



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Escola Tècnica
Superior d'Enginyeria
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

REPLICACIÓN DE SERVIDOR VOIP ACTUANDO DE FAILOVER

Trabajo Fin de Grado

Grado en Ingeniería Informática

Alumno: David Navarrete López

Tutor: Carlos Miguel Tavares de Araujo Cesariny Calafate

Curso: 2019-2020

Contenido

Resumen	3
Abstract.....	4
Agradecimientos	5
1. Introducción	6
1.1. Motivación	6
1.2. Objetivos	6
1.3. Estructura de la memoria	7
2. Explicación introductoria acerca de los servicios a utilizar	8
3. Replicación Amazon Web Services	9
3.1. Aplicación de opciones AWS para la replicación	9
3.2. Motivo de no usar AWS como mecanismo de replicación:	10
4. Instalación Hipervisor ESXI VMware e instancias	11
4.1. Utilización de la herramienta Putty	16
4.2. Instalación de Asterisk.....	16
4.3. Utilización del servicio Asterisk	18
4.3.1. Teléfono VoIP - Softphone.....	23
4.3.2. Demostración servicio Asterisk	24
4.3.3. Servicio Asterisk Realtime	27
5. Replicación en Servidores.....	30
5.1. Cluster DRBD	38
5.1.1. Instalación	38
5.1.2. Configuración	38
6. Gestión Alta Disponibilidad	42
6.1. Instalación y servicios.....	42
6.1.1. Pacemaker	42
6.1.2. Corosync.....	42
6.1.3. Ejecución de ambos servicios	43
6.1.4. Testing Replicación.....	48
7. Conclusiones y trabajo futuro	49
Bibliografía	51



Índice de Imágenes

Ilustración 1. Instalación ISO VMware Workstation	12
Ilustración 2. Acceso VMware ESXI	12
Ilustración 3. Acceso mediante Interfaz Web.....	13
Ilustración 4. Configuración e Instalación máquinas virtuales	13
Ilustración 5. Base de datos VMware ESXI	13
Ilustración 6. Instalación Centos 7.....	14
Ilustración 7. Configuración de Centos 7.....	14
Ilustración 9. Configuración VMware ESXI RED.....	15
Ilustración 10. Configuración Red VMware Workstation	15
Ilustración 11. Configuración de la Herramienta Putty.	16
Ilustración 13. Desactivación SELINUX	17
Ilustración 14. Comprobación del estado de Asterisk.	18
Ilustración 15 . Esquema de directorios Asterisk.....	18
Ilustración 16 . Teléfono VoIP	23
Ilustración 17. Softphone Zoiper	24
Ilustración 18. Estado firewall	24
Ilustración 19. Estado firewall desactivado	25
Ilustración 20. Configuración de Zoiper – Extensión.	25
Ilustración 21. Configuración de SIP Zoiper.....	26
Ilustración 22 . Debug Asterisk - Registro Zoiper.....	26
Ilustración 23. Realización llamadas internas Zoiper.....	27
Ilustración 24. Registros Realtime: Base de datos.....	29
Ilustración 25. Esquema Principal de Replicación.	30
Ilustración 26. Discos conectados en cada Servidor	32
Ilustración 27. Comprobación BD Server 1.....	34
Ilustración 28. Comprobación BD server 2	35
Ilustración 29. Esquema Orientativo Servicios	43
Ilustración 30. Comprobación estado servidor.....	45
Ilustración 31. Comprobación estado asignación servidores	45
Ilustración 32. Comprobación estado servidor.....	46
Ilustración 33. Comprobación final estado servidor.....	47



Resumen

En este trabajo de final de grado de ingeniería informática, se pretende diseñar e implementar un sistema de replicación de servidores, procurando una gran alta disponibilidad del servicio Asterisk. El proyecto propone una serie de mejoras y opciones que garanticen el correcto funcionamiento de estos servicios. Para ello, se han investigado varias maneras de ejecutar el objetivo de este proyecto, empezando por utilizar servicios como la plataforma Amazon Web Services, y decantándose al final por la herramienta de virtualización VMware ESXI, en servidores privados.

A lo largo del proyecto se explicarán cada uno de los servicios, las distintas posibilidades que tiene cada servicio empleado, y la toma de decisiones que se tuvieron que realizar para conseguir el objetivo final.

Palabras Clave: AWS, Amazon, VMware, ESXI, Asterisk, replicación, etc.



Abstract

This final degree project in computer engineering is intended to design and implement a server replication system, ensuring high availability of the Asterisk service. The project proposes a series of improvements and options that guarantee the correct functioning of these services. To accomplish this, several ways of achieving the objective of this project have been investigated, starting by using services such as the Amazon Web Services platform, and finally opting for the VMware ESXI virtualization tool, on private servers.

Throughout the project each of the services will be explained, along with the different possibilities that each service has, and the decision making that had to be applied to achieve the final objective.

Keywords: AWS, Amazon, VMware, ESXI, Asterisk, replication, etc.



Agradecimientos

En primer lugar, dar las gracias a todos mis familiares y amigos que han sido un gran apoyo y pilar, para realizar este proyecto. Con carácter especial, a mis compañeros de trabajo, y jefe, que han sido un gran apoyo y ayuda a la hora de ciertos problemas que se han desarrollado a lo largo de todo el proyecto.

Dar las gracias también a mi tutor Carlos Tavares Calafate, que desde el primer momento apoyó el desarrollo del proyecto, y poder guiarme en muchos aspectos. Creyó en la idea desde el principio, e incluso aplicó cambios de nombre del proyecto, y reorientación del mismo, ayudando a seguir un objetivo en concreto, obteniendo el resultado deseado.

Parte del proyecto se desarrolla en una plataforma que se dedicó mucho tiempo y esfuerzo, y es una gran alternativa a lo que finalmente se ha realizado, y agradecer el curso de AWS de Germán Moltó, que incluso con preguntas realizadas por correo me pudo ayudar.

En general, ha costado mucho trabajo finalizar la carrera, y sin la gente adecuada, como son los amigos, o buenos profesores, se hubiera hecho cuesta arriba.

Gracias a todos porque, sin vosotros, esto no hubiera sido posible.



1. Introducción

1.1. Motivación

Hoy en día, a nivel laboral, se utilizan con más frecuencia los servicios de VoIP.

Prácticamente a diario, existen nuevas vulnerabilidades o ataques incumpliendo la política de protección de datos, y estos problemas suceden tanto en los servicios VoIP, como en las líneas fijas convencionales.

La principal diferencia entre ambas opciones es la forma en que los datos se manejan, se almacenan, o se transmiten. Las líneas fijas funcionan a través de circuitos y cables, por lo que, para alguien que quiera escuchar una llamada, solo necesitan acceso físico a estos componentes.

Por otra parte, las llamadas VoIP son digitales y fluyen a través de Internet. De hecho, muchos expertos en estos temas afirman que un servicio de telefonía por VoIP correctamente configurado, y con la seguridad adecuada, es más fiable que cualquier sistema de línea fija.

Una de las posibilidades que tiene la VoIP en cuanto a la seguridad ante escuchas sería separar estas llamadas del resto de la red a través de Vlans. De hecho, muchas empresas en la actualidad llegan a tener una red física separada solamente para sus llamadas telefónicas, especialmente notarías.

Gracias a la vida laboral, se entiende mejor que es necesario que exista una alta disponibilidad del servicio, ya que un problema o error puede suponer grandes costes empresariales. Por todo esto se decidió montar un sistema con una gran disponibilidad, donde se intente tener activo el servicio constantemente, en este caso, un servicio de llamadas por VoIP.

1.2. Objetivos

El proyecto trata de realizar un sistema con servidores activos, proporcionando una alta disponibilidad de un servicio, denominado Asterisk. Este servicio proporciona la posibilidad de realizar llamadas por VoIP, que más adelante se explicarán cada uno con más detalle. El objetivo principal consiste en que este servicio, ocurra cualquier inconveniente o error, siempre se garantice el servicio de llamadas por Asterisk.

Para ello, se utiliza una plataforma como es VMware ESXI, donde conste de dos instancias o máquinas virtuales, haciendo uso de varias herramientas, para asegurar la disponibilidad del servicio comentado, realizando una replicación de ambos servidores, que actúen de failover, en caso de cualquier problema.

Al principio se pretendía usar la plataforma Cloud más conocida en la actualidad, Amazon Web Services, para realizar la instalación de las máquinas virtuales donde se replicaría el servicio de Asterisk. Debido a varios inconvenientes que se encontraron a lo largo de la investigación y explicado a continuación, se optó por usar la plataforma de VMware ESXI, ya que cumplía mejor los objetivos propuestos.



1.3. Estructura de la memoria

Esta memoria se divide en los siguientes bloques:

En el apartado 2 se ofrece información de los distintos servicios que se van a ir utilizando a lo largo del proyecto.

En el apartado 3 se estudia la posibilidad de realizar el proyecto en una plataforma en cloud como es Amazon Web Services, y se comentan las posibilidades que ofrece este servicio, y la explicación a posteriori de porque no se utiliza esta primera opción.

En el apartado 4 se detalla la base del proyecto, y se explica la instalación de la plataforma a utilizar y los servicios empleados para el objetivo final, como es VMware ESXI, o el servicio principal Asterisk.

En el apartado 5 se ofrece una explicación detallada sobre el proceso de replicación de servidores, y de como se han ido configurando los puntos más fundamentales para que este servicio fuera posible.

En el apartado 6 se describe como definir los principales servicios para realizar y garantizar una alta disponibilidad del servicio principal y de rendimiento de los servidores.

Finalmente, en el apartado 7, se presenta un resumen de los resultados obtenidos, así como de las futuras ampliaciones que se pueden realizar en próximos proyectos.



2. Explicación introductoria acerca de los servicios a utilizar

En este apartado se procede a comentar el significado de los servicios que se van a utilizar en este proyecto.

Para empezar, se tiene que destacar el servicio de **VoIP**, el cual es un acrónimo de Voz sobre el Protocolo de Internet (Voice Over Internet Protocol). Es una tecnología que proporciona la comunicación de voz y sesiones multimedia (tales como vídeo) sobre el Protocolo IP.

Las soluciones VoIP permiten una interacción dinámica entre los usuarios de cualquier dominio en Internet cuando desean realizar una llamada. Para realizar llamadas a través de VoIP el usuario necesita un teléfono IP basado en software (Softphone), o un Teléfono VoIP basado en hardware. Las llamadas telefónicas pueden hacer a cualquier lugar y a cualquier persona, tanto a números VoIP como a números de teléfono PSTN.

En este punto conviene aclarar los conceptos de PBX, PSTN y Asterisk:

- **PBX**: Se refiere a cualquier central telefónica conectada a la red pública por medio de troncales para gestionar llamadas internas, entrantes y salientes sobre otra central telefónica. Las siglas quieren decir “Central Privada Automática”.
- **PSTN**: es la red de redes públicas de telefonía en el mundo de conmutación de circuitos. [1].
- **Asterisk** es un programa de software libre (bajo licencia GPL) que proporciona funcionalidades de una central telefónica (PBX). Como cualquier PBX, se puede conectar un número determinado de teléfonos para hacer llamadas entre sí dentro de una misma organización e incluso acceder a comunicaciones fuera de la misma a la PSTN o conectando a un proveedor de VoIP [2].

Algunas de las funciones que le permite realizar una centralita basada en Asterisk serían:

- Grabación de llamadas de manera automática o bien a demanda del agente de atención. Escucha de llamadas.
- Agentes locales y remotos que le permitirá establecer su centro de atención al cliente en cualquier lugar del mundo.
- Recepcionista Digital (IVR). Operadora automatizada para la recepción y gestión de llamadas entrantes.
- ACD (Sistema para realizar y gestionar colas de llamadas de manera efectiva entre los agentes disponibles).

Todos estos recursos y servicios que se tiene pensado favorecen el desarrollo del objetivo principal realizando copias incrementales de datos, para que cuando se activara el servidor de repuesto no tenga que buscar los datos necesarios para el funcionamiento del servicio.



3. Replicación Amazon Web Services

3.1. Aplicación de opciones AWS para la replicación

La idea principal de realización del proyecto, y el comienzo del mismo se realizó en AWS, por lo que se investigó acerca de las diferentes ventajas y opciones que ofrecía este servicio, Amazon Web Services.

-AWS RDS:

Lo que puede realizar AWS es asegurar una réplica respecto a la base de datos, y todo este tema era bastante limitado. Consistía en configurar varias instancias (máquinas virtuales), la principal conocida como Instancia Maestra, y la réplica como una estancia en stand-by, estando ambas sincronizadas. El objetivo era que, en caso de caída de la instancia maestra, la instancia réplica que está esperando pasara automáticamente a estar operativa sin tener que realizar actualizaciones previas. [3]

-Amazon S3 y Amazon SimpleDB:

Gestionan automáticamente la replicación de los datos como mecanismo de tolerancia a fallos y alta disponibilidad. En el caso de bases de datos relacionales, Amazon RDS también ofrece replicación transparente de datos.

La ventaja de utilizar los esquemas de replicación ofrecidos por el proveedor Cloud es que éstos son transparentes para el usuario, y que trabajan como si únicamente existiese una única copia de los datos.

-Snapshot:

Respecto a las propias instancias existen Snapshots, que son imágenes de un momento determinado de la máquina virtual/instancia que hace que, si entra algún malware o virus, o si se borran archivos que no deseabas, u ocurre cualquier otro tipo de error o problema, puedas recuperar la misma máquina en un instante de tiempo anterior al problema en cuestión.

Amazon EBS (Elastic Block Store) permite la creación de estas instantáneas (snapshots).

