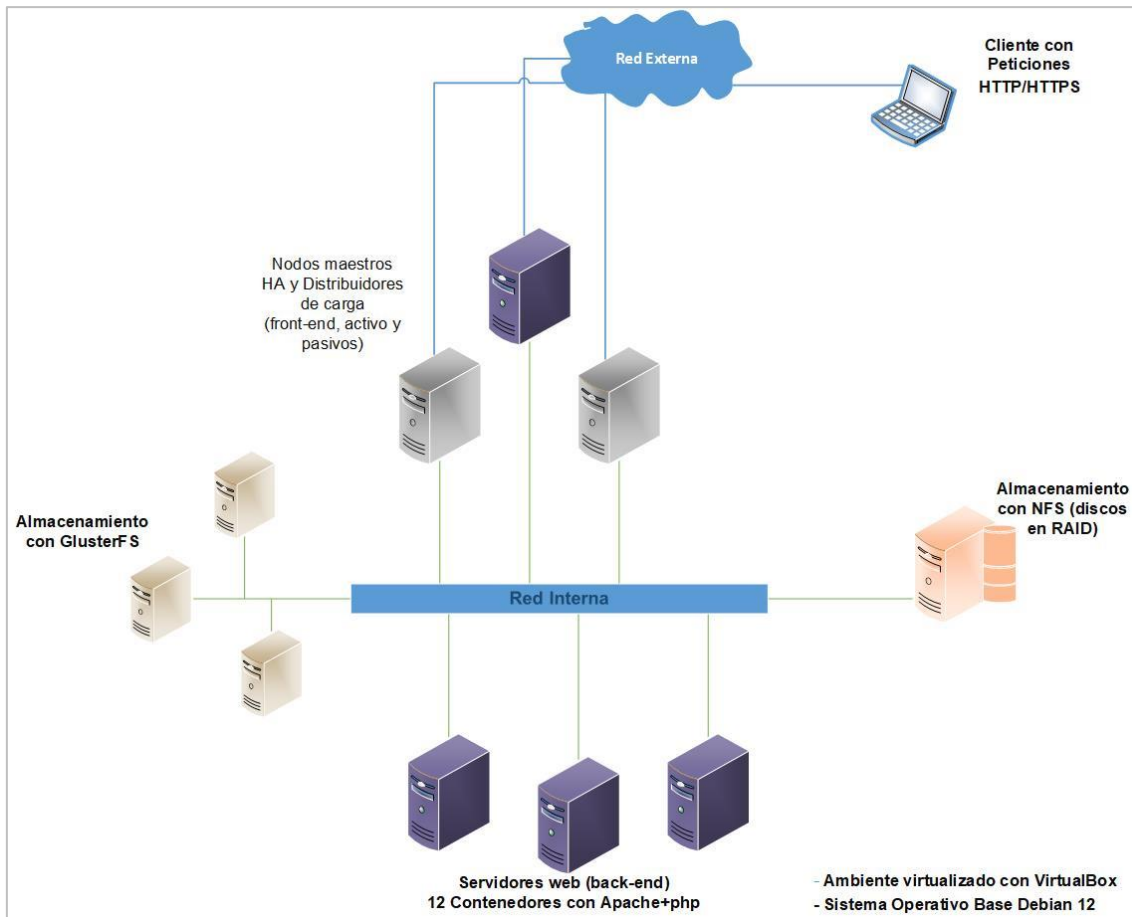


Figura 9 - Diseño Lógico de Clúster TFM

Los servicios y/o funciones que se encuentran activos en cada uno de los nodos se detallan a continuación:

- **Master y dos equipos Stand-by:** nodos maestros, están triplicados para conseguir alta disponibilidad, componiendo un servicio activo-pasivo-pasivo (**master-standby1-standby2**). Conforman el *front-end* del sistema. El nodo máster tiene 3 tarjetas de red: una en la red externa y dos en la red interna (una para comunicación exclusiva del clúster y la otra para comunicación con el restante de los equipos). Los nodos stand-by1 y stand-by2 tienen dos tarjetas de red interna (una para

comunicación exclusiva del clúster, y la otra para comunicación con el resto de los equipos).

Hacia el exterior (Internet), ofrecen el servicio web en una dirección IP virtual asociada al nodo que en cada momento está activo. Similar sucede hacia la red interna, donde el nodo activo responde a una dirección IP virtual relacionada al nodo activo en el momento de la petición. Además de las funciones propias que cumple el nodo maestro de un clúster, también se ocupará de distribuir la carga, repartiendo las peticiones HTTP recibidas entre los servidores reales del *back-end*.

Inicialmente, el maestro activo es **máster**, con mayor prioridad dentro del clúster. Cuando **standby1** detecta la caída de **máster** o del proceso de distribución de carga que allí se ejecuta, toma el relevo pues es el siguiente en prioridad. Si durante la caída del **máster**, **standby1** no estuviera disponible, entonces entra en operaciones el equipo **stand-by2**, quién es el nodo de menor prioridad dentro del clúster.

- ▶ **Cluster2, Cluster3, Cluster4:** son los servidores reales y base que forman el *back-end* del sistema. Tienen instalado Docker y sobre cada uno de ellos se levantan 4 contenedores con software Apache-PHP, que serán los encargados de atender las peticiones del usuario final. Es decir, finalmente tenemos un conjunto de “12 servidores” en el *back-end* para atender las peticiones que deriva el nodo maestro activo. La comunicación entre el nodo maestro y los servidores se realiza a través de la red interna del clúster.
- ▶ **NFS:** servidor de almacenamiento. Es un servidor NFS, cuyos discos se encuentran configurados con un RAID 5 por software. Las páginas web proporcionadas por 9 de los equipos contenedores se encuentran almacenadas aquí. Adicional, este equipo tiene sus dos interfaces de red con un link aggregation configurado, para brindar mayor ancho de banda y conmutación por error.
- ▶ **Gluster1, Gluster2 y Gluster3:** Servidores de almacenamiento replicado y distribuido. Exportan dos recursos para uso variado de los servidores del clúster, uno con esquema distribuido y otro con esquema replicado entre los 3 servidores que forman los volúmenes de Gluster. El repositorio replicado se utilizará, entre otras cosas, para albergar las páginas web proporcionadas por 3 de los equipos contenedores que ejecutarán el servicio de Apache-PHP.

### 3.3 IMPLEMENTACION DEL CLÚSTER

El clúster se implementará según el siguiente esquema:

#### Instalación de los nodos

1. El clúster se implementará mediante máquinas virtuales con VirtualBox versión 7.0.20

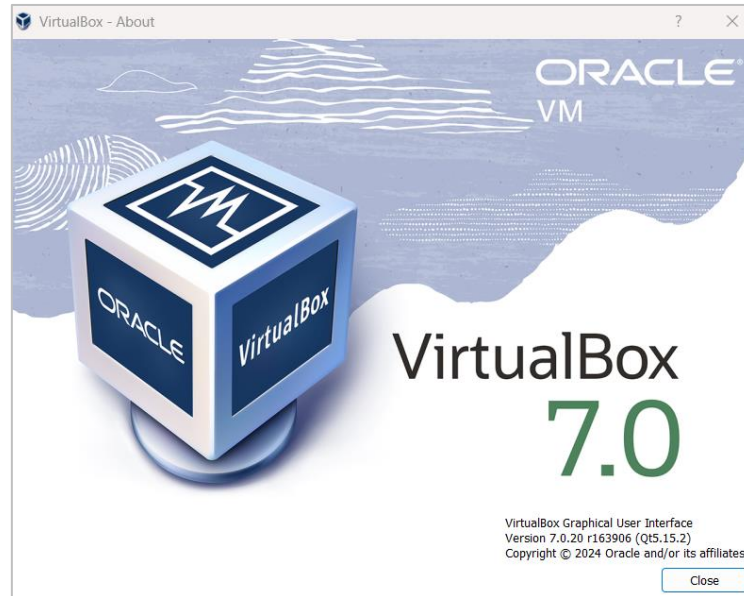


Figura 10 - Versión de VirtualBox

2. Este clúster “virtual” tendrá la configuración mostrada en la figura 11:

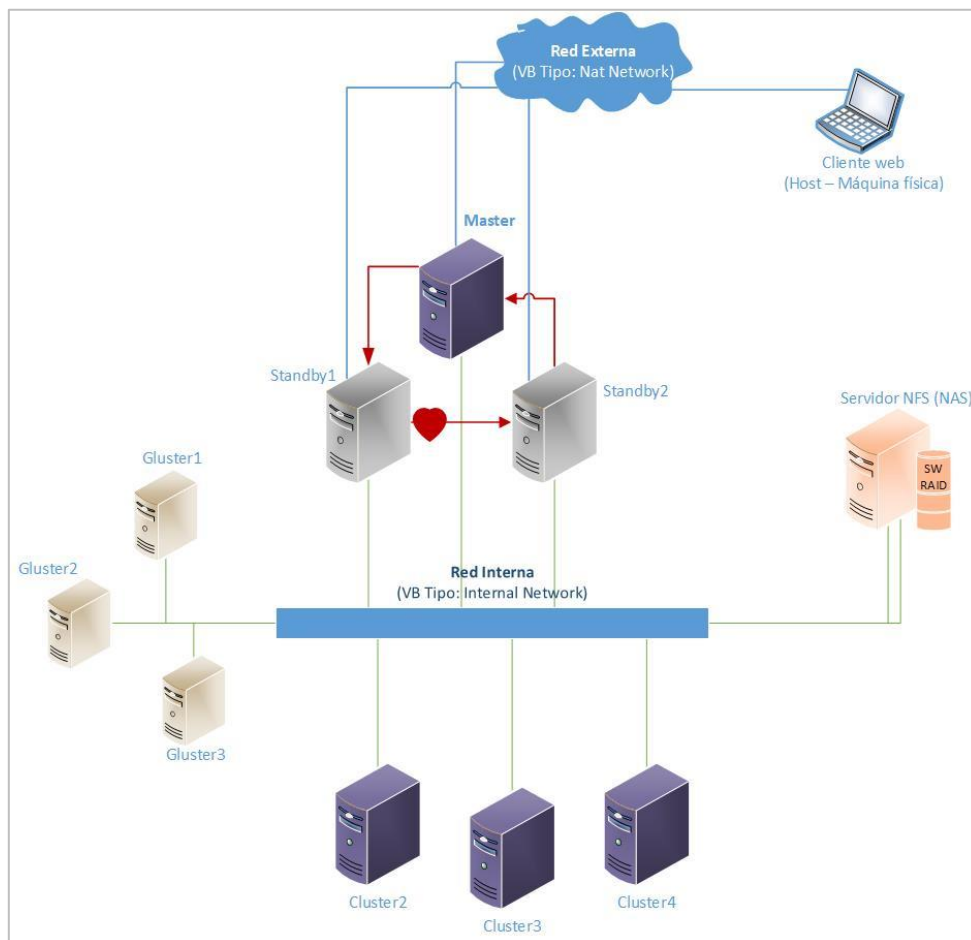


Figura 11 - Diseño Lógico de Clúster TFM

- ▶ La **red interna** corresponde a la red del clúster. A través de esta red se comunican los nodos maestros y los servidores. Los nodos maestros también se comunican entre sí para monitorizar su funcionamiento. Asimismo, permite el acceso al servidor de almacenamiento. Se configura en VirtualBox como una red de tipo *internal*.
  - ▶ La **red externa** permite el acceso a las máquinas del clúster a la Internet real. Se configura en VirtualBox con una red de tipo *nat-network*. Este tipo de red también permite la comunicación entre las máquinas virtuales y la máquina anfitrión, y viceversa, en este último caso redirigiendo puertos de la máquina anfitrión hacia las máquinas virtuales.
3. Se agregarán 3 discos al servidor de almacenamiento NFS, donde se configurará un RAID5 por software. Adicional se asignarán dos tarjetas de red a este servidor, donde posteriormente se configurará un link aggregation, esto con el fin de añadir redundancia en la red para evitar puntos únicos de fallo.
  4. Se agregarán 2 discos a cada uno de los servidores que forman parte del sistema de almacenamiento por GlusterFS, esto permitirá definir dos volúmenes tipo Glusters, uno para datos distribuidos y otro para datos replicados.

### Instalación del sistema

5. El sistema operativo que usarán todos los nodos del clúster es Debian 12 "bookworm".
6. Se realizará una instalación individual personalizada para los nodos maestros (front-end), los servidores para reparto de carga (back-end), el servidor de almacenamiento y los servidores glusters.
7. Una vez que los nodos del back-end estén listos, se levantarán los contenedores sobre cada uno de ellos.

### Instalación de los servicios

8. En el servidor máster se instalará y configurará los servicios necesarios para el correcto funcionamiento del clúster (DHCP, NAT).
9. El servidor de almacenamiento ofrecerá un servicio NFS a todos los nodos del sistema. El servidor de almacenamiento montará un RAID 5 software.
10. La distribución de la carga en los nodos maestros se realizará mediante HAProxy.
11. Los mecanismos de alta disponibilidad en los nodos directores se implementarán mediante Keepalived en primera instancia y luego se configurará pacemaker con corosync y stonith.
12. Se instalará el paquete de dockers en los equipos de back-end y se levantarán 3 contenedores en cada servidor, cada uno de esos contenedores iniciará una imagen de Apache+PHP. En total serán 9 contenedores en el ambiente.



13. Las páginas web residirán en el espacio compartido ofrecido por el servidor de almacenamiento NFS y también en el espacio replicado ofrecido por GlusterFS.
14. Se tendrá un segundo espacio compartido ofrecido a todos los nodos del clúster, para almacenar información variada, tanto en un volumen distribuido como en un volumen replicado, y será ofrecido por tres servidores Gluster.

### 3.4 DESARROLLO DE LA IMPLEMENTACIÓN

En esta sección mostraremos pantallas obtenidas del proceso de implementación de cada nodo y servicio dentro del clúster.

Partimos de las máquinas con el disco duro vacío, sin particionado y sin sistema operativo. La distribución Linux elegida para todos los nodos del clúster es **Debian12** (64 bits) "bookworm".

La instalación completa se realiza con base al siguiente diseño físico:

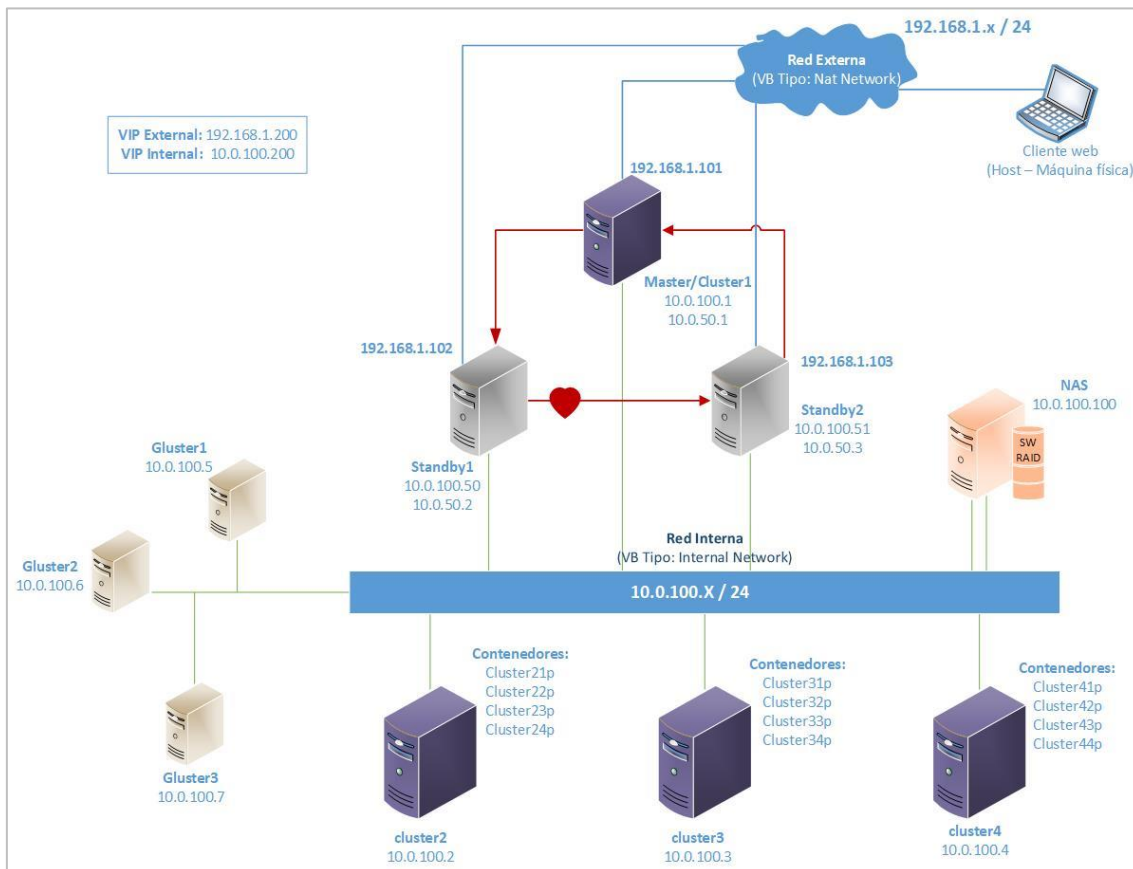


Figura 12 - Diseño Físico de Clúster TFM

### 3.4.1 Instalación del nodo servidor de almacenamiento NFS

La configuración de red de este servidor se detalla en la tabla 1:

Aspecto	Valor
Hostname	nas
Tarjeta	bond0 (interna)
IP	10.0.100.100 (estática)
Mascara	255.255.255.0
Gateway	10.0.100.200
DNS	10.0.100.200

Tabla 1 - Configuración de red nodo NFS

A este servidor se le agregaron 3 discos, con el fin de crear un RAID 5 por software y compartir este recurso a todos los servidores del clúster vía NFS.

El procedimiento que se siguió se detalla a continuación:

- Validar que los discos sean reconocidos por el sistema operativo. Los comandos **fdisk** o **lsblk** ayudan en esta tarea.

```

root@nas:/etc/init.d# fdisk -l
Disk /dev/sda: 10 GiB, 10737418240 bytes, 20971520 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xc41ca385

Device      Boot      Start          End  Sectors   Size Id Type
/dev/sda1   *          2048        9764863   9762816   4.7G 83 Linux
/dev/sda2             9764864   19529727   9764864   4.7G 83 Linux
/dev/sda3             19529728   20969471  1439744    703M 82 Linux swap / Solaris

Disk /dev/sdb: 200 MiB, 209715200 bytes, 409600 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/sdc: 200 MiB, 209715200 bytes, 409600 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/sdd: 200 MiB, 209715200 bytes, 409600 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
root@nas:/etc/init.d#

```

Figura 13 - Diseño de Clúster TFM

- ▶ Crear las particiones primarias en cada disco para posteriormente crear el RAID. Usaremos la utilidad de “**mdadm**” para crear el dispositivo md0 (un RAID de nivel 5) a partir de los **tres** primeros discos que añadimos al servidor.

```

root@nas:~# lsblk
NAME MAJ:MIN RM  SIZE RO  TYPE MOUNTPOINTS
sda   8:0    0   10G  0  disk
├─sda1 8:1    0   4.7G  0  part /
├─sda2 8:2    0   4.7G  0  part
└─sda3 8:3    0   703M  0  part [SWAP]
sdb   8:16   0   200M  0  disk
└─sdb1 8:17   0   199M  0  part
sdc   8:32   0   200M  0  disk
└─sdc1 8:33   0   199M  0  part
sdd   8:48   0   200M  0  disk
└─sdd1 8:49   0   199M  0  part
sr0   11:0   1  1024M  0  rom
root@nas:~# cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# systemd generates mount units based on this file, see systemd.mount(5).
# Please run 'systemctl daemon-reload' after making changes here.
#
# <file system> <mount point> <type> <options>          <dump> <pass>
# / was on /dev/sda1 during installation
UUID=19e10c55-a781-4da1-bbf1-d32f0ea48174 /          ext4      errors=remount-ro 0      1
# swap was on /dev/sda3 during installation
UUID=a77cce01-0bd1-4595-9610-8914ec30b102 none      swap      sw         0      0
/dev/sr0    /media/cdrom0  udf,iso9660 user,noauto 0      0
root@nas:~# apt-get install mdadm
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
mdadm is already the newest version (4.2-5).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@nas:~# mdadm --create /dev/md0 --level=5 --raid-devices=3 /dev/sdb1 /dev/sdc1 /dev/sdd1

```

Figura 14 - Creación de RAID

En el RAID5 se aplicará la distribución por defecto (left-symmetric). Asimismo, se usará el valor por defecto, 512 KB, para el tamaño de la *stripe unit*.

- ▶ Comprobar el estado del dispositivo **md0** revisando el contenido del fichero **/proc/mdstat**

```

root@nas:~# cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4] [raid10]
md0 : active raid5 sdd1[3] sdc1[1] sdb1[0]
      403456 blocks super 1.2 level 5, 512k chunk, algorithm 2 [3/3] [UUU]

unused devices: <none>
root@nas:~# _

```

Figura 15 - Fichero de Creación del RAID

Adicional, es posible obtener más información sobre la configuración del RAID con el mismo comando “**mdadm**”:

```

root@nas:~# mdadm --detail /dev/md0
/dev/md0:
  Version : 1.2
  Creation Time : Thu Jul  4 18:09:42 2024
  Raid Level : raid5
  Array Size : 403456 (394.00 MiB 413.14 MB)
  Used Dev Size : 201728 (197.00 MiB 206.57 MB)
  Raid Devices : 3
  Total Devices : 3
  Persistence : Superblock is persistent

  Update Time : Thu Jul  4 18:09:44 2024
  State : clean
  Active Devices : 3
  Working Devices : 3
  Failed Devices : 0
  Spare Devices : 0

  Layout : left-symmetric
  Chunk Size : 512K

Consistency Policy : resync

  Name : nas:0 (local to host nas)
  UUID : 2d1f1052:a6ecdc46:8c1564f7:67686e43
  Events : 18

   Number Major Minor RaidDevice State
    0         8   17        0  active sync  /dev/sdb1
    1         8   33        1  active sync  /dev/sdc1
    3         8   49        2  active sync  /dev/sdd1
root@nas:~# _

```

Figura 16 - Detalles del RAID

- Luego de instalar un sistema de ficheros en el dispositivo **md0** y montarlo en un directorio, podemos verificar su capacidad. En nuestro caso, montamos el sistema de ficheros sobre el directorio **/raid** y configuramos **el servicio NFS** para exportar el directorio y hacerlo disponible a todos los nodos del clúster.

```

File Machine View Input Devices Help
root@nas:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            458M   0  458M   0% /dev
tmpfs           97M   604K   96M   1% /run
/dev/sda1       4.6G  1.5G  2.8G  35% /
tmpfs           481M   0  481M   0% /dev/shm
tmpfs           5.0M   0   5.0M   0% /run/lock
/dev/md0        359M  5.1M  331M   2% /raid
tmpfs           97M   0   97M   0% /run/user/0
root@nas:~# _

```

Figura 17 - Directorio del RAID

Exportamos el directorio **/raid** modificando el fichero **/etc/exports** y añadiendo al final la línea siguiente:

```

root@nas:~# cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
# to NFS clients. See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
#
/raid 10.0.100.0/24(rw,no_root_squash,sync,no_subtree_check)
root@nas:~# _

```

Figura 18 - Configuración NFS Server

Luego de reiniciar el servicio **nfs-kernel-server**, comprobamos que el directorio **/raid** forma parte del listado de directorios exportados por el servicio NFS:

```

TFMNas [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@nas:~# exportfs
/raid 10.0.100.0/24
root@nas:~# _

```

Figura 19 - Directorio Exportado por NFS Server

- ▶ Queda comprobar el acceso al directorio exportado desde el resto de los nodos del clúster, y para eso se debe configurar el NFS Client.

La siguiente línea nos permite observar los directorios que exporta el servidor NFS:

```

LXTerminal
File Edit Tabs Help
root@cluster1:~# showmount -e nas
Export list for nas:
/raid 10.0.100.0/24
root@cluster1:~# █

```

Figura 20 - Listado de Directorios Exportados por NFS Server

Y dejamos el directorio agregado al fichero **/etc/fstab**, con lo que cada vez que se inicie el sistema, el directorio remoto se montará automáticamente. Este proceso se debe hacer en cada uno de los nodos del clúster.

```

root@cluster1:~# cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# systemd generates mount units based on this file, see systemd.mount(5).
# Please run 'systemctl daemon-reload' after making changes here.
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sdal during installation
UUID=19e10c55-a781-4da1-bbf1-d32f0ea48174 / ext4 errors=remount-ro 0 1
# swap was on /dev/sda3 during installation
UUID=a77cce01-0bd1-4595-9610-8914ec30b102 none swap sw 0 0
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
nas:/ /nfs nfs auto,nofail,bg,rsize=8192,wsz=8192 0 0

```

Figura 21 - Fichero /etc/fstab

De esta manera, está listo el sistema de archivos basado en RAID5 y exportado por el servidor NFS.

- ▶ Como último punto de preparación del nodo de almacenamiento NFS, configuraremos el **link aggregation**. El servidor dispone de 2 tarjetas de red, modificaremos el archivo de configuración “**interfaces**” ubicado en el directorio **/etc/network**. El archivo debe lucir así:

```

File Machine View Input Devices Help
source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

#Slave
#auto enp0s3
#iface enp0s3 inet manual

#Slave
#auto enp0s8
#iface enp0s8 inet manual

#Bond
auto bond0
iface bond0 inet static
    address 10.0.100.100
    netmask 255.255.255.0
    gateway 10.0.100.200
    dns-nameservers 10.0.100.200
    slaves enp0s3 enp0s8
#
# bond-primary enp0s3 enp0s8
#
# bond-mode balance-rr
# bond-mode active-backup
# bond-mode balance-xor
# bond-mode broadcast
# bond-mode 802.3ad
# bond-mode balance-tlb
# bond-mode balance-alb
#
# bond-miimon 100
# bond-downdelay 200
# bond-updelay 200
# bond-fail_over_mac 1

root@nas:/etc/network# _

```

Figura 22 - Configuración link aggregation

Luego de reiniciar el **servicio de networking**, las tarjetas lucen así:

```

File  Machine  View  Input  Devices  Help
root@nas:/etc/init.d# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc fq_codel master bond0 state UP group
up default qlen 1000
    link/ether 08:00:27:b2:5d:43 brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc fq_codel master bond0 state UP group
up default qlen 1000
    link/ether 08:00:27:65:10:f8 brd ff:ff:ff:ff:ff:ff
4: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen
1000
    link/ether 08:00:27:b2:5d:43 brd ff:ff:ff:ff:ff:ff
    inet 10.0.100.100/24 brd 10.0.100.255 scope global bond0
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feb2:5d43/64 scope link
        valid_lft forever preferred_lft forever
root@nas:/etc/init.d#

```

Figura 23 - Tarjeta bond0

### 3.4.2 Instalación del nodo Máster

El nodo máster es el servidor principal en el esquema de alta disponibilidad, siendo así tiene varios servicios configurados que permiten mantener el sistema de alta disponibilidad y el reparto de carga dentro del clúster. Adicional es un servidor DHCP para todos los equipos dentro de la red.

Este servidor tiene 3 tarjetas de red: una para la red externa y dos para la red interna, una de las cuales es exclusiva para la comunicación con los 2 servidores standbys.

En la siguiente tabla se especifica la configuración de red del nodo máster:

Aspecto	Valor
Hostname	cluster1
Tarjeta	enp0s3 (externa)
IP	192.168.1.101 (estática)
Máscara	255.255.255.0
Gateway	192.168.1.1
DNS	externos
Tarjeta	enp0s8 (interna)
IP	10.0.100.1 (estática)
Máscara	255.255.255.0

<b>Tarjeta</b>	enp0s9 (interna heartbeat)
<b>IP</b>	10.0.50.1 (estática)
<b>Máscara</b>	255.255.255.0

Tabla 2 - Configuración de red nodo Máster

Esta configuración se almacena en los respectivos archivos de configuración:

- ▶ Archivo de configuración de red /etc/network/interfaces:

```

LXTerminal
File Edit Tabs Help
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
#allow-hotplug enp0s3
#iface enp0s3 inet dhcp

auto enp0s3
iface enp0s3 inet static
    address 192.168.1.101
    netmask 255.255.255.0
    gateway 192.168.1.1

auto enp0s8
iface enp0s8 inet static
    address 10.0.100.1
    netmask 255.255.255.0

auto enp0s9
iface enp0s9 inet static
    address 10.0.50.1
    netmask 255.255.255.0

root@cluster1:/nfs/raid#

```

Figura 24 - Archivo de Configuración de red nodo máster

- ▶ Archivo de DNS

```

LXTerminal
File Edit Tabs Help
root@cluster1:/nfs/raid# cat /etc/resolv.conf
domain home
search home
nameserver 212.230.135.2
nameserver 212.230.135.1
root@cluster1:/nfs/raid#

```

Figura 25 - Archivo de Configuración de DNS nodo máster



► Rutas del sistema:

```

LXTerminal
File Edit Tabs Help
root@cluster1:/etc/network# route -n
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0          192.168.1.1    0.0.0.0         UG    0      0      0 enp0s3
10.0.50.0        0.0.0.0        255.255.255.0   U     0      0      0 enp0s9
10.0.100.0       0.0.0.0        255.255.255.0   U     0      0      0 enp0s8
169.254.0.0     0.0.0.0        255.255.0.0     U    1000   0      0 enp0s3
192.168.1.0     0.0.0.0        255.255.255.0   U     0      0      0 enp0s3
root@cluster1:/etc/network#

```

Figura 26 - Tabla de rutas nodo máster

### Configuración de servicio DHCP

El nodo máster desempeña el rol de servidor DNS/DHCP para todos los equipos de la red interna. Para configurar dicho servicio, se utilizó **DNSMASQ**, que es un servidor DHCP/DNS sencillo que puede ser utilizado en una red local de hasta unos 1.000 clientes. Sus características fundamentales son: una fácil configuración y un bajo consumo de recursos del sistema.

En la ruta **/etc/dnsmasq.conf** se guarda el archivo de configuración y el servicio puede ser manejado con:

**systemctl status/start/stop/restart dnsmasq.**

En las siguientes figuras se muestra la configuración del servicio DHCP y archivos relacionados:

```

File Edit Tabs Help
# Do not abort if the tftp-root is unavailable
#tftp-no-fail

# Make the TFTP server more secure: with this set, only files owned by
# the user dnsmasq is running as will be send over the net.
#tftp-secure

# This option stops dnsmasq from negotiating a larger blocksize for TFTP
# transfers. It will slow things down, but may rescue some broken TFTP
# clients.
#tftp-no-blocksize

# Set the boot file name only when the "red" tag is set.
#dhcp-boot=/pxelinux.0
dhcp-range=10.0.100.2,10.0.100.49,infinite

# An example of dhcp-boot with an external TFTP server: the name and IP
# address of the server are given after the filename.
# Can fail with old PXE ROMS. Overridden by --pxe-service.
#dhcp-boot=/var/ftpd/pxelinux.0,boothost,192.168.0.3

```

Figura 27 - Archivo de configuración DNSMASQ - parte 1

```
GNU nano 7.2 /etc/dnsmasq.conf
#log-queries

# Log lots of extra information about DHCP transactions.
log-dhcp

# Include another lot of configuration options.
#conf-file=/etc/dnsmasq.more.conf
#conf-dir=/etc/dnsmasq.d

# Include all the files in a directory except those ending in .bak
#conf-dir=/etc/dnsmasq.d,.bak

# Include all files in a directory which end in .conf
#conf-dir=/etc/dnsmasq.d/*.conf

# If a DHCP client claims that its name is "wpad", ignore that.
# This fixes a security hole. see CERT Vulnerability VU#598349
#dhcp-name-match=set:wpad-ignore,wpad
#dhcp-ignore-names=tag:wpad-ignore
dhcp-option=3,10.0.100.200
dhcp-option=6,10.0.100.200
interface=enp0s8
#read-ethers
dhcp-hostsfile=/etc/dnsmasq.hosts.conf
```

Figura 28 - Archivo de configuración DNSMASQ - parte 2

```
File Edit Tabs Help
GNU nano 7.2 /etc/dnsmasq.hosts.conf
08:00:27:*:*:
08:00:27:1E:BA:74,10.0.100.2,cluster2
08:00:27:0B:BE:50,10.0.100.3,cluster3
08:00:27:FA:F6:06,10.0.100.4,cluster4
08:00:27:5C:FB:CC,10.0.100.5,gluster1
08:00:27:EA:52:DC,10.0.100.6,gluster2
08:00:27:60:17:50,10.0.100.7,gluster3
```

Figura 29 - Archivo de asignación de nombres por DHCP

```
LXTerminal
File Edit Tabs Help
root@cluster1:/etc/init.d# cat /etc/ethers
08:00:27:1E:BA:74 10.0.100.2
08:00:27:0B:BE:50 10.0.100.3
08:00:27:FA:F6:06 10.0.100.4
08:00:27:5C:FB:CC 10.0.100.5
08:00:27:EA:52:DC 10.0.100.6
08:00:27:60:17:50 10.0.100.7

root@cluster1:/etc/init.d#
```

Figura 30 - Archivo de reservas de DHCP

```

root@cluster1:/etc/init.d# systemctl status dnsmasq
● dnsmasq.service - dnsmasq - A lightweight DHCP and caching DNS server
   Loaded: loaded (/lib/systemd/system/dnsmasq.service; enabled; preset: enabled)
   Active: active (running) since Wed 2024-08-28 00:45:14 CEST; 1 day 22h ago
     Process: 557 ExecStartPre=/etc/init.d/dnsmasq checkconfig (code=exited, status=0/SUCCESS)
     Process: 637 ExecStart=/etc/init.d/dnsmasq systemd-exec (code=exited, status=0/SUCCESS)
     Process: 687 ExecStartPost=/etc/init.d/dnsmasq systemd-start-resolvconf (code=exited, status=0/SUCCESS)
    Main PID: 684 (dnsmasq)
      Tasks: 1 (limit: 1098)
     Memory: 2.4M
        CPU: 91ms
    CGroup: /system.slice/dnsmasq.service
            └─684 /usr/sbin/dnsmasq -x /run/dnsmasq/dnsmasq.pid -u dnsmasq -7 /etc/dnsmasq.d,.dp

Aug 28 00:47:41 cluster1 dnsmasq-dhcp[684]: 3842470719 next server: 10.0.100.1
Aug 28 00:47:41 cluster1 dnsmasq-dhcp[684]: 3842470719 sent size: 1 option: 53 message-type 2
Aug 28 00:47:41 cluster1 dnsmasq-dhcp[684]: 3842470719 sent size: 4 option: 54 server-identifier
Aug 28 00:47:41 cluster1 dnsmasq-dhcp[684]: 3842470719 sent size: 4 option: 51 lease-time infin
Aug 28 00:47:41 cluster1 dnsmasq-dhcp[684]: 3842470719 sent size: 4 option: 1 netmask 255.255.255
Aug 28 00:47:41 cluster1 dnsmasq-dhcp[684]: 3842470719 sent size: 4 option: 28 broadcast 10.0.100.1
Aug 28 00:47:41 cluster1 dnsmasq-dhcp[684]: 3842470719 sent size: 4 option: 6 dns-server 10.0.100.1
Aug 28 00:47:41 cluster1 dnsmasq-dhcp[684]: 3842470719 sent size: 4 option: 3 router 10.0.100.1
Aug 28 00:47:41 cluster1 dnsmasq-dhcp[684]: 3842470719 available DHCP range: 10.0.100.2 -- 10.0.100.254
Aug 28 00:47:41 cluster1 dnsmasq-dhcp[684]: 3842470719 client provides name: localhost

```

Figura 31 - Servicio DNSMASQ

En el caso de un clúster formado por un número elevado de máquinas, sería deseable automatizar la creación de los archivos `/etc/ethers` y `/etc/dnsmasq.hosts.conf`.

