



S.I.G.T.

Sistemas Operativos III

Katsu enterprise (勝つ企業)

Rol	Apellido	Nombre	C.I	Email	Tel/Cel.
Coordinador	Macedo	Fiorella	5.503.612-7	fiorellamacedo22@gmail.com	095 256 351
Sub-Coordinador	Dávila	Oriana	5.074.874-1	orianadavila99@gmail.com	093 308 483
Integrante 1	Pérez	Lautaro	5.468.712-7	pabloramirez199221@gmail.com	097 967 986
Integrante 2	Budes	Agustín	5.121.247-6	agustinbudes@gmail.com	099 431 623

Docente: Dominguez, Walter.

Fecha de culminación

13/11/2023

TERCERA ENTREGA



Índice

Índice.....	1
Estudio de los diferentes roles de los usuarios del servidor.....	2
Instalación del servidor LAMP local.....	3
Logs de auditoría creados por el equipo de trabajo.....	23
Gestión de respaldos remotos.....	29
Replicación Master Slave de MySQL.....	31
REPLICAR MYSQL (MAESTRO/ESCLAVO) EN CENTOS.....	32
Archivos cuenta con rutinas de backup y sus correspondientes scripts para el administrador.....	41
Configuración del firewall de Gnu/Linux.....	42
Filtrado de Ips mediante Firewall.....	43
Generar un servidor de respaldo de datos.....	45
Para hacer el servidor de respaldo optamos por clonar el servidor principal,.....	45
Menú para el Operador del Centro de Cómputos (Administrador del Sistema).....	49
Script Gestión de Usuarios.....	49
Script grupos.sh.....	53
Script de Redes.....	57
Script de Servicios.....	62
Script de Firewall.....	65
Script de respaldo total.....	68
Configuraciones de red en las terminales y el servidor.....	74
Configuracion SSH.....	75
Bibliografia.....	76



Estudio de los diferentes roles de los usuarios del servidor

Operador de hardware.

Los usuarios operadores son aquellos quienes tienen control sobre el Hardware de los equipos que componen el sistema, por lo tanto, los trabajadores de Katsu Enterprise son considerados usuarios operadores.

Administrador (root)

El rol de usuario administrador es el encargado de gestionar la totalidad del servidor:

Debido a que en una empresa de informática cuando se trabaja en servidores, se cuentan con diferentes perfiles, se crearán grupos en los cuales se definirá los diferentes permisos dentro del archivo sudoers. Esta configuración se realizará mediante alias y reglas de acceso.

Los perfiles que se crearán serán los siguientes:

- Administrador de paquetes
- Administrador de base de datos
- Administrador de redes/firewall
- Otras configuraciones en el sistema.



Instalación del servidor LAMP local

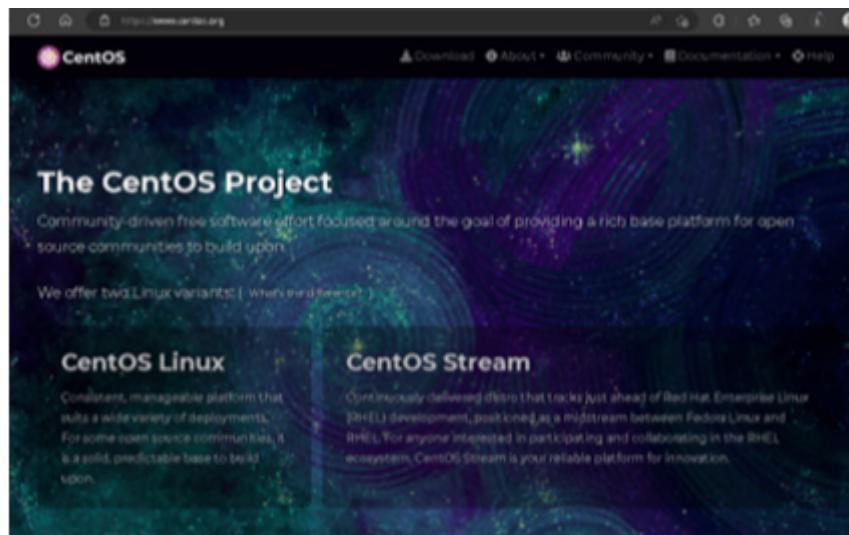
Instalación del Sistema Operativo Linux en el servidor

Manual Instalación CentOS 7

Los pasos que seguimos para la instalación del sistema operativo CentOS 7 en la versión de 64 bits fue la siguiente:

Primer Paso

- Ingresar a la página de CentOS 7: Desde el navegador de nuestra preferencia, ingresamos a la página oficial de CentOS (www.centos.org)



Segundo paso

- Descargar CentOS 7 en la versión de 64 bits:

Para eso debemos de seguir una serie de pasos:

- a) Ingresar a la pestaña que dice “Download” en la parte superior de la ventana.
- b) Elegir la opción de arquitectura “x86_64”, que corresponde a la versión de 64 bits.



Architectures	Packages	Others
x86_64	RPMs	Cloud Containers Vagrant
ARM64 (aarch64)	RPMs	Cloud Containers Vagrant
IBM Power BE (ppc64)	RPMs	Cloud Containers Vagrant
IBM Power (ppc64le)	RPMs	Cloud Containers Vagrant
ARM32 (armhf)	RPMs	Cloud Containers Vagrant
i386	RPMs	Cloud Containers Vagrant

c) Al clickear “x86_64”, nos llevará a la siguiente página, de la cual elegimos descargar el ISO del primer link disponible.

 CentOS Descargar Acerca de Comunidad Documentación Ayuda

Para conservar el ancho de banda limitado disponible, las imágenes ISO no se pueden descargar desde mirror.centos.org

Los siguientes espejos deben tener las imágenes ISO disponibles:

http://mirror.ufscar.br/centos/7.9.2009/isos/x86_64/

d) Posteriormente, elegiremos el primer link .iso, que corresponde al iso CentOS-7-x86_64-DVD-2009.iso.

Índice de /centos/7.9.2009/isos/x86_64

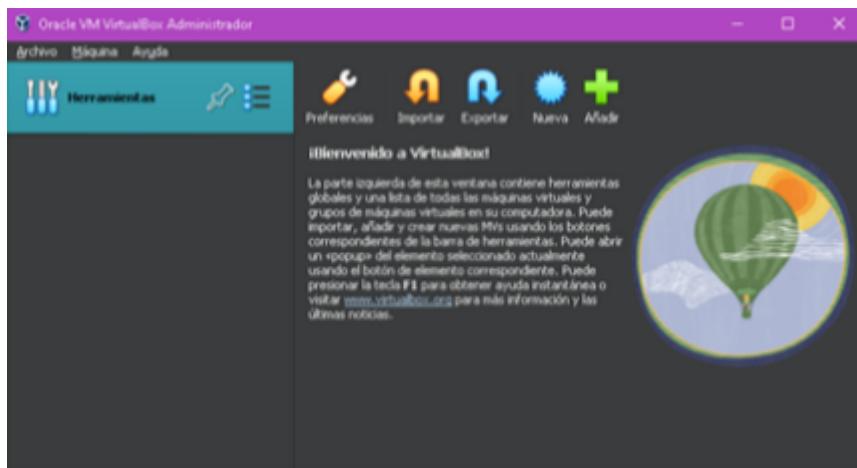
Nombre	Última modificación	Tamaño	Descripción
 directorio de padres		-	
 0_README.txt	2022-08-04 15:03	2.7K	
 CentOS-7-x86_64-DVD-2009.iso	2020-11-04 08:37	4.4G	