Amazon Elastic Block Store (EBS) es un servicio de almacenamiento de bloque de alto rendimiento con facilidad de uso diseñado para su uso con Amazon Elastic Compute Cloud (EC2) tanto en cargas de trabajo intensivas de rendimiento como de transacciones, a cualquier escala. [4].

Se pueden utilizar los snapshots para:

- Crear un nuevo volumen EBS a partir de alguno existente.
- Aumentar el tamaño de un volumen.
- Mover volúmenes entre diferentes zonas de disponibilidad.

En conclusión, los snapshots son copias de seguridad incrementales, es decir, sólo copia los datos que han variado desde la última operación de *backup* de cualquier tipo.



– Amazon S3:

“Amazon S3 es almacenamiento para Internet. Está diseñado para facilitar a los desarrolladores la informática a escala web.” [5]

Consiste en un sistema de almacenamiento de un número ilimitado de objetos para el entorno Cloud. Esta opción, copia de los datos del volumen en S3, donde se guardan de forma redundante en varias zonas de disponibilidad.

– Alarmas CloudWatch:

Existen alarmas para monitorizar los recursos utilizados en Amazon Web Services, es un tipo de monitorización similar al PRTG Network Monitor Software. El objetivo de utilizar este recurso básicamente es evitar daños mayores, es decir, en caso de cualquier caída del servicio utilizado, salta una alarma al correo avisándote del problema. Esto es una gran ventaja debido a que te puedes anticipar a los problemas con los clientes, y solucionarlos con mucha velocidad. [6]

Todas estas opciones, mantienen de una forma distinta la alta disponibilidad, que es el objetivo actual. No quiere decir que sea mejor o peor, de hecho, la utilización de recursos en Cloud como Amazon Web Services es cada vez mayor, y las empresas actuales tienden a utilizar más estos recursos debido a que los recursos hardware utilizados suelen estar asociados a un mayor coste económico y un mantenimiento constante, el cual puede entorpecer al objetivo principal, que es que el servicio sea perfecto.

Todos estos recursos anteriormente comentados, tienen limitaciones a la hora de realizar ciertas configuraciones. El objetivo principal del proyecto era realizar un “failover” en la plataforma AWS, sin embargo, como se comentará en el punto siguiente, no era posible conseguir este objetivo. Por lo que se optó por otra plataforma denominada VMware ESXI.

Toda esta primera parte, ha sido más de investigación que de ejecución, aunque haya utilizado otros recursos que no han servido para el objetivo final. Por todo esto, la elección final ha sido otra.

3.2. Motivo de no usar AWS como mecanismo de replicación:

Antes de explicar todo lo realizado en este proyecto, hay que conocer Asterisk, el servicio que se pretende desplegar. Asterisk es un programa de software libre (bajo licencia GPL) que proporciona funcionalidades de una central telefónica (PBX). Como cualquier PBX, se puede conectar un número determinado de teléfonos para hacer llamadas entre sí dentro de una misma organización e incluso acceder a comunicaciones fuera de la misma a la PSTN o conectando a un proveedor de VoIP o bien a una RDSI tanto básicos como primarios [1].

Viendo todo lo que se pretendía realizar en este proyecto, AWS estaba algo limitado de recursos para realizar todo lo explicado anteriormente.

Uno de los motivos principales para declinar esta opción fue la necesidad de utilizar una IP Flotante.



IP Flotante: es una dirección IP pública enrutable que no se asigna a ninguna instancia de manera automática. El propietario de un proyecto la pone a disposición de una o varias instancias de manera temporal. La instancia correspondiente dispone así tanto de una IP estática, que le ha sido concedida automáticamente para la comunicación en un ámbito de red privado no enrutable, como también de una IP flotante asignada manualmente. Esto hace accesible distintos servicios externos [7].

Por así decirlo, se trata de una IP pública que se puede asignar a instancias, y la cual, en este proyecto, se utilizará para la replicación y actuación de failover de los servidores, detectando esta IP flotante automáticamente, y permitiendo activar un servidor u otro, dependiendo de las necesidades o problemas que puedan existir a lo largo de su vida útil. Sin embargo, la aplicación de esta IP flotante en AWS no se puede realizar. AWS te permite asignar una IP a una instancia, pero no puede detectar la forma de cambiar de IP automáticamente en caso de caída del servidor, por lo que no se puede utilizar la IP flotante explicada en el párrafo anterior.

Después de toda la información y opciones encontradas, AWS no terminaba de ofrecer las especificaciones que se querían obtener, y las que se encontraron, eran un gasto de dinero extra, ya que por cada servicio que se utiliza en AWS se requiere de un gasto real de dinero. Por esto, en el proyecto se optó al final por un hipervisor ESXI de VMware donde se podía reproducir a la perfección lo que se pretende hacer, tal y como es especificado en el punto 4 en adelante.

4. Instalación Hipervisor ESXI VMware e instancias

Primero que nada, hay que explicar que usaremos una plataforma gratuita para instalar nuestro hipervisor ESXI. La plataforma que se va a utilizar se llama “VMware Workstation 15 Player”.

VMware Workstation Player es una aplicación de virtualización de escritorios que está disponible sin coste para uso personal. Se puede aplicar una licencia comercial para que Workstation Player ejecute las máquinas virtuales restringidas creadas por VMware Workstation Pro y Fusion Pro; es gratuito para un uso personal [8].

Una vez instalado, se procede a instalar la máquina virtual con la ISO del sistema operativo en cuestión, en este caso la ISO del Hipervisor ESXI.



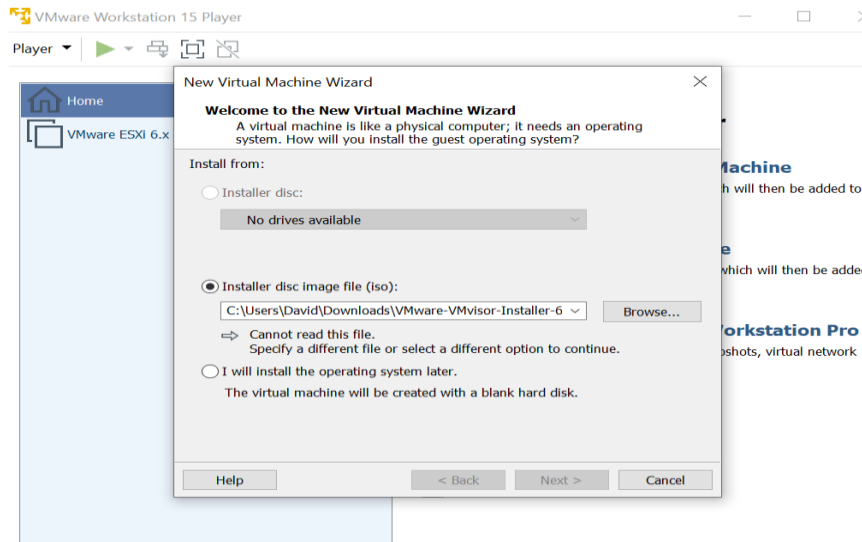


Ilustración 1. Instalación ISO VMware Workstation

Una vez realizado el proceso de instalación, se procede a iniciar la máquina virtual :

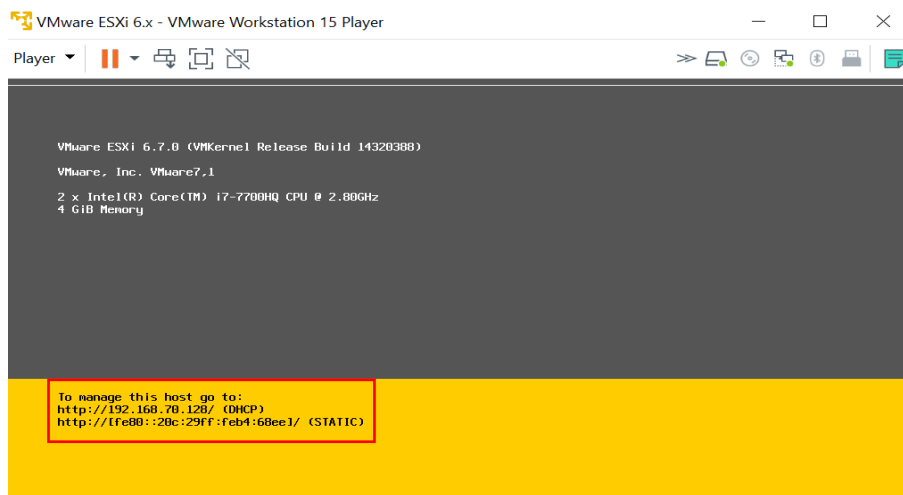


Ilustración 2. Acceso VMware ESXi

Al iniciarla, como podemos ver en la imagen anterior, proporciona una serie de datos, como el procesador utilizado, o la memoria RAM asignada a la máquina virtual, y una dirección IP para la configuración del hipervisor, y así proceder a realizar el proyecto explicado. Esta dirección de red está proporcionada con un servidor de DHCP, por lo que cada vez que se apague o inicie la máquina puede variar a la hora de iniciar el hipervisor.

Para acceder a la parte de configuración, se tiene que poner la dirección enmarcada en rojo en la imagen anterior en un navegador:

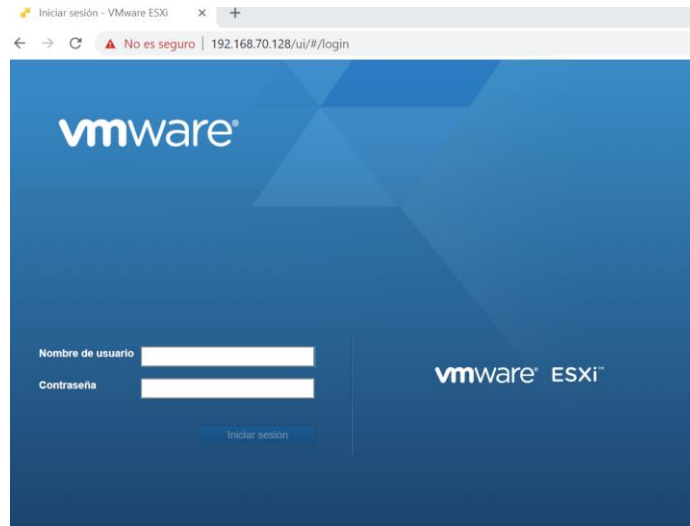


Ilustración 3. Acceso mediante Interfaz Web.

Una vez se accede a la interfaz del hipervisor ESXI, se tiene que agregar dos máquinas virtuales, donde se instalarán Centos 8 para nuestro servicio de Asterisk.

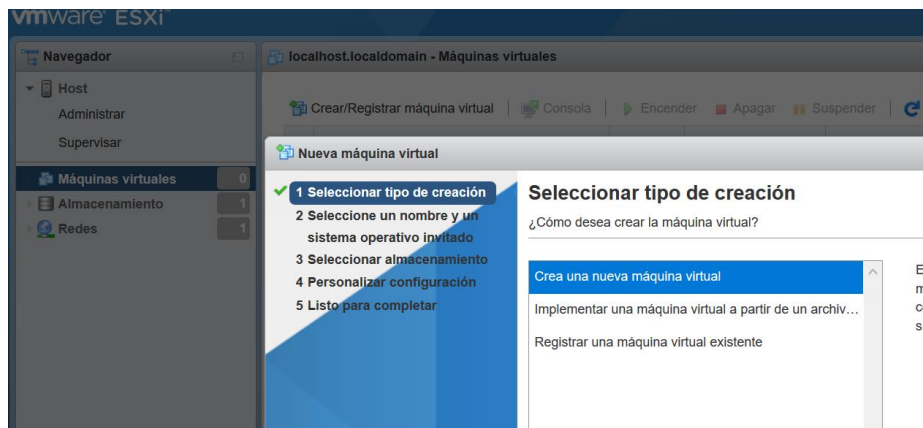


Ilustración 4. Configuración e Instalación máquinas virtuales

Se deberá seleccionar una imagen ISO referente al sistema operativo Centos 8, y subirla al mismo hipervisor, donde se tendrá que seleccionar en la máquina virtual para que arranque con la ISO para su instalación.