### Configuración de servicio de distribución de carga HA Proxy

En esta sección mostraremos los archivos de configuración relacionados a HA Proxy, para el nodo máster. Realizamos una configuración básica de todos los repartidores para que el clúster ofrezca un servicio web (HTTP). Con HAProxy probaremos el reparto de carga a nivel de transporte (modo TCP) y a nivel de aplicación (modo HTTP).

El archivo de configuración del servicio de HA Proxy se encuentra ubicado en la siguiente ruta:

**`/etc/haproxy/haproxy.cfg`**

```
global
log /dev/log      local0
log /dev/log      local1 notice
chroot /var/lib/haproxy
stats socket /run/haproxy/admin.sock mode 660 level admin
stats timeout 30s
user haproxy
group haproxy
daemon

# Default SSL material locations
ca-base /etc/ssl/certs
crt-base /etc/ssl/private

# See: https://ssl-config.mozilla.org/#server=haproxy&server-version=2.0.3&config=intermediate
ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384
ssl-default-bind-ciphersuites TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets

defaults
log      global
mode     http
option   httplog
option   dontlognull
timeout connect 5000

timeout client 3000
timeout server 3000
errorfile 400 /etc/haproxy/errors/400.http
errorfile 403 /etc/haproxy/errors/403.http
errorfile 408 /etc/haproxy/errors/408.http
errorfile 500 /etc/haproxy/errors/500.http
errorfile 502 /etc/haproxy/errors/502.http
errorfile 503 /etc/haproxy/errors/503.http
errorfile 504 /etc/haproxy/errors/504.http

frontend haproxy-http
mode http
option httplog
option forwardfor
bind 192.168.1.200:8000
default_backend http-servers

frontend haproxy-tcp
mode tcp
option tcplog
bind 192.168.1.200:80
default_backend tcp-servers

frontend haproxy-http_g
mode http
option httplog
option forwardfor
bind 192.168.1.200:9000
default_backend http-servers-g

frontend haproxy-tcp_g
mode tcp
option tcplog
bind 192.168.1.200:90
default_backend tcp-servers-g

backend http-servers
mode http
balance roundrobin
option redispatch
cookie SERVERID insert
option httpchk
server cluster21p cluster2:8081 cookie cluster21p check
server cluster22p cluster2:8082 cookie cluster22p check
server cluster23p cluster2:8083 cookie cluster23p check
server cluster31p cluster3:8081 cookie cluster31p check
server cluster32p cluster3:8082 cookie cluster32p check
server cluster33p cluster3:8083 cookie cluster33p check
server cluster41p cluster4:8081 cookie cluster41p check
```

Figura 32 - Archivo de Configuración HA Proxy - parte 1

```

server cluster42p cluster4:8082 cookie cluster42p check
server cluster43p cluster4:8083 cookie cluster43p check

backend tcp-servers
  mode tcp
  balance roundrobin
  server cluster21p cluster2:8081 check
  server cluster22p cluster2:8082 check
  server cluster23p cluster2:8083 check
  server cluster31p cluster3:8081 check
  server cluster32p cluster3:8082 check
  server cluster33p cluster3:8083 check
  server cluster41p cluster4:8081 check
  server cluster42p cluster4:8082 check
  server cluster43p cluster4:8083 check

backend http-servers-g
  mode http
  balance roundrobin
  option redispatch
  cookie SERVERID insert
  option httpchk
  server cluster24p cluster2:8084 cookie cluster24p check
  server cluster34p cluster3:8085 cookie cluster34p check
  server cluster44p cluster4:8086 cookie cluster44p check

backend tcp-servers-g
  mode tcp
  balance roundrobin
  server cluster24p cluster2:8084 check
  server cluster34p cluster3:8085 check
  server cluster44p cluster4:8086 check

listen stats
  bind 0.0.0.0:7000
  mode http
  stats show-desc Nodo cluster1
  stats refresh 5s
  stats uri /haproxy?stats
  stats realm HAProxy Statistics
  stats auth admin:password
  stats admin if TRUE
  
```

Figura 33 - Archivo de Configuración HA Proxy - parte 2

En las secciones *frontend* se definen los servicios ofrecidos en los que se va a repartir la carga, además se muestra la dirección IP virtual (VIP) externa en la que se van a recibir los requerimientos de los clientes finales. Las peticiones se envían a los servidores indicados en las secciones *backend*, que son los que realmente servirán las páginas web, en nuestro caso los contenedores.

En particular, se han definido los servicios siguientes:

- ▶ **192.168.1.200:8000.** Reparto de carga en modo HTTP con persistencia basada en cookies. Las peticiones se atienden siempre por el mismo nodo virtual. Las páginas web están almacenadas en el directorio compartido por el nodo NFS.
- ▶ **192.168.1.200:80.** Reparto de carga en modo TCP. Las peticiones se reparten de un modo equilibrado entre todos los servidores virtuales configurados. Las páginas web están almacenadas en el directorio compartido por el nodo NFS.
- ▶ **192.168.1.200:9000.** Reparto de carga en modo HTTP con persistencia basada en cookies. Las peticiones se atienden siempre por el mismo nodo virtual. Las páginas web están almacenadas en el sistema de archivos GlusterFS.
- ▶ **192.168.1.200:90.** Reparto de carga en modo TCP. Las peticiones se reparten de un modo equilibrado entre todos los servidores virtuales configurados. Las páginas web están almacenadas en el sistema de archivos GlusterFS.

Finalmente, en el último apartado se habilita la página web de estadísticas con los credenciales **admin:password**

Si se realiza cualquier cambio en el archivo de configuración, se puede ejecutar el siguiente comando para comprobar que el archivo de configuración está correcto:

```
# haproxy -f /etc/haproxy/haproxy.cfg -c
```

Finalmente, para manejar el servicio se debe ejecutar el siguiente comando, variando las instrucciones entre start, stop, restart. Ahora, reiniciamos el servicio para que trabaje con la nueva configuración:

```
# systemctl start / stop / restart haproxy
```

### Configuración de servicio de alta disponibilidad KeepAlived

En la sección anterior se configuró el distribuidor de carga sobre el nodo maestro. En esta sección, añadiremos dos nodos adicionales (standby1 y standby2) que asumirán la función de distribuidor en caso de que el nodo maestro falle.

Para ello, definiremos un par de **direcciones IP virtuales (VIP)**, una por cada interfaz de red de los nodos maestros o directores, de manera que el nodo que esté activo será el que las tendrá configuradas. En la siguiente tabla se especifican las direcciones IPs implicadas en la configuración:

	Hostname	Dirección IP	
<b>Repartidor de carga Master</b>	cluster1(enp0s3) cluster1 (enp0s8)	DIP: 192.168.1.101 DIP: 10.0.100.1	<b>VIP: 192.168.1.200</b> <b>VIP: 10.0.100.200</b>
<b>Repartidor de carga StandBy1</b>	standby1(enp0s3) standby1(enp0s8)	DIP: 192.168.1.102 DIP: 10.0.100.50	VIP: 192.168.1.200 VIP: 10.0.100.200
<b>Repartidor de carga StandBy2</b>	standby2(enp0s3) standby2(enp0s8)	DIP: 192.168.1.103 DIP: 10.0.100.51	VIP: 192.168.1.200 VIP: 10.0.100.200
<b>Servidores Reales</b>	cluster2 (enp0s8) cluster3 (enp0s8) cluster4 (enp0s8)	RIP: 10.0.100.2 RIP: 10.0.100.3 RIP: 10.0.100.4	
<b>Servidor NAS-NFS</b>	nas (bond0)	RIP: 10.0.100.100	
<b>Servidores de Gluster</b>	Gluster1 (enp0s3) Gluster2 (enp0s3) Gluster3 (enp0s3)	RIP: 10.0.100.5 RIP: 10.0.100.6 RIP: 10.0.100.7	

Tabla 3 - Direcciones IPs del Clúster

El archivo `/etc/hosts`, que se encuentra en todos los nodos del clúster, finalmente queda de la siguiente manera:

```

root@cluster1:/gluster/gfs-r/www# cat /etc/hosts
127.0.0.1    localhost

# The following lines are desirable for IPv6 capable hosts
::1        localhost ip6-localhost ip6-loopback
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
10.0.100.100  nas.cluster    nas
10.0.100.1   cluster1.cluster    cluster1    master lb1
10.0.100.2   cluster2.cluster    cluster2    server1
10.0.100.3   cluster3.cluster    cluster3    server2
10.0.100.4   cluster4.cluster    cluster4    server3
10.0.100.5   gluster1.cluster    gluster1
10.0.100.6   gluster2.cluster    gluster2
10.0.100.7   gluster3.cluster    gluster3
10.0.100.50  standby1.cluster    standby1
10.0.100.51  standby2.cluster    standby2
10.0.50.1    cluster1h.cluster   cluster1h
10.0.50.2    standby1h.cluster   standby1h
10.0.50.3    standby2h.cluster   standby2h

```

Figura 34 - Archivo Hosts

Para ofrecer alta disponibilidad, nuestro clúster debe tener dos direcciones VIP configuradas:

- ▶ La IP virtual en la red externa (Red-NAT). Como el servicio de distribución de carga puede migrar entre el máster (192.168.1.101) y los equipos standby (192.168.1.102 y 192.168.1.103), hay que definir una IP virtual (VIP: 192.168.1.200) que será un alias de la máquina que, en cada momento, esté dando el servicio. Esta será la dirección “pública” del servidor web y a la que accederán los clientes.
- ▶ La IP virtual en la red interna. Como el servicio de distribución de carga puede migrar entre el máster (10.0.100.1) y los equipos standby (10.0.100.50 y 10.0.100.51), hay que definir una IP virtual (VIP: 10.0.100.200) que será un alias en la red interna de la máquina que, en cada momento, esté dando servicio. Esto es necesario, para el caso de que falle uno de los distribuidores de carga. El distribuidor superviviente debe hacer de *router* de los servidores reales (cluster2, cluster3 y cluster4). Por tanto, será necesario también configurar estos servidores reales para que su *router* sea 10.0.100.200 (ver más adelante).

Para definir las direcciones VIP y que éstas se asignen dinámicamente al repartidor de carga que esté activo, utilizaremos el **paquete keepalived**, que implementa la función de *vrpp* (*virtual router redundancy protocol*). Para ello, lo instalaremos y modificaremos su archivo de configuración, añadiendo los interfaces de red virtuales al principio.

El archivo de configuración completo `/etc/keepalived/keepalived.conf` se muestra a continuación:

```
File Edit Tabs Help
vrrp_sync_group V61 {
    group {
        VI_1
        VI_2
    }
}

vrrp_instance VI_1 {
    state MASTER
    interface enp0s3
    virtual_router_id 51
    priority 150
    authentication {
        auth_type PASS
        auth_pass $ clustercac
    }
    virtual_ipaddress {
        192.168.1.200
    }
}

vrrp_instance VI_2 {
    state MASTER
    interface enp0s8
    virtual_router_id 52
    priority 150
    authentication {
        auth_type PASS
        auth_pass $ clustercac
    }
    virtual_ipaddress {
        10.0.100.200
    }
}

#Instancia VRRP para la interfaz de latidos enp0s9
vrrp_instance VI_3 {
    state MASTER
    interface enp0s9
    virtual_router_id 53
    priority 150
    authentication {
        auth_type PASS
        auth_pass $ clustercac
    }
}
```

Figura 35 - Archivo de Configuración Keepalived

En este caso, se han definido dos interfaces de red virtuales, correspondientes a la interfaz externa (VI\_1) e interna (VI\_2) de los nodos repartidores de carga. Además, están agrupadas, de manera que un fallo en cualquiera de ellas provocará la migración de las dos direcciones VIP.

Luego de concluir las configuraciones, debemos reiniciar el servicio para que trabaje con la nueva configuración:

**# systemctl start / stop / restart keepalived**



## Configuración de servicio de alta disponibilidad corosync + pacemaker

Otro sistema que ha sido probado dentro de este laboratorio de TFM es el servicio de alta disponibilidad con corosync y pacemaker. Tanto en el nodo máster como en los dos equipos standby, se procedió a instalar los paquetes que permiten levantar el servicio y administrarlo:

```
root@cluster1:~# apt-get install corosync pacemaker pcs
```

Figura 36 - Instalación de paquetes pacemaker

En la figura 36, se puede apreciar el comando que permite la instalación de los paquetes de corosync y pacemaker, y, además, instalamos pcs y crmsh (figura 37) como herramientas para administración del clúster vía comandos.

```
root@cluster1:/etc/corosync# apt-get install crmsh
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-parallax python3-yaml
Suggested packages:
  csync2 ocfs2-tools parted sbd ufw vim-addon-manager
The following NEW packages will be installed:
  crmsh python3-parallax python3-yaml
0 upgraded, 3 newly installed, 0 to remove and 10 not upgraded.
Need to get 737 kB of archives.
After this operation, 3,875 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://ftp.es.debian.org/debian bookworm/main amd64 python3-parallax all 1.0.6-4 [17.2 kB]
Get:2 http://ftp.es.debian.org/debian bookworm/main amd64 python3-yaml amd64 6.0-3+b2 [119 kB]
Get:3 http://ftp.es.debian.org/debian bookworm/main amd64 crmsh all 4.4.1-1+deb12u1 [601 kB]
Fetched 737 kB in 1s (1,271 kB/s)
Selecting previously unselected package python3-parallax.
(Reading database ... 63014 files and directories currently installed.)
Preparing to unpack ../python3-parallax_1.0.6-4_all.deb ...
Unpacking python3-parallax (1.0.6-4) ...
Selecting previously unselected package python3-yaml.
Preparing to unpack ../python3-yaml_6.0-3+b2_amd64.deb ...
```

Figura 37 - Instalación de paquetes de administración pacemaker

El primer paso para que nuestros servidores se puedan comunicar como un clúster es la modificación del archivo de configuración de corosync, el gestor de clúster en el que se basa Pacemaker, ubicado en la ruta `/etc/corosync/corosync.conf`. El archivo `corosync.conf` proporciona los parámetros de clúster utilizados por corosync, así que inicialmente lo haremos desde el nodo maestro `cluster1`, posterior se procederá a copiar dicho archivo en cada uno de los nodos que formarán parte del clúster.

En la siguiente figura se muestra dicho archivo:

```

GNU nano 7.2 corosync.conf
# Please read the corosync.conf.5 manual page
totem {
    version: 2

    # Corosync itself works without a cluster name, but DLM needs one.
    # The cluster name is also written into the VG metadata of newly
    # created shared LVM volume groups, if lvmlockd uses DLM locking.
    cluster_name: cluster_tfm
    transport: udpu
    interface {
        ringnumber: 0
        bindnetaddr: 10.0.50.0
        mcastport: 5405
        ttl: 1
    }

    # crypto_cipher and crypto_hash: Used for mutual node authentication.
    # If you choose to enable this, then do remember to create a shared
    # secret with "corosync-keygen".
    # enabling crypto_cipher, requires also enabling of crypto_hash.
    # crypto works only with knet transport
    crypto_cipher: none
    crypto_hash: none
}

logging {
    # Log the source file and line where messages are being
    # generated. When in doubt, leave off. Potentially useful for
    # debugging.
    fileline: off
    # Log to standard error. When in doubt, set to yes. Useful when
    # running in the foreground (when invoking "corosync -f")
    to_stderr: yes
    # Log to a log file. When set to "no", the "logfile" option
    # must not be set.
    to_logfile: yes
    logfile: /var/log/corosync/corosync.log
    # Log to the system log daemon. When in doubt, set to yes.
    to_syslog: yes
    # Log debug messages (very verbose). When in doubt, leave off.
    debug: off
    # Log messages with time stamps. When in doubt, set to hires (or on)
    #timestamp: hires
    logger_subsys {
        subsys: QUORUM
        debug: off
    }
}

quorum {
    # Enable and configure quorum subsystem (default: off)
    # see also corosync.conf.5 and votequorum.5
    provider: corosync_votequorum
}

nodelist {
    # Change/uncomment/add node sections to match cluster configuration

    node {
        # Hostname of the node
        name: cluster1
        # Cluster membership node identifier
        nodeid: 1
        # Address of first link
        ring0_addr: 10.0.50.1
        # When knet transport is used it's possible to define up to 8 links
        #ring1_addr: 192.168.1.1
    }
    node {
        name: standby1
        ring0_addr: 10.0.50.2
        nodeid: 2
    }
    node {
        name: standby2
        ring0_addr: 10.0.50.3
        nodeid: 3
    }
}
  
```

Figura 38 - Archivo de configuración corosync

Para una configuración básica, modificaremos la sección **totem**, que hace referencia al protocolo Totem que utiliza Corosync para definir la membresía del clúster y especifica cómo los miembros del clúster deben comunicarse entre sí. En nuestra configuración, las configuraciones importantes incluyen: **cluster\_name** (para definir un nombre para el cluster), **transport: udpu** (especifica el modo de unidifusión) y **bindnetaddr** (especifica a qué dirección de red debe vincularse Corosync), es la interfaz para heartbeat.

La sección de **nodelist**, especifica cada nodo del clúster y cómo se puede acceder a cada uno de ellos. Aquí, configuramos tanto el nodo primario como los secundarios y especificamos que se puede acceder a ellos a través de sus respectivas direcciones IP privadas, además les damos un id.

El segundo paso es copiar el archivo corosync.conf al resto de nodos que formarán el cluster, esto son los nodos standby1 y standby2.

```
Connection to 10.0.50.3 closed.
root@cluster1:/etc/corosync# nano corosync.conf
root@cluster1:/etc/corosync# scp corosync.conf 10.0.50.2:/etc/corosync/
corosync.conf                                100% 2025      1.0MB/s   00:00
root@cluster1:/etc/corosync# scp corosync.conf 10.0.50.3:/etc/corosync/
corosync.conf                                100% 2025      1.2MB/s   00:00
root@cluster1:/etc/corosync#
```

Figura 39 - Copiar archivo de configuración corosync

Antes de iniciar el servicio de corosync, es importante definir una contraseña para el usuario hacluster, y lo hacemos así:

```
root@cluster1:/etc/corosync# passwd hacluster
New password:
Retype new password:
passwd: password updated successfully
root@cluster1:/etc/corosync#
```

Figura 40 - Cambio de contraseña hacluster

Como tercer paso, procedemos a autorizar a los nodos del clúster. Para ello haremos uso de la herramienta de configuración "**pcs**", y ejecutaremos el comando que se muestra en la siguiente figura:

```
root@cluster1:/etc/corosync# pcs host auth cluster1 standby1 standby2
Username: hacluster
Password:
cluster1: Authorized
standby2: Authorized
standby1: Authorized
```

Figura 41 - Autorización nodos del clúster

Como cuarto paso, iniciar y habilitar el servicio de corosync, pacemaker y pcsd, con los siguientes comandos:

### systemctl status/start/stop/restart corosync

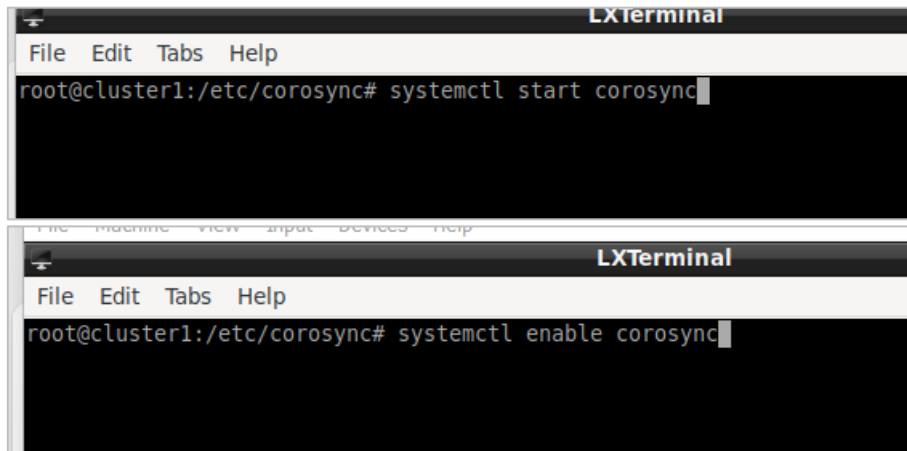


Figura 42 - Iniciar y Habilitar servicio de corosync

Para iniciar el servicio de pacemaker, se ejecuta el siguiente comando:

### systemctl status/start/stop/restart pacemaker

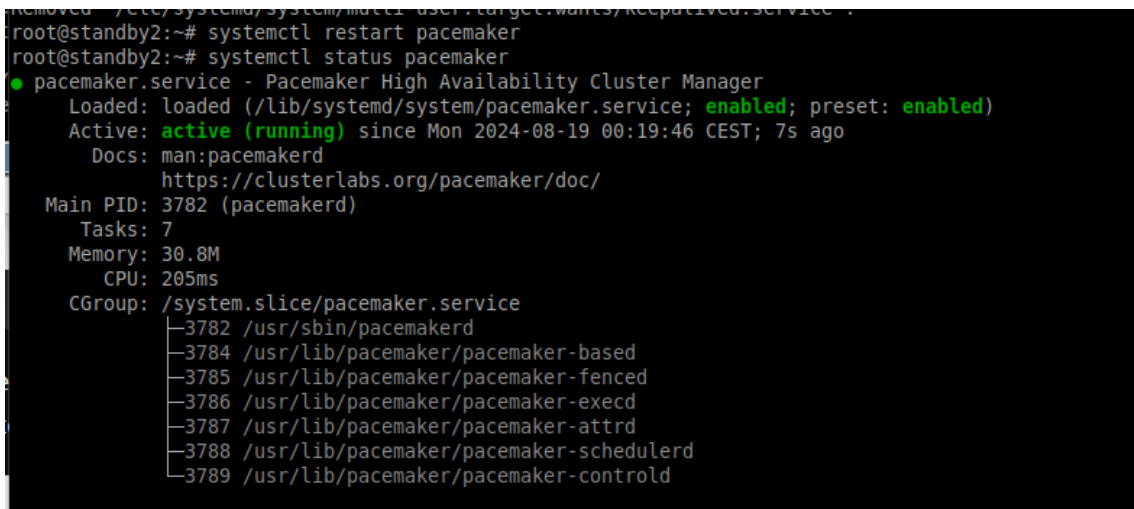


Figura 43 - Estado del servicio de pacemaker

Y para que se levante el servicio en tiempo de inicio del sistema, se debe ejecutar el siguiente comando:

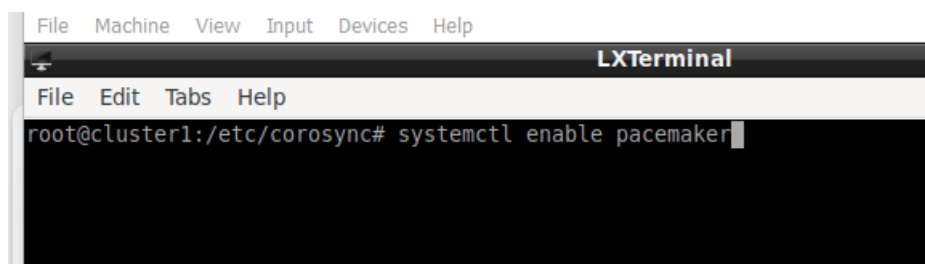


Figura 44 - Habilitar servicio de pacemaker

Finalmente, el servicio **pcds** para GUI administración del clúster, se inicia y habilita a través de los siguientes comandos:

**systemctl start pcsd**

**systemctl enable pcsd**

```

root@cluster1:/etc/corosync# systemctl status pcsd
● pcsd.service - PCS GUI and remote configuration interface
   Loaded: loaded (/lib/systemd/system/pcsd.service; enabled; preset: enabled)
   Active: active (running) since Sun 2024-08-18 22:18:23 CEST; 2h 23min ago
     Docs: man:pcsd(8)
           man:pcs(8)
  Main PID: 3452 (pcsd)
    Tasks: 29 (limit: 1098)
   Memory: 120.3M
      CPU: 1min 623ms
   CGroup: /system.slice/pcsd.service
           └─3452 /usr/bin/python3 -Es /usr/sbin/pcsd
             └─3459 /usr/bin/python3 -Es /usr/sbin/pcsd
               └─3469 /usr/bin/python3 -Es /usr/sbin/pcsd
                 └─3470 /usr/bin/python3 -Es /usr/sbin/pcsd
                   └─3471 /usr/bin/python3 -Es /usr/sbin/pcsd
                     └─3472 /usr/bin/python3 -Es /usr/sbin/pcsd
                       └─3474 /usr/bin/python3 -Es /usr/sbin/pcsd
                         └─3475 /usr/bin/python3 -Es /usr/sbin/pcsd
                           └─3481 /usr/bin/python3 -Es /usr/sbin/pcsd
                             └─3482 /usr/bin/python3 -Es /usr/sbin/pcsd
                               └─3483 /usr/bin/python3 -Es /usr/sbin/pcsd
                                 └─3491 /usr/bin/python3 -Es /usr/sbin/pcsd

Aug 18 22:18:21 cluster1 systemd[1]: Starting pcsd.service - PCS GUI and remote configuration interface:
Aug 18 22:18:23 cluster1 systemd[1]: Started pcsd.service - PCS GUI and remote configuration interface.
lines 1-25/25 (END)

```

Figura 45 - Estado del servicio pcsd

Con esto hemos conseguido armar el clúster y que los 3 nodos se encuentren comunicados y reportando su actividad. Podemos verificar el estado del clúster con algunos comandos que detallamos a continuación:

```

root@cluster1:/etc/corosync# pcs status
Cluster name: cluster_tfm

WARNINGS:
No stonith devices and stonith-enabled is not false

Status of pacemakerd: 'Pacemaker is running' (last updated 2024-08-19 00:22:51 +02:00)
Cluster Summary:
 * Stack: corosync
 * Current DC: cluster1 (version 2.1.5-a3f44794f94) - partition with quorum
 * Last updated: Mon Aug 19 00:22:52 2024
 * Last change: Mon Aug 19 00:16:24 2024 by hacluster via crmd on standby1
 * 3 nodes configured
 * 0 resource instances configured

Node List:
 * Online: [ cluster1 standby1 standby2 ]

Full List of Resources:
 * No resources

Daemon Status:
 corosync: active/enabled
 pacemaker: active/enabled
 pcsd: active/enabled
root@cluster1:/etc/corosync#

```

Figura 46 - Comandos de administración de pacemaker

```
root@standby1:/etc/corosync# pcs cluster status
Cluster Status:
Status of pacemakerd: 'Pacemaker is running' (last updated 2024-08-19 23:42:16 +02:00)
Cluster Summary:
 * Stack: corosync
 * Current DC: standby1 (version 2.1.5-a3f44794f94) - partition with quorum
 * Last updated: Mon Aug 19 23:42:16 2024
 * Last change: Mon Aug 19 02:11:04 2024 by root via cibadmin on cluster1
 * 3 nodes configured
 * 1 resource instance configured
Node List:
 * Online: [ cluster1 standby1 standby2 ]

PCSD Status:
cluster1: Online
standby1: Online
standby2: Online
root@standby1:/etc/corosync#
```

Figura 47 - Comandos de administración de pacemaker

```
root@cluster1:/etc/corosync# pcs status corosync

Membership information
-----
Nodeid      Votes Name
   1         1 cluster1 (local)
   2         1 standby1
   3         1 standby2
root@cluster1:/etc/corosync#
```

Figura 48 - Comandos de administración de pacemaker

Luego de configurar el servicio de corosync y validar que el clúster está arriba y corriendo, resta crear los recursos que se manejarán dentro del clúster. Lo haremos con el uso de las herramientas **pcs** o **crm** <sup>(12)</sup> (SUSE, 2024). Aunque pacemaker es bastante versátil y permite manejar una amplia variedad de tipos de recursos, esta vez nos enfocaremos en definir dos IP Virtuales “VIP” (externa e interna), que serán las que se transfieran entre nodos según la disponibilidad.

En las siguientes figuras mostraremos los comandos ejecutados, en secuencia, para crear los recursos de IP, grupo, entre otros:

```

root@cluster1:/var/log/pacemaker# pcs resource create IPVirtual_1 ocf:heartbeat:IPaddr2 ip=192.168
.1.200 cidr netmask=24 nic=enp0s3 op monitor interval=30s
root@cluster1:/var/log/pacemaker# pcs status
Cluster name: cluster_tfm
Status of pacemakerd: 'Pacemaker is running' (last updated 2024-08-20 00:28:12 +02:00)
Cluster Summary:
 * Stack: corosync
 * Current DC: standby2 (version 2.1.5-a3f44794f94) - partition with quorum
 * Last updated: Tue Aug 20 00:28:12 2024
 * Last change: Tue Aug 20 00:28:09 2024 by root via cibadmin on cluster1
 * 3 nodes configured
 * 1 resource instance configured

Node List:
 * Online: [ cluster1 standby1 standby2 ]

Full List of Resources:
 * IPVirtual_1 (ocf:heartbeat:IPaddr2):      Started cluster1

Daemon Status:
 corosync: active/enabled
 pacemaker: active/enabled
 pcsd: active/enabled
root@cluster1:/var/log/pacemaker# █

```

Figura 49 - Creación de recursos en el clúster

```

root@cluster1:/var/log/pacemaker# pcs resource create IPVirtual_2 ocf:heartbeat:IPaddr2 ip=10.0.10
0.200 cidr netmask=24 nic=enp0s8 op monitor interval=30s
root@cluster1:/var/log/pacemaker# pcs status
Cluster name: cluster_tfm
Status of pacemakerd: 'Pacemaker is running' (last updated 2024-08-20 00:30:00 +02:00)
Cluster Summary:
 * Stack: corosync
 * Current DC: standby2 (version 2.1.5-a3f44794f94) - partition with quorum
 * Last updated: Tue Aug 20 00:30:01 2024
 * Last change: Tue Aug 20 00:29:56 2024 by root via cibadmin on cluster1
 * 3 nodes configured
 * 2 resource instances configured

Node List:
 * Online: [ cluster1 standby1 standby2 ]

Full List of Resources:
 * IPVirtual_1 (ocf:heartbeat:IPaddr2):      Started cluster1
 * IPVirtual_2 (ocf:heartbeat:IPaddr2):      Started cluster1

Daemon Status:
 corosync: active/enabled
 pacemaker: active/enabled
 pcsd: active/enabled
root@cluster1:/var/log/pacemaker# █

```

Figura 50 - Creación de recurso IP en el clúster

```

root@cluster1:/var/log/pacemaker# pcs resource status
 * IPVirtual_1 (ocf:heartbeat:IPaddr2):      Started cluster1
 * IPVirtual_2 (ocf:heartbeat:IPaddr2):      Started cluster1
root@cluster1:/var/log/pacemaker# █

```

Figura 51 - Verificar estado de los recursos del clúster

Utilizando la herramienta **crm**, podemos editar también directamente la configuración de pacemaker y definir otros recursos. **crm** proporciona una función de "edición" que invoca al editor (predeterminado "vi") con el **cib** actual y permite la modificación de cualquier valor dentro del cib (Cluster Information Base) y, una vez guardado, aplica directamente la configuración del clúster (live). El contenido de la CIB se mantiene

automàticamente sincronizado en todo el cluster, de hecho, no se recomienda editar el archivo `cib.xml` directamente; en su lugar utilizar la interfaz `pcs`, `pcsd` o `crm`.