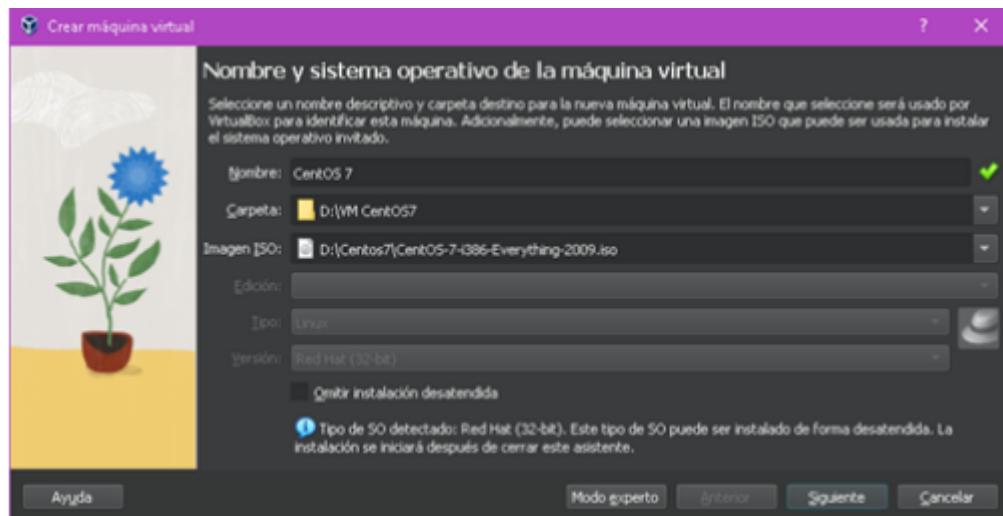
Creación de una Máquina virtual con CentOS 7 como sistema operativo

La creación de una máquina virtual con CentOS 7 debe seguir los siguientes pasos:

- 1) Abrir Oracle VM y apretar donde dice “nueva”



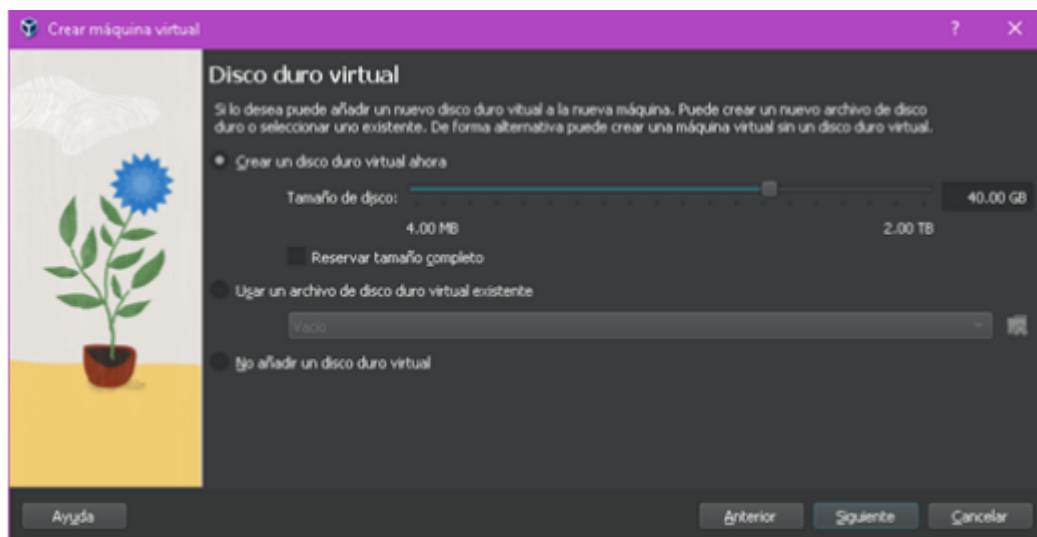
- 2) Clickear sobre el campo que dice “Imagen ISO” y seleccionar el CentOS 7 anteriormente instalado, apretamos también la opción de “Omitir instalación desatendida”.





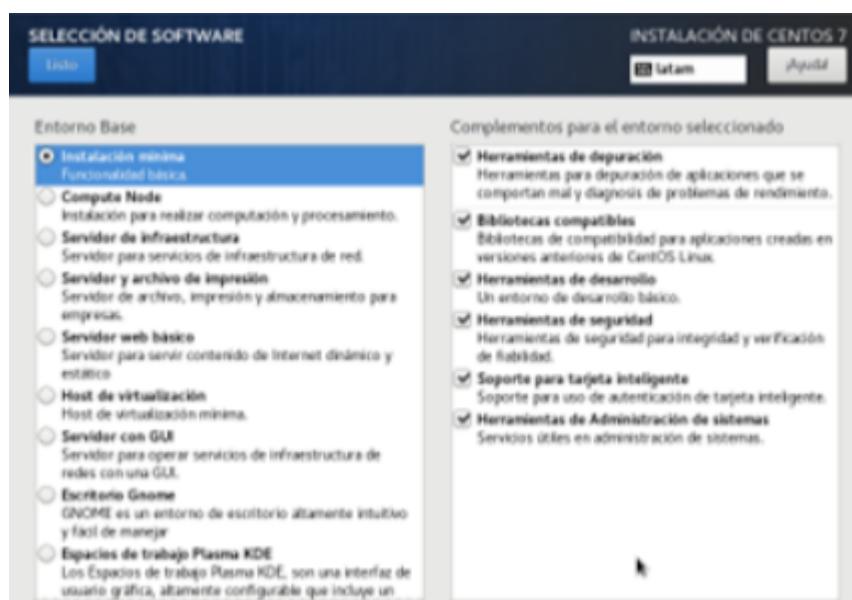
3) Elegimos la capacidad de memoria base y procesadores que queramos. En nuestro caso, decidimos optar por la opción estándar de 2048 MB y 1 procesador.

4) Al pulsar siguiente, nos proporciona la posibilidad de elegir el tamaño del disco duro, el cual cambiamos el predefinido por uno de 40GB



5) Posteriormente, solo es cliquear siguiente hasta volver a la pantalla principal, donde aparecerá nuestra máquina virtual ya creada.

6) Al ingresar a la máquina virtual, veremos una pantalla en la cual clickeamos en el apartado “Selección de Software” y elegiremos las opciones que aparecerán en las siguientes imágenes.





SELECCIÓN DE SOFTWARE

Listo

INSTALACIÓN DE CENTOS 7

latam Ayudar

Entorno Base

- Compute Node**
Instalación para realizar computación y procesamiento.
- Servidor de infraestructura**
Servidor para servicios de infraestructura de red.
- Servidor y archivo de impresión**
Servidor de archivo, impresión y almacenamiento para empresas.
- Servidor web básico**
Servidor para servir contenido de Internet dinámico y estático
- Host de virtualización**
Host de virtualización mínima.
- Servidor con GUI**
Servidor para operar servicios de infraestructura de redes con una GUI.
- Escritorio Gnome**
GNOME es un entorno de escritorio altamente intuitivo y fácil de manejar
- Espacios de trabajo Plasma KDE**
Los Espacios de trabajo Plasma KDE, son una interfaz de usuario gráfica, altamente configurable que incluye un panel, un escritorio, iconos y asistentes de escritorio y muchas aplicaciones potentes KDE.