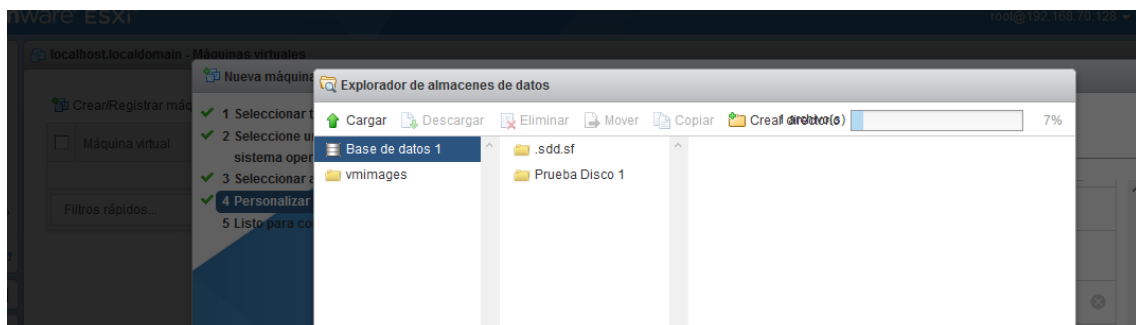


Ilustración 5. Base de datos VMware ESXI

Una vez se arranque la máquina virtual, se procederá a la instalación, como se ha comentado en la imagen anterior, configurando los parámetros necesarios para el correcto funcionamiento:

```

Server1
[ 11.290602] dracut-pre-udev[518]: nodprobe: FATAL: Module floppy not found in
[ 11.313143] dracut-pre-udev[518]: nodprobe: ERROR: could not insert 'edd': No
[ OK ] Started Show Plymouth Boot Screen.
[ OK ] Started Forward Password Requests to Plymouth Directory Watch.
[ OK ] Reached target Paths.
[ OK ] Reached target Local Encrypted Volumes.
[ 13.501903] sd 0:0:0:0: [sd] Assuming drive cache: write through
[ OK ] Started udev Wait for Complete Device Initialization.
      Starting Device-Mapper Multipath Device Controller...
[ OK ] Started Device-Mapper Multipath Device Controller.
[ OK ] Reached target Local File Systems (Pre).
[ OK ] Reached target Local File Systems.
      Starting Create Volatile Files and Directories...
      Starting Open-iSCSI...
[ OK ] Started Open-iSCSI.
      Starting dracut initqueue hook...
[ OK ] Started Create Volatile Files and Directories.
[ OK ] Reached target System Initialization.
[ OK ] Reached target Basic System.
[ 14.716526] dracut-initqueue[1066]: mount: /run/install/repo: WARNING: device write-protected, mounted
[ OK ] Started Show Plymouth Boot Screen.
[ OK ] Started Forward Password Requests to Plymouth Directory Watch.
[ OK ] Reached target Paths.
[ OK ] Reached target Local Encrypted Volumes.
[ OK ] Started udev Wait for Complete Device Initialization.
      Starting Device-Mapper Multipath Device Controller...
[ OK ] Started Device-Mapper Multipath Device Controller.
[ OK ] Reached target Local File Systems (Pre).
[ OK ] Reached target Local File Systems.
      Starting Create Volatile Files and Directories...
      Starting Open-iSCSI...
[ OK ] Started Open-iSCSI.
      Starting dracut initqueue hook...
[ OK ] Started Create Volatile Files and Directories.
[ OK ] Reached target System Initialization.
[ OK ] Reached target Basic System.
[ 14.716526] dracut-initqueue[1066]: mount: /run/install/repo: WARNING: device write-protected, mounted
[ OK ] Created slice system-checkisomd5.slice.
      Starting Media check on /dev/sr0...
/dev/sr0: 4f2afc516401969c32952697302fa041
Fragment sums: 2be83c6c3a66dd28aefa121889aaafc4cfd1759d618ac4e68d466b31f9789
Fragment count: 20
Supported ISO: yes
Press [Esc] to abort check.
Checking: 073.1%_
    
```

Ilustración 6. Instalación Centos 7



Ilustración 7. Configuración de Centos 7.



Por último, mostrar la prueba de que el sistema se ha instalado correctamente, en ambos servidores, con la misma versión de sistema operativo, que es: Centos 7.

Por otro lado, para que las máquinas virtuales tuvieran internet, había que entrar en las opciones del VMware ESXI y configurar una red adecuada:

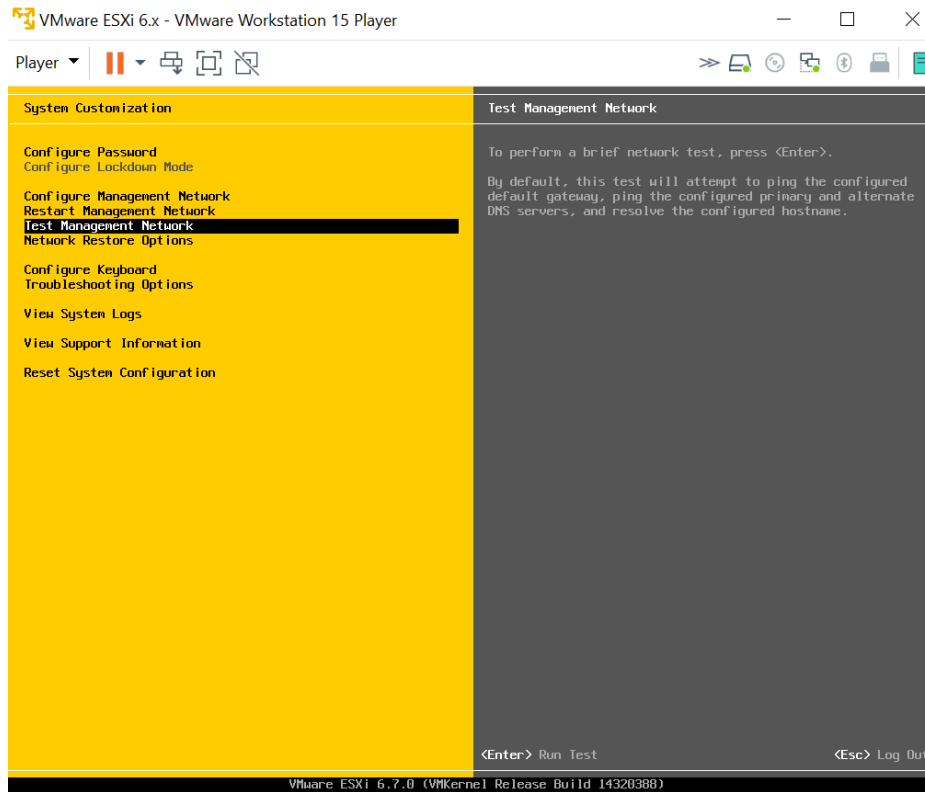


Ilustración 8. Configuración VMware ESXi RED

También habría que aclarar que es necesario poner la máquina virtual primaria en modo puente (bridge), para que tenga acceso a Internet mediante el router.

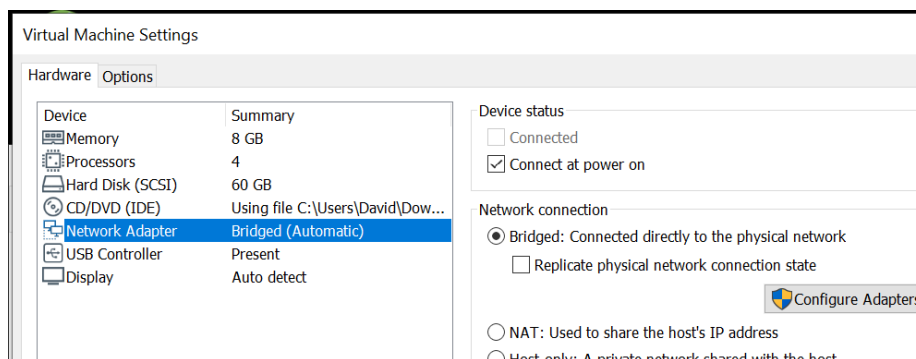


Ilustración 9. Configuración Red VMware Workstation

Una vez configurado el sistema operativo, se procede a instalar el servicio principal antes mencionado, Asterisk.



4.1. Utilización de la herramienta Putty

En este apartado se va a explicar una forma de acceder a las máquinas virtuales creadas en el hipervisor ESXI mediante la herramienta Putty.

Primero que nada, se va a explicar que es esta herramienta:

PuTTY es un cliente SSH, Telnet, rlogin,... con licencia libre. Es un programa que permite conectarse de una forma más cómoda a cualquier servidor o máquina virtual, en el cual se pueda interactuar de una forma más sencilla, tanto por la ejecución de comandos, como con herramientas como el copiar y pegar entre otras.

Para ello bastaría con ejecutar el programa e indicar la dirección IP correspondiente y el puerto.

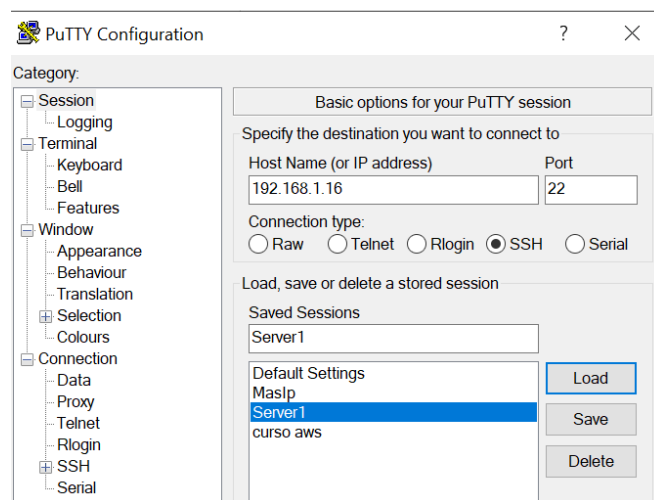


Ilustración 10. Configuración de la Herramienta Putty.

Y directamente te abre un terminal mediante ssh del servidor en cuestión. Para proceder a la instalación del servicio Asterisk, tal y como se explica a continuación, viene bastante bien la opción de copiar y pegar instrucciones, ya que suelen ser comandos muy extensos.

4.2. Instalación de Asterisk

Para toda la instalación del servicio, existe un libro en el que te especifica paso a paso como se tiene que instalar, y porqué necesitas cada uno de los comandos que se van a ejecutar.

Obviamente, no se van a explicar cada uno de los comandos que se ejecutan, ya que el proyecto pasaría a ser de más de 100 hojas, pero sí que se especificarán algunos de los más importantes.

Primero que nada, debido a que el sistema operativo se instala y no tiene prácticamente ningún servicio activado, se instala un editor de archivos conocido como "vim", ya que aporta una serie de ventajas y opciones que otros editores no ofrecen, siendo considerado el mejor editor de texto de Linux.

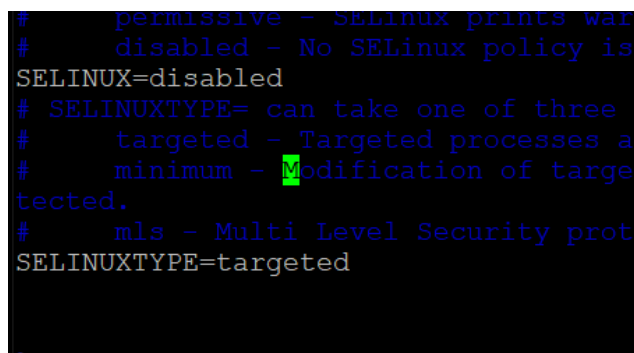
```
yum -y install vim
```

Algunos de los comandos necesarios para instalar Asterisk 13, podrían ser los especificados en el propio libro; sin embargo, habría que comentar que con el siguiente comando da problemas el servicio que se pretende instalar, por lo que es necesario aclarar y explicar cuál es:

Se tiene que editar la opción de Selinux. *Security-Enhanced Linux es un módulo de seguridad para el kernel Linux que proporciona el mecanismo para soportar políticas de seguridad para el control de acceso, incluyendo controles de acceso obligatorios como los del Departamento de Defensa de Estados Unidos. Se trata de un conjunto de modificaciones del núcleo y herramientas de usuario que pueden ser agregadas a diversas distribuciones Linux. Su arquitectura se enfoca en separar las decisiones de las aplicaciones de seguridad de las políticas de seguridad mismas y racionalizar la cantidad de software encargado de las aplicaciones de seguridad [9].*

Es imprescindible deshabilitarlo, caso contrario el servicio de Asterisk no funcionará correctamente. Para ello se edita el archivo de configuración:

```
vim /etc/selinux/config
```



```
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of three possible values:
# targeted - Targeted processes are protected.
# minimum - Modification of targeted processes. Allowed actions on
# targeted processes are denied. Modifications that are not targeted are
# still allowed.
# mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Ilustración 11. Desactivación SELINUX

Una vez instalado el servicio de Asterisk, habría que especificar la forma correcta para poder reiniciar el servicio, ya que de ello puede depender su correcto funcionamiento:

```
systemctl start asterisk
```

Instalado todo servicio necesario, samples, etc. comprobamos con el siguiente comando de que Asterisk está activo, lo cual corresponde al texto en verde, como se puede apreciar en la imagen.