La lnea de comandos que se utiliza es la siguiente:

### crm configure edit

Con esta herramienta definiremos un grupo para las IPs virtuales previamente creadas, y, ademàs otras configuraciones relacionadas al location, stonith, entre otras.

```
node 1: cluster1
node 2: standby1
node 3: standby2
primitive IPVirtual_1 IPAddr2 \
  params cidr_netmask=24 ip=192.168.1.200 nic=enp0s3 \
  op monitor interval=30s \
  op start interval=0s timeout=20s \
  op stop interval=0s timeout=20s
primitive IPVirtual_2 IPAddr2 \
  params cidr_netmask=24 ip=10.0.100.200 nic=enp0s8 \
  op monitor interval=30s \
  op start interval=0s timeout=20s \
  op stop interval=0s timeout=20s
primitive st-null stonith:null \
  params hostlist="cluster1 standby1 standby2"
group grupo1 IPVirtual_1 IPVirtual_2
clone fencing st-null
order orden1 Mandatory: IPVirtual_1:start IPVirtual_2:start
location pref-cluster1 grupo1 50: cluster1
property cib-bootstrap-options: \
  have-watchdog=false \
  dc-version=2.1.5-a3f44794f94 \
  cluster-infrastructure=corosync \
  cluster-name=debian \
  stonith-enabled=false \
  no-quorum-policy=ignore
~/
~/
~/
"/tmp/tmphiqyk7or.pcmk" 26 lines, 852 bytes
```

Figura 52 - Uso de Herramienta `crm configure`

Nótese que se resalta en amarillo la definici3n de los recursos de IPs que se realiz3 con la ayuda de la herramienta `pcs` <sup>(13)</sup> (Documentation, 2024), es evidencia de que todo se guarda en la base CIB.

El comando **group**, nos permite agrupar recursos que necesitamos se localicen siempre juntos, en nuestra configuraci3n las direcciones IPs virtuales deben estar juntas cuando se transfieren entre nodos disponibles en el clúster. El comando **location**, define en qu3 nodos se puede ejecutar un recurso, no se puede ejecutar o se prefiere ejecutarlo, en nuestro caso preferimos que se ejecute en el nodo máster "cluster1" el recurso grupo1, que tiene las IPs virtuales.

Escapa de nuestro laboratorio la definici3n de un método de fencing, por eso el recurso **stonith** està configurado con el valor **false** y **stonith:null**, en la vida real si disponemos



de algún método para aplicar fencing, por ejemplo una PDU, podemos definir un script que se ejecute para enviar señales de apagado o reinicio de los equipos. El comando **clone**, se utilizó para probar iniciar el recurso de fencing en todos los nodos del clúster, por tanto, no aplica por ahora. Hay distribuciones de Linux que no soportan la definición de un clúster sin stonith, por tanto, antes de configurar siempre se recomienda revisar las guías de configuración.

Finalmente, con la opción **no-quorum-policy=ignore**, le indicamos que ignore el hecho de que no exista quorum, por lo que seguirá ofreciendo sus recursos aún con un solo nodo activo en el clúster, pues deseamos alta disponibilidad y, por tanto, no tiene sentido que se deje de ofrecer los servicios por problemas en uno de los nodos.

Luego de hacer todas las configuraciones del cluster y sus recursos, podemos hacer comprobaciones con el comando "**pcs config**" o "**crm configure show**", y tenemos las siguientes salidas:

```

root@cluster1:/var/lib/pcs# crm configure show
node 1: cluster1
node 2: standby1
node 3: standby2
primitive IPVirtual_1 IPAddr2 \
    params cidr_netmask=24 ip=192.168.1.200 nic=enp0s3 \
    op monitor interval=30s \
    op start interval=0s timeout=20s \
    op stop interval=0s timeout=20s
primitive IPVirtual_2 IPAddr2 \
    params cidr_netmask=24 ip=10.0.100.200 nic=enp0s8 \
    op monitor interval=30s \
    op start interval=0s timeout=20s \
    op stop interval=0s timeout=20s
primitive st-null stonith:null \
    params hostlist="cluster1 standby1 standby2"
group grupo1 IPVirtual_1 IPVirtual_2
clone fencing st-null
order orden1 Mandatory: IPVirtual_1:start IPVirtual_2:start
location pref-cluster1 grupo1 50: cluster1
property cib-bootstrap-options: \
    have-watchdog=false \
    dc-version=2.1.5-a3f44794f94 \
    cluster-infrastructure=corosync \
    cluster-name=debian \
    stonith-enabled=false \
    no-quorum-policy=ignore
root@cluster1:/var/lib/pcs#

```

Figura 53 - Verificar configuración del clúster herramienta crm

```
root@cluster1:/var/lib/pcs# pcs config
Cluster Name: cluster_tfm
Corosync Nodes:
 cluster1 standby1 standby2
Pacemaker Nodes:
 cluster1 standby1 standby2

Resources:
Group: grup01
Resource: IPVirtual_1 (class=ocf provider=heartbeat type=IPAddr2)
Attributes: IPVirtual_1-instance_attributes
 cidr_netmask=24
 ip=192.168.1.200
 nic=enp0s3
Operations:
 monitor: IPVirtual_1-monitor-interval-30s
 interval=30s
 start: IPVirtual_1-start-interval-0s
 interval=0s
 timeout=20s
 stop: IPVirtual_1-stop-interval-0s
 interval=0s
 timeout=20s
Resource: IPVirtual_2 (class=ocf provider=heartbeat type=IPAddr2)
Attributes: IPVirtual_2-instance_attributes
 cidr_netmask=24
 ip=10.0.100.200
 nic=enp0s8
Operations:
 monitor: IPVirtual_2-monitor-interval-30s
 interval=30s
 start: IPVirtual_2-start-interval-0s
 interval=0s
 timeout=20s
 stop: IPVirtual_2-stop-interval-0s
 interval=0s
 timeout=20s
Clone: fencing
Resource: st-null (class=stonith type=null)
Attributes: st-null-instance_attributes
 hostlist="cluster1 standby1 standby2"

Stonith Devices:
Resource: st-null (class=stonith type=null)
Attributes: st-null-instance_attributes
 hostlist="cluster1 standby1 standby2"
Fencing Levels:

Location Constraints:
Resource: grup01
Enabled on:
Node: cluster1 (score:50) (id:pref-cluster1)
Ordering Constraints:
 start IPVirtual_1 then start IPVirtual_2 (kind:Mandatory) (id:orden1)
Colocation Constraints:
Ticket Constraints:

Alerts:
No alerts defined

Resources Defaults:
No defaults set
Operations Defaults:
No defaults set

Cluster Properties:
cluster-infrastructure: corosync
cluster-name: debian
dc-version: 2.1.5-a3f44794f94
have-watchdog: false
no-quorum-policy: ignore
stonith-enabled: false

Tags:
No tags defined

Quorum:
Options:
root@cluster1:/var/lib/pcs# █
```

Figura 54 - Verificar configuración del clúster herramienta pcs

```

root@cluster1:/var/lib/pcsd# pcs status
Cluster name: cluster_tfm
Status of pacemaker: 'Pacemaker is running' (last updated 2024-09-03 20:43:57 +02:00)
Cluster Summary:
 * Stack: corosync
 * Current DC: cluster1 (version 2.1.5-a3f44794f94) - partition with quorum
 * Last updated: Tue Sep 3 20:43:58 2024
 * Last change: Mon Sep 2 20:28:27 2024 by root via cibadmin on cluster1
 * 3 nodes configured
 * 5 resource instances configured

Node List:
 * Online: [ cluster1 standby1 standby2 ]

Full List of Resources:
 * Resource Group: grup01:
 * IPVirtual_1 (ocf:heartbeat:IPaddr2): Started cluster1
 * IPVirtual_2 (ocf:heartbeat:IPaddr2): Started cluster1
 * Clone Set: Fencing [st-null]:
 * Started: [ cluster1 standby1 standby2 ]

Daemon Status:
 corosync: active/enabled
 pacemaker: active/enabled
 pacemaker_remote: inactive/enabled
 pcsd: active/enabled
root@cluster1:/var/lib/pcsd#

```

Figura 55 - Verificar estado del clúster herramienta pcs

En el capítulo 4, se mostrará la funcionalidad del clúster con corosync+pacemaker y HAProxy como repartidor de carga.

### 3.4.3 Instalación de nodos Standby

Dentro del clúster de alta disponibilidad se disponen de dos nodos standby, en modo activo-pasivo-pasivo. Ambos servidores standby son similares al nodo máster en su infraestructura, cada uno dispone de 3 tarjetas de red, una para la red externa y dos para la red interna, una de las cuales es exclusiva para la comunicación con los 2 servidores del clúster HA.

La configuración de ambos equipos es muy similar en servicios en ejecución y sus archivos de configuración, modificando las prioridades con respecto al nodo máster.

En las siguientes tablas se detalla la configuración de red de cada uno de los equipos:

Aspecto	Valor
Hostname	standby1
Tarjeta	enp0s3 (externa)
Dirección IP	192.168.1.102 (estática)
Máscara	255.255.255.0
Gateway	192.168.1.1
DNS	externos

<b>Tarjeta</b>	enp0s8 (interna)
<b>Dirección IP</b>	10.0.100.50 (estática)
<b>Máscara</b>	255.255.255.0
<b>Tarjeta</b>	enp0s9 (interna heartbeat)
<b>Dirección IP</b>	10.0.50.2 (estática)
<b>Máscara</b>	255.255.255.0

Tabla 4 - Configuración de red Nodo Standby1

Aspecto	Valor
<b>Hostname</b>	standby2
<b>Tarjeta</b>	enp0s3 (externa)
<b>Dirección IP</b>	192.168.1.103 (estática)
<b>Máscara</b>	255.255.255.0
<b>Gateway</b>	192.168.1.1
<b>DNS</b>	externos
<b>Tarjeta</b>	enp0s8 (interna)
<b>Dirección IP</b>	10.0.100.51 (estática)
<b>Máscara</b>	255.255.255.0
<b>Tarjeta</b>	enp0s9 (interna heartbeat)
<b>Dirección IP</b>	10.0.50.3 (estática)
<b>Máscara</b>	255.255.255.0

Tabla 5 - Configuración de red Nodo Standby2

### Reparto de carga con alta disponibilidad (Keepalived + HAProxy)

En la sección previa, se realizó la configuración del distribuidor de carga sobre el nodo maestro. Ahora mostraremos la configuración realizada sobre los nodos adicionales (standby) que asumirán la función de distribuidor en caso de que el nodo maestro falle.

Para ello, recordemos que definimos dos direcciones IP's virtuales (VIP), una por cada interfaz de red (interna y externa), de manera que el nodo que esté activo será el que las tendrá configuradas. (Ver Tabla 3 – Direcciones IP's del Clúster).

Dado que ya realizamos la configuración del nodo maestro, ahora mostraremos los cambios de configuración que se realizaron por parte de los nodos adicionales, tanto en el servicio de HAProxy como en la parte de keepalived.

### **Modificaci3n de la configuraci3n de HAProxy.**

El archivo de configuraci3n de HAProxy es id3ntico al del nodo M3ster. El 3nico cambio que realizaremos nos permitir3 que, al visualizar las estadísticas de funcionamiento, podamos diferenciar cu3l de los tres nodos directores est3 ofreciendo el servicio:

```
listen stats
  bind 0.0.0.0:7000
  mode http
  stats show-desc Nodo Standby1
  stats refresh 5s
  stats uri /haproxy?stats
  stats realm HAProxy Statistics
  stats auth admin:password
  stats admin if TRUE
```

Figura 56 - Modificaci3n HAProxy Nodo Standby1

```
listen stats
  bind 0.0.0.0:7000
  mode http
  stats show-desc Nodo Standby2
  stats refresh 5s
  stats uri /haproxy?stats
  stats realm HAProxy Statistics
  stats auth admin:password
  stats admin if TRUE
```

Figura 57 - Modificaci3n HAProxy Nodo Standby2

### **Modificaci3n de la configuraci3n de keepalived.**

En el archivo de configuraci3n de keepalived, */etc/keepalived/keepalived.conf*, editaremos la configuraci3n de *VRRP (virtual router redundancy protocol)* para indicar que el nodo Standby1/Standby2 actúan de *backup*, y que, en caso de estar los tres nodos directores activos, tiene m3s prioridad el nodo M3ster, y entre los dos nodos standby tiene mayor prioridad el nodo Standby1. Tambi3n indicaremos que el servicio web se ofrece en la VIP.

Por otra parte, y con el fin de ańadir un sistema exclusivo que detecte cuando existen nodos en el cl3ster que no est3n disponibles, se ańade una tercera interfaz por donde los 3 nodos se comunicarán de forma continua, con esta implementaci3n de heartbeat tratamos de prevenir un split-brain cuando se pierde la comunicaci3n entre nodos y cada uno puede tomar decisiones por su cuenta que afecten al cl3ster.

```
GNU nano 7.2 /etc/keepalived/keepalived.conf
vrrp_instance VI_1 {
    state BACKUP
    interface enp0s3
    virtual_router_id 51
    priority 100
    authentication {
        auth_type PASS
        auth_pass $ clustercac
    }
    virtual_ipaddress {
        192.168.1.200
    }
}

vrrp_instance VI_2 {
    state BACKUP
    interface enp0s8
    virtual_router_id 52
    priority 100
    authentication {
        auth_type PASS
        auth_pass $ clustercac
    }
    virtual_ipaddress {
        10.0.100.200
    }
}

vrrp_instance VI_3 {
    state BACKUP
    interface enp0s9
    virtual_router_id 53
    priority 100
    authentication {
        auth_type PASS
        auth_pass $ clustercac
    }
}
```

Figura 58 - Modificaci3n Keepalived Nodo Standby1

```
GNU nano 7.2 /etc/keepalived/keepalived.conf
vrrp_instance VI_1 {
    state BACKUP
    interface enp0s3
    virtual_router_id 51
    priority 90
    authentication {
        auth_type PASS
        auth_pass $ clustercac
    }
    virtual_ipaddress {
        192.168.1.200
    }
}

vrrp_instance VI_2 {
    state BACKUP
    interface enp0s8
    virtual_router_id 52
    priority 90
    authentication {
        auth_type PASS
        auth_pass $ clustercac
    }
    virtual_ipaddress {
        10.0.100.200
    }
}

vrrp_instance VI_3 {
    state BACKUP
    interface enp0s9
    virtual_router_id 53
    priority 90
    authentication {
        auth_type PASS
        auth_pass $ clustercac
    }
}
```

Figura 59 - Modificaci3n Keepalived Nodo Standby2

### 3.4.4 Instalaci3n de sistema de almacenamiento GlusterFS

Para configurar el sistema de ficheros GlusterFS usaremos 3 nuevos nodos **gluster1**, **gluster2** y **gluster3**, con sistema operativo Debian 12 y 2 discos de 500 MB, un disco se usar3 para levantar un sistema distribuido y el otro estar3 disponible para un sistema replicado. Ambos vol3menes estar3n disponibles para todos los nodos del cl3ster con alta disponibilidad y reparto de carga.

Luego de validar que el sistema operativo detecta los nuevos discos y crear una 3nica partici3n primaria en cada disco, procedemos a formatear y utilizar el sistema de archivos XFS. Los puntos de montaje para cada uno de los nuevos dispositivos ser3n:

**/export/brick1**

**/export/brick2**

Cada nodo ofrecerá dos bricks, además crearemos un directorio “data”, que es donde directamente montaremos los volúmenes de gluster para compartir y almacenar información. Adicional modificamos el archivo /etc/fstab, de cada nodo, para que cada vez que se inicie el sistema monte las particiones.

```

TFMGluster1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 7.2 /etc/fstab *
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# systemd generates mount units based on this file, see systemd.mount(5).
# Please run 'systemctl daemon-reload' after making changes here.
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda1 during installation
UUID=19e10c55-a781-4da1-bbf1-d32f0ea48174 / ext4 errors=remount-ro 0 1
# swap was on /dev/sda3 during installation
UUID=a77cce01-0bd1-4595-9610-8914ec30b102 none swap sw 0 0
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
/dev/sdb1 /export/brick1 xfs defaults 0 0
/dev/sdc1 /export/brick2 xfs defaults 0 0
  
```

Figura 60 - Fichero /etc/fstab nodos gluster

### Configuración de *GlusterFS*

Luego de instalar los paquetes necesarios para ofrecer el servicio *glusterFS*, iniciar y habilitar el servicio, configuraremos el grupo de nodos:

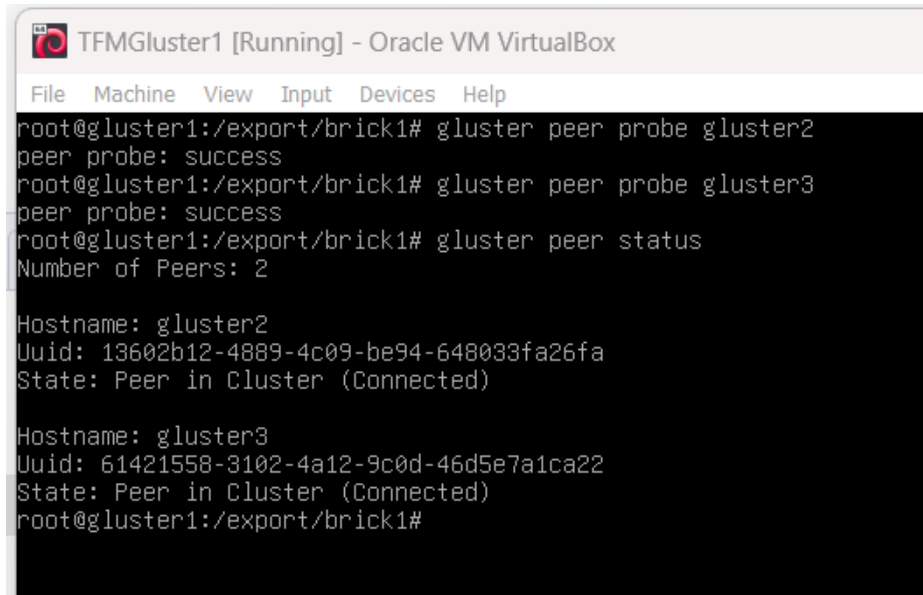
```

root@gluster2:~# systemctl status glusterd
* glusterd.service - GlusterFS, a clustered file-system server
   Loaded: loaded (/lib/systemd/system/glusterd.service; disabled; preset: enabled)
   Active: inactive (dead)
     Docs: man:glusterd(8)
root@gluster2:~# systemctl restart glusterd
root@gluster2:~# systemctl status glusterd
* glusterd.service - GlusterFS, a clustered file-system server
   Loaded: loaded (/lib/systemd/system/glusterd.service; disabled; preset: enabled)
   Active: active (running) since Tue 2024-08-06 02:09:36 CEST; 1s ago
     Docs: man:glusterd(8)
  Process: 651 ExecStart=/usr/sbin/glusterd -p /var/run/glusterd.pid --log-level $LOG_LEVEL $GLUS
 Main PID: 652 (glusterd)
    Tasks: 9 (limit: 1099)
   Memory: 26.2M
      CPU: 1.524s
   CGroup: /system.slice/glusterd.service
           └─652 /usr/sbin/glusterd -p /var/run/glusterd.pid --log-level INFO

Aug 06 02:09:34 gluster2 systemd[1]: Starting glusterd.service - GlusterFS, a clustered file-system
Aug 06 02:09:36 gluster2 systemd[1]: Started glusterd.service - GlusterFS, a clustered file-system
root@gluster2:~# systemctl enable glusterd
Created symlink /etc/systemd/system/multi-user.target.wants/glusterd.service → /lib/systemd/system/g
lusterd.service.
root@gluster2:~#
  
```

Figura 61 - Estado de servicio glusterd





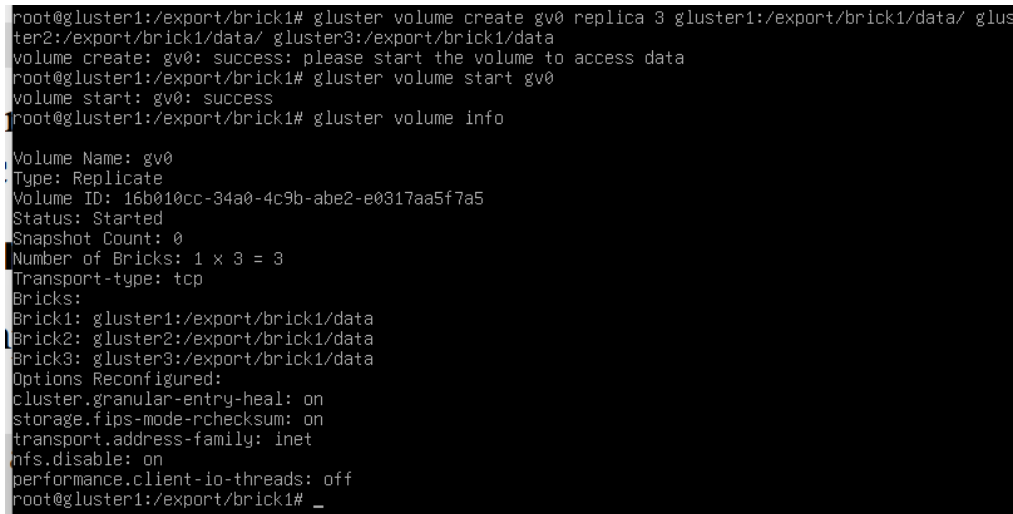
```
TFMGluster1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@gluster1:/export/brick1# gluster peer probe gluster2
peer probe: success
root@gluster1:/export/brick1# gluster peer probe gluster3
peer probe: success
root@gluster1:/export/brick1# gluster peer status
Number of Peers: 2

Hostname: gluster2
Uuid: 13602b12-4889-4c09-be94-648033fa26fa
State: Peer in Cluster (Connected)

Hostname: gluster3
Uuid: 61421558-3102-4a12-9c0d-46d5e7a1ca22
State: Peer in Cluster (Connected)
root@gluster1:/export/brick1#
```

Figura 62 - Creaci3n y estado de grupo gluster

Posterior, procedemos a crear cada uno de los volúmenes: replicado y distribuido. Los iniciamos y lucen así:



```
root@gluster1:/export/brick1# gluster volume create gv0 replica 3 gluster1:/export/brick1/data/ gluster2:/export/brick1/data/ gluster3:/export/brick1/data
volume create: gv0: success: please start the volume to access data
root@gluster1:/export/brick1# gluster volume start gv0
volume start: gv0: success
root@gluster1:/export/brick1# gluster volume info

Volume Name: gv0
Type: Replicate
Volume ID: 16b010cc-34a0-4c9b-abe2-e0317aa5f7a5
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x 3 = 3
Transport-type: tcp
Bricks:
Brick1: gluster1:/export/brick1/data
Brick2: gluster2:/export/brick1/data
Brick3: gluster3:/export/brick1/data
Options Reconfigured:
cluster.granular-entry-heal: on
storage.fips-mode-rchecksum: on
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
root@gluster1:/export/brick1# _
```

Figura 63 - Creaci3n e Inicio de Volumen Replicado

```

volume start: gv1: success
root@gluster1:/export/brick1# gluster volume info

Volume Name: gv0
Type: Replicate
Volume ID: 16b010cc-34a0-4c9b-abe2-e0317aa5f7a5
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x 3 = 3
Transport-type: tcp
Bricks:
Brick1: gluster1:/export/brick1/data
Brick2: gluster2:/export/brick1/data
Brick3: gluster3:/export/brick1/data
Options Reconfigured:
cluster.granular-entry-heal: on
storage.fips-mode-rchecksum: on
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off

Volume Name: gv1
Type: Distribute
Volume ID: 4db6a1ab-2fa6-488e-9b2b-023f6c9e6617
Status: Started
Snapshot Count: 0
Number of Bricks: 3
Transport-type: tcp
Bricks:
Brick1: gluster1:/export/brick2/data
Brick2: gluster2:/export/brick2/data
Brick3: gluster3:/export/brick2/data
Options Reconfigured:
storage.fips-mode-rchecksum: on
transport.address-family: inet
nfs.disable: on
root@gluster1:/export/brick1# _

```

Figura 64 - Información de los volúmenes

En la siguiente tabla se muestra un resumen de los volúmenes creados:

Aspecto	Valor
<b>Nombre Volumen</b>	<b>gv0</b>
<b>Tipo</b>	Replicado
<b>Tamaño</b>	500 MB
<b>Numero de bricks</b>	3 gluster1:/export/brick1/data gluster2:/export/brick1/data gluster3:/export/brick1/data
<b>Tipo de Transporte</b>	TCP
<b>Nombre Volumen</b>	<b>gv1</b>
<b>Tipo</b>	Distribuido
<b>Tamaño</b>	500 MB
<b>Numero de bricks</b>	3 gluster1:/export/brick2/data gluster2:/export/brick2/data gluster3:/export/brick2/data
<b>Tipo de Transporte</b>	TCP

Tabla 6 - Información de los volúmenes

Ya del lado de los clientes, resta instalar el paquete de glusterfs-client, crear los puntos de montaje y montar los volúmenes replicado y distribuido para su uso. Este proceso se debe realizar en cada uno de los nodos del clúster donde se utilizará dicho sistema de almacenamiento.

```
root@cluster1:/mnt# mkdir -p /gluster/gfs-r
root@cluster1:/mnt# mkdir -p /gluster/gfs-d
root@cluster1:/mnt# mount -t glusterfs gluster1:gv0 /gluster/gfs-r
root@cluster1:/mnt# mount -t glusterfs gluster1:gv1 /gluster/gfs-d
root@cluster1:/mnt#
```

Figura 65 - Configuración de Gluster-Client

De esta manera, reconocemos que el directorio **/gluster/gfs-r** corresponde al volumen **gv0 replicado** y el directorio **/gluster/gfs-d** corresponde al volumen **gv1 distribuido**.

Se modifica el archivo **/etc/fstab**, para que se monten los volúmenes cada vez que se inicia el sistema:

```
GNU nano 7.2 /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# systemd generates mount units based on this file, see systemd.mount(5).
# Please run 'systemctl daemon-reload' after making changes here.
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda1 during installation
UUID=19e10c55-a781-4da1-bbf1-d32f0ea48174 / ext4 errors=remount-ro 0 1
# swap was on /dev/sda3 during installation
UUID=a77cce01-0bd1-4595-9610-8914ec30b102 none swap sw 0 0
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
nas:/ /nfs nfs auto,nofail,bg,rsize=8192,wsz=8192 0 0
gluster1:gv0 /gluster/gfs-r glusterfs defaults,_netdev 0 0
gluster1:gv1 /gluster/gfs-d glusterfs defaults,_netdev 0 0
```

Figura 66 - Montaje de volúmenes Gluster

### 3.4.5 Configuración de contenedores

Para este laboratorio, usaremos contenedores como una parte esencial de la infraestructura del clúster, aprovechando las ventajas que ofrecen en términos de flexibilidad, escalabilidad y eficiencia. En concreto, se configurarán cuatro contenedores por cada nodo que forma parte del clúster (cluster2, cluster3 y cluster4), permitiendo una distribución eficiente de la carga y una alta disponibilidad de los servicios.

Con esta configuración, dispondremos de un total de doce contenedores operativos, listos para recibir y gestionar las solicitudes de los clientes. Esta estrategia incrementa significativamente la resiliencia del sistema. Aún si un nodo “físico” falla, los

contenedores en los otros nodos seguirán operativos, garantizando la continuidad del servicio, manejando las cargas de trabajo, minimizando la interrupción del servicio y manteniendo la integridad y la disponibilidad del sistema.

La utilización de contenedores también facilita el despliegue y la actualización de aplicaciones, permitiendo una mayor agilidad en la gestión del entorno. Esto es crucial en un clúster de alta disponibilidad, donde la capacidad de responder rápidamente a cambios y fallos es esencial. Cada contenedor opera de manera aislada pero coordinada con los demás, lo que mejora la eficiencia en el uso de recursos y permite un equilibrio de carga más efectivo.

En la siguiente Figura, se puede apreciar la creación de un contenedor, para lo cual se utilizó la imagen **php:apache** y el directorio sobre el cual se monta el servicio de apache `/var/www/html` es el almacenamiento compartido NFS del servidor NAS. Para ello, montamos el directorio **/nfs/raid** del "sistema host" sobre el contenedor (-v option). Cabe resaltar, que uno de los contenedores tendrá su almacenamiento para el servicio Apache+PHP en el sistema replicado de GlusterFS (`/gluster/gfs-r/www`), sin embargo, el proceso de creación es el mismo.

```

root@cluster2:/nfs/raid# docker run -dit --name cluster21p -p 8081:80 -v /nfs/raid/www:/var/www/html
php:apache
Unable to find image 'php:apache' locally
apache: Pulling from library/php
f11c1adaa26e: Pull complete
91c1fd48de30: Pull complete
c3b3bda7c6d1: Pull complete
55a68eb681dd: Pull complete
85406f9afc7f: Pull complete
a7d29e357509: Pull complete
d497b137ced8: Pull complete
bbc68bc20094: Pull complete
7401f0ea9603: Pull complete
7cccaf419fb3: Pull complete
86825edd5f6e: Pull complete
b4e2c5a7b120: Pull complete
209a72e837ef: Pull complete
Digest: sha256:cb992558faa44e6ed0800740f8fd3dac5ba4ac26644c46f8dc53e97f70095838
Status: Downloaded newer image for php:apache
80ad8478bea29c81f16534d11f3e9fea2f625e02bea66ea30ea4a73c1b2d6147
[ 1965.681884] docker0: port 1(vethecaad5b) entered blocking state
[ 1965.682056] docker0: port 1(vethecaad5b) entered disabled state
[ 1965.682693] device vethecaad5b entered promiscuous mode
[ 1966.307024] eth0: renamed from veth9128221
[ 1966.316163] IPv6: ADDRCONF(NETDEV_CHANGE): vethecaad5b: link becomes ready
[ 1966.316310] docker0: port 1(vethecaad5b) entered blocking state
[ 1966.316418] docker0: port 1(vethecaad5b) entered forwarding state
[ 1966.316543] IPv6: ADDRCONF(NETDEV_CHANGE): docker0: link becomes ready
root@cluster2:/nfs/raid#

root@cluster2:/nfs/raid/www# docker run -dit --name cluster24p -p 8084:80 -v /gluster/gfs-r/www:/var
/www/html php:apache
124ae763e8e50dabc8c885ed323210c029a76906f98b013fad1dafd789a6cc21
[14134.507540] docker0: port 4(vethd9b3d5d) entered blocking state
[14134.507579] docker0: port 4(vethd9b3d5d) entered disabled state
[14134.507642] device vethd9b3d5d entered promiscuous mode
[14134.890133] eth0: renamed from vethff63253
[14134.899809] IPv6: ADDRCONF(NETDEV_CHANGE): vethd9b3d5d: link becomes ready
[14134.899845] docker0: port 4(vethd9b3d5d) entered blocking state
[14134.899856] docker0: port 4(vethd9b3d5d) entered forwarding state

```

Figura 67 - Creación de contenedor

Además, hacemos uso de la red bridge por defecto del sistema Docker, con lo que cada contenedor tendrá una dirección IP asignada y se realizará un mapeo de puertos para comunicación con el host, lo que permitirá la comunicación con el exterior en el puerto detallado. Cabe recalcar que la red bridge es muy adecuada para aislar contenedores de la conexión exterior y para organizar los servicios expuestos, lo que nos permite tener

seguridad en el ambiente. Sin embargo, no proporciona conexión entre contenedores de hosts diferentes.