Complementos para el entorno seleccionado

- Cliente de respaldo**
Herramientas de cliente para conectarse a un servidor de respaldo y hacer copias de seguridad.
- Aplicaciones de Internet**
Software de correo-e, chat, y video conferencias.
- Aplicaciones KDE**
Un set de las aplicaciones KDE más utilizadas.
- Soporte multimedia para KDE**
Soporte multimedia para KDE
- Compatibilidad de sistema de ventanas con legado de X**
Programas de compatibilidad para migrar desde o funcionar con sistemas de entornos X.
- Office Suite y Productividad**
Un propósito total de office suite y otras herramientas de productividad.
- Soporte para tarjeta inteligente**
Soporte para uso de autenticación de tarjeta inteligente.
- Bibliotecas compatibles**
Bibliotecas de compatibilidad para aplicaciones creadas en versiones anteriores de CentOS Linux.
- Herramientas de desarrollo**
Un entorno de desarrollo básico.

SELECCIÓN DE SOFTWARE

Listo

INSTALACIÓN DE CENTOS 7

latam Ayudar

Entorno Base

- Instalación mínima**
Funcionalidad básica.
- Compute Node**
Instalación para realizar computación y procesamiento.
- Servidor de infraestructura**
Servidor para servicios de infraestructura de red.
- Servidor y archivo de impresión**
Servidor de archivo, impresión y almacenamiento para empresas.
- Servidor web básico**
Servidor para servir contenido de Internet dinámico y estático
- Host de virtualización**
Host de virtualización mínima.
- Servidor con GUI**
Servidor para operar servicios de infraestructura de redes con una GUI.
- Escritorio Gnome**
GNOME es un entorno de escritorio altamente intuitivo y fácil de manejar
- Espacios de trabajo Plasma KDE**
Los Espacios de trabajo Plasma KDE, son una interfaz de usuario gráfica, altamente configurable que incluye un

Complementos para el entorno seleccionado

- Alta disponibilidad**
Infraestructura para servicios altamente disponibles y/o almacenaje compartido.
- Servidor de administración de identidad**
Administración centralizada de usuarios, servidores y políticas de autenticación.
- Soporte para Infiniband**
Software diseñado para soportar agrupamiento y conectividad de rejilla mediante telas InfiniBand e iWARP basadas en RDMA.
- Plataforma de Java**
Soporte de Java para Servidor de CentOS Linux y plataformas de escritorio.
- Rendimiento de grandes sistemas**
Herramientas de soporte de rendimiento para grandes sistemas.
- Equilibrador de carga**
Soporte de equilibrio de carga para tráfico de red.
- Servidor de base de datos MariaDB**
El servidor de base de datos MariaDB SQL y paquetes asociados.
- Cliente de sistema de archivos de red**
Permite al sistema conectarse a un almacenamiento de



SELECCIÓN DE SOFTWARE

[Listo](#)

Entorno Base

- Instalación mínima**
Funcionalidad básica.
- Compute Node**
Instalación para realizar computación y procesamiento.
- Servidor de infraestructura**
Servidor para servicios de infraestructura de red.
- Servidor y archivo de impresión**
Servidor de archivo, impresión y almacenamiento para empresas.
- Servidor web básico**
Servidor para servir contenido de Internet dinámico y estático
- Host de virtualización**
Host de virtualización mínima.
- Servidor con GUI**
Servidor para operar servicios de infraestructura de redes con una GUI.
- Escritorio Gnome**
GNOME es un entorno de escritorio altamente intuitivo y fácil de manejar
- Espacios de trabajo Plasma KDE**
Los Espacios de trabajo Plasma KDE, son una interfaz de usuario gráfica, altamente configurable que incluye un

INSTALACIÓN DE CENTOS

[latam](#) [Ayuda](#)

Complementos para el entorno seleccionado

- Servidor de respaldo**
Software para centralizar sus respaldos de infraestructura.
- Servidor de nombres DNS**
Este paquete le permite ejecutar un servidor de nombres DNS (BIND) en el sistema.
- Herramientas de depuración**
Herramientas para depuración de aplicaciones que se comportan mal y diagnóstico de problemas de rendimiento.
- Cliente de directorio**
Cílientes para integración en una red administrada por un servicio de directorio.
- Servidor de correo-e**
Permite al sistema actuar como un servidor SMTP y/o servidor de correo-e IMAP.
- Servidor FTP**
Permite al sistema actuar como un servidor FTP.
- Servidor y archivo de almacenamiento**
CIFS, SMB, NFS, iSCSI, iSER, y servidor de almacenamiento de red iSNS.
- Agentes de huésped**
Agentes utilizados al ejecutarse en un hipervisor.
- Herramientas de monitorización de hardware**

SELECCIÓN DE SOFTWARE

[Listo](#)

Entorno Base

- Instalación mínima**
Funcionalidad básica.
- Compute Node**
Instalación para realizar computación y procesamiento.
- Servidor de infraestructura**
Servidor para servicios de infraestructura de red.
- Servidor y archivo de impresión**
Servidor de archivo, impresión y almacenamiento para empresas.
- Servidor web básico**
Servidor para servir contenido de Internet dinámico y estático
- Host de virtualización**
Host de virtualización mínima.
- Servidor con GUI**
Servidor para operar servicios de infraestructura de redes con una GUI.
- Escritorio Gnome**
GNOME es un entorno de escritorio altamente intuitivo y fácil de manejar
- Espacios de trabajo Plasma KDE**
Los Espacios de trabajo Plasma KDE, son una interfaz de usuario gráfica, altamente configurable que incluye un

INSTALACIÓN DE CENTOS 7

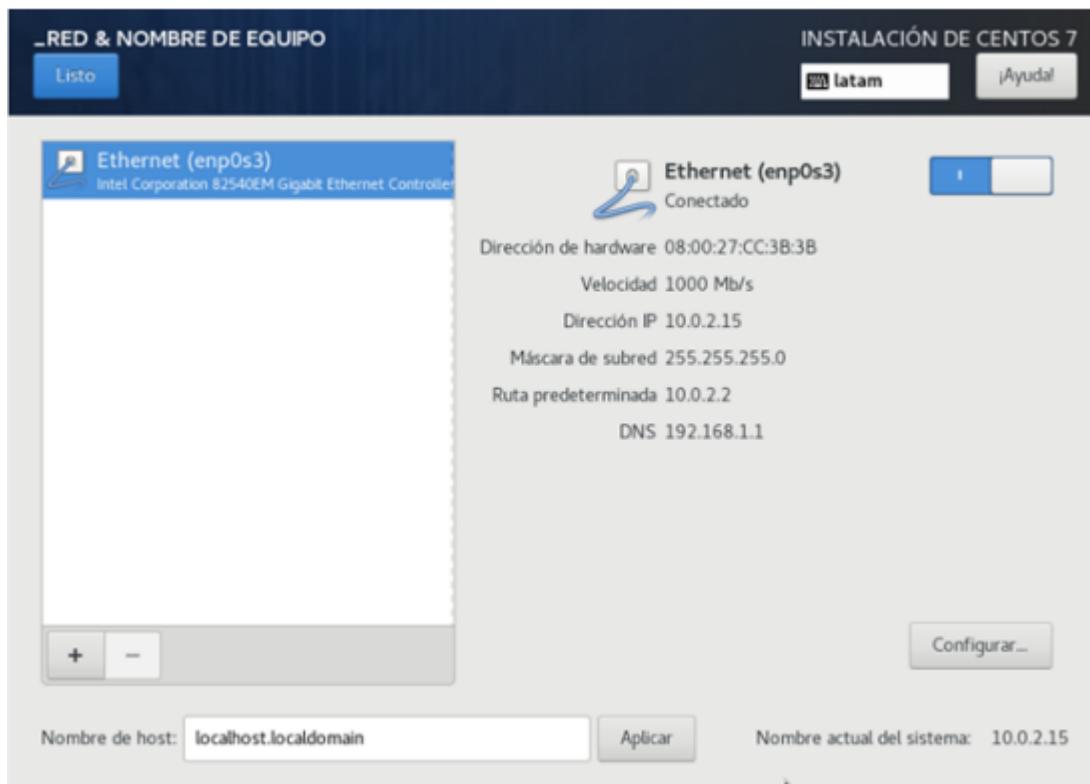
[latam](#) [Ayuda](#)