```
systemctl status asterisk
```

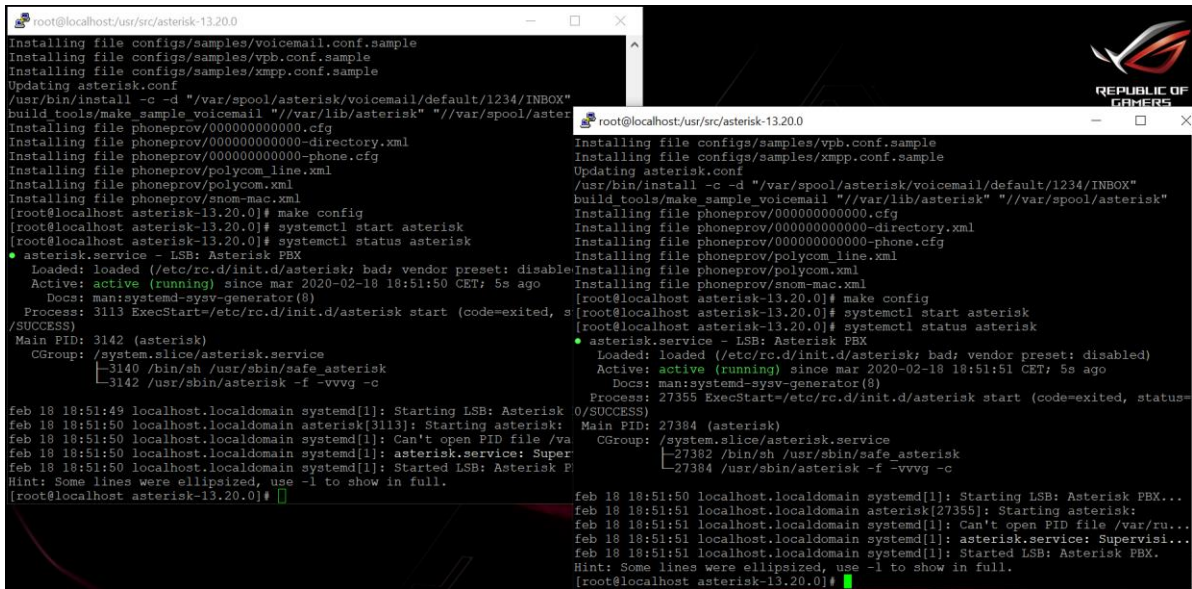


Ilustración 12. Comprobación del estado de Asterisk.

4.3. Utilización del servicio Asterisk

Antes de proceder al asunto principal del proyecto, es necesario ver cómo funciona este servicio, poniendo ejemplos reales de una configuración básica del servicio, y la diferente terminología que se va a emplear. Esto es necesario para ver la capacidad del servicio, y las posibilidades que ofrece a nivel profesional y empresarial.

Primero que nada, se va a comentar la estructura que puede tener el directorio principal “Asterisk”.

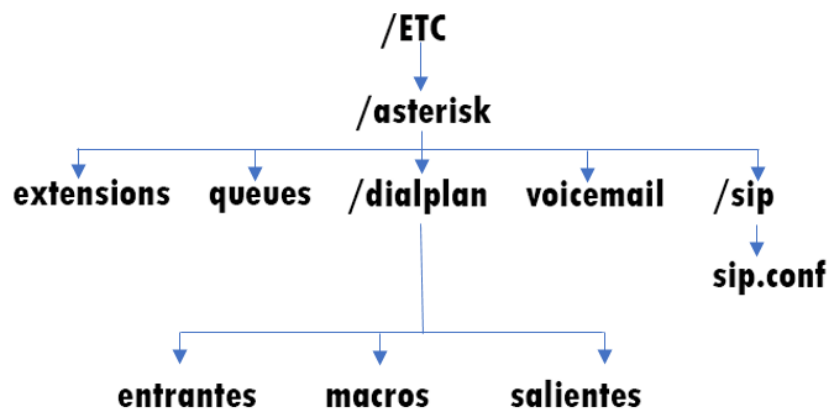


Ilustración 13 . Esquema de directorios Asterisk.



Se han puesto los archivos más relevantes a la hora de realizar la configuración. Esta es la estructura básica en la que se suele basar los sistemas Asterisk a la hora de organizar el flujo de información. Perfectamente se podría hacer todo en la misma carpeta dentro del directorio Asterisk, pero estructuralmente no queda tan claro como un sistema bien organizado.

Primero habría que aclarar el directorio dialplan. ¿Que és un Dialplan? Se refiere al plan de marcado de nuestra centralita Asterisk, donde se definirá que se tiene que hacer con las llamadas que entran, que tengo que marcar para poder realizar llamadas al exterior, definir el horario laborable del cliente, definir locuciones cuando reciban llamadas, cuando esperen, cuando tengan a todos los agentes ocupados, y permite la posibilidad de crear un IVR, que es un menú que puedes asignar a la llamada (como, por ejemplo, que apretando un número en concreto te lleve a un departamento seleccionado), y una larga lista de posibilidades que habilita este servicio.

Hay que explicar con más detalle, que para realizar la configuración completa de la centralita sería necesario contratar un troncal, por donde salgan las llamadas. Esto quiere decir, que al contratar este servicio se facilitaría un número al que poder llamar y donde se pueda recibir llamadas. Aquí entra en acción lo que sería el archivo de *entrantes.dialplan*, o incluso una parte del *salientes.dialplan*.

La idea consiste en que se contrata una numeración pagando un dinero al mes y, por supuesto, te cobre cada llamada realizada como cualquier operadora actual, y la centralita te tramite esta llamada y puedas configurarla un poco según preferencias o como el cliente desee.

Debido a que este proyecto no supone de ningún coste económico para realizarlo, no se ha contratado este servicio, sin embargo, se comentarán las distintas posibilidades y el dialplan que se podría llevar a nivel profesional. Y se realizará ejemplos prácticos de llamadas internas entre extensiones que se crearán, para la comprobación del correcto funcionamiento del sistema.

Puesto que el servidor puede contener varios dialplan referente a empresas o clientes distintos, se suele organizar en directorios con nombres específicos. En este caso, la empresa creada ha sido "tfg", y es donde contendrá estos archivos comentados en el organigrama del árbol del directorio dialplan.

Entrantes.dialplan:

```
[in-tfg]
exten => 997452141,1,noop(NÚMERO PRINCIPAL DEL CLIENTE)
    same => n,Answer()
    same => n,Macro(tfg-festivos)
        same => n,Macro(horario-tfg)
    same => n,Playback(${locuciones-tfg}/bienvenidaTFG)
    same => n,Queue cola-tfg,,60)
    same => n,Voicemail(tfg@default,s)
    same => n,Hangup()
```

Este sería un ejemplo de un entrante del dialplan, con un número inventado puesto para el proyecto. En él te sigue un orden, donde primero contesta la llamada, después comprueba el archivo de macros que se especificará más tarde, comprobando que siga el horario de jornada



laboral para ver si están disponibles, y ver que no sea un día festivo. Luego, suena una locución de bienvenida, y después entra a una cola de espera de extensiones.

Las colas de llamadas suelen ser muy importantes a la hora de recibir las llamadas, con esta configuración se puede organizar para que le entre a una extensión en concreto, luego a otra, y luego a otra. Este apartado se comentará más adelante.

Si durante 60 segundos no lo coge nadie, pasaría a un buzón de voz, configurado en el archivo `voicemail.conf` dentro del directorio propio de Asterisk. Y, por último, cuando haya realizado el mensaje de buzón de voz, la llamada se colgará.

Con este primer ejemplo se puede comenzar ya a tener una idea de la finalidad de este servicio y el valor que puede suponer.

El siguiente archivo que se va a comentar es "*macros.dialplan*", aquí es donde se suele definir varias funciones relacionado con el *dialplan*. Principalmente, se suele poner el horario y como debe actuar si está o no dentro del horario laboral. Aquí, se pueden definir otras muchas funciones, como por ejemplo la posibilidad de realizar un desvío de un número, es decir, si la empresa en cuestión cierra a una hora en concreto, pasada ya esta hora se puede hacer que cuando llamen a la numeración definida en el archivo entrantes, se desvíe a un móvil u otro teléfono.

Permite infinidad de funciones y complejidades este sistema, como habilitar también una numeración para que el cliente pueda grabar su propia locución y guardarla en nuestro servidor, o la posibilidad de crear una función donde te cree una "blacklist" que es donde se guardan los números no deseados que se bloquean, etc.

En este caso se explicará una macro sencilla para hacerse un poco a la idea de cómo funciona:

```
[macro-horario-tfg]
exten => s,1,GotoIfTime(9:00-14:00,mon-fri,*,*?abierto)
        same => n,GotoIfTime(16:00-19:00,mon-
fri,*,*?abierto:cerrado)
        same => n(cerrado),Noop(Cerrado)
                same => n,Answer()
                same => n,Playback(${locuciones-tfg}/fuera_horario)
                same => n,Hangup()
        same =>
n(abierto),goto(${MACRO_CONTEXT},${MACRO_EXTEN},${${MACRO_PRIORITY} +
1])
        same => n,Hangup()
```

Este sería un ejemplo fundamental del horario que sigue en nuestra empresa creada denominada "tfg". Lo que realmente hace es que genera un horario laboral, definiendo que de 9:00 a 14:00 de cualquier día y de cualquier mes (definido con "*"), esté abierto y disponible para que reciban llamadas, y de 16:00 a 19:00 siga con su jornada laboral. Habría que hacer hincapié en varias etiquetas, como *abierto:cerrado*, que quiere decir que si cumple lo antes comentado entrará en la etiqueta abierto, por lo que seguirá el *dialplan* de entrantes donde se ha explicado en el punto anterior los pasos que va siguiente cuando entra cualquier llamada a esa numeración.



Y, por último, poner un ejemplo de cómo sería la definición de los festivos. Cualquiera que use este sistema tiene que tener claro que, si no se programa algún día festivo como el día de año nuevo, seguiría sonando la llamada en la empresa determinada.

```
[macro-tfg-festivos]
exten => s,1,GotoIfTime(*,*,1,jan?cerrado:abierto)
    same => n,GotoIfTime(*,*,19,mar?cerrado)
    same => n,GotoIfTime(*,*,4,apr?cerrado:abierto)
        same => n(cerrado),Playback(${locuciones-tfg}/festividadlocal)
        same => n,Voicemail(tfg@default,s)
        same => n,Hangup()
        same =>
n(abierto),goto(${MACRO_CONTEXT},${MACRO_EXTEN},${MACRO_PRIORITY} +
1])
    same => n,Hangup()
```

Esta macro es bastante sencilla de entender, básicamente te detalla que, si es un día en concreto del año, entre a la etiqueta “cerrado”, donde primero sonaría una locución explicando que están fuera de su jornada laboral debido a una festividad. Luego pasaría el buzón de voz, y, por último, colgase la llamada.

En caso contrario, debería saltar a la etiqueta “abierto”, y seguir con el dialplan del archivo “entrantes.dialplan”.

Para terminar con este tema, faltaría comentar el archivo “salientes.dialplan”:

```
[from-tfg]
include => tfg_internas
include => tfg_fijos
include => tfg_moviles

[tfg_internas]
exten => _1XX,1,Dial(SIP/${EXTEN}-tfg)
    same => n,Hangup()

[tfg_fijos]
exten => _[9]XXXXXXXX,1,Noop(llamada nacional)
    same => n,Dial(${mor_tfg}34${EXTEN})
    same => n,Hangup()

[tfg_moviles]
exten => _[67]XXXXXXXX,1,Noop(llamada nacional)
    same => n,Dial(${mor_tfg}34${EXTEN})
    same => n,Hangup()
```

Éste sería un ejemplo básico de que botones se pueden apretar para realizar llamadas. Existen dos tipos, internas y externas. Las llamadas internas son a nivel local, y son llamadas entre extensiones dentro de una empresa. Y las externas, que son las definidas para llamar a un determinado número de teléfono fijo, o un número de móvil.



Respecto a las llamadas internas, que son las que se van a explicar con más detalle porque se pueden realizar y demostrar, procede aclarar que aquí se define una programación determinada para extensiones que se crearán en el archivo sip.conf. En este caso, como se puede apreciar en el código, más concreto en el contexto [tfg-internas]: exten => _1XX, significa que se podrá llamar a todas las extensiones internas que comiencen por 1, por ejemplo, 100,101,150,175, etc.

Para concluir este apartado, destacar la programación necesaria para realizar llamadas a fijos o móviles haciendo referencia al mor (troncal contratado, y por donde salen las llamadas) que sin esta definición no se podrían realizar llamadas hacia fuera.

Una vez explicado el dialplan, faltaría por explicar el sip.conf, que es donde se definen las extensiones que se van a utilizar para realizar las llamadas internas.

```
[TFG-DAVID](!)
type=friend
transport=udp
context=from-audinercop
host=dynamic
nat=force_rport,comedia
port=7075
ignoreexpiry=yes
qualify=yes
dtmfmode=rfc2833
disallow=all
allow=alaw
;allow=g729
language=es
callgroup=38
pickupgroup=38
canreinvite=no
subscribecontext=hints-tfg
musicclass=masip_tecnico
musiconhold=masip_tecnico
notifycid = yes
rpid_update=yes
secret=tf2020
```

```
;;CLIENTES SIP
[100-tfg](TFG-DAVID)
callerid=Puesto 100 <100>
```

```
[101-tfg](TFG-DAVID)
callerid=Puesto 101 <101>
```

Aquí, en el código anterior, se definen las extensiones para poder llamar, y donde se registrarán en teléfonos por VoIP, o Softphone.



4.3.1. Teléfono VoIP - Softphone

Antes de seguir con la explicación, cabe destacar estos dos términos que significan.

Un teléfono VoIP, también conocido como SIP phone o Softphone, utiliza Voice Over IP (VoIP) para realizar y transmitir llamadas telefónicas a través de una red IP, como Internet. VoIP convierte el audio telefónico estándar en un formato digital que puede ser transmitido a través de Internet, y también convierte señales digitales entrantes de teléfono procedentes de Internet en audio telefónico estándar.

Un teléfono VoIP permite a los usuarios hacer llamadas telefónicas utilizando VoIP, a cualquier softphone, teléfono fijo o móvil. Un teléfono VoIP puede ser un simple teléfono virtual basado en software o un dispositivo de hardware que se parece a un teléfono común [10].

Generalmente, existen teléfonos físicos como son los SIP phone, que tienen la forma de cualquier teléfono cotidiano, pero exigen una configuración en concreto para estos dispositivos. Dicha configuración, es la mostrada en el apartado anterior, con la configuración del servicio Asterisk. Un ejemplo de teléfono sería:



Ilustración 14 . Teléfono VoIP

Por otra parte están los Softphone, o software que actúa como teléfono físico, instalado en un ordenador. Hace la misma función, o muy parecida, a los teléfonos “sip phone”. Existen muchos tipos de Softphone utilizados hoy en día en muchísimas empresas. Como, por ejemplo, BRIA, X-LITE, 3CX, o Zoiper, que es el que se va a tratar en este proyecto.

Zoiper, es uno de los programas más utilizados mundialmente para el uso de teléfonos SIP. Sin embargo, hay que destacar que se necesita de unos recursos mínimos de ordenador, y sobre todo una buena conexión a Internet. Un ejemplo de zoiper sería:

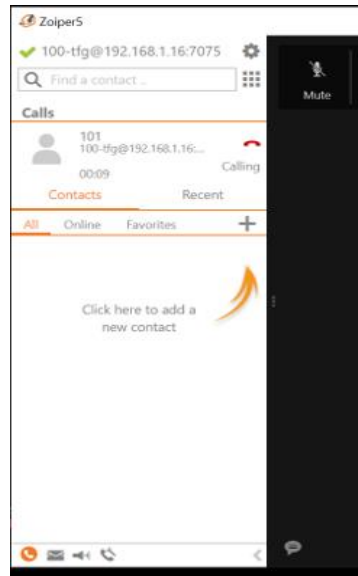


Ilustración 15. Softphone Zoiper

4.3.2. Demostración servicio Asterisk

Como se quiere demostrar que las llamadas internas funcionan correctamente, primero se desactiva cualquier tipo de firewall que tenga la máquina virtual con el sistema operativo instalado, y se comprueba el estado:

```

root@localhost:~
login as: root
root@192.168.1.16's password:
Last login: Tue Feb 25 19:25:20 2020 from 192.168.1.221
[root@localhost ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor
   eset: enabled)
   Active: active (running) since mar 2020-02-25 19:24:59 CET; 54min ago
     Docs: man:firewalld(1)
    Main PID: 857 (firewalld)
    CGroup: /system.slice/firewalld.service
           └─857 /usr/bin/python2 -Es /usr/sbin/firewalld --nofork --nopid

feb 25 19:24:57 localhost.localdomain systemd[1]: Starting firewalld - dynami.
feb 25 19:24:59 localhost.localdomain systemd[1]: Started firewalld - dynamic.
Hint: Some lines were ellipsized, use -l to show in full.
[root@localhost ~]#
    
```

Ilustración 16. Estado firewall

Una vez comprobado que el servicio está activado, es necesario pararlo e incluso deshabilitarlo para no interferir en nuestro sistema de telefonía.