En el caso mostrado en la figura 68, el contenedor "cluster21p" tiene mapeado el puerto 8081 del host a su puerto 80 del servicio apache local. En la figura 69 se muestran todos los contenedores creados en el nodo cluster2:

```
root@cluster2:/# docker ps -a
CONTAINER ID   IMAGE          COMMAND
NAMES
124ae763e8e5   php:apache    "docker-php-entrypoi..."
:8084->80/tcp,  :::8084->80/tcp  cluster24p
5dffa0efcc41   php:apache    "docker-php-entrypoi..."
:8083->80/tcp,  :::8083->80/tcp  cluster23p
4ed7e1aa52cb   php:apache    "docker-php-entrypoi..."
:8082->80/tcp,  :::8082->80/tcp  cluster22p
d0ad8478bea2   php:apache    "docker-php-entrypoi..."
:8081->80/tcp,  :::8081->80/tcp  cluster21p
```

Figura 68 - Listado de Contenedores Cluster2

Luego de crear los 4 contenedores, así luce el mapeo de puertos en el nodo host:

```
root@cluster2:/# netstat -atunp |grep 808*
tcp        0      0 0.0.0.0:8084          0.0.0.0:*           LISTEN     2974/docker-proxy
tcp        0      0 0.0.0.0:8082          0.0.0.0:*           LISTEN     1069/docker-proxy
tcp        0      0 0.0.0.0:8083          0.0.0.0:*           LISTEN     1175/docker-proxy
tcp        0      0 0.0.0.0:8081          0.0.0.0:*           LISTEN     1688/docker-proxy
tcp6       0      0 :::8084              :::*                 LISTEN     2979/docker-proxy
tcp6       0      0 :::8082              :::*                 LISTEN     1074/docker-proxy
tcp6       0      0 :::8083              :::*                 LISTEN     1180/docker-proxy
tcp6       0      0 :::8081              :::*                 LISTEN     1693/docker-proxy
root@cluster2:/# _
```

Figura 69 - Mapeo de puertos con host

En las siguientes figuras, se muestran los contenedores creados en los nodos cluster3 y cluster4:

```
root@cluster3:/etc/init.d# docker run -dit --name cluster23p -p 8083:80 -v /nfs/raid/www:/var/www/html php:apache
3f8fd25ef8a9f869146829f380fba6e8a09d77544c246a52e9b42eee37c9c6fc
[ 3523.149024] docker0: port 3(veth16bfc5b) entered blocking state
[ 3523.149304] docker0: port 3(veth16bfc5b) entered disabled state
[ 3523.149804] device veth16bfc5b entered promiscuous mode
[ 3523.546671] eth0: renamed from vetha57db45
[ 3523.553258] IPv6: ADDRCONF(NETDEV_CHANGE): veth16bfc5b: link becomes ready
[ 3523.553774] docker0: port 3(veth16bfc5b) entered blocking state
[ 3523.554097] docker0: port 3(veth16bfc5b) entered forwarding state
```

```

root@cluster3:~# docker run -dit --name cluster34p -p 8085:80 -v /gluster/gfs-r/www:/var/www/html php:apache
a175217a9e4faa37b5efc49fb46efa5b6194047c76f9d6f852e1831f2ed6032f
[181464.852687] docker0: port 4(veth3ac7181) entered blocking state
[181464.852942] docker0: port 4(veth3ac7181) entered disabled state
[181464.854545] device veth3ac7181 entered promiscuous mode
[181465.458818] eth0: renamed from veth6641306
[181465.469682] IPv6: ADDRCONF(NETDEV_CHANGE): veth3ac7181: link becomes ready
[181465.470272] docker0: port 4(veth3ac7181) entered blocking state
[181465.470810] docker0: port 4(veth3ac7181) entered forwarding state
root@cluster3:~# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
NAMES
a175217a9e4f   php:apache     "docker-php-entrypoi..." 6 hours ago   Up 6 hours   0.0.0.0
:8085->80/tcp, :::8085->80/tcp   cluster34p
f831b4ae2e0b   php:apache     "docker-php-entrypoi..." 6 weeks ago   Up 2 days    0.0.0.0
:8083->80/tcp, :::8083->80/tcp   cluster33p
4ce9eac20a9    php:apache     "docker-php-entrypoi..." 6 weeks ago   Up 2 days    0.0.0.0
:8082->80/tcp, :::8082->80/tcp   cluster32p
95a1abc3f55d   php:apache     "docker-php-entrypoi..." 6 weeks ago   Up 2 days    0.0.0.0
:8081->80/tcp, :::8081->80/tcp   cluster31p
ee8f51a0af09   hello-world    "/hello"                  6 weeks ago   Exited (0) 6 weeks ago
                               stoic_mirzakhani
root@cluster3:~#

```

Figura 70 - Listado de Contenedores Cluster3

```

PORTS          NAMES
b08b1b4366bf   php:apache     "docker-php-entrypoi..." About a minute ago   Up About a minute
0.0.0.0:8081->80/tcp, :::8081->80/tcp   cluster41p
e958d852cfe2   hello-world    "/hello"                  10 minutes ago      Exited (0) 10 minutes ago
                               vigilant_jennings
root@cluster4:~# docker run -dit --name cluster42p -p 8082:80 -v /nfs/raid/www:/var/www/html php:apache
b893b3340b1feb460c80269683baaa396bc952a74aac2620c7a262f9b7972f34
[ 1886.580445] docker0: port 2(veth3fd584e) entered blocking state
[ 1886.580759] docker0: port 2(veth3fd584e) entered disabled state
[ 1886.580971] device veth3fd584e entered promiscuous mode
[ 1886.879796] eth0: renamed from veth98a32af
[ 1886.892561] IPv6: ADDRCONF(NETDEV_CHANGE): veth3fd584e: link becomes ready
[ 1886.893207] docker0: port 2(veth3fd584e) entered blocking state
[ 1886.893508] docker0: port 2(veth3fd584e) entered forwarding state
root@cluster4:~# docker run -dit --name cluster43p -p 8083:80 -v /nfs/raid/www:/var/www/html php:apache
312d3e16e21eb046cc28aaa574091efd4cbc2d494071df58a202784d0dfb55cb9
[ 1898.234641] docker0: port 3(veth9f0a235) entered blocking state
[ 1898.234823] docker0: port 3(veth9f0a235) entered disabled state
[ 1898.235028] device veth9f0a235 entered promiscuous mode
[ 1898.517821] eth0: renamed from vethbda658a
[ 1898.529635] IPv6: ADDRCONF(NETDEV_CHANGE): veth9f0a235: link becomes ready
[ 1898.529987] docker0: port 3(veth9f0a235) entered blocking state
[ 1898.530290] docker0: port 3(veth9f0a235) entered forwarding state
root@cluster4:~# docker run -dit --name cluster44p -p 8086:80 -v /gluster/gfs-r/www:/var/www/html php:apache
11a21a115ffbd1d60d3a0c9694016237d132899f151823aa6517259a672c7a31
[181560.662744] docker0: port 4(vethade23d5) entered blocking state
[181560.662825] docker0: port 4(vethade23d5) entered disabled state
[181560.663961] device vethade23d5 entered promiscuous mode
[181561.911229] eth0: renamed from veth8bcace3
[181561.921008] IPv6: ADDRCONF(NETDEV_CHANGE): vethade23d5: link becomes ready
[181561.921043] docker0: port 4(vethade23d5) entered blocking state
[181561.921051] docker0: port 4(vethade23d5) entered forwarding state
root@cluster4:~# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
NAMES
11a21a115ffbd   php:apache     "docker-php-entrypoi..." 6 hours ago   Up 6 hours   0.0.0.0
:8086->80/tcp, :::8086->80/tcp   cluster44p
312d3e16e21e    php:apache     "docker-php-entrypoi..." 5 weeks ago   Up 2 days    0.0.0.0
:8083->80/tcp, :::8083->80/tcp   cluster43p
b893b3340b1f    php:apache     "docker-php-entrypoi..." 5 weeks ago   Up 2 days    0.0.0.0
:8082->80/tcp, :::8082->80/tcp   cluster42p
b08b1b4366bf    php:apache     "docker-php-entrypoi..." 5 weeks ago   Up 2 days    0.0.0.0
:8081->80/tcp, :::8081->80/tcp   cluster41p
e958d852cfe2    hello-world    "/hello"                  5 weeks ago   Exited (0) 5 weeks ago
                               vigilant_jennings
root@cluster4:~#

```

Figura 71 - Listado de Contenedores Cluster4

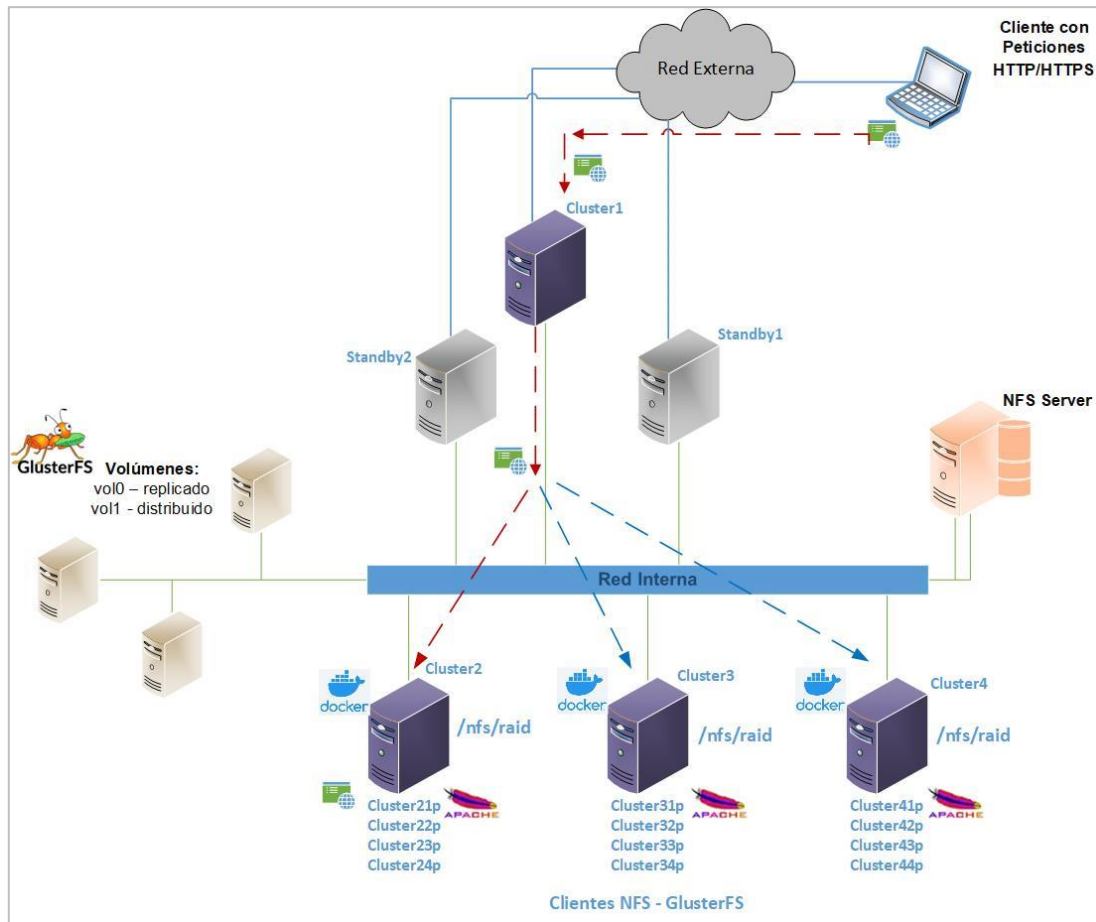


Figura 72 - Flujo de peticiones web server

La figura 72 muestra el flujo de una petición al servicio web, lo recibe el nodo activo del clúster HA, quien según considere, reparte la petición entre los nodos que forman parte del esquema de reparto de carga.

En la tabla adjunta se muestra el mapeo de puertos final entre los servidores reales y los contenedores:

Hostname	Puerto host	Hostname contenedor	Puerto contenedor
<b>Cluster2</b>	8081	Cluster21p	80
	8082	Cluster22p	80
	8083	Cluster23p	80
	8084	Cluster24p	80
<b>Cluster3</b>	8081	Cluster31p	80
	8082	Cluster32p	80
	8083	Cluster33p	80
	8085	Cluster34p	80

<b>Cluster4</b>	8081	Cluster41p	80
	8082	Cluster42p	80
	8083	Cluster43p	80
	8086	Cluster44p	80

Tabla 7 - Mapeo de puertos hosts y contenedores

## 4 PRUEBAS Y RESULTADOS

En este capítulo, nos adentraremos en las pruebas y resultados obtenidos al evaluar las prestaciones del clúster implementado. Aquí, mostraremos cómo cada componente del clúster, desde las tecnologías de alta disponibilidad como Keepalived y Corosync+Pacemaker, hasta las soluciones de reparto de carga como HAProxy, se comporta en distintos escenarios.

Para empezar, realizaremos pruebas exhaustivas de alta disponibilidad, verificando cómo el clúster maneja fallos y recuperaciones utilizando Keepalived y Corosync+Pacemaker. Posteriormente, evaluaremos el rendimiento del equilibrio de carga con HAProxy, asegurándonos de que las solicitudes se distribuyan de manera eficiente entre los nodos del clúster.

Además, exploraremos el funcionamiento y la eficiencia de las soluciones de almacenamiento implementadas. Esto incluye pruebas con GlusterFS para almacenamiento distribuido y replicado, NFS sobre RAID para almacenamiento compartido y Link Aggregation para mejorar el rendimiento de la red. También examinaremos cómo los contenedores gestionan las cargas de trabajo y su impacto en la disponibilidad y la respuesta del clúster.

Finalmente, utilizaremos la herramienta Apache Benchmark para medir el rendimiento global del clúster. Presentaremos indicadores clave como el número de peticiones por segundo y los tiempos de respuesta bajo diferentes condiciones de carga. Estos datos nos permitirán evaluar de manera objetiva la capacidad del clúster para manejar altas demandas y su eficiencia en términos de respuesta y estabilidad.

Este capítulo no solo demostrará la robustez y la resiliencia del clúster, sino que también proporcionará una visión clara de cómo se comporta en situaciones reales, ofreciendo una base sólida para futuras mejoras y optimizaciones



## 4.1 PRUEBAS DE FUNCIONALIDAD

En esta sección se mostrarán las siguientes funcionalidades del clúster:

- ▶ Alta disponibilidad con Corosync y Pacemaker
- ▶ Alta disponibilidad con KeepAlived
- ▶ Reparto de la carga, con HAProxy, cuando todos los nodos y sus contenedores están operativos
- ▶ Reparto de la carga, con HAProxy, cuando algunos contenedores están offline
- ▶ Reparto de la carga, con HAProxy, cuando un nodo, que alberga los 3 contenedores, está “caído”.
- ▶ Funcionamiento de acceso al recurso compartido con NFS cuando una de las tarjetas del link aggregation esta deshabilitada.
- ▶ Acceso al recurso compartido y replicado con GlusterFS, aun cuando uno de los nodos no se encuentra disponible
- ▶ Acceso al recurso compartido y distribuido con GlusterFS desde uno de los nodos del clúster
- ▶ Acceso a una página web almacenada en el recurso compartido y replicado con GlusterFS

### 4.1.1 Alta Disponibilidad con Corosync y Pacemaker

En primera instancia, vamos a mostrar como el sistema funciona cuando los tres nodos que forman el clúster HA están activos, esto es, el nodo maestro haciendo su tarea de repartir carga a los nodos con contenedores, mientras los otros dos nodos (standby1 y standby2) están esperando alguna acción que les alerte que deben entrar a operar.

```
root@cluster1:~# pcs cluster status
Cluster Status:
Status of pacemakerd: 'Pacemaker is running' (last updated 2024-09-04 00:45:05 +02:00)
Cluster Summary:
 * Stack: corosync
 * Current DC: standby2 (version 2.1.5-a3f44794f94) - partition with quorum
 * Last updated: Wed Sep  4 00:45:05 2024
 * Last change: Mon Sep  2 20:28:27 2024 by root via cibadmin on cluster1
 * 3 nodes configured
 * 5 resource instances configured
Node List:
 * Online: [ cluster1 standby1 standby2 ]
PCSD Status:
 standby2: Online
 standby1: Online
 cluster1: Online
root@cluster1:~#
```

Figura 73 - Estado del clúster HA

```

root@cluster1:~# pcs status
Cluster name: cluster_tfm
Status of pacemaker: 'Pacemaker is running' (last updated 2024-09-04 00:46:50 +02:00)
Cluster Summary:
 * Stack: corosync
 * Current DC: standby2 (version 2.1.5-a3f44794f94) - partition with quorum
 * Last updated: Wed Sep  4 00:46:50 2024
 * Last change: Mon Sep  2 20:28:27 2024 by root via cibadmin on cluster1
 * 3 nodes configured
 * 5 resource instances configured

Node List:
 * Online: [ cluster1 standby1 standby2 ]

Full List of Resources:
 * Resource Group: grupol1:
 * IPVirtual_1      (ocf:heartbeat:IPaddr2):      Started cluster1
 * IPVirtual_2      (ocf:heartbeat:IPaddr2):      Started cluster1
 * Clone Set: Fencing [st-null]:
 * Started: [ cluster1 standby1 standby2 ]

Daemon Status:
corosync: active/enabled
pacemaker: active/enabled
pacemaker_remote: inactive/enabled
pcsd: active/enabled

```

Figura 74 - Recursos del clúster en nodo máster

En las figuras 73 y 74, se puede observar que los tres nodos están activos y el recurso **Grupo1**, que incluye las **dos IPs virtuales**, está localizado en el nodo máster “cluster1”. Por tanto, al ejecutar el comando “**ip a**”, podemos observar que el nodo maestro tiene asignadas las IP virtuales.

```

root@cluster1:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 08:00:27:7d:37:7d brd ff:ff:ff:ff:ff:ff
   inet 192.168.1.101/24 brd 192.168.1.255 scope global enp0s3
       valid_lft forever preferred_lft forever
   inet 192.168.1.200/24 brd 192.168.1.255 scope global secondary enp0s3
       valid_lft forever preferred_lft forever
   inet6 fe80::a00:27ff:fe7d:377d/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 08:00:27:53:9f:cd brd ff:ff:ff:ff:ff:ff
   inet 10.0.100.1/24 brd 10.0.100.255 scope global enp0s8
       valid_lft forever preferred_lft forever
   inet 10.0.100.200/24 brd 10.0.100.255 scope global secondary enp0s8
       valid_lft forever preferred_lft forever
   inet6 fe80::a00:27ff:fe53:9fcd/64 scope link
       valid_lft forever preferred_lft forever
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 08:00:27:31:9c:c4 brd ff:ff:ff:ff:ff:ff
   inet 10.0.50.1/24 brd 10.0.50.255 scope global enp0s9
       valid_lft forever preferred_lft forever
   inet6 fe80::a00:27ff:fe31:9cc4/64 scope link

```

Figura 75 - Asignación de IPs Virtuales nodo máster

Adicional, ante el requerimiento de la página web, en los logs del servicio de **haproxy** se puede observar cual es el nodo que se encuentra haciendo el reparto de carga, para ello ejecutar el siguiente comando que envía el log de haproxy a un archivo que le indiquemos:

```
root@cluster1:/# journalctl -u haproxy > /root/haproxy.log
```

Figura 76 - Revisión logs de servicio HAProxy

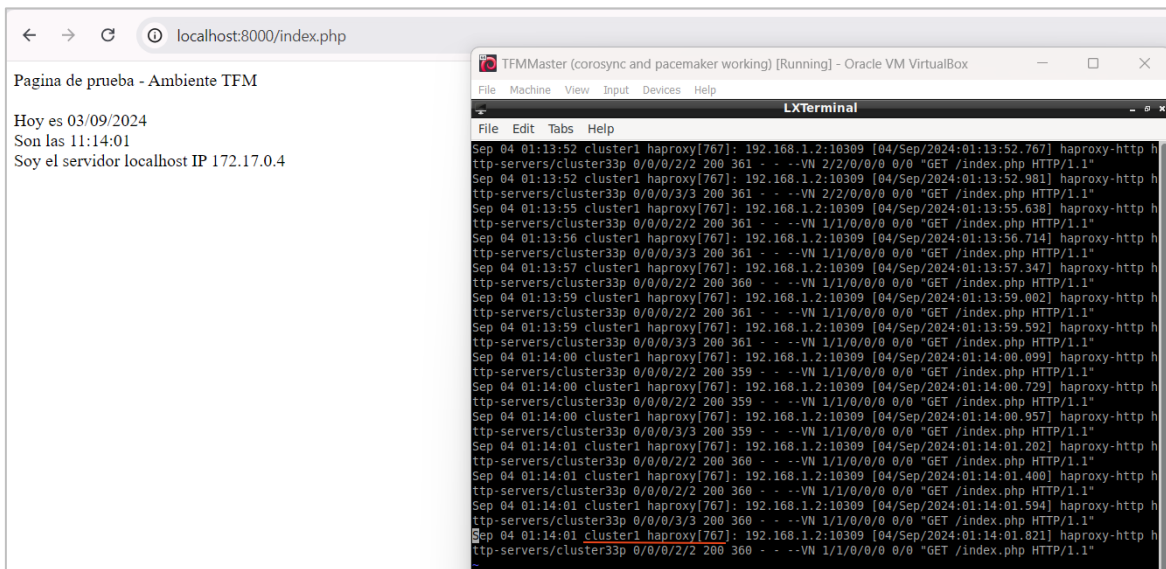


Figura 77 - Logs de servicio HAProxy - nodo máster activo

Simulamos la “caída” del nodo máster, desconectando (virtualmente) el cable conectado a la interfaz de red (a través de la configuración gráfica de red de la máquina virtual correspondiente).

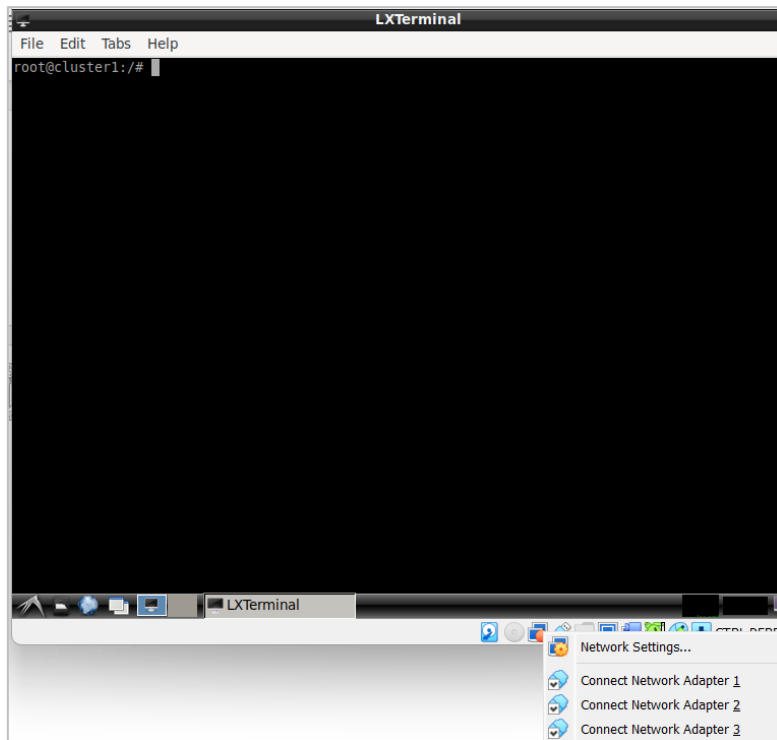


Figura 78 - Prueba de desconexión nodo máster - I

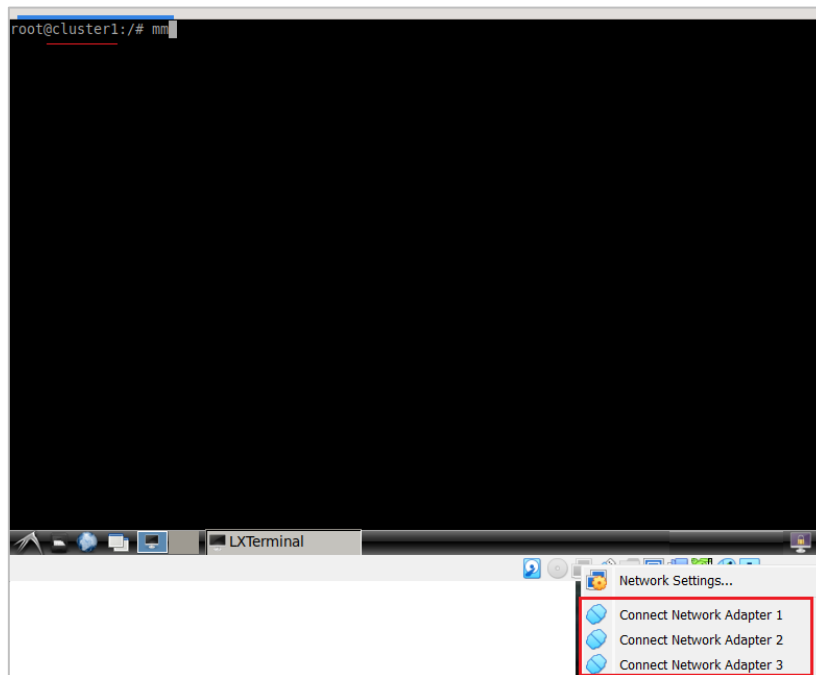


Figura 79 - Prueba de desconexión nodo máster - II

Entonces podemos observar el siguiente comportamiento:

- ▶ El recurso **Grupo1**, se transfiere o migra al nodo **standby1**

- ▶ Dado que el nodo máster luce offline, el reparto de la carga para el servicio web lo ejecuta el nodo standby1. Esto se puede evidenciar en los logs del servicio haproxy.

```

root@standby1:~# pcs status
Cluster name: cluster1
Status of pacemakerd: 'Pacemaker is running' (last updated 2024-09-04 01:33:47 +02:00)
Cluster Summary:
 * Stack: corosync
 * Current DC: standby2 (version 2.1.5-a3f44794f94) - partition with quorum
 * Last updated: Wed Sep 4 01:33:47 2024
 * Last change: Mon Sep 2 20:28:27 2024 by root via cibadmin on cluster1
 * 3 nodes configured
 * 5 resource instances configured

Node List:
 * Online: [ standby1 standby2 ]
 * OFFLINE: [ cluster1 ]

Full List of Resources:
 * Resource Group: grupol:
 * IPVirtual_1 (ocf:heartbeat:IPaddr2): Started standby1
 * IPVirtual_2 (ocf:heartbeat:IPaddr2): Started standby1
 * Clone Set: fencing [st-null]:
 * Started: [ standby1 standby2 ]
 * Stopped: [ cluster1 ]

Daemon Status:
 corosync: active/enabled
 pacemaker: active/enabled
 pcsd: active/enabled
root@standby1:~#

```

Figura 80 - Prueba nodo máster offline

```

root@standby1:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
   link/ether 08:00:27:9e:2a:06 brd ff:ff:ff:ff:ff:ff
   inet 192.168.1.102/24 brd 192.168.1.255 scope global enp0s3
       valid_lft forever preferred_lft forever
   inet 192.168.1.200/24 brd 192.168.1.255 scope global secondary enp0s3
       valid_lft forever preferred_lft forever
   inet6 fe80::a00:27ff:fe9e:2a06/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
   link/ether 08:00:27:2f:41:ad brd ff:ff:ff:ff:ff:ff
   inet 10.0.100.50/24 brd 10.0.100.255 scope global enp0s8
       valid_lft forever preferred_lft forever
   inet 10.0.100.200/24 brd 10.0.100.255 scope global secondary enp0s8
       valid_lft forever preferred_lft forever
   inet6 fe80::a00:27ff:fe2f:41ad/64 scope link
       valid_lft forever preferred_lft forever
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
   link/ether 08:00:27:ef:a6:72 brd ff:ff:ff:ff:ff:ff
   inet 10.0.50.2/24 brd 10.0.50.255 scope global enp0s9
       valid_lft forever preferred_lft forever

```

Figura 81 - Prueba nodo máster offline

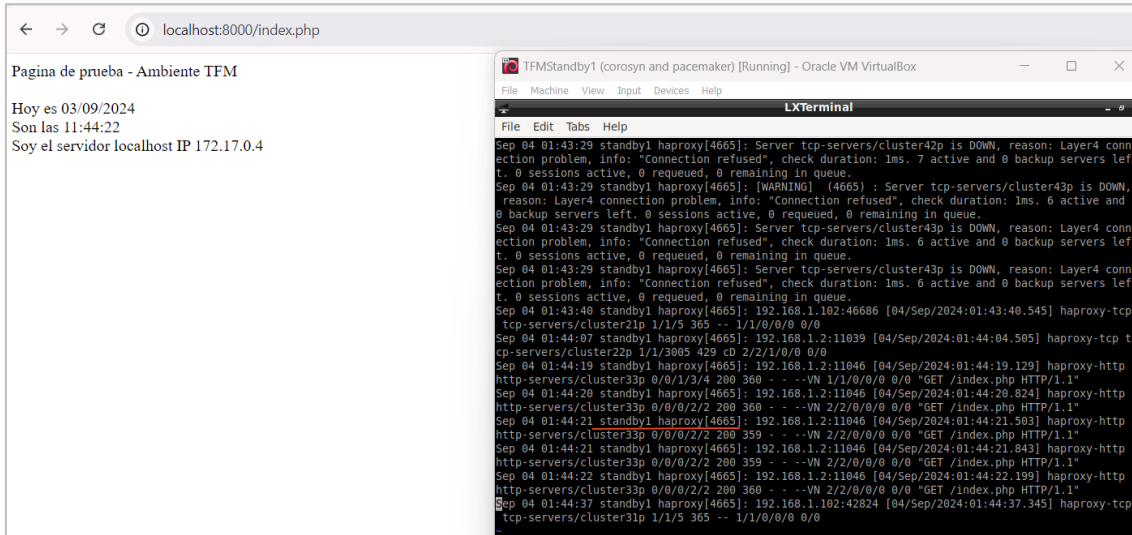


Figura 82 - Prueba nodo máster offline

Finalmente, si simulamos la caída del nodo standby1, queda operativo solo el nodo standby2 y eso se evidencia en que hereda las IPs virtuales y en los logs del servicio de haproxy se muestra el nodo que está realizando el reparto de carga.

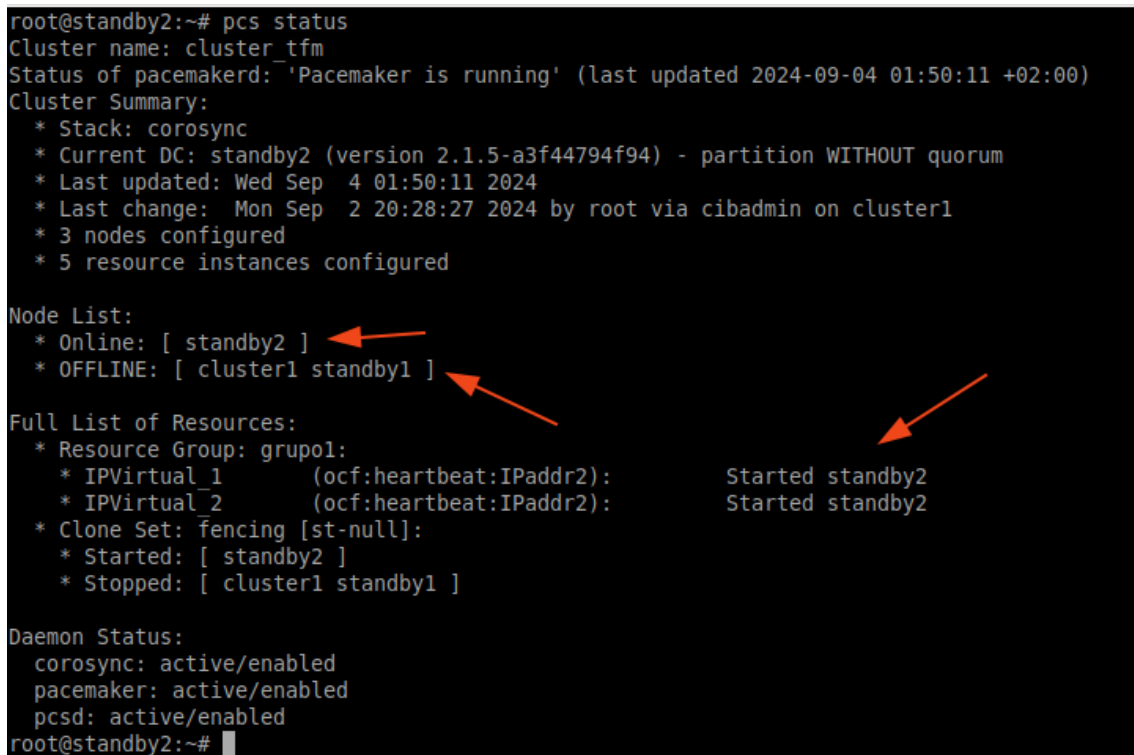


Figura 83 - Prueba nodo máster y standby1 offline

```

root@standby2:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:df:72:d8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.103/24 brd 192.168.1.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet 192.168.1.200/24 brd 192.168.1.255 scope global secondary enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fedf:72d8/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:fd:ad:9e brd ff:ff:ff:ff:ff:ff
    inet 10.0.100.51/24 brd 10.0.100.255 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet 10.0.100.200/24 brd 10.0.100.255 scope global secondary enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fed:ad9e/64 scope link
        valid_lft forever preferred_lft forever
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:70:fa:f3 brd ff:ff:ff:ff:ff:ff
    inet 10.0.50.3/24 brd 10.0.50.255 scope global enp0s9
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe70:faf3/64 scope link
    
```

Figura 84 - Prueba nodo máster y standby1 offline

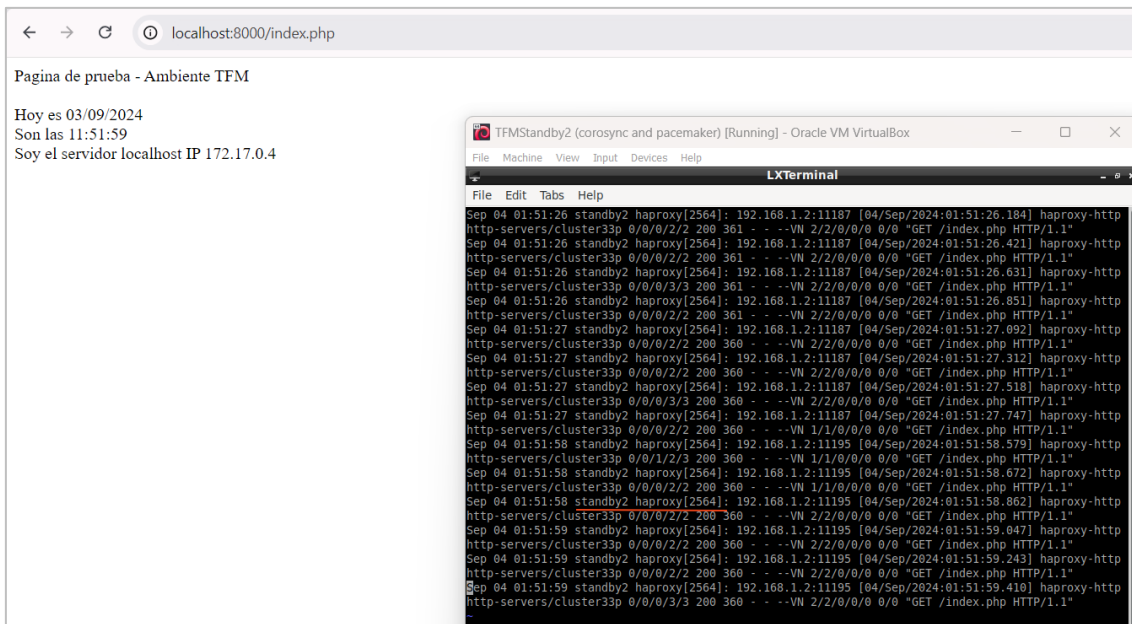


Figura 85 - Prueba nodo máster y standby1 offline

Según configuración del servicio de Corosync y Pacemaker, basta que un nodo esté operativo para continuar con el servicio web arriba y ejecutándose el proceso de reparto de carga.