Complementos para el entorno seleccionado

- Servidor de impresión**
Permite al sistema actuar como servidor de impresión.
- Administración remota para Linux**
Interfaz de administración remota para CentOS Linux, incluidos OpenLMI y SNMP.
- Almacenamiento resistente**
Almacenaje agrupado, incluido el sistema de archivos GFS2.
- Hipervisor de virtualización**
La instalación de host de virtualización más pequeña posible.
- Bibliotecas compatibles**
Bibliotecas de compatibilidad para aplicaciones creadas en versiones anteriores de CentOS Linux.
- Herramientas de desarrollo**
Un entorno de desarrollo básico.
- Herramientas de seguridad**
Herramientas de seguridad para integridad y verificación de fiabilidad.
- Soporte para tarjeta inteligente**
Soporte para uso de autenticación de tarjeta inteligente.
- Herramientas de Administración de sistemas**
Servicios útiles en administración de sistemas.



7) Vamos al apartado de "Red y nombre del equipo" y deslizamos el "switch" que está a la derecha de "Ethernet" (enp0s3).



8) Debemos de ir a "Origen de instalación" y darle al botón verificar.



FUENTE DE INSTALACIÓN

INSTALACIÓN DE CENTOS 7

Listo **latam** **Ayuda**

¿Qué fuente de instalación desea usar?

Medio de instalación detectado automáticamente:
Dispositivo: sr0
Etiqueta: CentOS_7_i386 **Verificar**

En la red:

 Esta URL se refiere a una lista de espejos.

Repositorios adicionales

Activado	Nombre
<input type="checkbox"/>	

Nombre: Este URL apunta a una lista de espejos.

URL del proxy:

Nombre de usuario:

Contraseña:

A Debes configurar una red para usar el medio de instalación en red.

9) Una vez se haya verificado, clickeamos “Listo” e iremos al botón que dice “Siguiente”, donde empezará a descargarse todo lo que elegimos en la selección de software, y de mientras, podemos ir creando un usuario y una contraseña de root.



Con esto daríamos como finalizada la instalación completa del CentOS 7.



Instalación de Apache en CentOS 7

Saber qué es apache es esencial para ver la utilidad. Apache realmente no es un servidor web, Apache es un software que permite a nuestro ordenador convertirse en un servidor web HTTP de código abierto.

Por ejemplo, cuando un usuario escribe en su navegador una url, esa petición llegará al servidor Apache, quien permitirá que las imágenes, textos y demás componentes de la página funcionen correctamente.

Instalación:

1.

```
● ● ●  
[root@10 ~]# sudo yum update httpd
```

Éste comando actualizará todos los paquetes apache para tener la versión más reciente.

2.

```
● ● ●  
[root@10 ~]# sudo yum install httpd
```

Como se explicó anteriormente, “yum install” se utiliza para instalar paquetes o archivos, mientras que “httpd” es lo que, en este caso, queremos instalar.

3.

```
● ● ●  
[root@10 ~]# sudo firewall-cmd --permanent --add-service=http
```

Activaremos el servicio http de firewall con este comando, principalmente para que Apache pueda enviar peticiones a http.



4.

```
● ● ●  
[root@10 ~]# sudo firewall-cmd --reload
```

Comando esencial para decirle a nuestro equipo que cargue el servicio http de firewall establecido anteriormente.

5.

Iniciaremos httpd para luego ver su estado. La captura deja a la vista que el servidor está activo y corriendo.

```
[root@10 ~]# sudo systemctl start httpd  
[root@10 ~]# sudo systemctl status httpd  
● httpd.service - The Apache HTTP Server  
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)  
  Active: active (running) since jue 2023-07-20 14:54:35 -03; 2min 49s ago  
    Docs: man:httpd(8)  
          man:apachectl(8)  
  Main PID: 5447 (httpd)  
    Status: "Total requests: 0; Current requests/sec: 0; Current traffic: 0 B/sec"  
     Tasks: 6  
    CGroup: /system.slice/httpd.service  
           ├─5447 /usr/sbin/httpd -DFOR...  
           ├─5478 /usr/sbin/httpd -DFOR...  
           ├─5479 /usr/sbin/httpd -DFOR...  
           ├─5480 /usr/sbin/httpd -DFOR...  
           ├─5481 /usr/sbin/httpd -DFOR...  
           └─5482 /usr/sbin/httpd -DFOR...
```

6.

Para conocer nuestra IP de servidor, podemos utilizar el siguiente comando:

```
[root@10 ~]# hostname -I  
10.0.2.15 192.168.122.1
```



7.

Apache por defecto se iniciará cuando el servidor lo haga. En nuestro caso, no queremos que eso ocurra, por lo que pondremos el siguiente comando.

```
[root@10 ~]# sudo systemctl disable httpd
```

8.

Para finalizar la instalación, apagaremos el servidor apache.

```
[root@10 ~]# sudo systemctl stop httpd
```



Instalación de MYSQL en CentOS 7

Antes de realizar la instalación, consideramos esencial saber qué es lo que estamos instalando y para qué, por ende, a continuación definiremos qué es MYSQL y por qué lo estamos instalando.

MySQL es un sistema de gestión de bases de datos relacionales de código abierto con un modelo cliente-servidor(<https://www.hostinger.es/>), para nuestro proyecto es indispensable tener una base de datos (lugar que almacene de forma estructurada todos los datos básicos de los usuarios) para que, valga la redundancia, los datos de nuestros usuarios estén seguros.

Nuestra empresa utiliza MYSQL porque considera que es el sistema más confiable e intuitivo.

Instalación

1. Siendo usuario root en nuestra computadora (Podemos ser usuario root poniendo “su -”) ingresamos el comando que se muestra en la imagen.

yum es un comando de instalación, actualización o eliminación de grupos de paquetes, mientras que upgrade funciona para actualizar los paquetes junto con sus dependencias, continuando con la eliminación de los paquetes obsoletos y las dependencias obsoletas.