```
[root@localhost ~]# systemctl stop firewalld
[root@localhost ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since mar 2020-02-25 20:21:12 CET; 2s ago
     Docs: man:firewalld(1)
   Process: 857 ExecStart=/usr/sbin/firewalld --nofork --nopid $FIREWALLD_ARGS (code=exited, status=0/SUCCESS)
  Main PID: 857 (code=exited, status=0/SUCCESS)

feb 25 19:24:57 localhost.localdomain systemd[1]: Starting firewalld - dynamic firewall daemon...
feb 25 19:24:59 localhost.localdomain systemd[1]: Started firewalld - dynamic firewall daemon.
feb 25 20:21:11 localhost.localdomain systemd[1]: Stopping firewalld - dynamic firewall daemon...
feb 25 20:21:12 localhost.localdomain systemd[1]: Stopped firewalld - dynamic firewall daemon.
[root@localhost ~]# systemctl disable firewalld
Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
[root@localhost ~]#
```

Ilustración 17. Estado firewall desactivado

Una vez deshabilitado el servicio que pueda entorpecer al proceso de llamadas internas entre extensiones definidas en el archivo de configuración sip.conf, se procede a utilizar el Softphone comentado en el punto anterior, configurándolo y realizando una llamada.

Para empezar, te pide un usuario y contraseña para iniciar sesión, y comenzar a utilizar el Softphone; este usuario será el definido en el archivo sip.conf (cualquiera de ellos), seguido de la dirección IP local donde está ubicado el servicio de Asterisk.

Hay que explicar que, en este proyecto, como no se utilizan troncales para poder salir al exterior, ya que tiene un coste, se utiliza la IP local del servidor. Al estar en la misma red, no se ha encontrado ningún tipo de problema a la hora de configurar los Zoiper dentro de cualquier ordenador de la red. Sin embargo, si se realizara a nivel empresarial y funcional, la IP por defecto que debería tener sería una IP pública, haciendo referencia al propio servidor.

Una vez definida la IP y el puerto en el archivo sip.conf, se pone la contraseña del usuario, y se hace clic en “siguiente”:

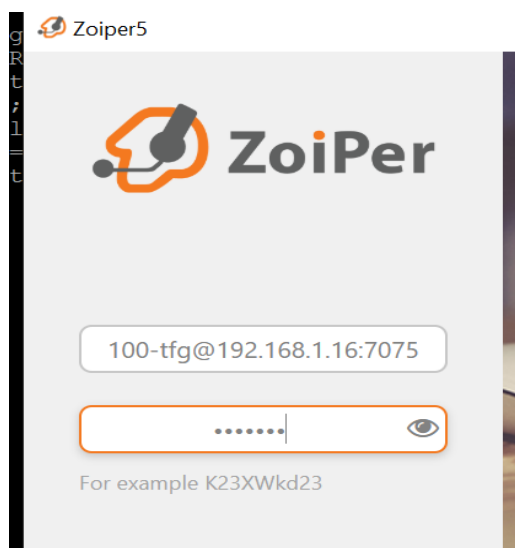


Ilustración 18. Configuración de Zoiper – Extensión.

Como se puede apreciar, el propio Softphone empieza a detectar los servicios utilizados y en qué configuración está realizada. En este caso, se tratará de SIP UDP.

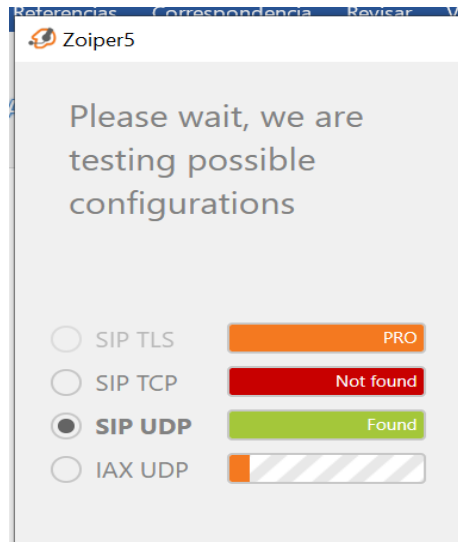


Ilustración 19. Configuración de SIP Zoiper.

Es curioso comentar que el debug del servicio Asterisk detecta cualquier situación nueva, problemas, conexiones, llamadas, etc.

Como se aprecia en la imagen siguiente, se puede ver como se registra la extensión correspondiente, y el proceso que conlleva:

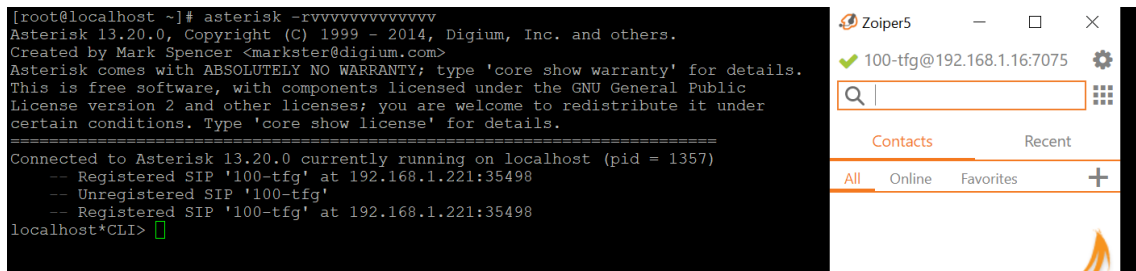


Ilustración 20 . Debug Asterisk - Registro Zoiper

Para finalizar esta demostración, se han instalado varios Softphone de Zoiper (versiones distintas) para poder registrar extensiones diferentes, y poder realizar llamadas entre ellas.

Como se puede ver en la imagen, se llama de la extensión 100 a la 101; éste recibe la llamada, y se establece la conexión necesaria para poder realizar esta operación con éxito.

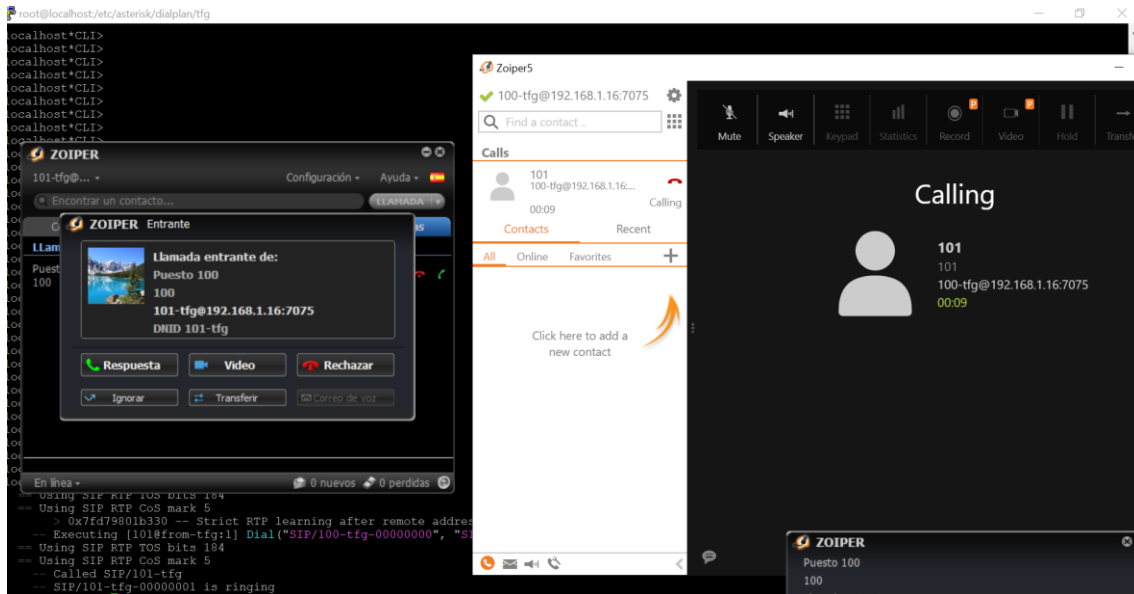


Ilustración 21. Realización llamadas internas Zoiper

Puesto que esto es la explicación básica, faltaría por definir más archivos como se han mostrado en el organigrama del servicio. Sin embargo, no se van a explicar con más detalle ya que no son tan fundamentales como los archivos anteriores explicados, puesto que es información adicional.

4.3.3. Servicio Asterisk Realtime

ARA (The Asterisk Realtime Architecture) es una funcionalidad que permite almacenar los archivos de configuración de Asterisk y/u objetos en una base de datos. Existen dos tipos de Realtime:

- Estático
- Dinámico

Realtime estático:

Con el Realtime estático es posible guardar los archivos de configuración de Asterisk en una base de datos. Se dice estático porque, cuando se realizan cambios en la base de datos, hay que recargar la configuración de Asterisk como si la configuración estuviera en los archivos de texto. Otra característica del Realtime estático es que, si las configuraciones se guardan en la base de datos, ya no se podrán utilizar los archivos de configuración textuales, es decir que, si por ejemplo se guarda la configuración del dialplan en Realtime estático, la PBX no leerá mas el archivo extensions.conf.

Realtime Dinámico:

El Realtime dinámico permite guardar objetos en una base de datos. Estos objetos pueden ser: extensiones/endpoints, colas de espera, agentes, buzones de voz, etc.



Cuando se efectúa un cambio en la base de datos, Asterisk actualizará la configuración en tiempo real. A diferencia del Realtime estático, los objetos creados en la base de datos pueden convivir con aquellos presentes en los archivos de configuración.

A efectos prácticos, se va a proceder a utilizar el Realtime para llevar un registro de llamadas dónde quede reflejada cada llamada realizada, recibida etc. Para esto es necesario configurar varios archivos:

-Primero es necesario crear una base de datos con el servicio instalado en el principio del proyecto (MariaDB), que incluye Asterisk:

```
Mysql -> create database asteriskcdr;
```

Se otorgan los permisos de acceso a la base de datos creada al usuario asterisk (clave tfg), desde local:

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON asteriskcdr.* TO 'asterisk'@'localhost'  
IDENTIFIED BY '1234';
```

Luego se crea la tabla correspondiente, que cogerá los datos necesarios como son el ID, número origen, número destino, etc.

Creada la base de datos y la tabla, se configura el conector ODBC. Esto se realiza en dos archivos de configuración:

- odbinst.ini
- odbc.ini

En el primero se configuran las conexiones ODBC → motores base de datos.

En el segundo se configuran la base de datos.

Open DataBase Connectivity (ODBC) es un estándar de acceso a las bases de datos desarrollado por SQL Access Group (SAG) en 1992. El objetivo de ODBC es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué sistema de gestión de bases de datos (DBMS) almacene los datos [11].

Del primer archivo se deja la configuración predefinida, mientras que, en el segundo, al final del archivo se copian las siguientes líneas que crean una conexión a la base de datos MariaDB cuyo nombre es asteriskcdr:

```
vim /etc/odbc.ini
```

```
[asteriskcdr]  
Description = MySQL Asterisk  
Driver = MySQL  
Database = asterisk  
Server = localhost  
User = asterisk  
Password = tfg  
Port = 3306  
Option = 3
```

Para probar la conexión entre ODBC y la base de datos Asterisk se usa el comando:

```
“isql asteriskcdr asterisk tfg”
```



El siguiente paso es configurar el archivo de configuración `res_odbc.conf`:

```
vim /etc/asterisk/res_odbc.conf
```

```
[asterisk]
enabled => yes
dsn => asteriskcdr
username => asterisk
password => tfg
pre-connect => yes
```

Dónde la línea más importante es `dsn => asteriskcdr`. El valor `asteriskcdr` es el que da inicio al bloque que se acaba de crear en el archivo `odbc.ini`, haciendo referencia a la propia base de datos definida anteriormente.

Una vez configurados estos archivos, se reinicia el servicio asterisk: `systemctl restart asterisk`.