Para finalizar esta prueba, al volver a estar online los nodos máster y standby1, los recursos de IPs virtuales retornan al nodo máster y el proceso de reparto de carga lo inicia el mismo nodo.

### 4.1.2 Alta Disponibilidad con Keepalived

Las pruebas de funcionalidad con el servicio de Keepalived son muy similares a las ya realizadas con el servicio de alta disponibilidad de Corosync más Pacemaker.

Por tal motivo, vamos a mostrar varias gráficas que permiten observar el comportamiento del clúster cuando todos los nodos están activos vs lo que sucede cuando alguno de los nodos está fuera de servicio.

El nodo cluster1, tiene el rol de nodo maestro, esto se puede comprobar leyendo los logs de keepalived. Para ello ejecutar el siguiente comando:

```
root@cluster1:/etc/keepalived# journalctl -u keepalived > /root/keepalived.log
```

Figura 86 - Comando revisión de logs Keepalived

En el contenido del log, para el servidor que tiene el rol maestro, se ven las siguientes líneas:

```
Sep 09 01:03:15 cluster1 Keepalived_vrrp[477109]: (VI_2) Entering MASTER STATE
Sep 09 01:03:15 cluster1 Keepalived_vrrp[477109]: VRRP_Group(VG1) Syncing instances to MASTER state
Sep 09 01:03:15 cluster1 Keepalived_vrrp[477109]: (VI_1) Entering MASTER STATE
Sep 09 01:03:15 cluster1 Keepalived_vrrp[477109]: (VI_3) Entering MASTER STATE
root@cluster1:/etc/keepalived#
```

Figura 87 - Keepalived Nodo Maestro

Mientras que para los servidores que son stand-by tiene líneas así:

```
File Edit Tabs Help
Sep 09 01:03:50 standby1 Keepalived_healthcheckers[8775]:
Sep 09 01:03:50 standby1 Keepalived_vrrp[8776]: (VI_1) Entering BACKUP STATE (init)
Sep 09 01:03:50 standby1 Keepalived_vrrp[8776]: (VI_2) Entering BACKUP STATE (init)
Sep 09 01:03:50 standby1 Keepalived_vrrp[8776]: (VI_3) Entering BACKUP STATE (init)
Sep 09 01:04:12 standby2 Keepalived_vrrp[5877]: (VI_1) Entering BACKUP STATE (init)
Sep 09 01:04:12 standby2 Keepalived_vrrp[5877]: (VI_2) Entering BACKUP STATE (init)
Sep 09 01:04:12 standby2 Keepalived_vrrp[5877]: (VI_3) Entering BACKUP STATE (init)
```

Figura 88 - Keepalived Nodos Stand-by



De esta manera, como primera prueba, cuando los tres nodos están disponibles, vamos a comprobar que el nodo maestro, que tiene mayor prioridad, maneja las dos direcciones IP virtuales (IPV), por tanto, en la siguiente figura mostramos dicha asignación:

```

inet6 ::1/128 scope host noprefixroute
    valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:7d:37:7d brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.101/24 brd 192.168.1.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet 192.168.1.200/32 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe7d:377d/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:53:9f:cd brd ff:ff:ff:ff:ff:ff
    inet 10.0.100.1/24 brd 10.0.100.255 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet 10.0.100.200/32 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe53:9fcd/64 scope link
        valid_lft forever preferred_lft forever
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:31:9c:c4 brd ff:ff:ff:ff:ff:ff
    inet 10.0.50.1/24 brd 10.0.50.255 scope global enp0s9
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe31:9cc4/64 scope link
        valid_lft forever preferred_lft forever
root@cluster1:/etc/keepalived#
    
```

Figura 89 - Keepalived Manejo de IPV

En el caso de que el servidor con el rol maestro, esto es el cluster1, esté fuera de línea, el grupo de IPV migrarán al siguiente nodo stand-by con mayor prioridad, esto es, standby1 (según la configuración en el archivo /etc/keepalived/keepalived.conf).

Para hacer esta prueba vamos a desconectar las interfaces de red del nodo cluster1 y comprobar que se migran las direcciones IPV y el rol maestro:

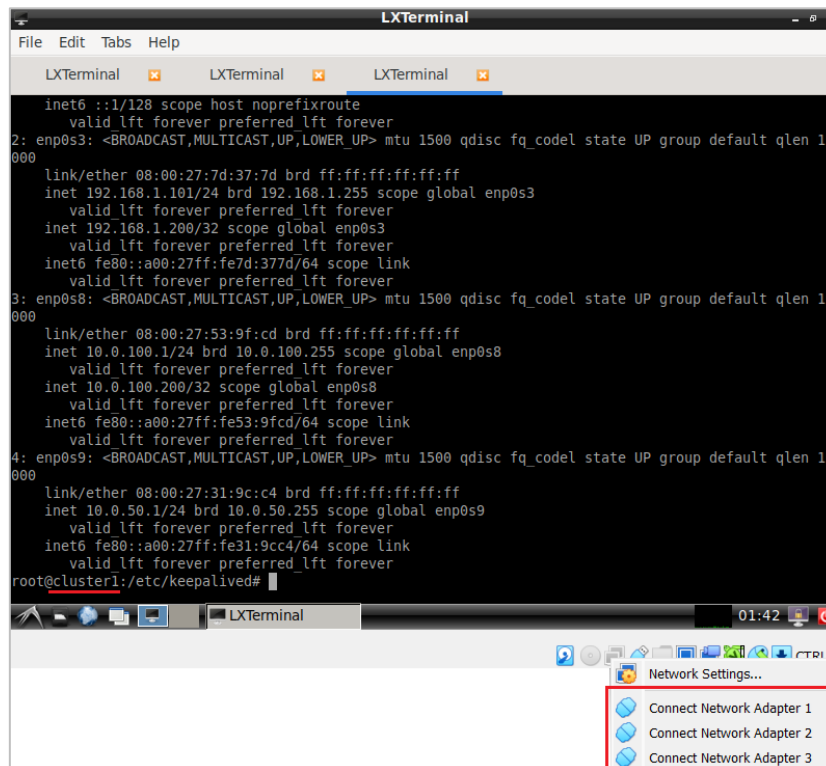


Figura 90 - Desconexión Nodo Maestro

```

LXTerminal
File Edit Tabs Help
inet 127.0.0.1/8 scope host lo
    valid lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
    valid lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
link/ether 08:00:27:9e:2a:06 brd ff:ff:ff:ff:ff:ff
inet 192.168.1.102/24 brd 192.168.1.255 scope global enp0s3
    valid lft forever preferred_lft forever
inet 192.168.1.200/32 scope global enp0s3
    valid lft forever preferred_lft forever
inet6 fe80::a00:27ff:fe9e:2a06/64 scope link
    valid lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
link/ether 08:00:27:2f:41:ad brd ff:ff:ff:ff:ff:ff
inet 10.0.100.50/24 brd 10.0.100.255 scope global enp0s8
    valid lft forever preferred_lft forever
inet 10.0.100.200/32 scope global enp0s8
    valid lft forever preferred_lft forever
inet6 fe80::a00:27ff:fe2f:41ad/64 scope link
    valid lft forever preferred_lft forever
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
link/ether 08:00:27:ef:a6:72 brd ff:ff:ff:ff:ff:ff
inet 10.0.50.2/24 brd 10.0.50.255 scope global enp0s9
    valid lft forever preferred_lft forever
inet6 fe80::a00:27ff:feef:a672/64 scope link
    valid lft forever preferred_lft forever
root@standby1:/etc/keepalived#
root@standby1:/etc/keepalived#
Sep 09 01:42:38 standby1 Keepalived_vrrp[8776]: (VI_1) Entering MASTER STATE
Sep 09 01:42:38 standby1 Keepalived_vrrp[8776]: VRRP_Group(VG1) Syncing instances to MASTER state
Sep 09 01:42:38 standby1 Keepalived_vrrp[8776]: (VI_2) Entering MASTER STATE
Sep 09 01:42:44 standby1 Keepalived_vrrp[8776]: (VI_3) Entering MASTER STATE
root@standby1:/etc/keepalived#

```

Figura 91 - Keepalived transferencia de rol e IPv6 - Standby1

Ahora, si el nodo standby1 sufre un percance y está fuera de línea, el que debe tomar su lugar y continuar con el rol de maestro es el nodo standby2:

```

File Edit Tabs Help
inet 127.0.0.1/8 scope host lo
    valid lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
    valid lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
link/ether 08:00:27:9e:2a:06 brd ff:ff:ff:ff:ff:ff
inet 192.168.1.102/24 brd 192.168.1.255 scope global enp0s3
    valid lft forever preferred_lft forever
inet 192.168.1.200/32 scope global enp0s3
    valid lft forever preferred_lft forever
inet6 fe80::a00:27ff:fe9e:2a06/64 scope link
    valid lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
link/ether 08:00:27:2f:41:ad brd ff:ff:ff:ff:ff:ff
inet 10.0.100.50/24 brd 10.0.100.255 scope global enp0s8
    valid lft forever preferred_lft forever
inet 10.0.100.200/32 scope global enp0s8
    valid lft forever preferred_lft forever
inet6 fe80::a00:27ff:fe2f:41ad/64 scope link
    valid lft forever preferred_lft forever
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
link/ether 08:00:27:ef:a6:72 brd ff:ff:ff:ff:ff:ff
inet 10.0.50.2/24 brd 10.0.50.255 scope global enp0s9
    valid lft forever preferred_lft forever
inet6 fe80::a00:27ff:feef:a672/64 scope link
    valid lft forever preferred_lft forever
root@standby1:/etc/keepalived#

```

Figura 92 - Desconexión Nodo Standby1

```

LXTerminal
File Edit Tabs Help
inet 127.0.0.1/8 scope host lo
  valid_lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
  valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
link/ether 08:00:27:df:72:d8 brd ff:ff:ff:ff:ff:ff
inet 192.168.1.103/24 brd 192.168.1.255 scope global enp0s3
  valid_lft forever preferred_lft forever
inet 192.168.1.200/32 scope global enp0s3
  valid_lft forever preferred_lft forever
inet6 fe80::a00:27ff:fedf:72d8/64 scope link
  valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
link/ether 08:00:27:fd:ad:9e brd ff:ff:ff:ff:ff:ff
inet 10.0.100.51/24 brd 10.0.100.255 scope global enp0s8
  valid_lft forever preferred_lft forever
inet 10.0.100.200/32 scope global enp0s8
  valid_lft forever preferred_lft forever
inet6 fe80::a00:27ff:fedf:ad9e/64 scope link
  valid_lft forever preferred_lft forever
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc fq_codel state UP group default qlen 1000
link/ether 08:00:27:70:fa:f3 brd ff:ff:ff:ff:ff:ff
inet 10.0.50.3/24 brd 10.0.50.255 scope global enp0s9
  valid_lft forever preferred_lft forever
inet6 fe80::a00:27ff:fe70:faf3/64 scope link
  valid_lft forever preferred_lft forever
root@standby2:~#
Sep 09 01:49:58 standby2 Keepalived_vrrp[5877]: (VI_2) Entering MASTER STATE
Sep 09 01:49:58 standby2 Keepalived_vrrp[5877]: VRRP_Group(VG1) Syncing instances to MASTER state
Sep 09 01:49:58 standby2 Keepalived_vrrp[5877]: (VI_1) Entering MASTER STATE
Sep 09 01:50:00 standby2 Keepalived_vrrp[5877]: (VI_3) Entering MASTER STATE
root@standby2:~#

```

Figura 93 - Keepalived transferencia de rol e IPv6s - Standby2

Para finalizar esta prueba, al restablecerse el nodo maestro, el rol e IPv6s son migradas hacia él:

```

Sep 09 01:55:58 cluster1 Keepalived_vrrp[477109]: (VI_1) Entering MASTER STATE
Sep 09 01:55:58 cluster1 Keepalived_vrrp[477109]: VRRP_Group(VG1) Syncing instances to MASTER state
Sep 09 01:55:58 cluster1 Keepalived_vrrp[477109]: (VI_2) Entering MASTER STATE
Sep 09 01:55:58 cluster1 Keepalived_vrrp[477109]: (VI_3) Entering MASTER STATE
root@cluster1:/etc/keepalived#

```

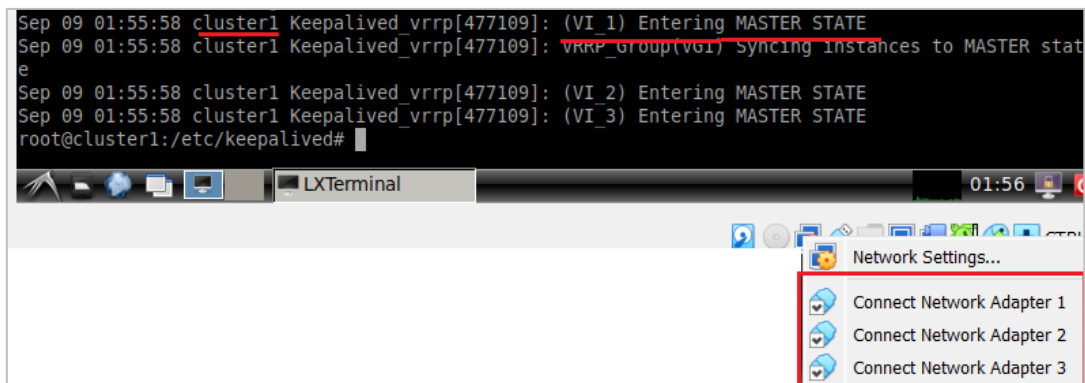


Figura 94 - Conexión Nodo Maestro

```

LXTerminal  LXTerminal  LXTerminal
inet6 ::1/128 scope host noprefixroute
    valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:7d:37:7d brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.101/24 brd 192.168.1.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet 192.168.1.200/32 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe7d:377d/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:53:9f:cd brd ff:ff:ff:ff:ff:ff
    inet 10.0.100.1/24 brd 10.0.100.255 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet 10.0.100.200/32 scope global enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe53:9fcd/64 scope link
        valid_lft forever preferred_lft forever
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:31:9c:c4 brd ff:ff:ff:ff:ff:ff
    inet 10.0.50.1/24 brd 10.0.50.255 scope global enp0s9
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe31:9cc4/64 scope link
        valid_lft forever preferred_lft forever
root@cluster1:/etc/keepalived#
    
```

Figura 95 - Keepalived - IPv6 en Nodo Maestro

De esta manera hemos completado las pruebas de funcionalidad del servicio Keepalived, los roles e IPv6 se transfieren con naturalidad y sin dejar fuera de servicio el cluster.

### 4.1.3 Reparto de carga con HAProxy

Para el tema de reparto de carga, en nuestro clúster usamos el servicio de HAProxy. En el capítulo 3 realizamos una configuración básica de los tres repartidores para que el clúster ofrezca un servicio web (HTTP). Por tal motivo, definimos un par de direcciones IP virtuales (VIP), una por cada interfaz de red de los nodos maestros o directores, de manera que el nodo que esté activo será el que las tendrá configuradas.

Ahora, con HAProxy probaremos el reparto de carga a nivel de transporte (modo TCP) y a nivel de aplicación (modo HTTP), en el primero las peticiones se repartirán equitativamente entre los contenedores disponibles y para la segunda opción, esto es modo HTTP, se tiene persistencia según el nodo en que caiga inicialmente el requerimiento. En ambos casos, los paquetes de los clientes llegarán al repartidor de carga, que cambiará su dirección IP externa VIP (Virtual IP) por la dirección RIP (Real IP) de uno de los **contenedores en los servidores reales**. A su vez, las respuestas de los contenedores en los servidores reales pasan por el director que sustituye su dirección RIP por la dirección VIP.

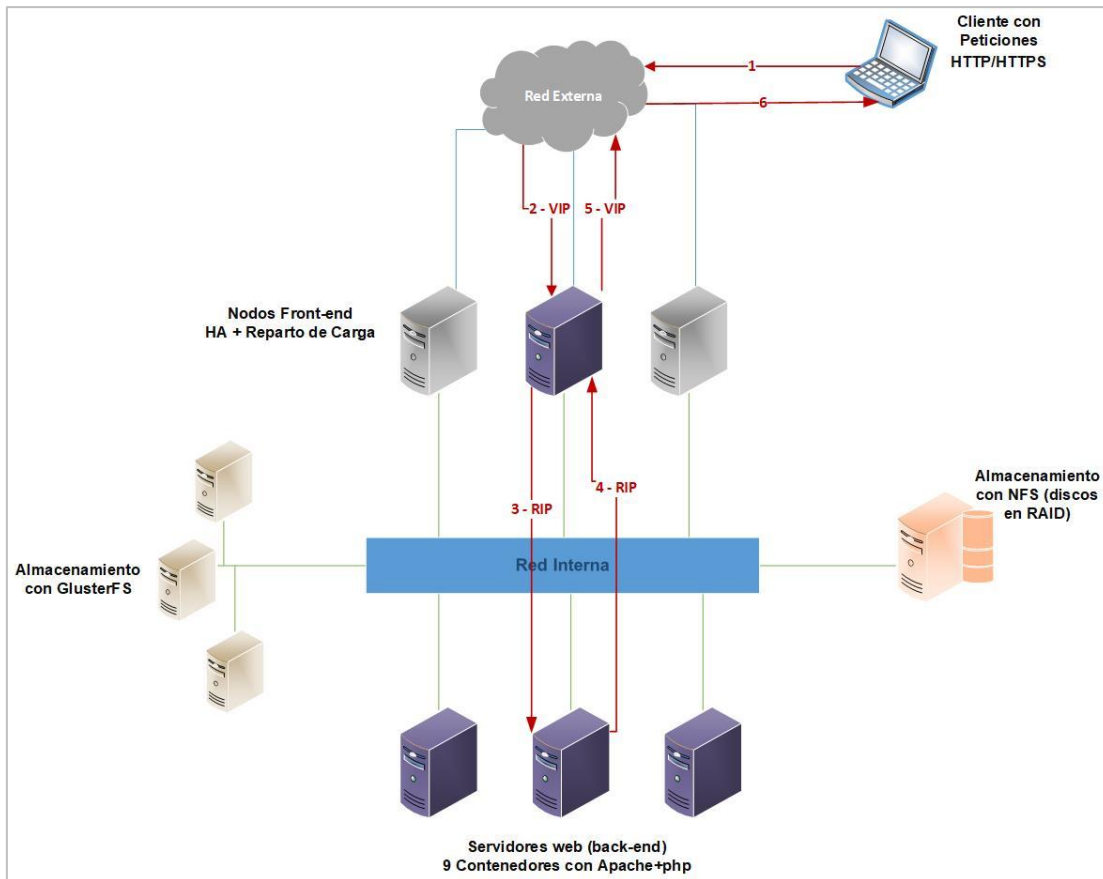


Figura 96 - Flujo de Requerimientos y Respuesta HA Proxy

Usaremos el nodo maestro “cluster1” como repartidor de carga, los equipos standby1 y standby2 estarán en modo pasivo mientras el nodo cluster1 permanezca activo, y el resto de los nodos (cluster2, cluster3 y cluster4), que levantan contenedores, pasarán como los servidores reales. El cliente del servicio web se ejecutará en la máquina física (host) donde se ubican las máquinas virtuales, que en nuestro caso es un equipo con sistema operativo Windows 11. Como queremos ofrecer acceso web a través de dos tipos de reparto de carga diferentes en cluster1 (HTTP y TCP), los diferenciamos definiendo puertos distintos.

De esta manera, para poder probar con facilidad los servicios que se desplegarán en el clúster, procederemos a configurar un reenvío de puertos desde la máquina anfitrión (el equipo Windows que ejecuta VirtualBox) hacia la máquina virtual del clúster que ofrece el servicio.

Podemos hacer la redirección de puertos en las opciones de configuración de la Red NAT de VirtualBox. Desde la misma ventana de red-nat-1, en la pestaña “**Reenvío de puertos**”:

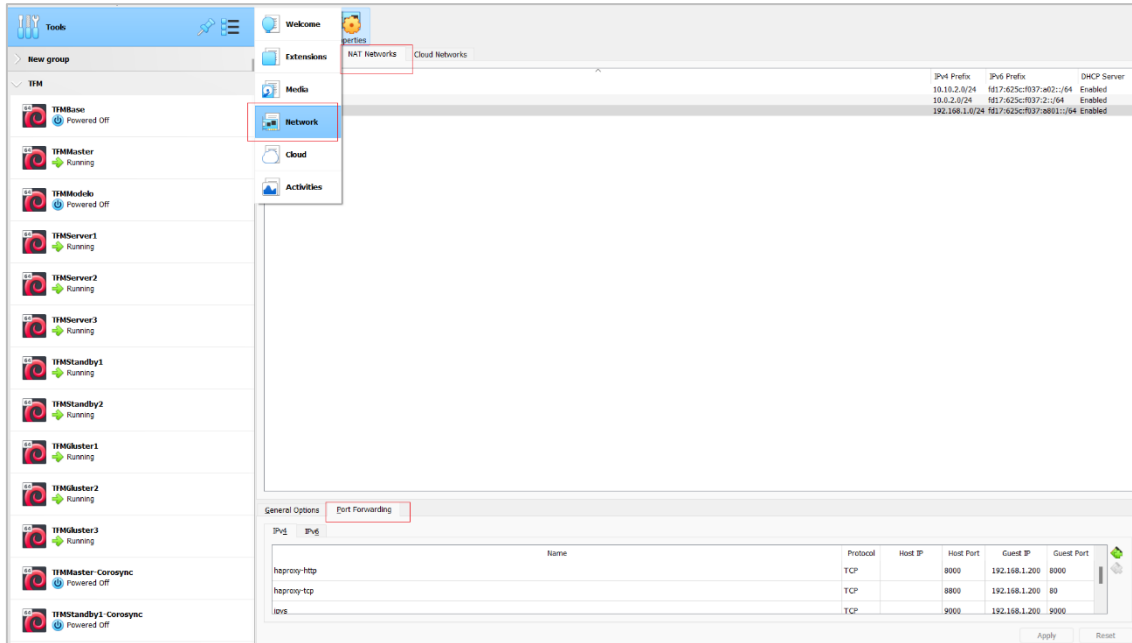


Figura 97 - Redirección de Puertos en VirtualBox

Realizamos las siguientes redirecciones:

Nombre	Protocolo	IP Anfitrión	Puerto Anfitrión	IP Invitado	Puerto Invitado
HAProxy-HTTP	TCP		8000	192.168.1.200	8000
HAProxy-TCP	TCP		8800	192.168.1.200	80
HAProxy-HTTP-gluster	TCP		9000	192.168.1.200	9000
HAProxy-TCP-gluster	TCP		9900	192.168.1.200	90
HA-Estado	TCP		7000	192.168.1.200	7000

Tabla 8 - Redirección de Puertos VirtualBox

Donde el “Puerto anfitrión” podría ser cualquier puerto no privilegiado y el “Puerto invitado” no tiene porqué coincidir con el del anfitrión, pero si debe estar relacionado a la configuración realizada en el servicio de HAProxy. Utilizamos cinco puertos diferentes en el clúster (7000, 9000, 90, 8000 y 80) para ofrecer tres servicios:

- ▶ “ha-estado”: Permite obtener las estadísticas de utilización de HAproxy.
- ▶ “haproxy-http”: Repartidor de carga basado en HAProxy funcionando en modo http (repartidor de nivel 7), en el puerto 8000 se muestran las páginas que se almacenan en NFS y en el puerto 9000 las páginas que se guardan en GlusterFS.

- ▶ “haproxy-tcp”: Repartidor de carga basado en HAProxy funcionando en modo tcp (repartidor de nivel 4), en el puerto 80 se muestran las páginas que se almacenan en NFS y en el puerto 90 las páginas que se guardan en GlusterFS.

En este caso, hemos redirigido el puerto 8000 del host al puerto 8000 de 192.168.1.200 (VIP), el puerto 9000 del host al puerto 9000 de 192.168.1.200, el puerto 8800 del host al puerto 80 de 192.168.1.200, el puerto 9900 del host al puerto 90 de 192.168.1.200 y finalmente, el puerto 7000 del host al puerto 7000 de 192.168.1.200.

Antes de proceder a las pruebas de reparto de carga, como recordatorio, en la siguiente tabla se muestran las IP's implicadas en la configuración del clúster (DIP: Director IP; VIP: Virtual IP y RIP: Real IP):

	Hostname	Dirección IP	
<b>Repartidor de carga Master</b>	cluster1(externa) cluster1 (interna)	DIP: 192.168.1.101 DIP: 10.0.100.1	VIP: 192.168.1.200 VIP: 10.0.100.200
<b>Repartidor de carga StandBy1</b>	standby1(externa) standby1(interna)	DIP: 192.168.1.102 DIP: 10.0.100.50	VIP: 192.168.1.200 VIP: 10.0.100.200
<b>Repartidor de carga StandBy2</b>	standby2(externa) standby2(interna)	DIP: 192.168.1.103 DIP: 10.0.100.51	VIP: 192.168.1.200 VIP: 10.0.100.200
<b>Servidores Reales</b>	cluster2 (interna) cluster3 (interna) cluster4 (interna)	RIP: 10.0.100.2 RIP: 10.0.100.3 RIP: 10.0.100.4	Puertos: 8081, 8082, 8083, 8084 8081, 8082, 8083, 8085 8081, 8082, 8083, 8086
<b>Contenedores</b>	Cluster2 <ul style="list-style-type: none"> <li>• Cluster21p</li> <li>• Cluster22p</li> <li>• Cluster23p</li> <li>• Cluster24p</li> </ul> Cluster3 <ul style="list-style-type: none"> <li>• Cluster31p</li> <li>• Cluster32p</li> <li>• Cluster33p</li> <li>• Cluster34p</li> </ul> Cluster4 <ul style="list-style-type: none"> <li>• Cluster41p</li> <li>• Cluster42p</li> <li>• Cluster43p</li> <li>• Cluster44p</li> </ul>	<ul style="list-style-type: none"> <li>• RIP: 172.17.0.2</li> <li>• RIP: 172.17.0.3</li> <li>• RIP: 172.17.0.4</li> <li>• RIP: 172.17.0.5</li> </ul>	

Tabla 9 - Listado de Direcciones IPs/Puertos del Clúster

**En primera instancia**, comprobaremos que nuestra instalación de HAProxy funciona correctamente, es decir, realizaremos **pruebas de reparto de carga cuando todos los contenedores en los servidores reales se encuentran operativos**, es decir, tenemos doce “servidores” listos para procesar los requerimientos de los usuarios.

Cabe resaltar que, salvo que se diga lo contrario, las pruebas se realizarán en modo TCP por el puerto 80, donde las peticiones se reparten entre los nueve contenedores disponibles y no se tiene persistencia. Las pruebas de HAProxy en modo TCP puerto 90 y HTTP puertos 8000 y 9000, son similares, y estarán los nodos contenedores listos para atender los requerimientos según la configuración realizada.

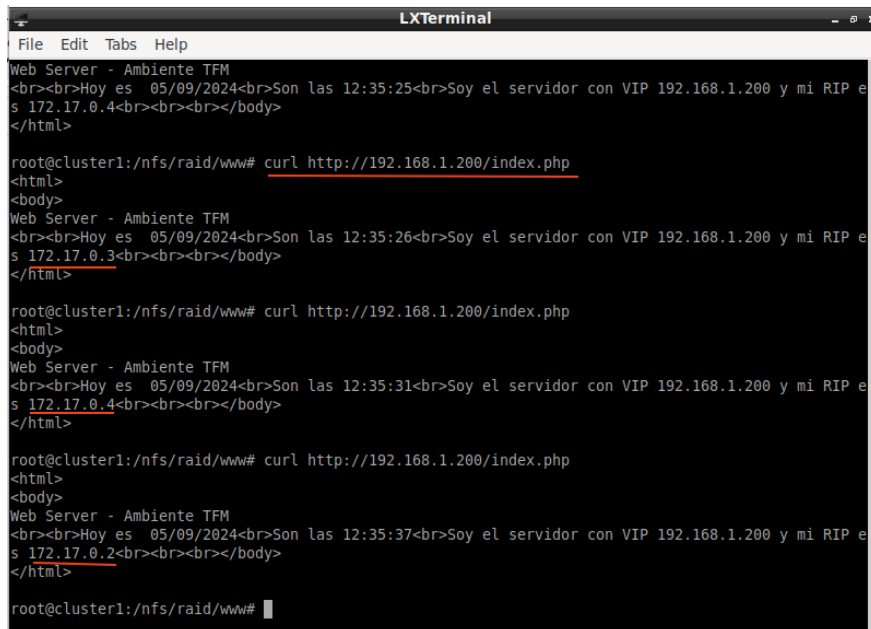
Para arrancar la prueba, primero comprobaremos que podemos acceder desde el nodo maestro y que la carga se reparte correctamente, después lo comprobaremos desde un cliente situado en el exterior, en nuestro caso un navegador desde el host anfitrión.

Para ello, accedemos desde **cluster1** al servidor web con un navegador en modo texto, ejecutamos la siguiente orden que repartirá las peticiones mediante HAProxy en modo TCP:

**curl** <http://192.168.1.200/index.php>

**w3m** <http://192.168.1.200/index.php>

Lo ejecutamos varias veces comprobando que en cada iteración responda un servidor distinto.



```
LXTerminal
File Edit Tabs Help
Web Server - Ambiente TFM
<br><br>Hoy es 05/09/2024<br>Son las 12:35:25<br>Soy el servidor con VIP 192.168.1.200 y mi RIP es 172.17.0.4<br><br><br></body>
</html>

root@cluster1:/nfs/raid/www# curl http://192.168.1.200/index.php
<html>
<body>
Web Server - Ambiente TFM
<br><br>Hoy es 05/09/2024<br>Son las 12:35:26<br>Soy el servidor con VIP 192.168.1.200 y mi RIP es 172.17.0.3<br><br><br></body>
</html>

root@cluster1:/nfs/raid/www# curl http://192.168.1.200/index.php
<html>
<body>
Web Server - Ambiente TFM
<br><br>Hoy es 05/09/2024<br>Son las 12:35:31<br>Soy el servidor con VIP 192.168.1.200 y mi RIP es 172.17.0.4<br><br><br></body>
</html>

root@cluster1:/nfs/raid/www# curl http://192.168.1.200/index.php
<html>
<body>
Web Server - Ambiente TFM
<br><br>Hoy es 05/09/2024<br>Son las 12:35:37<br>Soy el servidor con VIP 192.168.1.200 y mi RIP es 172.17.0.2<br><br><br></body>
</html>

root@cluster1:/nfs/raid/www#
```

Figura 98 - Prueba interna de funcionamiento de HAProxy



Dado que funciona la prueba desde el “interior del clúster”, procedemos a comprobar el funcionamiento desde “fuera del clúster”. Para ello debemos generar tráfico dirigido a la dirección VIP (192.168.1.200) en los puertos 8000 (modo HTTP) u 80 (modo TCP) que son los que hemos definido en el servidor HAProxy.

Como hemos hecho un túnel entre el puerto 8000 de la máquina física y el puerto 8000 de la máquina virtual cluster1, podemos generar el tráfico desde la máquina física (host) accediendo con un navegador a la URL <http://localhost:8000> o desde otra máquina a la dirección <http://dirección IP host:8000>. De igual manera podemos generar tráfico para el puerto 80 del nodo maestro, a través del puerto 8800 de la máquina anfitrión (<http://localhost:8800>)

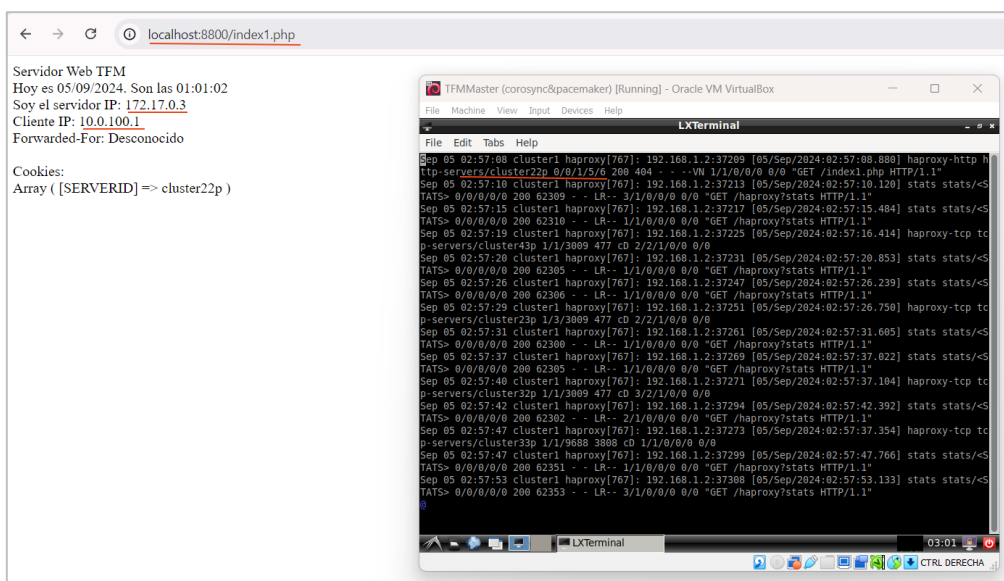


Figura 99 - Prueba externa de funcionamiento de HAProxy - Modo TCP

En la figura se muestra en la parte izquierda la página web con información como “servidor IP” que es el servidor que realmente responde a la petición (el nodo contenedor), “cliente IP” que es el nodo maestro (“cluster1”) y su IP interna, información en el campo “Forwarded-For” que se muestra como “Desconocido” en el modo TCP de HAProxy (<http://localhost:8800>) pero corresponde a la dirección IP del cliente real (el que hace el requerimiento a la página web), este campo será posible de ver en el modo HTTP de HAProxy. Finalmente, la última línea muestra la cookie enviada por el cliente y que corresponde a su hostname.