-y no realiza preguntas a la hora de actualizar.

```
[root@10 ~]# yum upgrade -y
```

2. Ingresamos El siguiente comando para instalar epel, que es un repositorio usado para instalar paquetes de software de terceros en sistemas basados en RedHat como RHEL y CentOS. Es esencial para descargar MYSQL, y hasta el momento, no existe para máquinas de 32 bits, por ende, se necesita un equipo de 64 bits.

```
[root@10 ~]# yum install -y epel-release
```



3. yum repolist nos ayuda a ver las listas de los repositorios que se tengan de yum.

```
[root@10 ~]# yum repolist
```

4. wget es un comando útil para poder guardar, instalar o recuperar archivos de cualquier sitio web, por eso se ingresó el link del archivo que queremos obtener.

```
[root@10 ~]# wget https://dev.mysql.com/get/mysql80-community-release-el7-3.noarch.rpm
```

5. rpm es un comando de código abierto, utilizado para cumplir con la gestionar paquetes de sistemas Red Hat Linux y derivados, siendo útil ahora para obtener información y datos de archivos .rpm. -Uvh son los parámetros.

```
[root@10 ~]# rpm -Uvh mysql80-community-release-el7-3.noarch.rpm
```

6. Comando que permitirá instalar mysql.

```
[root@10 ~]# yum install mysql-server
```

7. El comando mostrado a continuación nos será eficaz para saber si en el repositorio yum, está instalado mysql.

```
[root@10 ~]# yum list installed | grep mysql
```



8. Quitamos mysql80-community-release.noarch

```
[root@10 ~]# yum remove mysql80-community-release.noarch
```

9. yum clean all --verbose lo que hará es mostrar toda información sobre el número total de errores del compilador, notificando qué ensamblados carga un módulo y mostrando los archivos que se están compilando actualmente.

```
[root@10 ~]# yum clean all --verbose
```

10. En este paso, eliminaremos de forma rápida el archivo que se encuentra en la ruta marcada.

```
[root@10 ~]# rm -R -f /var/cache/yum/x86_64/7/mysql*
```

11. Se actualiza el repositorio yum para que se guarden las modificaciones.

```
[root@10 ~]# yum update
```

12. Repetimos la instalación de mysql, mencionado en el paso 5.

```
[root@10 ~]# rpm -Uvh https://repo.mysql.com/mysql80-community-release-el7-3.noarch.rpm
```



13. Este comando permite sustituir información en una cadena de carácter, en este caso, sustituimos el enabled=1 por enabled=0.

```
[root@10 ~]# sed -i 's(enabled=1)enabled=0/' /etc/yum.repos.d/mysql-community.repo
```

14. Comando para habilitar el repositorio que distribuye el software mysql.

```
[root@10 ~]# yum --enablerepo=mysql80-community install mysql-community-server
```

15. Importamos todos los archivos que posee el link.

```
[root@10 ~]# rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022
```

16. En nuestro directorio yum, permitiremos que se instale el servidor de mysql.

```
[root@10 ~]# yum --enablerepo=mysql80-community install mysql-community-server
```

17. mysql -V nos mostrará la versión descargada de mysql.

```
[root@10 ~]# mysql -V
```



18. Necesitaremos escribir el comando yum update para actualizar todos los paquetes instalados.

```
[root@10 ~]# yum update
```

19. Los siguientes comandos son necesarios para continuar con la instalación.

El primero es necesario para encender la base de datos mysql, mientras que el segundo comando nos muestra su estado.

```
[root@10 ~]# systemctl start mysqld
[root@10 ~]# systemctl status mysqld
● mysqld.service - MySQL Server
  Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; vendor preset: disabled)
  Active: active (running) since mié 2023-07-19 14:49:59 -03; 1min 9s ago
    Docs: man:mysqld(8)
          http://dev.mysql.com/doc/refman/en/using-systemd.html
   Process: 21786 ExecStartPre=/usr/bin/mysqld_pre_systemd (code=exited, status=0/SUCCESS)
   Main PID: 21868 (mysqld)
     Status: "Server is operational"
       Tasks: 37
      CGroup: /system.slice/mysqld.service
```

20. Aquí, habilitaremos el acceso a mysql, para que solo el root pueda acceder.

```
[root@10 ~]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.34

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

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

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



Usuarios del proyecto creados de acuerdo al estudio de roles

Para agregar un usuario llamado "SIGTadmin" con la contraseña "SIGTadmin" y permisos de root en CentOS 7, debes seguir estos pasos. Ten en cuenta que otorgar permisos de root a un usuario es una práctica riesgosa desde el punto de vista de la seguridad, ya que le otorga acceso completo al sistema. Deberías considerar cuidadosamente si esto es necesario en tu caso.

1. Inicia sesión en tu servidor CentOS 7 con una cuenta que ya tenga permisos de administrador.
2. Abre una terminal o utiliza SSH para acceder al servidor.
3. Para agregar un usuario, puedes utilizar el comando `useradd`. Para crear un usuario llamado "SIGTadmin" y establecer su contraseña como "admin", puedes usar los siguientes comandos:

```
sudo useradd SIGTadmin  
sudo passwd SIGTadmin
```

Después de ejecutar el segundo comando, se te pedirá que ingreses la contraseña "admin" dos veces.

4. Ahora, para otorgarle permisos de root al usuario, debes agregarlo al grupo "wheel". El grupo "wheel" se utiliza en CentOS para permitir el acceso con privilegios de root. Puedes hacerlo mediante el comando `usermod`:

```
sudo usermod -aG wheel SIGTadmin
```

Este comando agrega al usuario "SIGTadmin" al grupo "wheel".

5. Para asegurarte de que el usuario pueda usar sus privilegios de root, debes permitir que los miembros del grupo "wheel" tengan acceso al comando `sudo`. Editaremos el archivo de configuración de sudo utilizando el comando `visudo`:

```
sudo visudo
```



6. Busca la línea que dice `## Allow root to run any commands anywhere` y debajo de ella, agrega la siguiente línea para permitir que los miembros del grupo "SIGTadmin" ejecuten comandos con privilegios de root:

```
%SIGTadmin ALL=(ALL) ALL
```

Guarda el archivo y ciérralo. Esto permitirá a los miembros del grupo "SIGTadmin" usar `sudo` para ejecutar comandos con privilegios de root.

Ahora, el usuario "SIGTadmin" tiene permisos de root en CentOS 7 y puede utilizar el comando `sudo` para ejecutar comandos con privilegios elevados. Sin embargo, ten en cuenta que otorgar permisos de root de esta manera debe hacerse con precaución, ya que puede representar un riesgo de seguridad si no se gestiona adecuadamente. Asegúrate de que el usuario siga prácticas seguras de administración del sistema y solo utilice los privilegios de root cuando sea necesario.

Script que permita manejar los logs del sistema operativo

El código es un script en Awk que procesa líneas de registros de eventos relacionados con tareas programadas (cron) y extrae información como la fecha, el número de registro, el usuario original (usrOri), el tipo de operación (BEGIN, END o LIST), y el usuario final (userFin).

- El script utiliza una expresión regular (`\$7 ~ "LIST|CMD|INFO" || \$8=="EDIT"`) para filtrar las líneas que contienen las palabras clave "LIST," "CMD," o "INFO" en el campo 7 o "EDIT" en el campo 8. Estas líneas representan eventos de interés relacionados con las tareas programadas (cron).
- El script utiliza un array llamado `meses` para mapear los nombres de los meses a sus números correspondientes, lo cual es útil para formatear la fecha en un formato que se pueda procesar fácilmente.
- Luego, el script extrae información como el mes, el día, la hora, el número de registro, el usuario original, el tipo de operación (BEGIN, END o LIST), y el usuario final.
- El script también permite la depuración al imprimir información adicional si la variable `imprDep` está configurada en "Si."
- Finalmente, el script imprime la información formateada en un formato específico.