Al terminar con toda la configuración, y archivos enlazados con la base de datos, cada llamada realizada se guardará en la propia base de datos creada como `asteriskcdr`, que se podrá consultar con una simple consulta en SQL.

id	accountcode	src	dst	dcontext	clid	channel	dstchannel	lastapp
1	NULL	100	101	EMPRESA1	"100" <100>	SIP/100-00000000	SIP/101-00000001	Dial
2	NULL	101	100	EMPRESA1	"101" <101>	SIP/101-00000002	SIP/100-00000003	Dial

Ilustración 22. Registros Realtime: Base de datos.

Esta tabla (CDR) es una tabla que obtiene ese nombre, y que está definida a propósito para guardar estos registros de llamadas.

El CDR (Call Detail Record) es un registro «log» que gestiona y almacena todo el detalle de llamadas que se realizan a través de Asterisk por lo que, tanto para las empresas que necesitan llevar un control riguroso de llamadas, como para los proveedores que utilizan el CDR de Asterisk para poder facturar a sus clientes, este registro es de vital importancia. El propio servicio de Asterisk puede ofrecer uno por defecto, pero no es muy completo, por lo que generalmente las empresas suelen optar por realizar su propio CDR con más detalles [12].

Esta definición será muy importante para el objetivo del proyecto, al cual se va proceder en el siguiente punto, centrándonos en conseguir la replicación de nuestro servidor.



5. Replicación en Servidores

Los principales objetivos de este apartado son los siguientes:

1. Replicar la base de datos para que el CDR no se detenga.
2. Crear un cluster donde se ubicará “/etc/asterisk/”, “/tftboot” y “/var/lib/asterisk”.
3. Configurar una IP Virtual que se asignará a una u otra centralita según su estado, aplicado como “failover”.

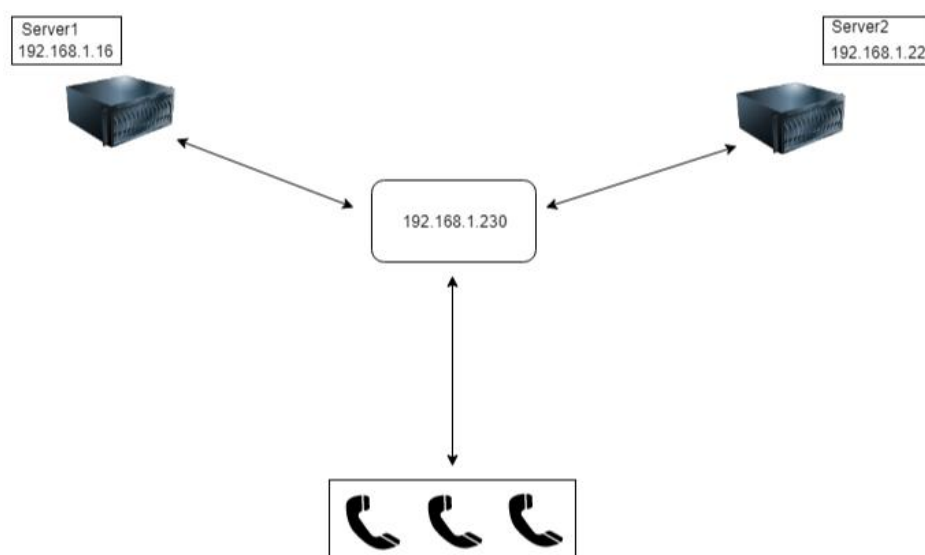


Ilustración 23. Esquema Principal de Replicación.

Antes de empezar, es necesario que se aclare el concepto de cluster, y en qué contexto se va a utilizar. El término clúster (del inglés cluster, que significa grupo) se aplica a los conjuntos de servidores unidos entre sí normalmente por una red de alta velocidad y que se comportan como si fuesen un único servidor. Esta tecnología ha ido evolucionando según la frecuencia de actividades que han ido ocurriendo, desde aplicaciones de supercómputo, como software para aplicaciones más críticas, servidores web, comercio electrónico, o bases de datos muy potentes.

El cómputo con clústeres surge como resultado de la convergencia de varias tendencias actuales que incluyen la disponibilidad de microprocesadores económicos de alto rendimiento y redes de alta velocidad, el desarrollo de herramientas de software para cómputo distribuido de alto rendimiento, así como la creciente necesidad de potencia computacional para aplicaciones que la requieran.

Simplemente, un clúster es un grupo de múltiples ordenadores unidos mediante una red de alta velocidad, de tal forma que el conjunto es visto como un único ordenador, más potente que un PC común [13].

Configuración general:

A continuación, se irán describiendo puntos principales de la configuración necesaria que requiere ambos servidores para que la replicación se realice correctamente:

-Nombre del host:

Se va a editar el nombre en la dirección: `"/etc/hostname"` de cada máquina, y se pondrá el correspondiente a nuestro esquema. Por ejemplo, en la 192.168.1.16 pondremos el host `"Server1"`.

-DNS local:

Es necesario escribir en el fichero `"/etc/hosts"` de cada servidor lo siguiente:

```
192.168.1.16 server1
```

```
192.168.1.22 server2
```

El archivo hosts es usado por el sistema operativo para guardar la correspondencia entre dominios de Internet y direcciones IP. Este es uno de los diferentes métodos que usa el sistema operativo para resolver nombres de dominio.

-Seguridad:

Se tiene que deshabilitar la opción selinux en el directorio: `"/etc/selinux/config"` y el firewall: `"systemctl stop firewalld"`, `"systemctl disable firewalld"`, `"systemctl stop iptables"`, `"systemctl disable iptables"`. Estas opciones ya se habían definido en la instalación del propio servicio de Asterisk, pero es necesario remarcarlos, ya que esta replicación tampoco será satisfactoria sin aplicar estos cambios.

-Red:

La configuración de red para estas máquinas debe ser estática. Esto se realizará en el fichero `"/etc/sysconfig/network-scripts/nombretarjetadared"`, aunque esta parte ya fue explicada en el punto de instalación de las máquinas virtuales del VMware ESXI, para poder tener conexión a Internet.

-Disco o partición:

Para la configuración del cluster va a ser necesario un disco o una partición vacía en cada servidor. Para ello, ejecutar el comando `"lsblk"` para ver qué hay conectado.




```
[root@server1 etc]# lsblk
-bash: lsblk: no se encontró la orden
[root@server1 etc]# sblk
-bash: sblk: no se encontró la orden
[root@server1 etc]# lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                  8:0    0   50G  0 disk
├─sda1                8:1    0    1G  0 part /boot
├─sda2                8:2    0   15G  0 part
├─centos-root        253:0    0  13,4G  0 lvm  /
├─centos-swap        253:1    0    1,6G  0 lvm  [SWAP]
├─sda3                8:3    0   34G  0 part
├─drbd1             147:1    0   34G  0 disk /home/data
└─sr0                 11:0    1   4,3G  0 rom
```

Ilustración 24. Discos conectados en cada Servidor

Para realizar la partición se ejecutan los siguientes comandos:

`Fdisk /dev sda , P, n, Primary`

y los demás parámetros toman los valores por defecto. Una vez configurada la partición, volverá al principio del proceso, y se guardarán los cambios (w).

Luego hay que darle un tipo de extensión al disco duro:

`mkfs.ext4 /dev/sda1` → sda1 será la partición creada de disco duro “sda” antes mencionado.

Luego se procede a montar esta partición creada:

`mount /dev/sdb1 /mnt`

Para comprobar que todo es correcto, entrar en el directorio “`cd /mnt`” y, si aparece un fichero “lost+found”, es que está montada la partición.

-Base de datos:

Esta replicación será del tipo Master-Master. En los dos servidores la base de datos se llamará “asteriskcdr”, el usuario será “root”, y la contraseña “tfg”.

Para ello se inicia el servicio de MariaDB, dedicada a base de datos, se inicia sesión, y seleccionamos la base de datos creada (asteriskcdr):

```
systemctl start mariadb
systemctl enable mariadb
mysql o mysql -u root -p y presionamos enter.
create database asteriskcdr;
use asteriskcdr
```

A partir de este punto, se tendrá que realizar la configuración en ambos servidores, por lo que se irá alternando la configuración necesaria y más importante para el funcionamiento de esta replicación:

Server1:

1. Se crea una carpeta donde guardar los archivos “logs” binarios de MariaDB, y se cambia el propietario.

1.1 `mkdir -p /var/log/mysql`

1.2 `chown mysql:mysql /var/log/mysql`

2. Después, en la consola de la Base de datos en MariaDB, se crea un usuario con privilegios de replicación:

2.1 `mysql -u root -p`

2.2 `GRANT REPLICATION SLAVE ON *.* TO 'server2'@'192.168.1.22' IDENTIFIED BY 'tfg';`

2.3 `flush privileges;` (flush es como si fuera un commit, confirma los cambios producidos en la base de datos).

3. Se modifica el fichero “/etc/my.cnf”, y en la etiqueta [mysqld] y se añaden las siguientes líneas, especificando parámetros de configuración previos de la base de datos:

```
server-id = 10
auto_increment_increment = 10
auto_increment_offset = 1
log_bin = /var/log/mysql/mysql-bin.log
expire_logs_days = 10
max_binlog_size = 100M
replicate-do-db = asteriskcdr
sync_binlog = 1
Reinicio de MariaDB: “systemctl restart mariadb”
```

4. Acto seguido, en la consola, se bloquea la lectura de todas las bases de datos, para garantizar que no se produce ningún cambio mientras se realiza el proceso de replicación:

1. `mysql -u root -pIntecna009`

2. `FLUSH TABLES WITH READ LOCK;`

3. `SHOW MASTER STATUS;`

4. Aparecerá algo parecido a la captura siguiente, que se tendrá que recordar o apuntar para más adelante:



File	Position	Binlog_Do_DB	Binlog_Ignore_DB
mysql-bin.000001	15174		

Ilustración 25. Comprobación BD Server 1.

5. **SIN CERRAR LA VENTANA ANTERIOR**, se abre otro terminal ssh mediante el programa “putty” explicado al principio del proyecto, y se hace un backup de la base de datos.
 1. `mysqldump -u root -ptfg asteriskcdr > asteriskcdr.sql`
 2. Se copia de la máquina virtual Server1 a la otra máquina Server2.
 3. `scp *.sql root@192.168.1.22:/tmp`
6. Se regresa al terminal ssh del paso 4 y se vuelven a desbloquear las tablas:

6.1 UNLOCK TABLES;

En el Server2:

1. Se crea una carpeta donde guardar los archivos “logs” binarios de MariaDB, y se cambia su propietario, tal y como se ha realizado en el apartado anterior en referencia al servidor 1.
2. Se crea de nuevo la base de datos “asteriskcdr”, donde se importará la base de datos del Server1. El procedimiento es exactamente igual que con el servidor 1.
3. Una vez creada la base de datos, se importa la misma que la del servidor 1.
 - 3.1 `cd /tmp`
 - 3.2 `mysql -u root -ptfg asteriskcdr < asteriskcdr.sql`
4. Se crea un usuario con permisos de replicación:
 - 4.3 `mysql -u root -ptfg`
 - 4.4 `GRANT REPLICATION SLAVE ON *.* TO 'server1'@'192.168.1.16' IDENTIFIED BY 'tfg';`
 - 4.5 `Flush privileges;`
4. Después se edita el fichero “/etc/my.cnf” y la etiqueta [mysqld], tal y como se explicó con el anterior servidor.



5 Se entra a la consola y se vuelve a bloquear la lectura de todas las bases de datos:

5.1 FLUSH TABLES WITH READ LOCK;

5.2 SHOW MASTER STATUS;

File	Position	Binlog_Do_DB	Binlog_Ignore_DB
mysql-bin.000001	245		

Ilustración 26. Comprobación BD server 2

5.3 UNLOCK TABLES;

Como se puede ver con las dos imágenes anteriores, se puede comprobar que la posición de ambas es distinta, pero, sin embargo, el archivo ya es el mismo.

6. El siguiente paso es conectar la máquina virtual Server2 a la máquina virtual Server1, desde el propio Server2.

6.1 mysql -u root -ptfg

```
6.2 CHANGE MASTER TO MASTER_HOST='192.168.1.16',
    MASTER_USER='server2',
    MASTER_PASSWORD='tfg',
    MASTER_LOG_FILE='mysql-bin.000001',
    MASTER_LOG_POS=15174;
```

Los datos de la consulta anterior están sacados de la captura de la configuración del Server1.