Podemos recargar la página varias veces y observar que, en modo TCP (puerto 80), las peticiones se reparten entre todos los nodos y por tal motivo, veremos diferentes direcciones IPs en el campo “servidor IP”. Por el contrario, en modo HTTP (8000) podemos recargar varias veces la página y el mismo servidor aparecerá en el campo “Servidor IP” pues recordemos que las conexiones al **puerto 8000** son tratadas

mediante el modo http de HAProxy que hemos configurado para que utilice **persistencia de sesión mediante cookies**.



Figura 100 - Prueba externa de funcionamiento de HAProxy - Modo HTTP

Otro recurso que podemos utilizar para visualizar el **reparto de carga** es la página de **estadísticas de HAProxy**. Podemos conectarnos desde el equipo host, con el navegador que utilicemos, a la dirección **http://localhost:7000/haproxy?stats** y comprobar la página de estadísticas de HAProxy. Se ha configurado esa dirección en el fichero de configuración de HAProxy con el parámetro **stats uri**. Adicional, para poder visualizar el contenido debemos identificarnos con el usuario y contraseña establecidos con **stats auth** en el mismo fichero de configuración, en nuestro caso **admin** y **password**.

<http://localhost:7000/haproxy?stats>

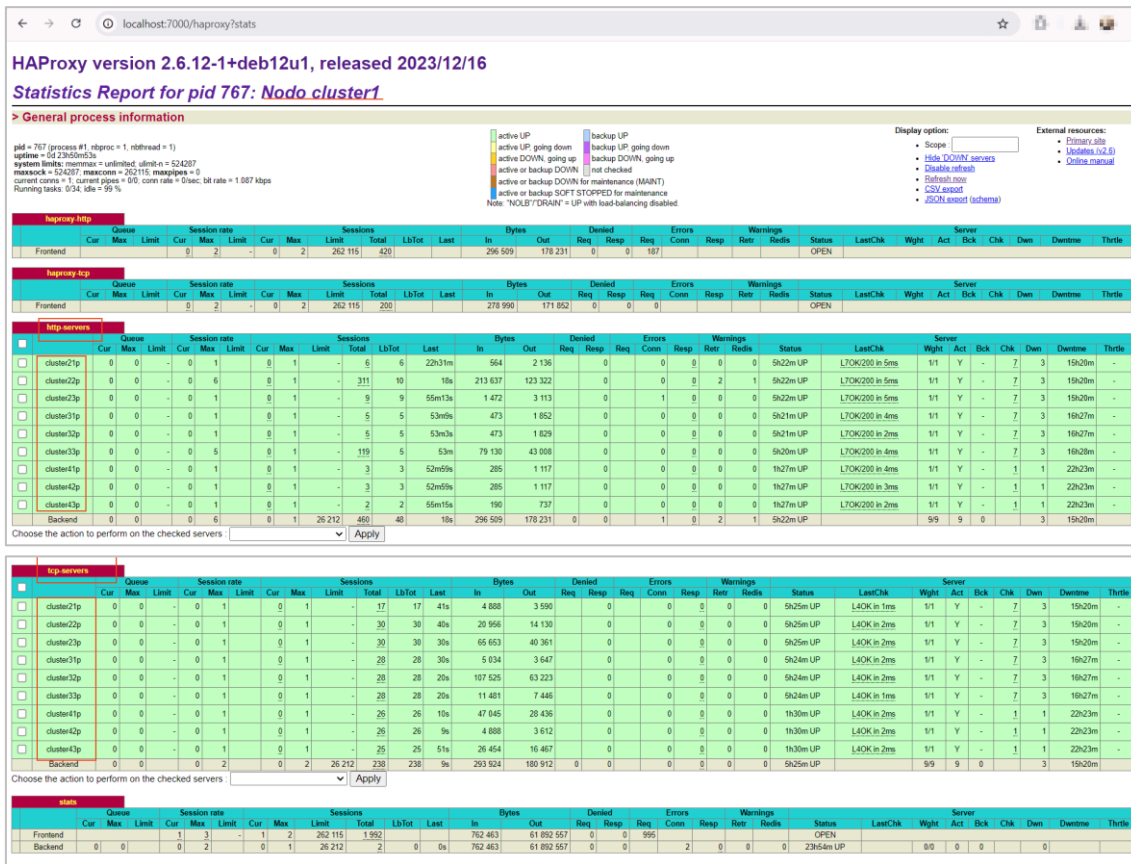


Figura 101 - Estadísticas de HAProxy

Aquí podemos observar el reparto de carga en modo TCP y en modo http, además de que todos los nodos se ven en color verde y con status UP.

Podemos hacer una prueba de generar picos de tráfico, desde el nodo maestro, mediante la siguiente secuencia de órdenes para el modo HTTP:

```
while true; do curl http://192.168.1.200:8000/index1.php --cookie SERVERID=cluster23p; done
```

```
while true; do curl http://192.168.1.200:9000/imagenes.html --cookie SERVERID=cluster23p; done
```

(La opción --cookie "SERVERID=cluster23p" permite que el cliente envíe al servidor web la cookie indicada. Es necesario añadirla así, ya que, por defecto, el navegador curl no utiliza cookies).

Y para generar picos de tráfico en el modo TCP, las líneas a ejecutar son las siguientes:

```
while true; do curl http://192.168.1.200:80/index1.php; done
while true; do curl http://192.168.1.200:90/imagen.html; done
```

Los resultados se muestran a continuación:

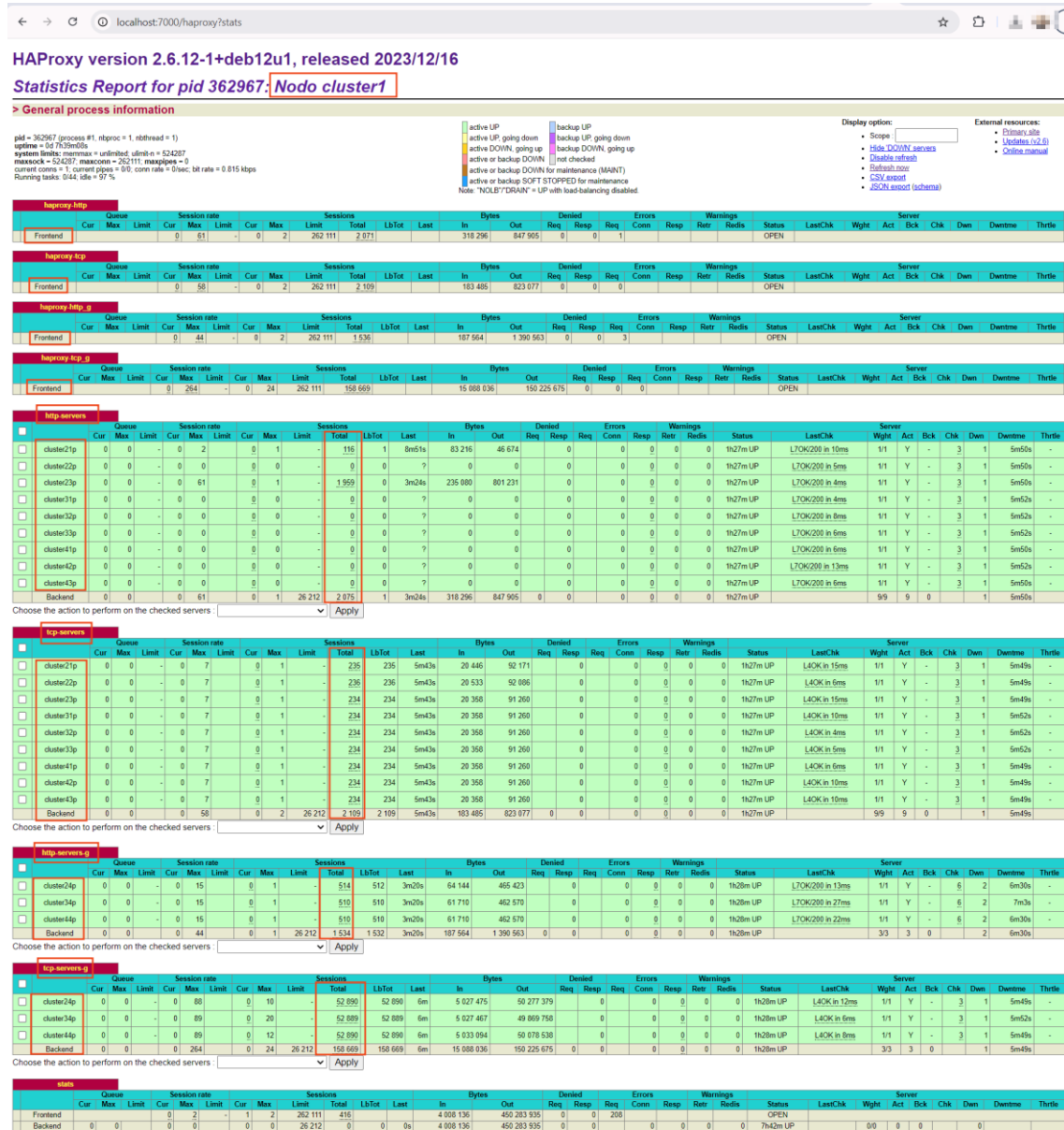


Figura 102 - Estadísticas de HAProxy con sobrecarga

Aquí es importante visualizar como se ha distribuido la carga entre los nodos disponibles, tanto cuando se tienen requerimientos de modo HTTP como cuando se tienen aquellos de modo TCP. Así mismo, se puede observar que por un lado existen 9 contenedores escuchando requerimientos en modo TCP (80) y HTTP (8000), y por otro, tenemos 3 contenedores escuchando requerimientos en modo TCP (90) y HTTP (9000). La diferencia entre estos equipos back-end es el repositorio de datos que utilizan, los servidores http-servers y tcp-servers utilizan NFS y los servidores http-server-g y tcp-servers-g utilizan GlusterFS. En todo caso, las pruebas de funcionalidad son similares.

Hasta aquí hemos comprobado que el sistema funciona correctamente cuando todos sus elementos están activos. Sin embargo, HAProxy es capaz de detectar la falla de alguno de los nodos contenedores y seguir operando y repartiendo la carga entre los demás nodos disponibles. Para ello, podemos detener uno o dos de los contenedores del nodo cluster2, y comprobar en la página de estadísticas de HAProxy su estado DOWN. Ante un nuevo requerimiento, el nodo master repartirá la carga entre los demás nodos:

```

File Machine View Input Devices Help
root@cluster2:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED    STATUS    PORTS
124ae763e8e5       php:apache         "docker-php-entrypoi..." 4 days ago Up 2 days 0.0.0.0
:8084->80/tcp, :::8084->80/tcp cluster24p
5dffa0efcc41       php:apache         "docker-php-entrypoi..." 7 weeks ago Up 2 days 0.0.0.0
:8083->80/tcp, :::8083->80/tcp cluster23p
4ed7e1aa52cb       php:apache         "docker-php-entrypoi..." 7 weeks ago Up 2 days 0.0.0.0
:8082->80/tcp, :::8082->80/tcp cluster22p
d0ad8478bea2       php:apache         "docker-php-entrypoi..." 7 weeks ago Up 2 days 0.0.0.0
:8081->80/tcp, :::8081->80/tcp cluster21p
dd7a0ae87d7f       hello-world        "/hello"                 8 weeks ago Exited (0) 8 weeks ago
pensive_shamir
root@cluster2:~# docker stop cluster21p
[211875.428248] docker0: port 1(vetha14d259) entered disabled state
[211875.428395] veth8d5e24f: renamed from eth0
[211875.477739] docker0: port 1(vetha14d259) entered disabled state
[211875.478689] device vetha14d259 left promiscuous mode
[211875.478719] docker0: port 1(vetha14d259) entered disabled state
cluster21p
root@cluster2:~# docker stop cluster22p
[211880.134228] docker0: port 2(veth1f3cff8) entered disabled state
[211880.134283] veth014f663: renamed from eth0
[211880.182637] docker0: port 2(veth1f3cff8) entered disabled state
[211880.183052] device veth1f3cff8 left promiscuous mode
[211880.183098] docker0: port 2(veth1f3cff8) entered disabled state
cluster22p
root@cluster2:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED    STATUS    PORTS
124ae763e8e5       php:apache         "docker-php-entrypoi..." 4 days ago Up 2 days 0.0.0.0
:8084->80/tcp, :::8084->80/tcp cluster24p
5dffa0efcc41       php:apache         "docker-php-entrypoi..." 7 weeks ago Up 2 days 0.0.0.0
:8083->80/tcp, :::8083->80/tcp cluster23p
4ed7e1aa52cb       php:apache         "docker-php-entrypoi..." 7 weeks ago Exited (0) 34 seconds ago
cluster22p
d0ad8478bea2       php:apache         "docker-php-entrypoi..." 7 weeks ago Exited (0) 39 seconds ago
cluster21p
dd7a0ae87d7f       hello-world        "/hello"                 8 weeks ago Exited (0) 8 weeks ago
pensive_shamir
root@cluster2:~# _

```

Figura 103 - Prueba de contenedores detenidos

http-servers																						
	Queue			Session rate			Sessions			LbTot	Last	Bytes		Denied		Errors		Warnings		Status		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit			Total	In	Out	Req	Resp	Req	Conn	Resp		Retr	Redis
<input checked="" type="checkbox"/>	cluster21p	0	0	-	0	2	0	1	-	116	1	42m9s	83 216	46 674	0	0	0	0	0	0	4m35s DOWN	
<input checked="" type="checkbox"/>	cluster22p	0	0	-	0	0	0	0	-	0	0	?	0	0	0	0	0	0	0	0	4m30s DOWN	
<input type="checkbox"/>	cluster23p	0	0	-	0	61	0	1	-	1 959	0	36m42s	235 080	801 231	0	0	0	0	0	0	2h31s UP	
<input type="checkbox"/>	cluster31p	0	0	-	0	0	0	0	-	0	0	?	0	0	0	0	0	0	0	0	2h29s UP	
<input type="checkbox"/>	cluster32p	0	0	-	0	0	0	0	-	0	0	?	0	0	0	0	0	0	0	0	2h29s UP	
<input type="checkbox"/>	cluster33p	0	0	-	0	0	0	0	-	0	0	?	0	0	0	0	0	0	0	0	2h29s UP	
<input type="checkbox"/>	cluster41p	0	0	-	0	0	0	0	-	0	0	?	0	0	0	0	0	0	0	0	2h31s UP	
<input type="checkbox"/>	cluster42p	0	0	-	0	0	0	0	-	0	0	?	0	0	0	0	0	0	0	0	2h29s UP	
<input type="checkbox"/>	cluster43p	0	0	-	0	0	0	0	-	0	0	?	0	0	0	0	0	0	0	0	2h31s UP	
	Backend	0	0	-	0	61	0	1	-	26 212	2 075	1	36m42s	318 296	847 905	0	0	0	0	0	0	2h31s UP

tcp-servers																					
	Queue			Session rate			Sessions			LbTot	Last	Bytes		Denied		Errors		Warnings		Status	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit			Total	In	Out	Req	Resp	Req	Conn	Resp		Retr
<input checked="" type="checkbox"/>	cluster21p	0	0	-	0	7	0	1	-	235	235	39m1s	20 446	92 171	0	0	0	0	0	0	4m35s DOWN
<input checked="" type="checkbox"/>	cluster22p	0	0	-	0	7	0	1	-	236	236	39m1s	20 533	92 086	0	0	0	0	0	0	4m31s DOWN
<input type="checkbox"/>	cluster23p	0	0	-	0	7	0	1	-	234	234	39m1s	20 358	91 260	0	0	0	0	0	0	2h31s UP
<input type="checkbox"/>	cluster31p	0	0	-	0	7	0	1	-	234	234	39m1s	20 358	91 260	0	0	0	0	0	0	2h29s UP
<input type="checkbox"/>	cluster32p	0	0	-	0	7	0	1	-	234	234	39m1s	20 358	91 260	0	0	0	0	0	0	2h29s UP
<input type="checkbox"/>	cluster33p	0	0	-	0	7	0	1	-	234	234	39m1s	20 358	91 260	0	0	0	0	0	0	2h29s UP
<input type="checkbox"/>	cluster41p	0	0	-	0	7	0	1	-	234	234	39m1s	20 358	91 260	0	0	0	0	0	0	2h31s UP
<input type="checkbox"/>	cluster42p	0	0	-	0	7	0	1	-	234	234	39m1s	20 358	91 260	0	0	0	0	0	0	2h31s UP
<input type="checkbox"/>	cluster43p	0	0	-	0	7	0	1	-	234	234	39m1s	20 358	91 260	0	0	0	0	0	0	2h31s UP
	Backend	0	0	-	0	58	0	2	-	2 109	2 109	39m1s	183 485	823 077	0	0	0	0	0	0	2h31s UP

Figura 104 - HAProxy detección de contenedores caídos

Nos resta comprobar que el repartidor de carga es capaz de detectar cuando falla alguno/s de los servidores reales y con ello todos los contenedores que se ejecutan en él. Si ocurre, los servidores supervivientes se harán cargo del servicio. El fallo de un servidor real lo conseguiremos desconectando (virtualmente) el cable conectado al interfaz de red, por ejemplo del nodo cluster3.

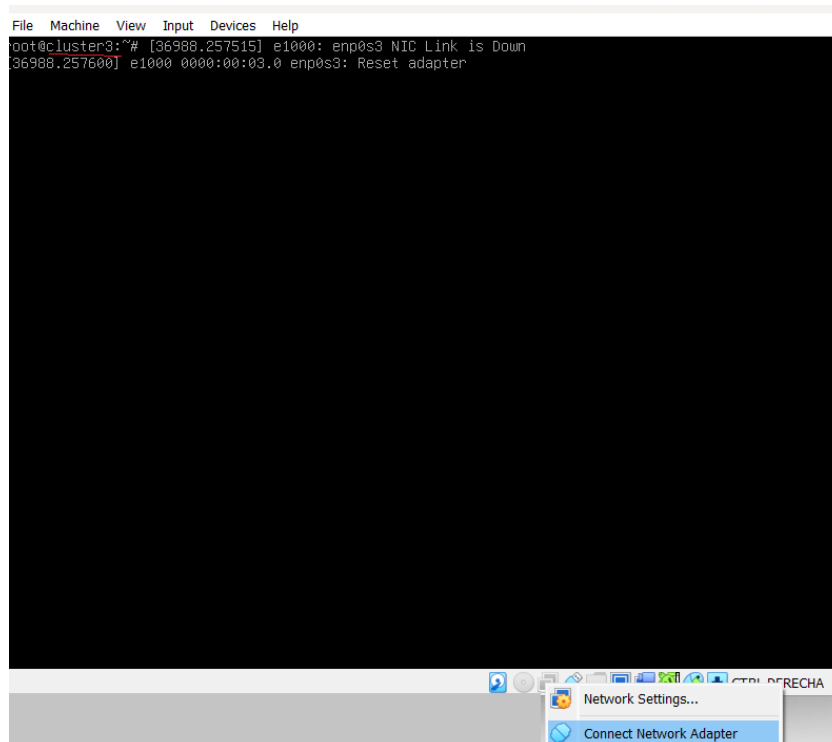


Figura 105 - Simulación caída nodo cluster3

http.servers																								
	Cur	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Status		
		Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis			
<input type="checkbox"/>	0	0	-	0	47		0	1	-	270	6	12m26s	32 244	110 112		0	0	0	0	0	0	5h52m UP		
<input type="checkbox"/>	0	0	-	0	6		0	1	-	311	10	30m38s	213 637	123 322		0	0	0	2	1	0	5h52m UP		
<input type="checkbox"/>	0	0	-	0	48		0	1	-	3 059	9	12m55s	367 472	1 250 563		0	1	0	0	0	0	5h52m UP		
<input checked="" type="checkbox"/>	0	0	-	0	61		0	1	-	144	5	12m8s	17 153	58 703		0	0	0	0	0	0	1m12s DOWN		
<input checked="" type="checkbox"/>	0	0	-	0	61		0	1	-	345	5	12m36s	41 273	140 889		0	0	0	0	0	0	1m12s DOWN		
<input checked="" type="checkbox"/>	0	0	-	0	5		0	1	-	119	5	1h23m	79 130	43 008		0	0	0	0	0	0	1m12s DOWN		
<input type="checkbox"/>	0	0	-	0	60		0	1	-	523	3	12m13s	62 685	213 797		0	0	0	0	0	0	1h57m UP		
<input type="checkbox"/>	0	0	-	0	1		0	1	-	3	3	1h23m	285	1 117		0	0	0	0	0	0	1h57m UP		
<input type="checkbox"/>	0	0	-	0	58		0	1	-	194	2	12m47s	23 230	79 265		0	0	0	0	0	0	1h57m UP		
Backend	0	0	-	0	61		0	1	-	26 212	4 965	48	12m8s	837 109	2 020 776		0	1	0	2	1	5h52m UP		

Choose the action to perform on the checked servers :  Apply

tcp.servers																								
	Cur	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Status		
		Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis			
<input type="checkbox"/>	0	0	-	1	6		1	1	-	276	276	1s	38 524	99 532		0	0	0	0	0	0	5h52m UP		
<input type="checkbox"/>	0	0	-	1	6		1	2	-	289	289	1s	53 545	108 428		0	0	0	0	0	0	5h52m UP		
<input type="checkbox"/>	0	0	-	0	6		0	1	-	288	288	22s	99 289	136 306		0	0	0	0	0	0	5h52m UP		
<input checked="" type="checkbox"/>	0	0	-	0	6		0	2	-	284	284	1m34s	37 797	98 715		0	0	0	0	0	0	1m13s DOWN		
<input checked="" type="checkbox"/>	0	0	-	0	6		0	1	-	284	284	1m34s	139 589	158 212		0	0	0	0	0	0	1m13s DOWN		
<input checked="" type="checkbox"/>	0	0	-	0	6		0	2	-	284	284	1m24s	44 856	102 218		0	0	0	0	0	0	1m12s DOWN		
<input type="checkbox"/>	0	0	-	0	6		0	1	-	284	284	21s	79 022	123 031		0	0	0	0	0	0	1h57m UP		
<input type="checkbox"/>	0	0	-	0	6		0	1	-	284	284	11s	39 223	99 644		0	0	0	0	0	0	1h57m UP		
<input type="checkbox"/>	0	0	-	0	6		0	2	-	284	284	11s	59 130	111 149		0	0	0	0	0	0	1h57m UP		
Backend	0	0	-	2	54		2	4	-	26 212	2 557	2 557	1s	590 975	1 037 235		0	0	0	0	0	5h52m UP		

Choose the action to perform on the checked servers :  Apply

Figura 106 - Estadísticas de HAProxy detección de caída de nodos

Nótese que el fallo de los tres contenedores del servidor real cluster3 ha sido detectado. cluster31p, cluster32p y cluster33p se muestran con status DOWN, y por tanto, los nuevos requerimientos serán atendidos por el restante de contenedores supervivientes.

Otro comportamiento que observaremos, es el caso de una avería del nodo master, el que reparte la carga, entonces en la página de estadísticas de HAProxy se evidenciará el cambio de repartidor.

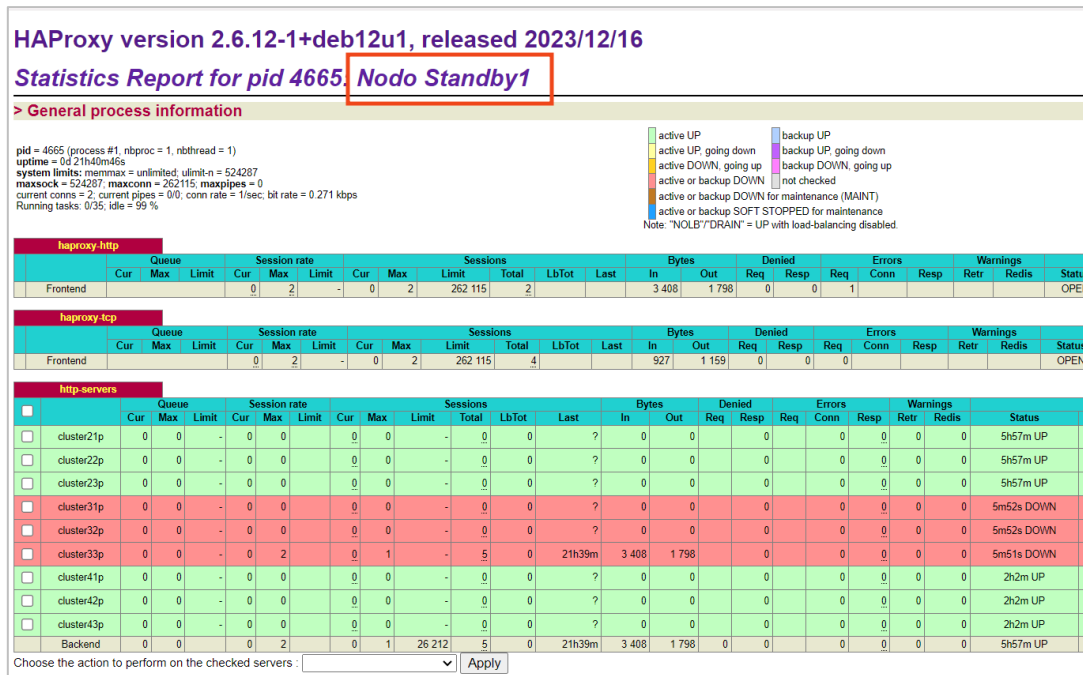


Figura 107 - Estadísticas de HAProxy detección de caída nodo máster

En la página de estadísticas de HAProxy, se resalta el nombre del nuevo repartidor, esto dado que se ha detectado que el nodo máster no está disponible.

Estas son algunas situaciones que se pueden presentar como parte del uso de HAProxy, servicio para el cual hemos comprobado su funcionalidad.

#### 4.1.4 Uso de sistema de almacenamiento con GlusterFS

Para la configuración del sistema de archivos basado en GlusterFS se utilizaron 3 servidores: gluster1, gluster2 y gluster3, donde cada uno publica dos bricks: uno para datos replicados y otro para datos distribuidos. Basados en eso, tenemos dos volúmenes que se comparten al resto de nodos del clúster:

Nombre del volumen	Tipo	Punto de Montaje
gv0	Replicado	/gluster/gfs-r
gv1	Distribuido	/gluster/gfs-d

Tabla 10 - Volúmenes GlusterFS



Podemos observar los puntos de montaje y el uso del sistema de archivos desde uno de los nodos del clúster, por ejemplo, desde el servidor cluster1.

```
root@cluster1:/# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=468860k,nr_inodes=117215,mode=755,inode64)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,noexec,relatime,size=98388k,mode=755,inode64)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k,inode64)
cgroup2 on /sys/fs/cgroup type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate,memory_recursiveprot)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=30,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=12746)
mqueue on /dev/mqueue type mqueue (rw,nosuid,nodev,noexec,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
debugfs on /sys/kernel/debug type debugfs (rw,nosuid,nodev,noexec,relatime)
tracefs on /sys/kernel/tracing type tracefs (rw,nosuid,nodev,noexec,relatime)
fusectl on /sys/fs/fuse/connections type fusectl (rw,nosuid,nodev,noexec,relatime)
configfs on /sys/kernel/config type configfs (rw,nosuid,nodev,noexec,relatime)
ramfs on /run/credentials/systemd-sysctl.service type ramfs (ro,nosuid,nodev,noexec,relatime,mode=700)
ramfs on /run/credentials/systemd-sysusers.service type ramfs (ro,nosuid,nodev,noexec,relatime,mode=700)
ramfs on /run/credentials/systemd-tmpfiles-setup-dev.service type ramfs (ro,nosuid,nodev,noexec,relatime,mode=700)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,nosuid,nodev,noexec,relatime)
ramfs on /run/credentials/systemd-tmpfiles-setup.service type ramfs (ro,nosuid,nodev,noexec,relatime,mode=700)
sunrpc on /run/rpc pipefs type rpc pipefs (rw,relatime)
gluster1:/gv1 on /gluster/gfs-d type fuse.glusterfs (rw,relatime,user_id=0,group_id=0,default_permissions,allow_other,max_read=131072,netdev)
gluster1:/gv0 on /gluster/gfs-r type fuse.glusterfs (rw,relatime,user_id=0,group_id=0,default_permissions,allow_other,max_read=131072,netdev)
nas:/ on /nfs type nfs4 (rw,relatime,vers=4.2,rsize=8192,wsize=8192,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=10.0.100.1,local_lock=none,addr=10.0.100.100)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,size=98384k,nr_inodes=24596,mode=700,inode64)
root@cluster1:/#
```

Figura 108 - Volúmenes glusters montados

Ahora podemos crear algunos archivos para observar el funcionamiento de GlusterFS.

La siguiente orden crea 100 copias de un archivo (por ejemplo /etc/hosts) en el volumen replicado:

```
for i in `seq 1 100`; do cp /etc/hosts /gluster/gfs-r/replica$i; done
```

Si obtenemos un listado del contenido del *brick1* correspondiente en cada una de las máquinas, observaremos que todas tienen los 100 archivos creados, puesto que el volumen funciona como replicado. Esto es, en las máquinas gluster1, gluster2 y gluster3, el listado mostrado debe ser el mismo:

```
ls /export/brick1/data
```

```
File Machine View Input Devices Help
root@gluster1:~# cd /export/brick1/data/
root@gluster1:/export/brick1/data# ls
replica1 replica2 replica30 replica41 replica52 replica63 replica74 replica85 replica96
replica10 replica20 replica31 replica42 replica53 replica64 replica75 replica86 replica97
replica100 replica21 replica32 replica43 replica54 replica65 replica76 replica87 replica98
replica11 replica22 replica33 replica44 replica55 replica66 replica77 replica88 replica99
replica12 replica23 replica34 replica45 replica56 replica67 replica78 replica89
replica13 replica24 replica35 replica46 replica57 replica68 replica79 replica9
replica14 replica25 replica36 replica47 replica58 replica69 replica8 replica90
replica15 replica26 replica37 replica48 replica59 replica7 replica80 replica91
replica16 replica27 replica38 replica49 replica6 replica70 replica81 replica92
replica17 replica28 replica39 replica5 replica60 replica71 replica82 replica93
replica18 replica29 replica4 replica50 replica61 replica72 replica83 replica94
replica19 replica3 replica40 replica51 replica62 replica73 replica84 replica95
root@gluster1:/export/brick1/data# _
```

Figura 109 - Volumen replicado gluster1

```
File Machine View Input Devices Help
root@gluster2:/export/brick1/data# ls
replica1 replica2 replica30 replica41 replica52 replica63 replica74 replica85 replica96
replica10 replica20 replica31 replica42 replica53 replica64 replica75 replica86 replica97
replica100 replica21 replica32 replica43 replica54 replica65 replica76 replica87 replica98
replica11 replica22 replica33 replica44 replica55 replica66 replica77 replica88 replica99
replica12 replica23 replica34 replica45 replica56 replica67 replica78 replica89
replica13 replica24 replica35 replica46 replica57 replica68 replica79 replica9
replica14 replica25 replica36 replica47 replica58 replica69 replica8 replica90
replica15 replica26 replica37 replica48 replica59 replica7 replica80 replica91
replica16 replica27 replica38 replica49 replica6 replica70 replica81 replica92
replica17 replica28 replica39 replica5 replica60 replica71 replica82 replica93
replica18 replica29 replica4 replica50 replica61 replica72 replica83 replica94
replica19 replica3 replica40 replica51 replica62 replica73 replica84 replica95
root@gluster2:/export/brick1/data#
```

Figura 110 - Volumen replicado gluster2

```
File Machine View Input Devices Help
root@gluster3:/export/brick1/data# ls
replica1 replica2 replica30 replica41 replica52 replica63 replica74 replica85 replica96
replica10 replica20 replica31 replica42 replica53 replica64 replica75 replica86 replica97
replica100 replica21 replica32 replica43 replica54 replica65 replica76 replica87 replica98
replica11 replica22 replica33 replica44 replica55 replica66 replica77 replica88 replica99
replica12 replica23 replica34 replica45 replica56 replica67 replica78 replica89
replica13 replica24 replica35 replica46 replica57 replica68 replica79 replica9
replica14 replica25 replica36 replica47 replica58 replica69 replica8 replica90
replica15 replica26 replica37 replica48 replica59 replica7 replica80 replica91
replica16 replica27 replica38 replica49 replica6 replica70 replica81 replica92
replica17 replica28 replica39 replica5 replica60 replica71 replica82 replica93
replica18 replica29 replica4 replica50 replica61 replica72 replica83 replica94
replica19 replica3 replica40 replica51 replica62 replica73 replica84 replica95
root@gluster3:/export/brick1/data#
```

Figura 111 - Volumen replicado gluster3

Y un vistazo del lado del cliente gluster, el directorio luce así:

```

root@cluster1:/gluster/gfs-r# ls
replica1  replica20  replica32  replica44  replica56  replica68  replica8  replica91
replica10 replica21  replica33  replica45  replica57  replica69  replica80 replica92
replica100 replica22  replica34  replica46  replica58  replica7  replica81 replica93
replica11  replica23  replica35  replica47  replica59  replica70  replica82 replica94
replica12  replica24  replica36  replica48  replica6  replica71  replica83 replica95
replica13  replica25  replica37  replica49  replica60  replica72  replica84 replica96
replica14  replica26  replica38  replica5  replica61  replica73  replica85 replica97
replica15  replica27  replica39  replica50  replica62  replica74  replica86 replica98
replica16  replica28  replica4  replica51  replica63  replica75  replica87 replica99
replica17  replica29  replica40  replica52  replica64  replica76  replica88
replica18  replica3  replica41  replica53  replica65  replica77  replica89
replica19  replica30  replica42  replica54  replica66  replica78  replica9
replica2  replica31  replica43  replica55  replica67  replica79  replica90
root@cluster1:/gluster/gfs-r#