```

# primer versión, cp1, codigo procesador 1
# extraemos fecha, usuario, etc
# Datos de prueba
# [SIGTadmin@centos bin]$ sudo cat /var/log/cron
# Oct 11 18:40:01 centos CROND[6653]: (root) CMD (/usr/lib64/sa/sa1 1 1)
# Oct 11 16:58:49 centos crond[1089]: (CRON) INFO (RANDOM_DELAY will be
scaled with factor 43% if used.)
# crontab -l
# Oct 11 19:12:25 centos crontab[7613]: (SIGTadmin) LIST (SIGTadmin)
# crontab -e
# Oct 11 19:13:38 centos crontab[7656]: (SIGTadmin) BEGIN EDIT (SIGTadmin)
# Oct 11 19:13:41 centos crontab[7656]: (SIGTadmin) END EDIT (SIGTadmin)
# crontab -l desde root (crontab -l)
# Oct 11 19:15:54 centos crontab[7757]: (root) LIST (root)
# crontab -e desde root (crontab -e)
# Oct 11 19:15:57 centos crontab[7759]: (root) BEGIN EDIT (root)
# Oct 11 19:16:00 centos crontab[7759]: (root) END EDIT (root)
# crontab -l desde root, pero sobre otro usuario (crontab -u SIGTadmin -l)
# Oct 11 19:16:25 centos crontab[7761]: (root) LIST (SIGTadmin)
# crontab -e desde root, pero sobre otro usuario (crontab -u SIGTadmin -e)
# Oct 11 19:16:29 centos crontab[7762]: (root) BEGIN EDIT (SIGTadmin)
# Oct 11 19:16:32 centos crontab[7762]: (root) END EDIT (SIGTadmin)
$7 ~ "LIST|CMD|INFO" || $8=="EDIT" {
    meses["Jan"]="01"; meses["Feb"]="02"; meses["Mar"]="03"; meses["Apr"]="04";
meses["May"]="05"; meses["Jun"]="06"
    #meses["Jul"]=7;   meses["Aug"]=8;   meses["Sep"]=9;   meses["Oct"]=10;
meses["Nov"]=11; meses["Dec"]=12
    # debo colocar el 0 delante de los meses menores que 10
    meses["Jul"]="07"; meses["Aug"]="08"; meses["Sep"]="09"; meses["Oct"]="10";
meses["Nov"]="11"; meses["Dec"]="12"
    mes=$1
    dia=$2
    hora=$3
    hor=substr(hora,1,2)
    min=substr(hora,4,2)
    seg=substr(hora,7,2)

    #print de toda la linea original
    #print

    nro=$5
    usrOri=$6
    oper=$7
}

```



```
if(oper=="BEGIN" || oper=="END") {
    BE=oper
    oper=$8
    userFin=$9
} else {
    BE=""
    userFin=$8
}

# imprimir ayuda depuración
imprDep="Si"
imprDep="No"

if( imprDep=="Si") {
    print "=====
    print "imprDep = " imprDep
    print "mes : " mes, meses[mes]
    print "dia : " dia
    print "año : " vani
    print "linea original = " $0
    print "fecha = " vani meses[mes] dia
    print "hora = " hor min seg
} else
    if(yalmpreso!=1) {
        print "imprimir Depuración = " imprDep; yalmpreso=1
    }

print vani meses[mes] dia hor min seg "|" nro "|" usrOri "|" BE "|" oper "|" userFin

}
```



Logs de auditoría creados por el equipo de trabajo

El código es un script de shell en Bash que realiza varias operaciones relacionadas con registros de eventos cron en un servidor CentOS. A continuación, proporcionó una descripción de lo que hace el código:

1. La función `tstCant()` se define para realizar pruebas de cantidad y comparar resultados utilizando `grep` y `awk` en diferentes situaciones. Estas pruebas se realizan en líneas que contienen las palabras clave "LIST," "EDIT," "CMD," o "INFO" en los registros de cron. El resultado de cada prueba se imprime y se le permite al usuario comparar los resultados.
2. Se definen variables como `ani` (para el año), `mes` (para el mes), `dia` (para el día), `hor` (para la hora), y `min` (para los minutos) utilizando comandos `date` para capturar la fecha y hora actual.
3. El script permite controlar si se ejecutan las pruebas de cantidad a través de la variable `testCantidad`.
4. Finalmente, el código invoca el script en `awk` llamado `cp1.awk` y pasa la variable `ani` como un valor utilizando el comando `awk`.

El código también contiene comentarios que proporcionan información adicional sobre las acciones que se están realizando y las pruebas que se llevan a cabo.

En general, el código es una herramienta para analizar registros de eventos cron y realizar pruebas en ellos.

```
tstCant()
{
echo grep LIST
sudo cat /var/log/cron* | grep LIST
echo -n "cantidad : "
sudo cat /var/log/cron* | grep LIST | wc -l
echo awk
sudo cat /var/log/cron* | awk '$7=="LIST"
echo -n "cantidad : "
sudo cat /var/log/cron* | awk '$7=="LIST"' | wc -l
read -p 'Compare LIST ...'

echo grep EDIT
```



```
sudo cat /var/log/cron* | grep EDIT
echo -n "cantidad : "
sudo cat /var/log/cron* | grep EDIT | wc -l
echo awk
sudo cat /var/log/cron* | awk '$8=="EDIT"
echo -n "cantidad : "
sudo cat /var/log/cron* | awk '$8=="EDIT" | wc -l
read -p 'Compare EDIT ...'

echo grep 'LIST\|EDIT'
sudo cat /var/log/cron* | grep 'LIST\|EDIT'
echo -n "cantidad : "
sudo cat /var/log/cron* | grep 'LIST\|EDIT' | wc -l
echo awk
sudo cat /var/log/cron* | awk '$7=="LIST" || $8=="EDIT"
echo -n "cantidad : "
sudo cat /var/log/cron* | awk '$7=="LIST" || $8=="EDIT" | wc -l
read -p 'Compare suma de LIST mas EDIT ...'
```



```

echo 'Las cuatro especies LECI (paginadas, q para salir) grep'
sudo cat /var/log/cron* | grep 'LIST\|EDIT\|CMD\|INFO' | less
echo -n "cantidad : "
sudo cat /var/log/cron* | grep 'LIST\|EDIT\|CMD\|INFO' | wc -l
echo las cuatro especies LECI \(paginadas, q para salir\) awk
sudo cat /var/log/cron* | awk '$7 ~ "LIST|CMD|INFO" || $8=="EDIT"' | less
echo -n "cantidad : "
sudo cat /var/log/cron* | awk '$7 ~ "LIST|CMD|INFO" || $8=="EDIT"' | wc -l
read -p 'Compare TOTALES ...'

```

```
}
```

```

# main (programa principal)
#
# Autenticación - /var/log/secure
# Mensajes del sistema - /var/log/messages
# Tareas cron - /var/log/cron
# Servicio de correo - /var/log/maillog
# Errores hardware en inicio - /var/log/dmesg
# Información de arranque - /var/log/boot.log
# Paquetes instalados con yum - /var/log/yum.log
# Info sobre el motor - /var/log/mysqld.log
# Horas en que una ip se conecto al servidor - /var/log/cups/access_log
# Quienes estuvieron en el sistema - last -f /var/log/wtmp | more
# Intentos de ingreso fallidos - last -f /var/log/btmp | more
# Usuarios conectados ahora - who
# Cuando ingreso por ultima vez un usuario - last -f /var/log/lastlog | more
#

```

```

# manejo de log de cron
# /var/log/cron*
# [SIGTadmin@localhost bin]$ ls -lrt /var/log/cron*
# -rw----- 1 root root 21981 Aug 23 18:33 /var/log/cron-20220823
# -rw----- 1 root root 8587 Sep  6 11:49 /var/log/cron-20220906
# -rw----- 1 root root 6977 Sep 27 12:32 /var/log/cron-20220927
# -rw----- 1 root root 3271 Oct 11 12:20 /var/log/cron-20221011
# -rw----- 1 root root 1213 Oct 11 12:40 /var/log/cron
#
# colocar permiso (lo hace root) :
# SIGTadmin ALL = NOPASSWD: /bin/cat /var/log/cron*
# puede ser mediante visudo, o echo :
# echo 'SIGTadmin ALL = NOPASSWD: /bin/cat /var/log/cron*' >>/etc/sudoers
#

```