6.3 START SLAVE;

6.4 Y se comprueba el resultado: "SHOW SLAVE STATUS\G":

Debe salir algo similar a lo siguiente:

```
*****row*****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.1.16
Master_User: server2
Master_Port: 3306
Connect_Retry: 60
Master_Log_File: mysql-bin.000001
Read_Master_Log_Pos: 1718240
Relay_Log_File: mariadb-relay-bin.000002
```



```
Relay_Log_Pos: 1703595
Relay_Master_Log_File: mysql-bin.000001
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB: asteriskcdr
Replicate_Ignore_DB:
Replicate_Do_Table:
Replicate_Ignore_Table:
Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
Last_Errno: 0
Last_Error:
Skip_Counter: 0
Exec_Master_Log_Pos: 1718240
Relay_Log_Space: 1703891
Until_Condition: None
Until_Log_File:
Until_Log_Pos: 0
Master_SSL_Allowed: No
Master_SSL_CA_File:
Master_SSL_CA_Path:
Master_SSL_Cert:
Master_SSL_Cipher:
Master_SSL_Key:
Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
Last_IO_Errno: 0
Last_IO_Error:
Last_SQL_Errno: 0
Last_SQL_Error:
Replicate_Ignore_Server_Ids:
Master_Server_Id: 10
```

Una vez comprobado el estado del servidor, se vuelve al servidor 1:

Server1:

1. Ahora se procede a conectar el Server1 al Server2:
 - 1.1 mysql -u root -ptfg
 - 1.2 CHANGE MASTER TO MASTER_HOST='192.168.1.22',
MASTER_USER='server1',
MASTER_PASSWORD='tfg',
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=245;
 - 1.3 START SLAVE;
 - 1.4 "SHOW SLAVE STATUS\G" y tal y como se ha visto en el anterior servidor,
ahora debería mostrar algo similar a esto:

```
*****row *****
Slave_IO_State: Waiting for master to send event
Master_Host: 192.168.1.22
Master_User: Server1
```



```
Master_Port: 3306
Connect_Retry: 60
Master_Log_File: mysql-bin.000001
Read_Master_Log_Pos: 245
Relay_Log_File: mariadb-relay-bin.000002
Relay_Log_Pos: 529
Relay_Master_Log_File: mysql-bin.000001
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB: asteriskcdr
Replicate_Ignore_DB:
Replicate_Do_Table:
Replicate_Ignore_Table:
Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
Last_Errno: 0
Last_Error:
Skip_Counter: 0
Exec_Master_Log_Pos: 245
Relay_Log_Space: 825
Until_Condition: None
Until_Log_File:
Until_Log_Pos: 0
Master_SSL_Allowed: No
Master_SSL_CA_File:
Master_SSL_CA_Path:
Master_SSL_Cert:
Master_SSL_Cipher:
Master_SSL_Key:
Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
Last_IO_Errno: 0
Last_IO_Error:
Last_SQL_Errno: 0
Last_SQL_Error:
Replicate_Ignore_Server_Ids:
Master_Server_Id: 20
```

Para probar esta replicación que se acaba de configurar, bastaría con parar el servicio de MariaDB: “systemctl stop mariadb” y realizar llamadas internas para que se inserten en el “cdr” los distintos registros. Para eso hay que seguir los apartados de configuración del servicio de Asterisk explicados en puntos anteriores.

Aquí sería el punto donde se configuraría el realtime de la base de datos y el CDR haciendo conexión con el ODBC, tal y como se ha explicado en el punto 4.3.3.

Después de estos apartados se prosigue con el proceso de replicación:

Siguiendo por el Server2, tendrá en su base de datos los datos registrados en el CDR al realizar las pruebas anteriores.



5.1. Cluster DRBD

5.1.1. Instalación

Explicada la terminología de “Cluster” en el principio del punto 5, ahora se procederá a la ejecución e instalación en ambos servidores, y se ejecutarán los siguientes comandos e instrucciones:

```
rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org
rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-3.el7.elrepo.noarch.rpm

yum install pcre-devel xmlrpc-c-devel lynx -y
yum install drbd84-utils kmod-drbd84
```

5.1.2. Configuración

La partición que se utilizará como cluster será “/dev/sda3” y será la misma en los dos servidores, al igual que el tamaño.

1. Se editará el fichero “/etc/drbd.conf” de AMBOS servidores y se le añaden las siguientes líneas, enlazando cada servidor con la partición del cluster, cada una con su respectiva IP:

```
1. global {
2.   usage-count yes;
3. }
4. common {
5.   protocol C;
6. }
7. resource data {
8.   net {
9.     allow-two-primaries;
10.    after-sb-0pri discard-zero-changes;
11.    after-sb-1pri discard-secondary;
12.    after-sb-2pri disconnect;
13.  }
14.  startup {
15.    become-primary-on both;
16.  }
17.  on server2 {
18.    device /dev/drbd1;
19.    disk /dev/sda3;
20.    address 192.168.1.22:7789;
21.    meta-disk internal;
22.  }
23.  on server1 {
24.    device /dev/drbd1;
25.    disk /dev/sda3;
26.    address 192.168.1.16:7789;
27.    meta-disk internal;
28.  }
29. }
```



2. Después se crearán los metadatos que utilizará el cluster: “drbdadm create-md data”

Esto debe ser el resultado:

```
initializing activity log
NOT initializing bitmap
Writing meta data...
New drbd meta data block successfully created.
success
```

3. Después se habilita el cluster para que se inicie cuando arranque la máquina

```
systemctl enable drbd
```

4. Se carga el módulo DRBD en el Kernel:

```
modprobe drbd
```

5. Y se crea la conexión entre ambos servidores servidores, mediante el comando:

```
drbdadm up data
```

Una vez configurado todos los aspectos importantes del cluster, seguimos con la sincronización:

En Server1

1. Se sincroniza los datos de la partición
 1. drbdadm -- --overwrite-data-of-peer primary data
 2. Si tarda mucho, se puede ver el progreso en el fichero “/proc/drbd”

Una vez acabado también se puede ver que ha escrito en el /proc/drbd del Server2.

En Server2

Se sincroniza los datos igualmente en el servidor 2, realizando los mismos pasos que con el servidor 1, especificado en el apartado anterior.

Una vez finalizado el proceso, se podrá observar en el fichero “/proc/drbd” que los dos archivos son “Primary”:

```
1: cs:Connected ro:Primary/Primary ds:UpToDate/UpToDate C r----- ns:0 nr:1024000
dw:1024000 dr:2072 al:16 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0
```



En Server1

1. Después se crea el directorio donde se montará este cluster configurado:

```
mkdir -p /home/data
```

2. Se le tendrá que dar formato al cluster:

```
mkfs.ext4 /dev/drbd1
```

3. Luego se procede a montar el disco cluster en el directorio creado anteriormente:

```
mount /dev/drbd1 /home/data
```

4. Para realizar la prueba, se copiará un fichero de prueba dentro del directorio para comprobar que se replican correctamente:

```
touch /home/data/prueba1
```

5. Y después de realizar las pruebas pertinentes, se tendrá que desmontar el disco cluster, ya que solo puede estar montado en 1 máquina simultáneamente:

```
umount /home/data
```

En Server2

1. Se vuelve a crear la carpeta donde se montó el cluster en el servidor 1:

```
mkdir -p /home/data
```

2. Se vuelve a montar cluster en el directorio, igual que con el servidor 1:

```
mount /dev/drbd1 /home/data
```

3. Si se puede obtener un listado de /home/data, debería mostrarse el fichero que se ha creado desde el Server1. Si no muestra ese fichero, habría que revisar la configuración del cluster, ya que algún aspecto de configuración tiene que estar con algún error.

En Server1

Ahora se procederá a poner los directorios principales del servicio Asterisk para replicarlos.

En este caso /etc/asterisk, /var/lib/asterisk/ y /tftboot en el cluster. Esto se realizará a través de enlaces simbólicos:

1. Se vuelve a montar el disco cluster

```
mount /dev/drbd1 /home/data
```



2. Se crea el directorio “etc”, “tftboot” y “var/lib” en el cluster:

```
mkdir -p /home/data/etc  
mkdir -p /home/data/tftboot  
mkdir -p /home/data/var/lib
```

3. Se mueve la carpeta de Asterisk, tftboot y librerías a la carpeta recién creada:

```
mv /etc/asterisk /home/data/etc/asterisk  
mv /tftboot /home/data/tftboot  
mv /var/lib/asterisk /home/data/var/lib/asterisk
```

4. Una vez se tiene todo bien localizado, se crean los enlaces simbólicos apuntando a las carpetas nuevas:

```
ln -s /home/data/etc/asterisk /etc/asterisk  
ln -s /home/data/tftboot /tftboot  
ln -s /home/data/var/lib/asterisk /var/lib/asterisk
```

En Server2

1. Se vuelve a montar el disco cluster:

```
mount /dev/drbd1 /home/data
```

2. Se borra los directorios “/etc/asterisk”, “/var/lib/asterisk” y “/tftboot”

```
rm -rf /etc/asterisk  
rm -rf /var/lib/asterisk  
rm -rf /tftboot
```

3. Y se crean los enlaces simbólicos al igual que con el servidor 1 que apuntan al cluster:

```
ln -s /home/data/etc/asterisk /etc/asterisk  
ln -s /home/data/tftboot /tftboot  
ln -s /home/data/var/lib/asterisk /var/lib/asterisk
```

Ahora ya estaría el contenido configurable de Asterisk y los autoprovisionamientos dentro del cluster como si Asterisk estuviese instalado ahí directamente. De esta manera se alterna la replicación de archivos en ambos servidores.



6. Gestión de Alta Disponibilidad

Para ir finalizando el proyecto, se diría que hasta este punto se ha configurado la replicación de la base de datos y del disco que está “conectado” en ambas máquinas, pero que solo puede estar utilizándose por una en tiempo real.

Ahora se configurará un servicio que monitorizará ambos servidores y, dependiendo de su estado, realizará unas acciones, como, montar el cluster si no está montado, lanzar el servicio de Asterisk, o asignar una IP Virtual.

6.1. Instalación y servicios

Introduciendo el tema de alta disponibilidad y del autoservicio, se deberá parar y deshabilitar el servicio de Asterisk y del DRBD (servicio del cluster explicado en el punto anterior) debido a que se van a gestionar gracias al software que se instalará a continuación: Pacemaker y Corosync.

6.1.1. Pacemaker

Pacemaker es un gestor de recursos de Alta Disponibilidad (High Availability) Open Source utilizado en clusters de ordenadores.

El fichero de configuración de pacemaker se encuentra en `/var/lib/pacemaker/cib/cib.xml`, aunque por su complejidad nunca debe editarse directamente, sino que debe utilizarse el comando “`pcs`” para su configuración y manejo.

6.1.2. Corosync

El motor de Cluster Corosync es el encargado de gestionar un sistema de comunicación implementando la alta disponibilidad entre aplicaciones. Podríamos decir que Corosync se encarga de la «mensajería» entre los nodos del cluster.

El servicio `pacemaker_remote` permite a los nodos, que no se ejecutan en `corosync`, integrarse al clúster y hacer que el clúster administre sus recursos como si fueran nodos de clúster reales. Esto significa que los clústeres Pacemaker ahora pueden manejar entornos virtuales (KVM/LXC) y recursos que viven dentro de dichos entornos, sin que los entornos ejecuten `pacemaker` o `corosync` [14].

Como resumen a los términos anteriores:



Corosync: se encargará de la comunicación entre los nodos del cluster.

Pacemaker: se encargará de la gestión de los recursos del cluster.

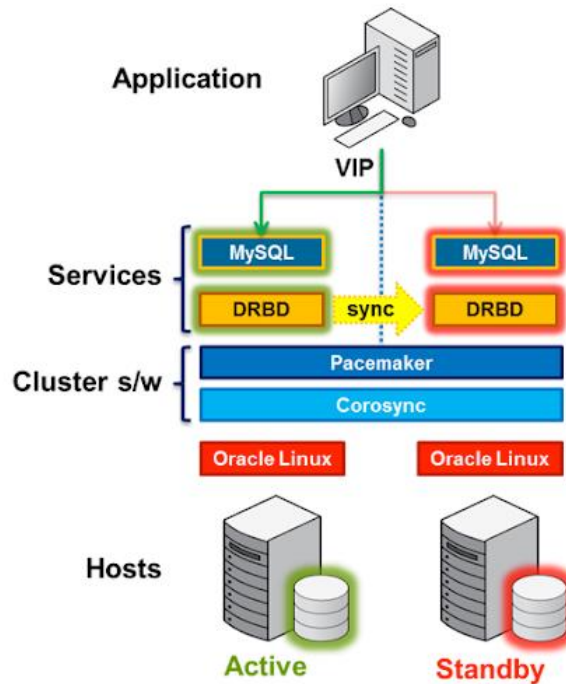


Ilustración 27. Esquema Orientativo Servicios

6.1.3. Ejecución de ambos servicios

Estas acciones se realizarán en ambos servidores:

1. Parar y deshabilitar servicios

```
systemctl stop asterisk
systemctl disable asterisk
systemctl stop drbd
systemctl disable drbd
```

2. Instalar servicio Pacemaker y sus distintas utilidades:

```
yum install pacemaker pcs psmisc policycoreutils-python -y
```

3. Lanzar y habilitar la utilidad de Pacemaker pcsd:

```
systemctl start pcsd
systemctl enable pcsd
```

4. Se crea una contraseña para el usuario hacluster, para el uso especial de este servicio:



```
passwd hacluster
```

5. Se ejecuta el comando de autenticación para el usuario recién creado:

```
pcs cluster auth Server1 Server2
```

En Server1

1. Se crea un cluster con nombre Asterisk entre los dos servidores:

```
pcs cluster setup --name asterisk --force Server1 Server2
```

Al comprobar el estado con el comando anterior, si ha ido todo correctamente tendrá que dar varios mensajes como que está todo "Success".

2. Iniciar cluster:

```
pcs cluster start --all
```

Este comando es necesario por si no te dejase acceder a la carpeta de asterisk, que es donde se encuentra toda la configuración del servicio Asterisk.