```

Figura 112 - Contenido de directorio replicado de glusterfs

Por otra parte, la siguiente orden crea 100 copias de un archivo (por ejemplo /etc/hosts) en el volumen distribuido:

```
for i in `seq 1 100`; do cp /etc/hosts /gluster/gfs-d/distr$i; done
```

Si obtenemos un listado del contenido del *brick* correspondiente en cada una de las máquinas, observaremos que los archivos creados se han repartido entre los tres *bricks* que componen el volumen. Esto es, en las máquinas gluster1, gluster2 y gluster3, el listado mostrado por la siguiente orden es distinto:

```
ls /export/brick2/data
```

```

File Machine View Input Devices Help
root@gluster1:/export/brick2/data# ls
distr10  distr16  distr26  distr38  distr52  distr6  distr67  distr73  distr78  distr88  distr98
distr100 distr19  distr31  distr45  distr56  distr61  distr68  distr74  distr79  distr90
distr14  distr24  distr32  distr47  distr58  distr62  distr7  distr75  distr8  distr91
distr15  distr25  distr33  distr48  distr59  distr63  distr70  distr76  distr82  distr96
root@gluster1:/export/brick2/data# _

```

Figura 113 - Volumen distribuido gluster1

```

File Machine View Input Devices Help
root@gluster2:/export/brick2/data# ls
distr1  distr18  distr29  distr36  distr44  distr53  distr64  distr81  distr87  distr95
distr11 distr2  distr3  distr39  distr5  distr54  distr71  distr84  distr9  distr99
distr12 distr21  distr35  distr43  distr50  distr55  distr72  distr86  distr94
root@gluster2:/export/brick2/data# _

```

Figura 114 - Volumen distribuido gluster2

```
File Machine View Input Devices Help
root@gluster3:/export/brick2/data# ls
distr13 distr22 distr28 distr37 distr41 distr49 distr60 distr69 distr83 distr92
distr17 distr23 distr30 distr4 distr42 distr51 distr65 distr77 distr85 distr93
distr20 distr27 distr34 distr40 distr46 distr57 distr66 distr80 distr89 distr97
root@gluster3:/export/brick2/data# _
```

Figura 115 - Volumen distribuido gluster3

De hecho, la unión de los archivos mostrados en cada una de las máquinas deben ser los 100 archivos que hemos creado, y eso lo podemos observar en uno de los nodos cliente, como el nodo cluster1:

```
root@cluster1:/gluster/gfs-d# ls
distr1 distr18 distr27 distr36 distr45 distr54 distr63 distr72 distr81 distr90
distr10 distr19 distr28 distr37 distr46 distr55 distr64 distr73 distr82 distr91
distr100 distr2 distr29 distr38 distr47 distr56 distr65 distr74 distr83 distr92
distr11 distr20 distr3 distr39 distr48 distr57 distr66 distr75 distr84 distr93
distr12 distr21 distr30 distr4 distr49 distr58 distr67 distr76 distr85 distr94
distr13 distr22 distr31 distr40 distr5 distr59 distr68 distr77 distr86 distr95
distr14 distr23 distr32 distr41 distr50 distr6 distr69 distr78 distr87 distr96
distr15 distr24 distr33 distr42 distr51 distr60 distr7 distr79 distr88 distr97
distr16 distr25 distr34 distr43 distr52 distr61 distr70 distr8 distr89 distr98
distr17 distr26 distr35 distr44 distr53 distr62 distr71 distr80 distr9 distr99
root@cluster1:/gluster/gfs-d# █
```

Figura 116 - Contenido de directorio distribuido de gluster

Otra prueba de funcionalidad es la disponibilidad de la información en el volumen replicado, aunque uno de los nodos gluster no estuviera disponible. Para ello, vamos a desconectar la interfaz de red del nodo gluster1 y comprobaremos el acceso a los datos desde el cliente cluster1.

```
TFMGluster1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@gluster1:/export/brick1/data# ls
replica1 replica2 replica30 replica41 replica52 replica63 replica74 replica85 replica96
replica10 replica20 replica31 replica42 replica53 replica64 replica75 replica86 replica97
replica100 replica21 replica32 replica43 replica54 replica65 replica76 replica87 replica98
replica11 replica22 replica33 replica44 replica55 replica66 replica77 replica88 replica99
replica12 replica23 replica34 replica45 replica56 replica67 replica78 replica89 test
replica13 replica24 replica35 replica46 replica57 replica68 replica79 replica9 vanessa
replica14 replica25 replica36 replica47 replica58 replica69 replica8 replica90 www
replica15 replica26 replica37 replica48 replica59 replica7 replica80 replica91
replica16 replica27 replica38 replica49 replica6 replica70 replica81 replica92
replica17 replica28 replica39 replica5 replica60 replica71 replica82 replica93
replica18 replica29 replica4 replica50 replica61 replica72 replica83 replica94
replica19 replica3 replica40 replica51 replica62 replica73 replica84 replica95
root@gluster1:/export/brick1/data# [71578.430256] e1000 0000:00:03:0 enp0s3: Reset adapter
```

Figura 117 - Prueba de desconexión nodo gluster1

```

root@cluster1:/gluster/gfs-r# ls
replica1 replica20 replica32 replica44 replica56 replica68 replica8 replica91
replica10 replica21 replica33 replica45 replica57 replica69 replica80 replica92
replica100 replica22 replica34 replica46 replica58 replica7 replica81 replica93
replica11 replica23 replica35 replica47 replica59 replica70 replica82 replica94
replica12 replica24 replica36 replica48 replica6 replica71 replica83 replica95
replica13 replica25 replica37 replica49 replica60 replica72 replica84 replica96
replica14 replica26 replica38 replica5 replica61 replica73 replica85 replica97
replica15 replica27 replica39 replica50 replica62 replica74 replica86 replica98
replica16 replica28 replica4 replica51 replica63 replica75 replica87 replica99
replica17 replica29 replica40 replica52 replica64 replica76 replica88 test
replica18 replica3 replica41 replica53 replica65 replica77 replica89 vanessa
replica19 replica30 replica42 replica54 replica66 replica78 replica9 www
replica2 replica31 replica43 replica55 replica67 replica79 replica90
root@cluster1:/gluster/gfs-r# touch newfile
root@cluster1:/gluster/gfs-r# ls
newfile replica2 replica31 replica43 replica55 replica67 replica79 replica90
replica1 replica20 replica32 replica44 replica56 replica68 replica8 replica91
replica10 replica21 replica33 replica45 replica57 replica69 replica80 replica92
replica100 replica22 replica34 replica46 replica58 replica7 replica81 replica93
replica11 replica23 replica35 replica47 replica59 replica70 replica82 replica94
replica12 replica24 replica36 replica48 replica6 replica71 replica83 replica95
replica13 replica25 replica37 replica49 replica60 replica72 replica84 replica96
replica14 replica26 replica38 replica5 replica61 replica73 replica85 replica97
replica15 replica27 replica39 replica50 replica62 replica74 replica86 replica98
replica16 replica28 replica4 replica51 replica63 replica75 replica87 replica99
replica17 replica29 replica40 replica52 replica64 replica76 replica88 test
replica18 replica3 replica41 replica53 replica65 replica77 replica89 vanessa
replica19 replica30 replica42 replica54 replica66 replica78 replica9 www
root@cluster1:/gluster/gfs-r# █

```

Figura 118 - Acceso a directorio replicado

En la figura, se puede observar que listamos el contenido del directorio replicado y adicional podemos continuar escribiendo en disco.

```

File Machine View Input Devices Help
root@gluster2:/export/brick1/data# gluster peer status
Number of Peers: 2

Hostname: gluster1.cluster
Uuid: 61e20a22-147e-481f-a312-aa3b0eada7ed
State: Peer in Cluster (Disconnected)

Hostname: gluster3
Uuid: 61421558-3102-4a12-9c0d-46d5e7a1ca22
State: Peer in Cluster (Connected)
root@gluster2:/export/brick1/data# _

```

Figura 119 - Estado de nodos GusterFS

El comando **gluster peer status**, nos permite observar el estado de cada uno de los nodos del sistema GlusterFS. En la figura, se puede apreciar la desconexión del nodo gluster1.

Finalmente, y dado que hemos creado contenedores adicionales en el clúster donde el directorio del servicio de Apache-PHP es un directorio en el sistema de almacenamiento

de GlusterFS, vamos a comprobar que podemos seguir accediendo a la página web ahí almacenada, aun cuando un nodo (gluster1) del esquema gluster no se encuentra disponible:

```
root@cluster1:/gluster/gfs-r/www# ls -l
total 403
-rw-r--r-x 1 root root 659 Sep 8 14:26 imagen.html
-rw-r--r-x 1 root root 204800 Sep 6 00:54 imagen.jpg
-rw----r-x 1 root root 512 Sep 6 17:25 index1.php
-rw-r--r-- 1 root root 298 Sep 6 17:24 index.php
-rw----r-x 1 root root 204800 Sep 8 14:24 lion.jpg
-rw----r-x 1 root root 473 Sep 6 17:24 pi.php
root@cluster1:/gluster/gfs-r/www#
```

Figura 120 - Directorio Apache en GusterFS



Figura 121 - Disponibilidad de página web con GlusterFS

Con esto hemos mostrado la funcionalidad del sistema de almacenamiento GlusterFS.

#### 4.1.5 Uso de sistema de almacenamiento con NFS

Dentro del clúster, utilizamos el sistema de almacenamiento NFS como directorio del servicio Apache-PHP que se ejecuta en cada uno de los contenedores, esto es para albergar las páginas webs que se publican. Por tanto, es un contenido que debe estar siempre disponible.

```

TFMNas [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@nas:/raid/www# cd ..
root@nas:/raid# ls -l
total 5133
-rw-r--r-- 1 root root 5242880 Jul  4 18:25 cincomegas
drwx----- 2 root root 12288 Jul  4 18:20 lost+found
drwxr-xr-x 2 root root 1024 Sep  4 02:58 www
root@nas:/raid# cd www
root@nas:/raid/www# ls -l
total 3
-rw---r-x 1 root root 512 Sep  4 02:25 index1.php
-rw-r--r-- 1 root root 287 Jul 29 01:05 index.php
-rw---r-x 1 root root 473 Sep  4 02:26 pi.php
-rw-r--r-- 1 root root  0 Jul 18 10:33 prueba.txt
root@nas:/raid/www#
    
```

Figura 122 - Contenido Directorio NFS

Dado que es un sistema basado en RAID5, vamos a evaluar que podemos acceder al directorio desde uno de los nodos clientes, aunque un disco falle.

Desde el nodo cluster2, observamos que se tiene montado el sistema **nfs** en el directorio **/nfs**:

```

File Machine View Input Devices Help
root@cluster2:~# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=468956k,nr_inodes=117239,mode=755,inode64)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,noexec,relatime,size=98388k,mode=755,inode64)
/dev/sda1 on / type ext4 (rw,relatime)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k,inode64)
cgroup2 on /sys/fs/cgroup type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate,memory_recursivepr
rot)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=30,pgrp=1,timeout=0,minproto=5,max
proto=5,direct,pipe_ino=12736)
debugfs on /sys/kernel/debug type debugfs (rw,nosuid,nodev,noexec,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
mqueue on /dev/mqueue type mqueue (rw,nosuid,nodev,noexec,relatime)
tracefs on /sys/kernel/tracing type tracefs (rw,nosuid,nodev,noexec,relatime)
configfs on /sys/kernel/config type configfs (rw,nosuid,nodev,noexec,relatime)
ramfs on /run/credentials/systemd-sysctl.service type ramfs (ro,nosuid,nodev,noexec,relatime,mode=70
0)
ramfs on /run/credentials/systemd-sysusers.service type ramfs (ro,nosuid,nodev,noexec,relatime,mode=
700)
fusectl on /sys/fs/fuse/connections type fusectl (rw,nosuid,nodev,noexec,relatime)
ramfs on /run/credentials/systemd-tmpfiles-setup-dev.service type ramfs (ro,nosuid,nodev,noexec,rela
time,mode=700)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,nosuid,nodev,noexec,relatime)
ramfs on /run/credentials/systemd-tmpfiles-setup.service type ramfs (ro,nosuid,nodev,noexec,relatime
,mode=700)
rpc_pipefs on /run/rpc_pipefs type rpc_pipefs (rw,relatime)
nas:/ on /nfs type nfs4 (rw,relatime,vers=4.2,rsize=131072,wsize=131072,namlen=255,hard,proto=tcp,tli
neo=600,retrans=2,sec=sys,clientaddr=10.0.100.2,local_lock=none,addr=10.0.100.100)
gluster1:gv1 on /gluster/gfs-d type fuse.gluusterfs (rw,relatime,user_id=0,group_id=0,default_permiss
ions,allow_other,max_read=131072,_netdev)
gluster1:gv0 on /gluster/gfs-r type fuse.gluusterfs (rw,relatime,user_id=0,group_id=0,default_permiss
ions,allow_other,max_read=131072,_netdev)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,size=98384k,nr_inodes=24596,mode=700,inode
64)
    
```

Figura 123 – Punto de Montaje sistema NFS

El contenido de dicho directorio se observa en la siguiente figura, nótese el punto de montaje **/nfs/raid/www**, que es el directorio utilizado por el servicio Apache-PHP instalado en los contenedores:

```
File Machine View Input Devices Help
root@cluster2:~# cd /nfs/
root@cluster2:/nfs# ls -l
total 1
drwxr-xr-x 4 root root 1024 Jul 18 00:51 raid
root@cluster2:/nfs# cd raid/
root@cluster2:/nfs/raid# ls -l
total 5133
-rw-r--r-- 1 root root 5242880 Jul  4 18:25 cincomegas
drwx----- 2 root root  12288 Jul  4 18:20 lost+found
drwxr-xr-x 2 root root  1024 Sep  4 02:58 www
root@cluster2:/nfs/raid# cd www
root@cluster2:/nfs/raid/www# ls -l
total 3
-rw---r-x 1 root root 512 Sep  4 02:25 index1.php
-rw-r--r-- 1 root root 287 Jul 29 01:05 index.php
-rw---r-x 1 root root 473 Sep  4 02:26 pi.php
-rw-r--r-- 1 root root  0 Jul 18 10:33 prueba.txt
root@cluster2:/nfs/raid/www# _
```

Figura 124 – Punto de Montaje sistema NFS

Vamos a hacer que un dispositivo del RAID falle, lo quitamos y lo añadimos otra vez, después de cada orden comprobaremos el resultado observando el contenido del **archivo /proc/mdstat** en el servidor NFS. Adicional comprobaremos el acceso desde el nodo cliente cluster2.

```
File Machine View Input Devices Help
root@nas:/raid/www# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4] [linear] [multipath] [raid0] [raid1] [raid10]
md0 : active raid5 sdb1[0] sdd1[1] sdc1[3]
      403456 blocks super 1.2 level 5, 512k chunk, algorithm 2 [3/3] [UUU]

unused devices: <none>
root@nas:/raid/www# mdadm /dev/md0 --fail /dev/sdc1
[102970.883623] md/raid:md0: Disk failure on sdc1, disabling device.
[102970.883808] md/raid:md0: Operation continuing on 2 devices.
mdadm: set /dev/sdc1 faulty in /dev/md0
root@nas:/raid/www# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4] [linear] [multipath] [raid0] [raid1] [raid10]
md0 : active raid5 sdb1[0] sdd1[1] sdc1[3](F)
      403456 blocks super 1.2 level 5, 512k chunk, algorithm 2 [3/2] [UU_]

unused devices: <none>
root@nas:/raid/www#
```

Figura 125 – Falla de disco en RAID

El dispositivo /dev/sdc1 ha fallado y se evidencia con **la etiqueta “F”** al observar el contenido del archivo /proc/mdstat. El RAID ha quedado funcionando con dos discos.



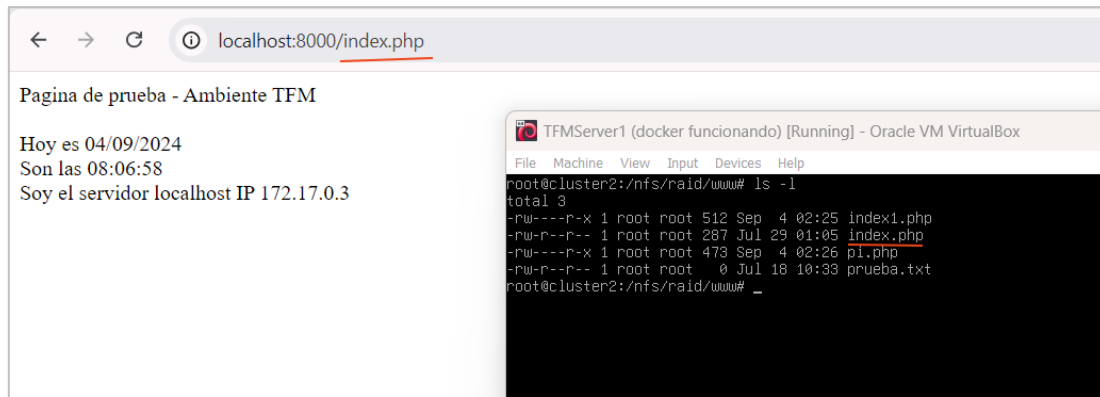


Figura 126 – Acceso a repositorio NFS con falla en RAID

A pesar de eso, se tiene acceso al repositorio y el servicio web se mantiene operativo. Ahora, vamos a reponer el disco y comprobamos resultados finales.

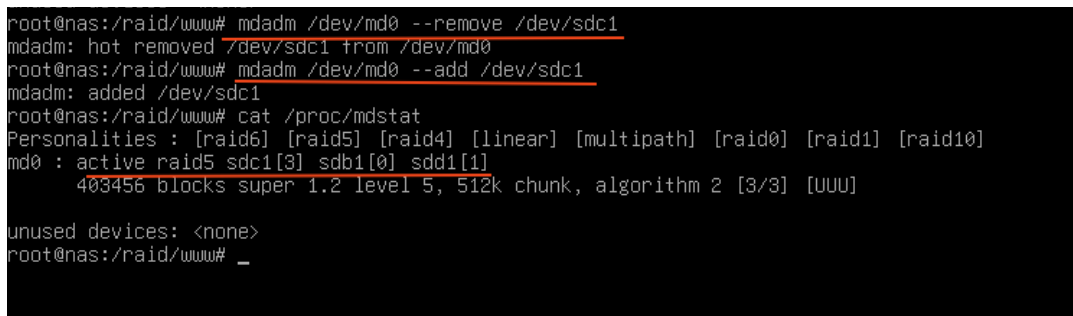


Figura 127 – Remover y Agregar Dispositivo al RAID

El RAID nuevamente tiene 3 discos funcionando con normalidad y los clientes tienen acceso al repositorio.

Por otra parte, para evaluar el rendimiento y la capacidad del servidor NAS dentro de nuestro clúster, recordemos que lo configuramos con dos interfaces de red que funcionan bajo un esquema de **link aggregation**. Este enfoque permite combinar el ancho de banda de ambas interfaces, optimizando la velocidad de transferencia de datos y aumentando la resiliencia ante posibles fallos de red.

En la siguiente prueba verificaremos la continuidad del servicio en caso de que una de las interfaces de red sufra una desconexión o falla, garantizando así que el almacenamiento compartido sigue estando **disponible** para todos los nodos del clúster, aunque es probable que se afecte el rendimiento cuando se trate de acceder a una página web ahí almacenada. Esta última parte queda fuera del alcance de la prueba.

En nuestro ambiente, desconectaremos una interfaz y comprobaremos la disponibilidad del servidor desde uno de los nodos clientes NFS, por ejemplo, cluster2:

```

File Machine View Input Devices Help
root@nas:/raid/www# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc fq_codel master bond0 state UP group default qlen 1000
    link/ether 08:00:27:b2:5d:43 brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc fq_codel master bond0 state UP group default qlen 1000
    link/ether 08:00:27:b2:5d:43 brd ff:ff:ff:ff:ff:ff
4: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 08:00:27:b2:5d:43 brd ff:ff:ff:ff:ff:ff
    inet 10.0.100.24/24 brd 10.0.100.255 scope global bond0
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feb2:5d43/64 scope link
        valid_lft forever preferred_lft forever
root@nas:/raid/www# [107311.784226] bond0: (slave enp0s8): invalid new link 1 on slave

ping (docker funcionando) [Running] - Oracle VM VirtualBox
View Input Devices Help
jm nas.cluster (10.0.100.100): icmp_seq=88 ttl=64 time=0.823 ms
jm nas.cluster (10.0.100.100): icmp_seq=89 ttl=64 time=1.33 ms
jm nas.cluster (10.0.100.100): icmp_seq=90 ttl=64 time=0.838 ms
jm nas.cluster (10.0.100.100): icmp_seq=91 ttl=64 time=0.713 ms
jm nas.cluster (10.0.100.100): icmp_seq=92 ttl=64 time=0.992 ms
jm nas.cluster (10.0.100.100): icmp_seq=93 ttl=64 time=0.913 ms
jm nas.cluster (10.0.100.100): icmp_seq=94 ttl=64 time=1.04 ms
jm nas.cluster (10.0.100.100): icmp_seq=95 ttl=64 time=0.864 ms
jm nas.cluster (10.0.100.100): icmp_seq=96 ttl=64 time=1.34 ms
jm nas.cluster (10.0.100.100): icmp_seq=97 ttl=64 time=1.85 ms
jm nas.cluster (10.0.100.100): icmp_seq=98 ttl=64 time=0.797 ms
jm nas.cluster (10.0.100.100): icmp_seq=99 ttl=64 time=0.560 ms
jm nas.cluster (10.0.100.100): icmp_seq=100 ttl=64 time=1.01 ms
jm nas.cluster (10.0.100.100): icmp_seq=101 ttl=64 time=2.00 ms
jm nas.cluster (10.0.100.100): icmp_seq=102 ttl=64 time=1.01 ms
jm nas.cluster (10.0.100.100): icmp_seq=103 ttl=64 time=0.384 ms
jm nas.cluster (10.0.100.100): icmp_seq=104 ttl=64 time=1.36 ms
jm nas.cluster (10.0.100.100): icmp_seq=105 ttl=64 time=0.947 ms
jm nas.cluster (10.0.100.100): icmp_seq=106 ttl=64 time=1.03 ms
jm nas.cluster (10.0.100.100): icmp_seq=107 ttl=64 time=0.985 ms
jm nas.cluster (10.0.100.100): icmp_seq=108 ttl=64 time=0.969 ms
jm nas.cluster (10.0.100.100): icmp_seq=109 ttl=64 time=0.999 ms
jm nas.cluster (10.0.100.100): icmp_seq=110 ttl=64 time=0.848 ms
jm nas.cluster (10.0.100.100): icmp_seq=111 ttl=64 time=0.635 ms
jm nas.cluster (10.0.100.100): icmp_seq=112 ttl=64 time=1.14 ms
jm nas.cluster (10.0.100.100): icmp_seq=113 ttl=64 time=1.38 ms
jm nas.cluster (10.0.100.100): icmp_seq=114 ttl=64 time=1.02 ms
64 bytes from nas.cluster (10.0.100.100): icmp_seq=115 ttl=64 time=1.03 ms
64 bytes from nas.cluster (10.0.100.100): icmp_seq=116 ttl=64 time=0.729 ms
64 bytes from nas.cluster (10.0.100.100): icmp_seq=117 ttl=64 time=0.747 ms
64 bytes from nas.cluster (10.0.100.100): icmp_seq=118 ttl=64 time=0.696 ms
64 bytes from nas.cluster (10.0.100.100): icmp_seq=119 ttl=64 time=0.517 ms
64 bytes from nas.cluster (10.0.100.100): icmp_seq=120 ttl=64 time=0.926 ms
64 bytes from nas.cluster (10.0.100.100): icmp_seq=121 ttl=64 time=0.621 ms
64 bytes from nas.cluster (10.0.100.100): icmp_seq=122 ttl=64 time=0.992 ms
64 bytes from nas.cluster (10.0.100.100): icmp_seq=123 ttl=64 time=1.72 ms
    
```

Figura 128 – Disponibilidad servidor NFS

Con esto hemos verificado los dos servicios importantes que brinda el nodo NAS con servicio NFS basado en RAID y link aggregation para disponibilidad de red.

## 4.2 EVALUACIÓN DE PRESTACIONES

En esta sección de nuestro trabajo fin de máster, nos enfocaremos en la evaluación de las prestaciones del clúster de alta disponibilidad (HA) y reparto de carga que hemos configurado. La finalidad de estas pruebas es medir la eficiencia y capacidad del clúster para manejar diferentes tipos de cargas y escenarios, asegurando así que el sistema pueda satisfacer las demandas de los usuarios en un entorno real.

Nuestro clúster está compuesto por tres equipos front-end, de los cuales uno se encuentra activo y los otros dos en estado pasivo, listos para asumir el control en caso de fallo del nodo activo. Además, el clúster incluye nueve equipos back-end, implementados como contenedores que se ejecutan sobre tres servidores físicos. Estos contenedores son los encargados de atender las solicitudes entrantes, proporcionando la capacidad de respuesta necesaria para mantener un servicio continuo y eficiente.

Para realizar las pruebas de rendimiento, hemos seleccionado Apache Benchmark (ab), una herramienta robusta y ampliamente utilizada en la industria para medir el rendimiento de servidores web. Apache Benchmark nos permitirá simular múltiples peticiones concurrentes a nuestro sistema, proporcionando datos precisos sobre la capacidad de manejo de tráfico del clúster.

En estas pruebas, evaluaremos tres tipos de páginas web alojadas en nuestro clúster en el sistema de almacenamiento NFS basado en RAID5:

1. **Página Estática:** Una página HTML simple sin contenido dinámico.
2. **Página Dinámica:** Una página que incluye un cálculo para mostrar resultados, específicamente una página que realiza el cálculo del valor de PI.
3. **Página Estática con Imagen:** Una página HTML simple que incluye una imagen para evaluar el manejo de recursos multimedia.

Los principales indicadores que obtendremos de estas pruebas serán:

- ▶ **Peticiones por Segundo (Requests per Second):** Este indicador nos mostrará la cantidad de peticiones que nuestro clúster puede manejar por segundo, dando una idea clara de la capacidad del sistema bajo diferentes tipos de carga.
- ▶ **Tiempo de Respuesta (Response Time):** Este indicador medirá el tiempo que tarda el sistema en responder a una solicitud, proporcionando una visión sobre la eficiencia del clúster en términos de velocidad de respuesta.

Al analizar estos indicadores, podremos obtener una comprensión de las capacidades y limitaciones del clúster, así como identificar áreas de mejora para optimizar su rendimiento. Las pruebas nos permitirán validar la eficacia de nuestra configuración de alta disponibilidad y reparto de carga, asegurando que el sistema esté bien preparado para enfrentar los desafíos del entorno real.

### 4.2.1 Resultados para página web estática

En esta primera prueba, evaluaremos el rendimiento de una página web estática, esto es **index.php**.

```
GNU nano 7.2 index.php
<html>
<body>
Web Server - Ambiente TFM
<?php
echo "<br><br>";
echo "Hoy es " . date ("d/m/Y") . "<br>";
echo "Son las " . date ("h:i:s") . "<br>";
echo "Soy el servidor con VIP " . $_SERVER['SERVER_NAME'] . " y mi RIP es " . $_SERVER['SERVER_ADDR'] . "<br><br>";
?>
</body>
</html>
```

Figura 129 – Página web estática

```
Web Server - Ambiente TFM
Hoy es 07/09/2024
Son las 12:00:06
Soy el servidor con VIP 192.168.1.200 y mi RIP es 172.17.0.2
```

Figura 130 – Muestra de página web estática

Esta página web a la que acceden los clientes se encuentra en el sistema de almacenamiento NFS, ubicado en el servidor NAS:

```
root@cluster1:/nfs/raid/www# ls -l
total 204
-rw-r--r-x 1 root root 659 Sep 7 00:53 imagen.html
-rw-r--r-x 1 root root 204800 Sep 7 00:53 imagen.jpg
-rw----r-x 1 root root 512 Sep 5 02:22 index1.php
-rw-r--r-- 1 root root 298 Sep 5 02:32 index.php
-rw----r-x 1 root root 473 Sep 4 02:26 pi.php
root@cluster1:/nfs/raid/www#
```

Figura 131 – Repositorio Apache2 en NFS

En la figura se puede apreciar el directorio raíz del servicio Apache2+PHP que se ejecuta en cada uno de los nodos contenedores.

Para la ejecución del benchmark, utilizaremos el software Apache Benchmark (comando `ab`), que nos ayudará a medir las prestaciones del clúster. Ejecutaremos esta prueba desde el **nodo maestro** utilizando los siguientes comandos:

```
apt-get install apache2-utils
```

```
ab -n 1000 -c 3 http://192.168.1.200/index.php
```

donde el parámetro `-n` indica el número de requerimientos a desarrollar para la sesión de benchmarking y `-c` el número de peticiones simultáneas (por default es una petición por tiempo). Finalmente, indicamos la página web que vamos a evaluar. En nuestro caso utilizaremos la publicada en el puerto 80, que corresponde al reparto de peticiones con HAProxy en modo TCP, por lo que las solicitudes HTTP se distribuirán directamente entre todos los nodos contenedores.

En la siguiente figura se muestra la salida del benchmark y se resaltan los parámetros a evaluar, requerimientos por segundo y tiempos por requerimiento.

```

root@cluster1:~# ab -n 1000 -c 1 http://192.168.1.200:8000/index.php
This is ApacheBench, Version 2.3 <$Revision: 1913912 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.1.200 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software:      Apache/2.4.59
Server Hostname:     192.168.1.200
Server Port:         8000

Document Path:       /index.php
Document Length:     179 bytes

Concurrency Level:   1
Time taken for tests: 6.469 seconds
Complete requests:   1000
Failed requests:     0
Total transferred:   466992 bytes
HTML transferred:    179000 bytes
Requests per second: 154.57 [#./sec] (mean)
Time per request:    6.469 [ms] (mean)
Time per request:    6.469 [ms] (mean, across all concurrent requests)
Transfer rate:       70.49 [kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median  max
Connect:     0    0  0.2      0    8
Processing:  2    6  3.4      5   34
Waiting:     2    6  3.4      5   34
Total:       2    6  3.4      5   34

Percentage of the requests served within a certain time (ms)
 50%    5
 66%    7
 75%    9
 80%    9
 90%   11
 95%   12
 98%   15
 99%   17
100%   34 (longest request)
root@cluster1:~#

```

Figura 132 – Ejecución de Apache Benchmark

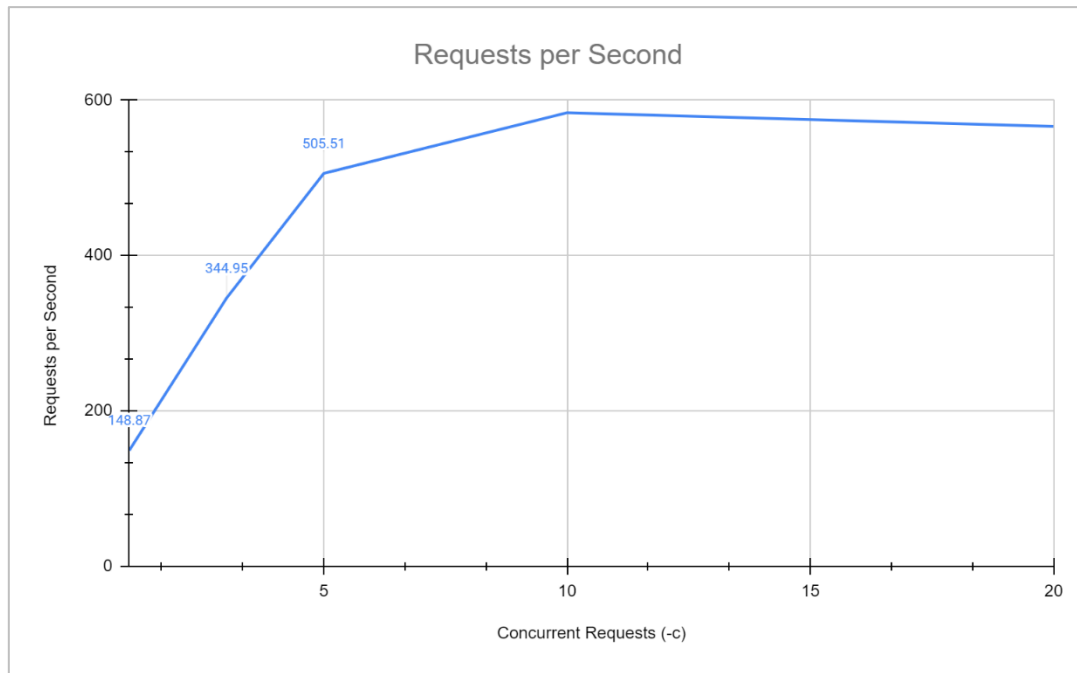
En la siguiente tabla se muestran los resultados de las evaluaciones, con diferentes valores de tareas concurrentes y número de requerimientos:

Total Requests (-n)	Concurrent Requests (-c)	Requests per Second	Time per Request (ms)
1000	1	165.58	6.04
	3	326.47	9.189
	5	487.88	10.25
	10	499.89	20
	20	517.86	38.62
5000	1	157.03	6.368

	3	335.55	8.94
	5	498.76	10.025
	10	530.82	18.84
	20	562.81	35.54
<b>10000</b>	1	148.87	6.717
	3	344.95	8.697
	5	505.51	9.89
	10	583.71	17.13
	20	565.92	35.34
<b>20000</b>	1	142.02	7.041
	3	340.32	8.814
	5	495.91	10.08
	10	233.89	42.76
	20	567.18	35.26
<b>25000</b>	1	154.9	6.456
	3	188.14	15.945
	5	322.83	15.49
	10	318.41	31.41
	20	191.38	104.5
<b>30000</b>	1	143.88	6.95
	3	180.77	16.596
	5	213.77	23.39
	10	220.58	45.34
	20	213.74	93.58
<b>40000</b>	1	156.52	6.389
	3	231.62	12.951
	5	196.06	25.505
	10	253.94	39.38
	20	259.74	77
<b>50000</b>	1	143.14	6.986
	3	186.69	16.068
	5	191.23	26.145
	10	186.78	53.54
	20	225.77	88.58

Tabla 11 - Resultados Evaluación AB Página Estática

Los gráficos que se generan a partir de esta tabla para los indicadores de “Request per Second” y “Time per Request (ms)” son:



**Figura 133 – Request per Second Pàgina Estàtica con n=10000**

Este gráfico muestra cómo varía el número de solicitudes que el clúster puede manejar por segundo a medida que se incrementa el número de solicitudes concurrentes y la cantidad total de solicitudes.