```
# Vamos a buscar estas acciones : LIST, EDIT, CMD, INFO (LECI)
#



# no funcionan !
#año=1964
#echo $año
#a#o=1964
#echo $a#o


# esto no !
#sudo grep LIST /var/log/cron*
# esto si !
#sudo cat /var/log/cron* | grep LIST


ani=$(date +%Y)
mes=$(date +%m)
dia=$(date +%d)
hor=$(date +%H)
min=$(date +%M)
#sudo cat /var/log/cron* | grep LIST >cronLIST-$ani$mes$dia$hor$min
#[SIGTadmin@localhost bin]$ sudo cat /var/log/cron* | grep LIST
# Oct 11 12:25:35 localhost crontab[3778]: (root) LIST (root)
# Oct 11 12:25:55 localhost crontab[3780]: (root) LIST (SISRDadmin)
# Aug 15 21:55:35 localhost crontab[19576]: (SISRDadmin) LIST (SISRDadmin)
# Aug 25 21:44:30 172 crontab[3368]: (SISRDadmin) LIST (SISRDadmin)
#



testCantidad=0 # 0 es no, 1 es si
#testCantidad=1
if test $testCantidad -eq 1
then
    # invoco función
    test Cant
else
    echo "control de cuenta de depuración deshabilitada"
fi

#exit


# primer código, extraemos la fecha
ani=$(date +%Y)
# test de archivo individual
```



```
#sudo cat /var/log/cron-20220817 | awk -f cp1.awk -v vani=$ani  
# total  
sudo cat /var/log/cron* | awk -f cp1.awk -v vani=$ani
```



```
tstCant()  
{  
echo grep LIST  
sudo cat /var/log/cron* | grep LIST  
echo -n "cantidad : "  
sudo cat /var/log/cron* | grep LIST | wc -l  
echo awk  
sudo cat /var/log/cron* | awk '$7=="LIST"'  
echo -n "cantidad : "  
sudo cat /var/log/cron* | awk '$7=="LIST"' | wc -l  
read -p 'Compare LIST ...'  
  
echo grep EDIT  
sudo cat /var/log/cron* | grep EDIT  
echo -n "cantidad : "  
sudo cat /var/log/cron* | grep EDIT | wc -l  
echo awk  
sudo cat /var/log/cron* | awk '$8=="EDIT"'  
echo -n "cantidad : "  
sudo cat /var/log/cron* | awk '$8=="EDIT"' | wc -l  
read -p 'Compare EDIT ...'  
  
echo grep 'LIST\|EDIT'  
sudo cat /var/log/cron* | grep 'LIST\|EDIT'  
echo -n "cantidad : "  
sudo cat /var/log/cron* | grep 'LIST\|EDIT' | wc -l  
echo awk  
sudo cat /var/log/cron* | awk '$7=="LIST" || $8=="EDIT"'  
echo -n "cantidad : "  
sudo cat /var/log/cron* | awk '$7=="LIST" || $8=="EDIT"' | wc -l  
read -p 'Compare suma de LIST mas EDIT ...'  
  
echo 'Las cuatro especies LECI (paginadas, q para salir) grep'  
sudo cat /var/log/cron* | grep 'LIST\|EDIT\|CMD\|INFO' | less  
echo -n "cantidad : "  
sudo cat /var/log/cron* | grep 'LIST\|EDIT\|CMD\|INFO' | wc -l  
echo las cuatro especies LECI \n(paginadas, q para salir\n) awk  
sudo cat /var/log/cron* | awk '$7 ~ "LIST|CMD|INFO" || $8=="EDIT"' | less  
echo -n "cantidad : "  
sudo cat /var/log/cron* | awk '$7 ~ "LIST|CMD|INFO" || $8=="EDIT"' | wc -l  
read -p 'Compare TOTALES ...'  
}
```



```
# main (programa principal)
#
# Autenticación - /var/log/secure
# Mensajes del sistema - /var/log/messages
# Tareas cron - /var/log/cron
# Servicio de correo - /var/log/maillog
# Errores hardware en inicio - /var/log/dmesg
# Información de arranque - /var/log/boot.log
# Paquetes instalados con yum - /var/log/yum.log
# Info sobre el motor - /var/log/mysqld.log
# Horas en que una ip se conectó al servidor - /var/log/cups/access_log
# Quienes estuvieron en el sistema - last -f /var/log/wtmp | more
# Intentos de ingreso fallidos - last -f /var/log/btmp | more
# Usuarios conectados ahora - who
# Cuando ingresó por última vez un usuario - last -f /var/log/lastlog | more
#
#
# manejo de log de cron
# /var/log/cron*
# [SIGTadmin@localhost bin]$ ls -lrt /var/log/cron*
# -rw-----. 1 root root 21981 Aug 23 18:33 /var/log/cron-20220823
# -rw-----. 1 root root 8587 Sep  6 11:49 /var/log/cron-20220906
# -rw-----. 1 root root 6977 Sep 27 12:32 /var/log/cron-20220927
# -rw-----. 1 root root 3271 Oct 11 12:20 /var/log/cron-20221011
# -rw-----. 1 root root 1213 Oct 11 12:40 /var/log/cron
#
# colocar permiso (lo hace root) :
# SIGTadmin ALL = NOPASSWD: /bin/cat /var/log/cron*
# puede ser mediante visudo, o echo :
# echo 'SIGTadmin ALL = NOPASSWD: /bin/cat /var/log/cron*' >>/etc/sudoers
#
# Vamos a buscar estas acciones : LIST, EDIT, CMD, INFO (LECI)
#
#
# no funcionan !
#año=1964
#echo $año
#a#o=1964
#echo $a#o
#
# esto no !
#sudo grep LIST /var/log/cron*
# esto si !
#sudo cat /var/log/cron* | grep LIST
```



```
ani=$(date +%Y)
mes=$(date +%m)
dia=$(date +%d)
hor=$(date +%H)
min=$(date +%M)
#sudo cat /var/log/cron* | grep LIST >cronLIST-$ani$mes$dia$hor$min
#[SIGTadmin@localhost bin]$ sudo cat /var/log/cron* | grep LIST
# Oct 11 12:25:35 localhost crontab[3778]: (root) LIST (root)
# Oct 11 12:25:55 localhost crontab[3780]: (root) LIST (SISRDadmin)
# Aug 15 21:55:35 localhost crontab[19576]: (SISRDadmin) LIST (SISRDadmin)
# Aug 25 21:44:30 172 crontab[3368]: (SISRDadmin) LIST (SISRDadmin)
#
testCantidad=0 # 0 es no, 1 es si
#testCantidad=1
if test $testCantidad -eq 1
then
    # invoco función
    tstCant
else
    echo "control de cuenta de depuración deshabilitada"
fi