3. Y se comprueba el resultado con:

```
pcs status
```

Tendría que salir algo parecido a las siguientes líneas:

```
1. Cluster name: asterisk
2.
3. WARNINGS:
4. No stonith devices and stonith-enabled is not false
5.
6. Stack: corosync
7. Current DC: Server2 (version 1.1.19-8.e17_6.2-c3c624ea3d) -
  partition with
8. quorum
9. Last updated: Sun Dec 23 17:24:16 2018
10. Last change: Sun Dec 23 17:23:07 2018 by hacluster via crmd on
  Server2
11.
12. 2 nodes configured
13. 0 resources configured
14.
15. Online: [ Server1 Server2 ]
16.
17. No resources
18.
19. Daemon Status:
20. corosync: active/disabled
21. pacemaker: active/disabled
22. pcsd: active/enabled
```



4. Después se tendrá que deshabilitar el servicio de STONITH para que no salten warnings y no interfieran en nuestro objetivo. Este servicio es irrelevante, ya que no se utiliza.

```
pcs property set stonith-enabled=false
```

```
pcs property set no-quorum-policy=ignore
```

5. Luego se crea un recurso de IP para el cluster:

```
pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.1.230
cidr_netmask=24 op_monitor interval=30s
```

Si se comprueba el estado del Pacemaker mediante el comando: “pcs status” debería haberse añadido algo parecido a la siguiente imagen:

```
VirtualIP (ocf::heartbeat:IPaddr2): Started server2
```

Ilustración 28. Comprobación del estado del servidor.

Además, si se ejecuta el comando “ip a l” mostrará la IP nueva asignada a la máquina. Esta ip nueva, será la referencia de la IP flotante explicada en el principio del proyecto, el cual se irá turnando de un servidor a otro.

6. Se crea el recurso de DRBD (cluster):

```
pcs cluster cib drbd_cfg
```

```
pcs -f drbd_cfg resource create Data ocf:linbit:drbd drbd_resource=data op
monitor interval=60s
```

```
pcs -f drbd_cfg resource master Data master-max=1 master-node-max=1 \
clone-max=2 clone-node-max=1 notify=true
```

```
pcs cluster cib-push drbd_cfg
```

Ahora si se realizara un “pcs status” debería haberse añadido algo parecido a esto:

```
Master/Slave Set: Data-master [Data]
Masters: [ server2 ]
Slaves: [ server1 ]
```

Ilustración 29. Comprobación del estado de asignación de servidores.

Con esto se puede distinguir los distintos servidores activos, haciendo uso de términos como maestro o esclavo.

7. Lo siguiente será crear un recurso "Filesystem", que montará y desmontará la partición DRBD (cluster):

```
pcs cluster cib fs_cfg

pcs -f fs_cfg resource create Partition Filesystem device="/dev/drbd1"
directory="/home/data" \ fstype="ext4"

pcs -f fs_cfg constraint colocation add Partition Data-master INFINITY with-rsc-
role=Master

pcs -f fs_cfg constraint order promote Data-master then start Partition

pcs cluster cib-push fs_cfg
```

Si se comprueba el estado haciendo otra vez un "pcs status", y debe haberse añadido algo parecido a esto:

```
Partition (ocf::heartbeat:Filesystem): Started server2
```

Ilustración 30. Comprobación del estado del servidor.

8. En consecuencia, siguiendo el objetivo del proyecto, se descargará un agente que monitorice el servicio de Asterisk que compruebe el estado:

```
cd /usr/lib/ocf/resource.d/heartbeat

wget https://raw.githubusercontent.com/ClusterLabs/resource-
agents/master/heartbeat/asterisk

mkdir -p /lib/heartbeat/

cp /usr/lib/ocf/lib/heartbeat/* /lib/heartbeat
```

Se edita el fichero "/usr/lib/ocf/resource.d/heartbeat/asterisk" y se sustituye este bloque:

```
1. else
2.     OCF_RESKEY_user_default="asterisk"
3.     OCF_RESKEY_group_default="asterisk"
   fi
```

Por este:

```
4. else
5.     OCF_RESKEY_user_default="root"
```



```
6.      OCF_RESKEY_group_default="root"
      fi
```

Y se le da permisos al ejecutable del fichero en ambos servidores:

```
chmod +x /usr/lib/ocf/resource.d/heartbeat/asterisk
```

9. Aquí ya se comenzarían a cambiar los parámetros del fichero sip.conf ubicado en el “/home/data/etc/asterisk”.

```
vim /home/data/etc/asterisk/sip.conf
```

Modificando las siguientes líneas:

```
1. udpbindaddr=0.0.0.0:5060
2. tcpbindaddr=0.0.0.0:5060
```

Y tienen que quedar así:

```
3. udpbindaddr=192.168.1.230:5060
4. tcpbindaddr=192.168.1.230:5060
```

Cambiando enlaces de conexión del servicio, de local a la IP flotante en base a la que se irán turnando ambos servidores.

10. Por último, se configura el recurso de monitorización de Asterisk:

```
pcs cluster cib asterisk_cfg

pcs -f asterisk_cfg resource create asterisk ocf:heartbeat:asterisk op monitor
timeout=5

pcs -f asterisk_cfg constraint colocation add asterisk with Partition
score=INFINITY

pcs -f asterisk_cfg constraint order Partition then asterisk

pcs cluster cib-push asterisk_cfg

pcs resource defaults resource-stickiness="INFINITY"
```

Al realizar el comando: “pcs status” debe haberse añadido algo parecido a esto indicando que está activo estos servicios ahora mismo en el server2, después de aplicar una caída del servidor 1, demostrando que cambia automáticamente el servicio ante una caída o problema del servidor:

```
asterisk      (ocf::heartbeat:asterisk):      Started server2
```

Ilustración 31. Comprobación final estado servidor



6.1.4. Testing Replicación

Para asegurarnos de que todo funciona correctamente, se puede simular una parada en uno de los servidores, apagando dicho servidor. Después, se lanza el comando “pc status” y se verá en qué servidor están los recursos con el estado “Started”, permitiendo verificar, como se ha comentado en el punto anterior, donde está activada la ejecución de los servicios.

También se puede apreciar este cambio lanzando este comando “pcs cluster standby server1”, por lo que se verá que la activación de los recursos y servicios ha cambiado de un servidor al otro, demostrando la replicación, y la garantía de un servicio con una gran disponibilidad.

Haciendo alusiones a las ilustraciones 30 y 31, donde se simula la parada en cada servidor, se puede observar qué servidor está activo en ese momento.

El tiempo de recuperación tras la pérdida del sistema depende de parámetros configurados por el administrador. Gracias a los servicios de Pacemaker y Corsync, puedes hacer que compruebe el estado del servicio Asterisk, por ejemplo a los 5 segundos, como en la instrucción siguiente:

```
pcs -f asterisk_cfg resource create asterisk ocf:heartbeat:asterisk op monitor timeout=5
```

Si esto ocurre, directamente el sistema detecta una caída, y se activa el otro servidor mediante los comandos comentados en el punto anterior.



7. Conclusiones y trabajo futuro

Analizando los resultados obtenidos de todo el proyecto, de la implementación de los servicios explicados anteriormente, y del recurso de replicación para garantizar una alta disponibilidad, se puede concluir lo siguiente:

Con el desarrollo presentado, se consigue obtener y alcanzar un mayor control sobre instalaciones con servidores y servicios concretos, más próximo a cualquier realidad empresarial, ajustándose a dicha realidad.

En primer lugar, se han comentado las posibilidades que pueden ofrecer los servicios de AWS, que son servicios en cloud, como bien se ha visto en su correspondiente punto. Permiten que el uso de componentes físicos (hardware) sea mucho menor, ya que puedes gestionar muchos servicios y dispositivos de una manera virtual, e procede además destacar la sustanciosa reducción de instalaciones de cada componente hardware empleado. Aunque este servicio no haya sido el utilizado finalmente, ha contribuido a un descubrimiento muy práctico de las distintas posibilidades que puede ofrecer. Y aunque supone unos costes por utilización de los servicios, compensa bastante más que tener todo configurado en dispositivos físicos. Por lo que se puede deducir que esta plataforma es muy aconsejable dependiendo del tipo de servicio que quieras realizar. Con el paso del tiempo, irá tomando más importancia a nivel empresarial.

La segunda parte del proyecto y desarrollada en el mismo, ha consistido en utilizar otra plataforma alternativa a la anterior, VMware ESXI. Esta herramienta ha hecho realidad la posibilidad de desarrollar una replicación de los servidores mencionados. Aunque a efectos prácticos se tenga que instalar en un servidor físico, el mantenimiento principal que se hace al servidor es mínimo, debido a que, con esta herramienta de virtualización, se han instalado ahí las máquinas virtuales simulando otros 2 servidores, por lo que el mantenimiento de estas máquinas se puede hacer virtualmente.

Por supuesto, al tener que instalarlo en un servidor físico, éste puede suponer su mantenimiento específico, como la posibilidad de un SAI, que no se ha mencionado en este proyecto, para poder garantizar la disponibilidad del servidor en caso de cortes de luz, o problemas de conexión. También comentar que esto supone un mantenimiento económico, por lo que independientemente de usar una plataforma u otra, el precio sería bastante similar.

Como conclusiones, se llega a entender que, dependiendo del servicio que se quiera implantar, se tiene que estudiar con detalle y precisión qué componentes utilizar, qué herramientas emplear, el coste económico que puede suponer utilizar un servicio u otro, etc.

Como opinión personal, podría decirse que ha sido un proyecto algo complejo, en cuanto a que no está replicado o desarrollado por ningún sitio en Internet. De hecho, al comenzar con la plataforma AWS y dedicarle horas de investigación y esfuerzo, se llegó a la conclusión de que no permitía lograr el objetivo principal, por lo que fue esfuerzo “perdido”, y por ello se tuvo que cambiar a otra herramienta que pudiera hacer posible el objetivo final.

Realizarlo en la herramienta VMware ESXI también supuso un esfuerzo extra, tanto de conocimientos de red, como de virtualización. A medida que se iban instalando máquinas, y el servicio principal (Asterisk), surgían dificultades varias, sobre todo a nivel de red por problemas de conexión, de firewall, e incluso del router, ya que se necesita una configuración especial, desactivando opciones que impiden el correcto funcionamiento de llamadas por VoIP con SIP.



A pesar de los problemas comentados, la información encontrada en distintos foros, y sobre todo en el libro del propio Servicio Asterisk, se ha podido ir superando y solucionando cada uno de estos problemas.

Para finalizar, en referencia a trabajos futuros, destacar que todo lo explicado de las distintas plataformas tiene finalidades y desarrollos totalmente ampliables y aplicables.

En el caso de AWS, existen miles de imágenes creadas por usuarios que permiten crear distintas instancias personalizadas según el servicio empleado, por lo que, dependiendo del uso que se quiera hacer, existen opciones desarrollables en esta plataforma.

Por otro lado, en cuanto al servicio Asterisk, se pueden hacer infinidad de ampliaciones:

- Es posible adjuntar software desarrollado en este servicio, como podrían ser colas dinámicas, es decir, opciones que se suelen dar para que los clientes, con pocos conocimientos de informática en general, puedan configurarse sus propias colas de llamadas.
- Existe también posibilidad de crear locuciones personalizadas, e introducirlas interactuando con éstas, es decir, existe la posibilidad de realizar menús interactivos, configurando lo que sería un IVR de Asterisk, haciendo que si se pulsa un número en concreto te vaya a una cola de extensiones en concreto, o incluso te llame a otro número de teléfono en concreto.
- También se puede configurar buzones de voz, realizar grabaciones de llamadas, crear listas negras de llamadas (números que bloqueas, que no quieres recibir llamadas), sacar estadísticas de los registros de las llamadas, etc.

Como se puede apreciar, existen infinidad de ampliaciones y posibilidades futuras, en caso de querer ampliar el proyecto inicial, principalmente mediante configuraciones más complejas del servicio de Asterisk, haciendo que finalmente, con años de experiencia y muchas horas de investigación, se pueda mejorar la calidad y el servicio de una manera empresarial y práctica, compitiendo así con empresas potenciales, como pueden ser Vodafone o Movistar.



Bibliografía

- [1] <http://welinfocto.blogspot.com/2012/03/123-pstn.html> : Explicación PSTN (28 Septiembre 2019)
- [2] <https://es.wikipedia.org/wiki/Asterisk> : Servicio Asterisk (3 Octubre 2019)
- [3] [Curso AWS Germán Moltó](#) : Curso Germán Moltó oficial RDS (6 Octubre 2019)
- [4] <https://aws.amazon.com/es/ebs/> : Explicación Servicios EBS de AWS (7 Octubre 2019)
- [5] [Curso AWS Germán Moltó S3](#) : Curso Germán Moltó oficial S3 (8 Octubre 2019)
- [6] [Curso AWS Germán Moltó](#) : Curso Germán Moltó oficial Cloudwatch AWS (8 Octubre 2019)
- [7] <https://www.ionos.es/digitalguide/servidores/know-how/ip-flotante-definicion-y-caracteristicas/> : Definición IP Flotante (20 Octubre 2019)
- [8] <https://www.vmware.com/es/products/player/faqs.html> : Productos VMware (24 Octubre 2019)
- [9] <https://es.wikipedia.org/wiki/SELinux> : Problemas y definición Selinux (29 Octubre 2019)
- [10] <https://www.3cx.es/voip-sip/voip-definicion/> : Definición VoIP (3 Noviembre 2019)
- [10] <https://www.3cx.es/voip-sip/telefono-voip-definicion/> : Definción Teléfonos VoIP (5 Noviembre 2019)
- [11] https://es.wikipedia.org/wiki/Open_Database_Connectivity : Explicación ODBC (6 Noviembre 2019)
- [12] <http://wlaruccia.blogspot.com/2010/03/asterisk-cdr-en-mysql.html> : Definición del servicio CDR de Asterisk (12 Noviembre 2019)
- [13] [https://es.wikipedia.org/wiki/Cl%C3%BAster_\(inform%C3%A1tica\)](https://es.wikipedia.org/wiki/Cl%C3%BAster_(inform%C3%A1tica)) : Definición Cluster (21 Noviembre)
- [14] https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/6/html/configuring_the_red_hat_high_availability_add-on_with_pacemaker/pacemaker_remote : Servicios Pacemaker and Corosync (16 Enero 2020)