A medida que aumenta el número de solicitudes concurrentes (-c), el número de solicitudes por segundo (Requests per Second) generalmente también aumenta, alcanzando su punto máximo alrededor de  $-c = 20$ .

El valor máximo de Request per Second se observa en las ejecuciones con  $-c = 20$  y  $-n$  de 20000 solicitudes, alcanzando 567.18 solicitudes por segundo.

Con un número muy alto de peticiones (e.g.,  $-n=50000$ ), el Requests per Second comienza a disminuir con un alto número de solicitudes concurrentes. Esto sugiere un punto de saturación del clúster.



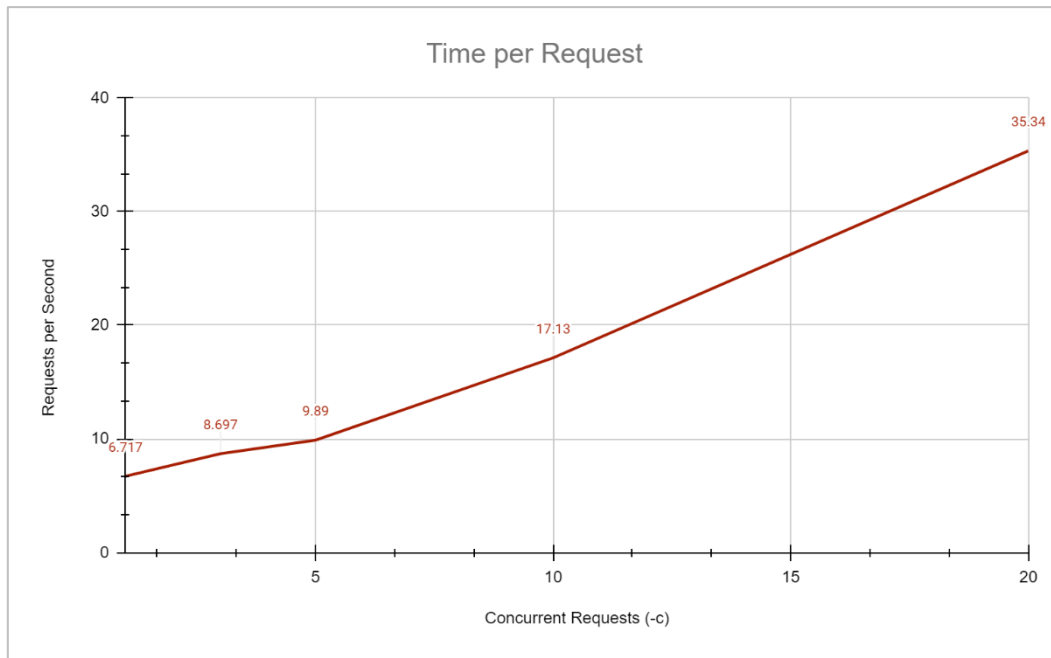


Figura 134 – Time per Request Pàgina Estàtica con n=10000

Este gràfico muestra el tiempo promedio por solicitud a medida que se incrementa el número de solicitudes concurrentes.

El tiempo necesario para procesar una petición es de 6.717 ms.

El tiempo permanece estable para los niveles más bajos de concurrencia, aumentando exponencialmente. Esto sugiere nuevamente la saturación del servidor.

#### 4.2.2 Resultados para página web dinámica

A continuación, evaluaremos el rendimiento de la página web dinámica, **pi.php**, que se encuentra en el sistema de almacenamiento NFS y realiza el cálculo del valor de PI:

```

GNU nano 7.2                                pi.php
<html>
<body>
<?php
$start=microtime(true);

$area=0.0;

$n=$_GET["n"];
// $n=100000000;

for ($i=0; $i<$n; $i++)
{
    $x=($i+0.5)/$n;
    $area=$area+4.0/(1.0+$x*$x);
}
$result=$area/$n;

$end=microtime(true);
$exectime=$end-$start;

echo "<br>Calculo de PI<br><br>";
printf ("La cte. PI con n= %d es igual a %f<br>", $n, $result);
printf ("Tiempo de ejecucion= %.5f segundos<br>",$exectime);
printf ("<br>El servidor es %s<br>", $_SERVER['SERVER_ADDR']);
?>
</body>
</html>
    
```

Figura 135 – Pagina web Dinámica

```

root@cluster1:/nfs/raid/www# w3m http://192.168.1.200/pi.php?n=10000
Calculo de PI
La cte. PI con n= 10000 es igual a 3.141593
Tiempo de ejecucion= 0.00067 segundos

El servidor es 172.17.0.2
    
```

Figura 136 – Muestra de página web dinámica

Antes de someter la página a evaluaciones con Apache Benchmark, es crucial determinar un valor adecuado para el número de intervalos utilizados en el cálculo del valor de PI. Dado que el tiempo de ejecución de la página web dinámica depende de este parámetro, debemos identificar un número de intervalos que nos permita alcanzar un tiempo de ejecución aceptable. Esto garantizará que las pruebas de rendimiento reflejen de manera precisa el comportamiento real de la página bajo condiciones normales de uso.

En la siguiente tabla se muestra el tiempo de ejecución requerido para un numero de intervalos dado:

Intervalos (n)	URI	Tiempo de ejecución (seg)
100.000	pi.php?n=100000	0.00940
1'000.000	pi.php?n=1000000	0.07770
5'000.000	pi.php?n=5000000	0.39048
10'000.000	pi.php?n=10000000	0.83823

Tabla 12 - Intervalos para uso de página web dinámica

```

root@cluster1:/nfs/raid/www# ab -n 1000 -c 1 http://192.168.1.200/pi.php?n=100000
This is ApacheBench, Version 2.3 <$Revision: 1913912 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.1.200 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software: Apache/2.4.61
Server Hostname: 192.168.1.200
Server Port: 80

Document Path: /pi.php?n=100000
Document Length: 177 bytes

Concurrency Level: 1
Time taken for tests: 21.561 seconds
Complete requests: 1000
  
```

Figura 137 – Evaluación de AB en páginas web dinámicas

Para evaluar el rendimiento de esta página web dinámica, vamos a elegir un valor de 100,000 intervalos para el cálculo. Este valor permite observar el comportamiento de los indicadores de “Requests per Second” y “Time per Request” sin impactar excesivamente en el tiempo de ejecución. En la siguiente tabla se muestran los resultados de las evaluaciones con diferentes valores de tareas concurrentes y número de requerimientos:

Total Requests (-n)	Concurrent Requests (-c)	Requests per Second	Time per Request (ms)
1000	1	46.38	21.561
	3	63.97	46.896
	5	82.41	60.67
	10	85.57	116.86
	20	84.12	237.76

<b>5000</b>	1	46.63	21.445
	3	66.34	45.225
	5	85.2	58.685
	10	78.22	127.85
	20	87.39	228.84
<b>10000</b>	1	44.38	22.533
	3	68.07	44.07
	5	83.84	59.635
	10	81.92	122.07
	20	79.88	250.36
<b>20000</b>	1	45.95	21.761
	3	71.3	42.078
	5	84.89	58.9
	10	85.67	116.73
	20	81.86	244.32
<b>30000</b>	1	47.3	21.144
	3	68.13	44.037
	5	84.2	59.38
	10	82.31	121.49
	20	77.35	258.56

Tabla 13 - Resultados Evaluación AB Página Dinámica

Los gráficos que se generan a partir de esta tabla para los indicadores de “Request per Second” y “Time per Request (ms)” son:

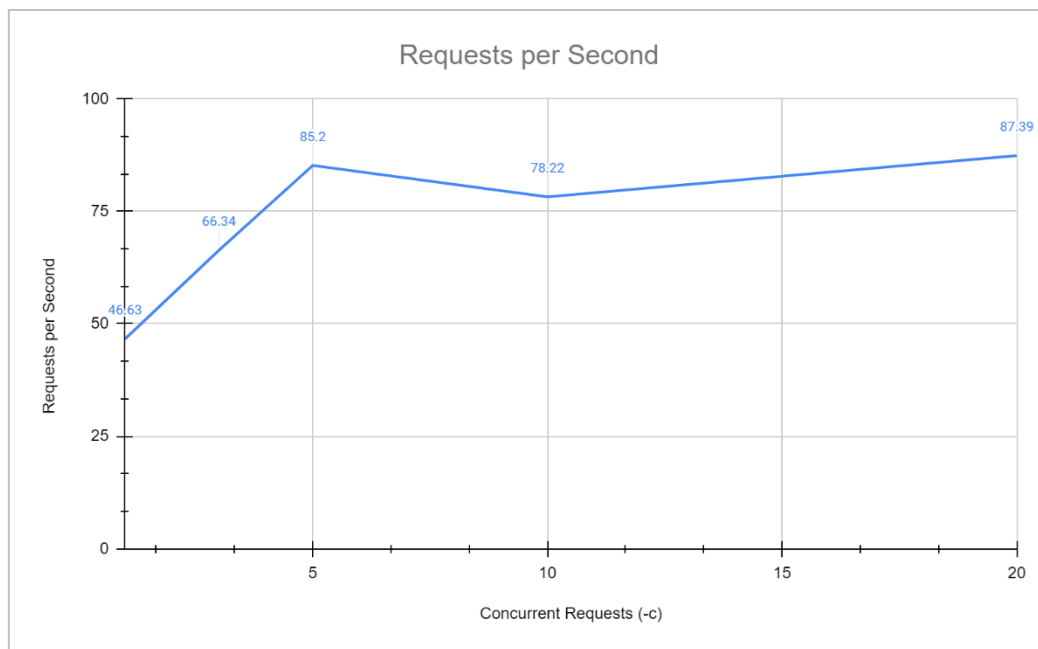


Figura 138 – Request per Second Página Dinámica con n=5000

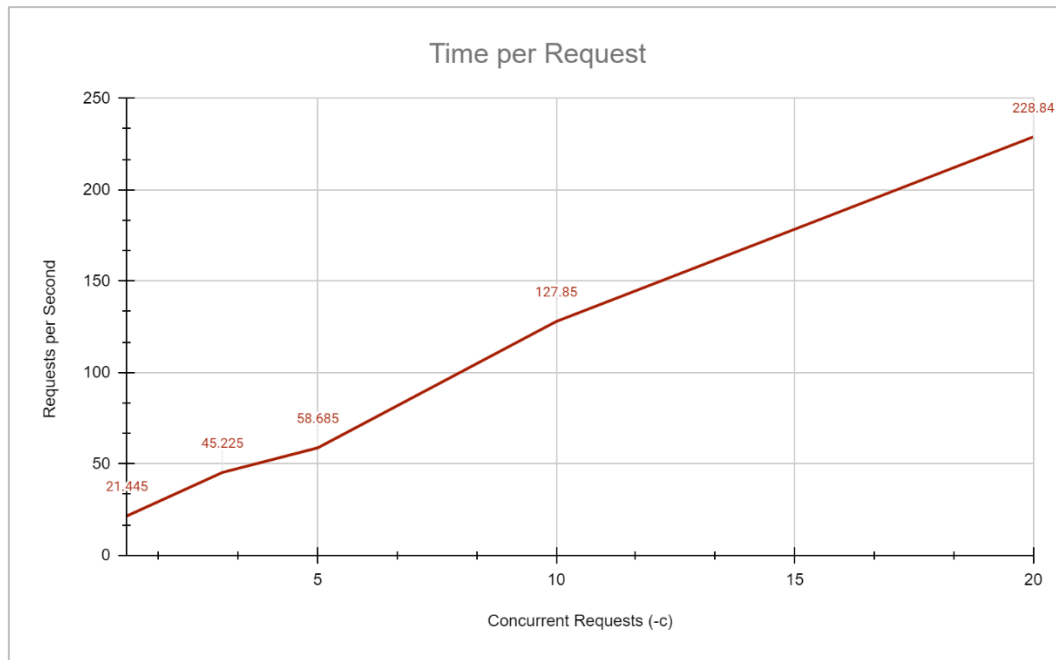


Figura 139 – Time per Request Página Dinámica con n=5000

En general, podemos observar que, a medida que aumentan las peticiones concurrentes, el número de Requests per Second aumenta (figura 138), alcanzando el valor máximo en torno a las 90 peticiones por segundo.

El time per Request permanece estable en torno a los 22 ms, hasta que aumenta la carga y se dispara.

#### 4.2.3 Resultados para página web estática con imagen

En esta prueba, evaluaremos el rendimiento de una página web estática, esto es **imagen.html**, con una imagen que pesa 10 MB, para valorar el manejo de recursos multimedia.

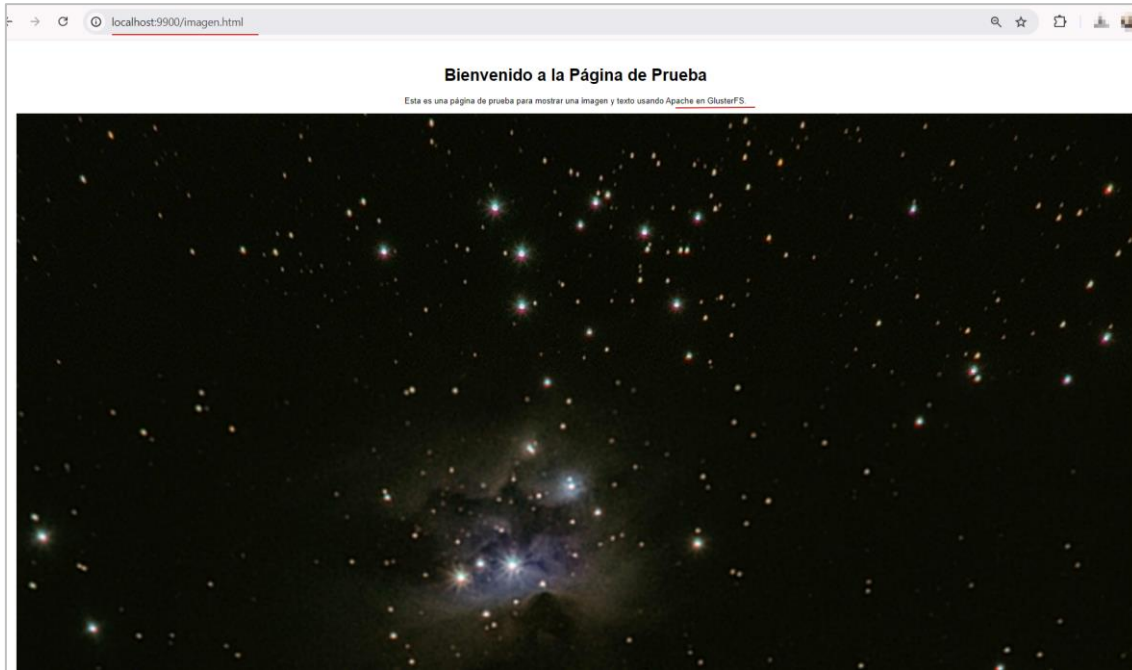


Figura 140 – Muestra de página web estática con imagen

La página web a la que acceden los clientes se encuentra en el sistema de almacenamiento GlusterFS, en el sistema de archivos replicado:

```
root@cluster1:/gluster/gfs-r/www# ls -l
total 403
-rw-r--r-x 1 root root 672 Sep 8 15:14 imagen.html
-rw-r--r-x 1 root root 204800 Sep 6 00:54 imagen.jpg
-rw----r-x 1 root root 512 Sep 6 17:25 index1.php
-rw-r--r-- 1 root root 298 Sep 6 17:24 index.php
-rw----r-x 1 root root 204800 Sep 8 14:24 lion.jpg
-rw----r-x 1 root root 473 Sep 6 17:24 pi.php
root@cluster1:/gluster/gfs-r/www#
```

Figura 141 – Repositorio Apache2 en GlusterFS

En la figura se puede apreciar el directorio raíz del servicio Apache2+PHP que se ejecuta en tres de los doce nodos contenedores del ambiente de cluster.

En la siguiente tabla se muestran los resultados de las evaluaciones, con diferentes valores de tareas concurrentes y número de requerimientos:

Total Requests (-n)	Concurrent Requests (-c)	Requests per Second	Time per Request (ms)
<b>1000</b>	1	67.41	14.836
	3	154.91	19.365
	5	138.26	36.165
	10	102.75	97.32
	20	93.7	213.44
<b>5000</b>	1	88.58	11.289
	3	153.89	19.494
	5	100.4	49.8
	10	106.97	93.48
	20	105.93	188.8
<b>10000</b>	1	49.27	20.297
	3	87.38	34.332
	5	90.98	54.96
	10	95.88	104.29
	20	94.81	210.94
<b>20000</b>	1	89.6	11.16
	3	115.97	25.869
	5	103.65	48.24
	10	103.34	96.77
	20	107.39	186.22
<b>30000</b>	1	70.61	14.163
	3	110.82	27.069
	5	103.14	48.48
	10	107.8	92.77
	20	106.23	188.28

Tabla 14 - Resultados Evaluación Página Estática con Imagen

Los gráficos que se generan a partir de esta tabla para los indicadores de “Request per Second” y “Time per Request (ms)” son:

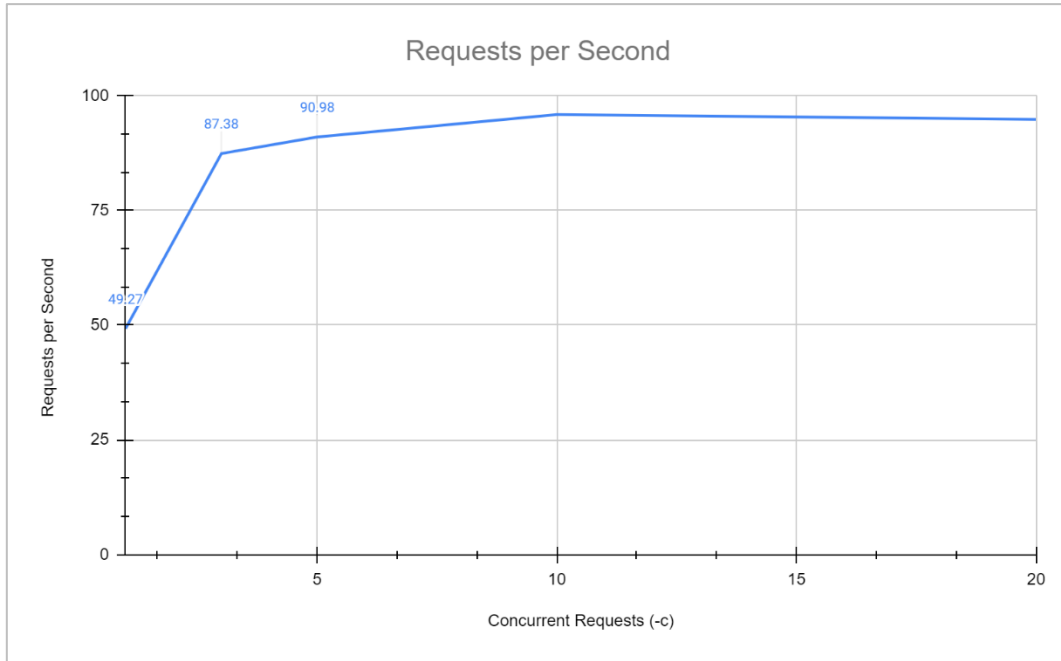


Figura 142 – Request per Second Página Estática con Imagen n=10000

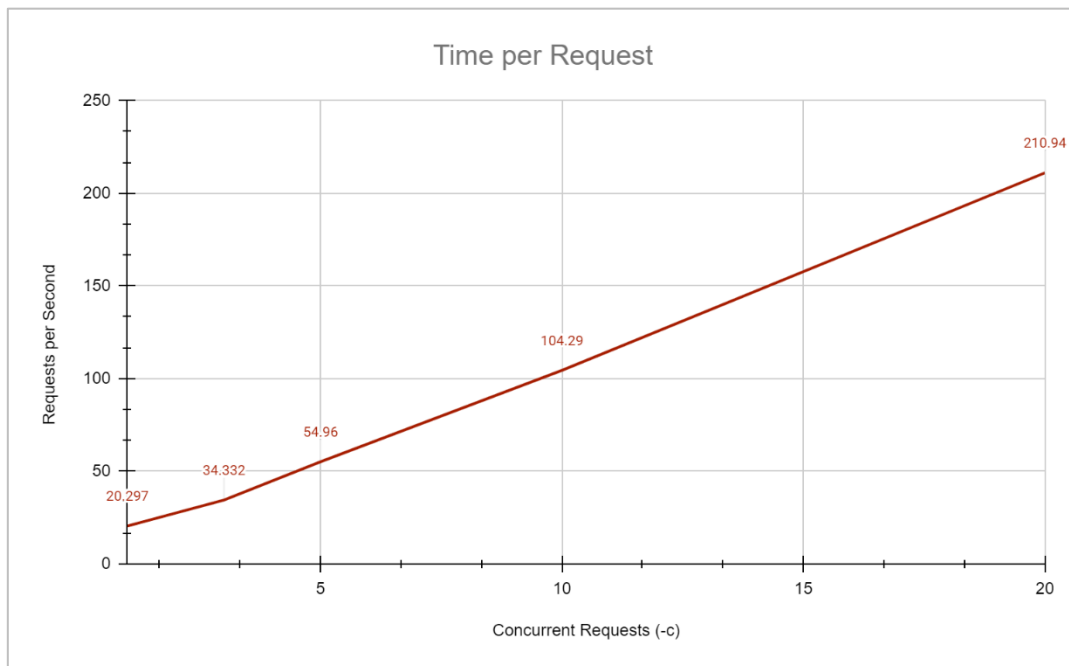


Figura 143 – Time per Request Página Estática con Imagen n=10000

En este caso, el clúster alcanza casi 100 peticiones por segundo con un tiempo mínimo en torno a los 20 ms.



Tras realizar y analizar las pruebas de desempeño con Apache Benchmark en tres tipos de páginas web (contenido estático, contenido dinámico, y contenido estático con una imagen de 10 MB), podemos indicar lo siguiente sobre el rendimiento del clúster con HA y reparto de carga:

- ▶ La página estática sin imágenes es la más eficiente en términos de rendimiento. El clúster de computadores maneja la carga de manera muy efectiva, proporcionando tiempos de respuesta rápidos y un alto throughput.
- ▶ El contenido dinámico impone una carga significativa sobre el clúster, resultando en menor throughput y mayores tiempos de respuesta. Aunque el clúster maneja la carga, el rendimiento es menos eficiente comparado con las páginas estáticas.
- ▶ La inclusión de una imagen grande en una página estática introduce una carga adicional, pero el clúster aún maneja la carga de manera eficiente. El rendimiento es superior al de la página dinámica, mostrando que el clúster puede manejar contenido estático pesado mejor que contenido dinámico computacionalmente intensivo. Es necesario hacer adecuaciones al clúster para mejorar el rendimiento para páginas con contenido dinámico.

Como comentario adicional, debemos recordar que todas estas pruebas se han realizado en un entorno virtualizado, siendo la máquina anfitrión (Host) un ordenador portátil con sistema operativo Windows 11 equipado con un procesador 12th Gen Intel(R) Core(TM) i7-1255U, 1700 Mhz, 10 Core(s), 12 Logical Processor(s), por lo que los resultados obtenidos están evidentemente limitados por la configuración empleada.

## 5 CONCLUSIONES Y RECOMENDACIONES

El proyecto tuvo como objetivo principal diseñar, implementar y evaluar un sistema de clúster de computadores que garantizara la alta disponibilidad y el equilibrado de carga para un servidor web. Para alcanzar este objetivo, se plantearon y lograron los siguientes objetivos específicos:

1. Diseñar una arquitectura de clúster que asegure la alta disponibilidad, implementando redundancia y mecanismos de conmutación por error para mantener el sistema siempre operativo.
2. Implementar y configurar un sistema de equilibrado de carga que reparta el tráfico de manera eficiente entre los diferentes nodos del clúster, optimizando así el uso de los recursos y mejorando la respuesta del sistema.
3. Configurar un sistema de archivos distribuido y paralelo que permita gestionar el almacenamiento de datos de manera eficiente dentro del clúster, garantizando que los datos estén siempre disponibles.
4. Evaluar el rendimiento del sistema a través de pruebas que midan aspectos clave como la disponibilidad y la capacidad de manejo de tráfico bajo distintas condiciones.

A continuación, se presentan las conclusiones detalladas y las recomendaciones asociadas a cada uno de estos objetivos:

### ► **Diseño de la Arquitectura de Clúster para Alta Disponibilidad**

La configuración de HAProxy en combinación con Corosync+Pacemaker y HAProxy con Keepalived demostró ser eficaz para asegurar la alta disponibilidad del clúster. Las configuraciones garantizaron la conmutación por error automática en caso de fallos, manteniendo el servicio disponible en todo momento.

#### **Recomendaciones:**

- Continuar utilizando HAProxy junto con Corosync+Pacemaker o Keepalived para asegurar una solución robusta y confiable para la alta disponibilidad.
- Implementar monitoreos adicionales para detectar y resolver fallos de manera proactiva.

### ► **Implementación y Configuración de un Sistema de Equilibrado de Carga**

El sistema de equilibrado de carga configurado logró repartir eficientemente el tráfico entre los diferentes nodos del clúster, optimizando el uso de recursos y mejorando la respuesta del sistema. El uso de HAProxy como balanceador de carga demostró ser efectivo para gestionar el tráfico y asegurar un rendimiento estable.

**Recomendaciones:**

- Continuar utilizando HAProxy como balanceador de carga debido a su eficacia y flexibilidad.
- Realizar ajustes periódicos en las políticas de balanceo para adaptarse a cambios en las cargas de trabajo y optimizar el rendimiento del sistema.

**► Configuración de un Sistema de Archivos Distribuido y Paralelo**

La implementación de NFS y Gluster permitió gestionar el almacenamiento de datos de manera eficiente dentro del clúster, asegurando que los datos fueran accesibles y consistentes en todo momento. Gluster, en particular, mostró una mejor capacidad de escalabilidad.

**Recomendaciones:**

- Optar por Gluster en escenarios donde se requiera una mayor escalabilidad y flexibilidad en el almacenamiento.
- Implementar políticas de monitoreo y mantenimiento regular para asegurar la integridad y disponibilidad de los datos.

**► Evaluación del Rendimiento del Sistema**

Se realizaron pruebas de rendimiento con Apache Benchmark para evaluar las prestaciones del clúster bajo diferentes cargas y tipos de contenido (páginas estáticas, dinámicas y estáticas con imágenes). Los resultados mostraron que el clúster es capaz de manejar grandes volúmenes de tráfico con un buen rendimiento, aunque el rendimiento varía según el tipo de contenido y la carga concurrente.

- **Páginas Estáticas:** Ofrecen el mejor rendimiento en términos de solicitudes por segundo y tiempo por solicitud, debido a su simplicidad y menor carga de procesamiento.
- **Páginas Dinámicas:** Tienen un rendimiento menor en comparación con las páginas estáticas, debido al procesamiento adicional requerido para generar contenido dinámico.
- **Páginas Estáticas con Imágenes:** El rendimiento disminuye en comparación con las páginas estáticas simples, debido al tamaño de las imágenes, lo que aumenta el tiempo de transferencia.

**Recomendaciones:**

- Para optimizar el rendimiento del clúster, considerar el uso de cachés y sistemas de entrega de contenido (CDN) para páginas con imágenes grandes o contenido estático que se accede con frecuencia.
- Optimizar el código de las páginas dinámicas y emplear técnicas de compresión y reducción de recursos para mejorar los tiempos de respuesta.

### ▶ Link Aggregation

Se implementaron técnicas de link aggregation para aumentar el ancho de banda y la redundancia en la red. Esta configuración demostró mejorar la capacidad de manejo de tráfico y aumentar la tolerancia a fallos de red.

#### Recomendaciones:

- Continuar utilizando link aggregation para maximizar el rendimiento de la red y asegurar una alta disponibilidad.
- Evaluar periódicamente la configuración de la red para identificar posibles cuellos de botella y optimizar la infraestructura.

### ▶ Contenedores

El uso de contenedores para manejar los requerimientos de diferentes tipos de páginas web permitió una gestión eficiente de los recursos y una fácil escalabilidad. Los contenedores demostraron ser una herramienta eficaz para aislar y gestionar aplicaciones web de diferentes naturalezas.

#### Recomendaciones:

- Adoptar el uso de contenedores en entornos de producción para facilitar la gestión y escalabilidad de los servicios web.
- Implementar prácticas de monitoreo y orquestación de contenedores para asegurar un despliegue eficiente y resiliente.

Como recomendaciones generales, destinadas a mejorar este proceso de implementación de un clúster HA con reparto de carga, tenemos:

- **Monitoreo y Mantenimiento Proactivo:** Implementar herramientas de monitoreo para supervisar el rendimiento del clúster en tiempo real y realizar mantenimientos regulares para prevenir fallos.
- **Optimización del Código y Recursos:** Revisar y optimizar el código de las aplicaciones web y los recursos asociados para mejorar el rendimiento y reducir los tiempos de respuesta.
- **Pruebas Continuas:** Realizar pruebas de rendimiento periódicas para evaluar la capacidad del clúster y ajustar configuraciones según sea necesario para asegurar un rendimiento óptimo.

Para este Trabajo Final de Master, el sistema de clúster implementado demostró ser robusto y capaz de asegurar la alta disponibilidad y un buen rendimiento para un servidor web. Las tecnologías utilizadas y las configuraciones aplicadas permitieron cumplir con los objetivos planteados, ofreciendo una solución escalable y resiliente. Las recomendaciones proporcionadas previamente ayudarán a mantener y mejorar el rendimiento del clúster en el futuro.

## 6 BIBLIOGRAFÍA

- [1] AWS. (10 de Agosto de 2024). *¿Qué es la virtualización? Explicacion de la virtualización*. Obtenido de *¿Qué es la virtualización? - AWS*: <https://aws.amazon.com/es/what-is/virtualization/>
- [2] Barraza, C. (4 de Agosto de 2023). *Ventajas y desventajas de la virtualización*. Obtenido de *Ventajas y desventajas de la virtualización*: <https://barrazacarlos.com/es/ventajas-e-inconvenientes-de-la-virtualizacion/>
- [3] VirtualBox. (2024, Agosto). *VirtualBox User Manual*. Retrieved from <https://www.virtualbox.org/manual/>
- [4] *Razones para escoger Debian*. (Agosto de 2024). Obtenido de *Razones para escoger Debian*: [https://www.debian.org/intro/why\\_debian.es.html](https://www.debian.org/intro/why_debian.es.html)
- [5] Wikipedia. (Noviembre de 2023). *Cluster de Alta Disponibilidad*. Obtenido de *Cluster de Alta Disponibilidad*: [https://es.wikipedia.org/w/index.php?title=Cl%C3%BAster\\_de\\_alta\\_disponibilidad&oldid=155419665](https://es.wikipedia.org/w/index.php?title=Cl%C3%BAster_de_alta_disponibilidad&oldid=155419665)
- [6] Keepalived. (2024, Agosto). *What is Keepalived*. Retrieved from *What is Keepalived*: <https://www.keepalived.org/>
- [7] RedHat. (2024, Agosto). *Configuración y gestión de clusters de alta disponibilidad*. Retrieved from *RedHat Documentation*: [https://docs.redhat.com/es/documentation/red\\_hat\\_enterprise\\_linux/8/html/configuring\\_and\\_managing\\_high\\_availability\\_clusters/index](https://docs.redhat.com/es/documentation/red_hat_enterprise_linux/8/html/configuring_and_managing_high_availability_clusters/index)
- [8] Morvan, A. L. (7 de Noviembre de 2022). *Cluster de alta disponibilidad con GlusterFS*. Obtenido de *Rocky-Linux Documentation*: [https://docs.rockylinux.org/es/guides/file\\_sharing/glusterfs/](https://docs.rockylinux.org/es/guides/file_sharing/glusterfs/)
- [9] HAProxy. (2024). *HAProxy*. Retrieved from *HAProxy*: <https://www.haproxy.org/>
- [10] Sinisterra, M. M. (2012). *Cluster de balanceo de carga y alta disponibilidad para servicios web y mail*. Retrieved from <https://dialnet.unirioja.es/descarga/articulo/4364562.pdf>
- [11] Docker. (2024). *Docker Docs*. Retrieved from *Docker Docs*: <https://docs.docker.com/get-started/docker-overview/>
- [12] SUSE. (2024). *SLES HA Docs*. Retrieved from *SLES HA Docs*: <https://documentation.suse.com/sle-ha/12-SP5/html/SLE-HA-all/cha-ha-manual-config.html>
- [13] Documentation, R. (2024). *The pcs Command Line Interface*. Retrieved from *The pcs Command Line Interface*: [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/6/html/configuring\\_the\\_red\\_hat\\_high\\_availability\\_add-on\\_with\\_pacemaker/ch-pcscommand-haar#ch-pcscommand-HAAR](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/6/html/configuring_the_red_hat_high_availability_add-on_with_pacemaker/ch-pcscommand-haar#ch-pcscommand-HAAR)