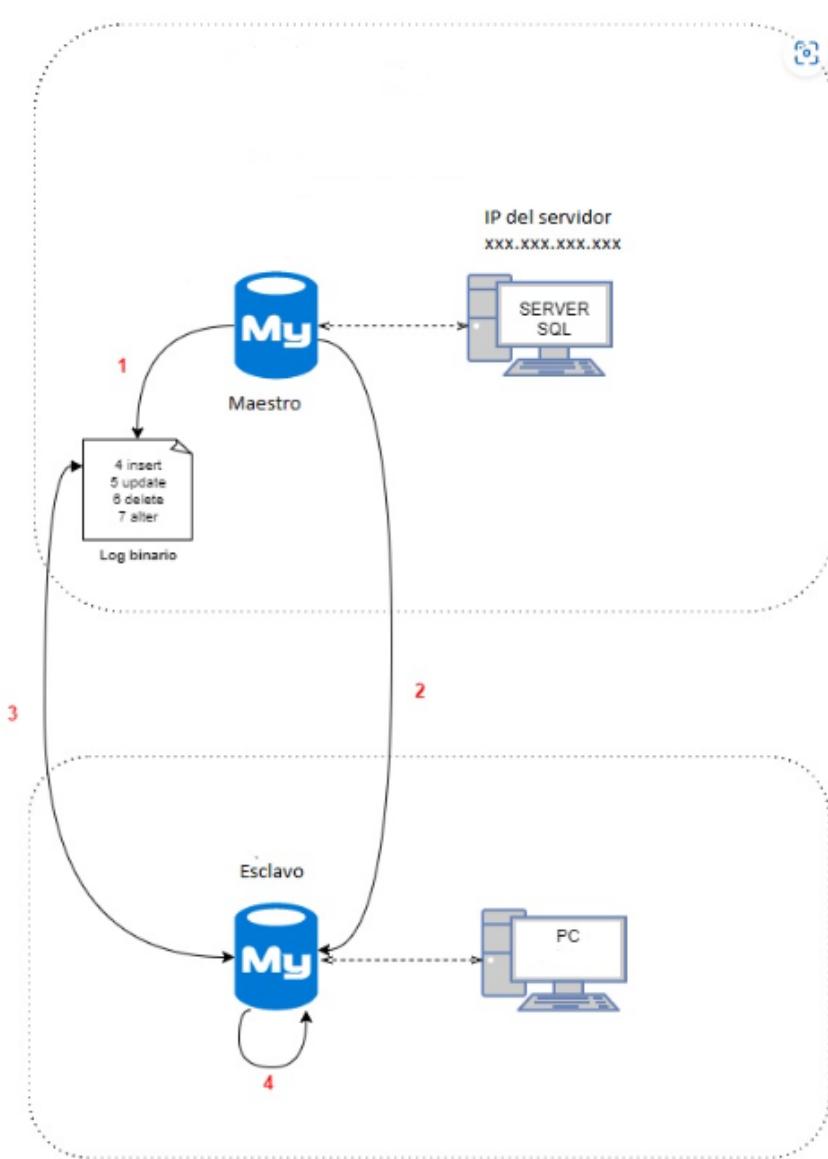
#exit

# primer código, extraemos la fecha
ani=$(date +%Y)
# test de archivo individual
#sudo cat /var/log/cron-20220817 | awk -f cpl.awk -v vani=$ani
# total
sudo cat /var/log/cron* | awk -f cpl.awk -v vani=$ani
```



Gestión de respaldos remotos

Replicación Master Slave de MySQL





REPLICAR MYSQL (MAESTRO/ESCLAVO) EN CENTOS.

La replicación MySQL resulta muy útil en términos de seguridad de los datos, solución Fail-Over, balanceo de carga, backups y para la realización de análisis sin perjudicar el rendimiento. En este post vamos a aprender cómo replicar nuestro servidor MySQL de manera que nuestras bases de datos sean idénticas en dos servidores distintos, y que únicamente necesitemos escribir en uno de ellos para que los datos sean transferidos automáticamente al otro servidor.

¿Qué beneficios ofrece la replicación MySQL?

Cuando escuchamos la expresión «replicar una base de datos», normalmente lo primero que nos viene a la cabeza es la posibilidad de tener varias copias de la misma información, a modo de backup, para poder recuperarla en caso de que se produzca un fallo en el servidor. Pero la replicación de una base de datos ofrece otras ventajas también, por ejemplo, nos permite realizar backups de las bases de datos directamente sobre el esclavo, sin que la escritura en el maestro afecte a nuestra copia de seguridad; nos permite disponer de la información duplicada, de forma que puedan balancearse las consultas entre los dos servidores en caso de ser muy pesadas; también permite realizar los análisis necesarios sobre el esclavo, que normalmente consumen muchos recursos, sin perjudicar el rendimiento en el maestro.

Nuestro entorno

En este ejemplo, vamos a funcionar con las siguientes características:

- Dos servidores Linux con CentOS 7
- Ambos servidores usan MySQL
- El servidor Maestro admite conexiones remotas por el puerto 3306.
- IP del Maestro: 192.168.1.2



- IP del Esclavo: 192.168.195.1 el servidor esclavo no necesariamente tiene que estar en el mismo establecimiento.

Configurando el servidor Maestro

El servidor maestro será el servidor donde se escriban nuestros datos, y desde aquí se replicarán al servidor esclavo. Para ello debemos realizar unos ajustes en su archivo de configuración, editándolo:

```
# nano /etc/my.cnf
```

Añadiremos las siguientes líneas en la sección [mysqld] de nuestro archivo de configuración:

```
server-id = 1  
  
relay-log = /var/lib/mysql/mysql-relay-bin  
  
relay-log-index = /var/lib/mysql/mysql-relay-bin.index  
  
master-info-file = /var/lib/mysql/mysql-master.info  
  
relay-log-info-file = /var/lib/mysql/mysql-relay-log.info  
  
log-bin = /var/lib/mysql/mysql-bin
```

A continuación, reiniciamos el servidor MySQL:

```
# service mysqld restart
```

Ahora nos conectamos al servidor MySQL como root, pues vamos a crear un usuario para el servidor esclavo y a otorgarle permisos para la replicación de los datos (reemplaza «esclavo» y «contraseña» por el usuario y contraseña que deseas):



```
mysql> GRANT REPLICATION SLAVE ON *.* TO 'esclavo'@'%' IDENTIFIED BY  
'contraseña';
```

```
mysql> FLUSH PRIVILEGES;
```

```
mysql> FLUSH TABLES WITH READ LOCK;
```

```
mysql> SHOW MASTER STATUS;
```

```
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
```

```
| mysql-bin.000002 | 1112 | |
```

```
1 row in set (0.00 sec)
```

```
mysql> quit;
```

Toma nota del nombre del archivo mostrado anteriormente (mysql-bin.000002) y de la posición (1112) ya que la necesitaremos después para la configuración del esclavo.
Ahora exportamos todas las bases de datos así como la configuración del maestro:

```
# mysqldump -u root -p --all-databases --master-data > /root/backup.db
```

Ahora que ya tenemos lista la exportación, nos conectaremos de nuevo como root al servidor MySQL y desbloquearemos las tablas:

```
mysql> UNLOCK TABLES;
```

```
mysql> quit;
```

Subimos el backup de MySQL que acabamos de generar al servidor esclavo, con el comando SCP:

```
# scp /root/backup.db root@192.168.195.1:/root/
```



Configurando el servidor Esclavo

Ya tenemos listo nuestro servidor maestro, y hemos subido al esclavo la base de datos, vamos entonces a dejar listo el servidor de esclavo, editando su archivo de configuración:

```
# nano /etc/my.cnf
```

Añadiremos las siguientes líneas en la sección [mysqld] de nuestro archivo de configuración:

```
server-id = 2  
master-host=192.168.1.2  
master-connect-retry=60  
master-user=esclavo  
master-password=contraseña  
relay-log = /var/lib/mysql/mysql-relay-bin  
relay-log-index = /var/lib/mysql/mysql-relay-bin.index  
master-info-file = /var/lib/mysql/mysql-master.info  
relay-log-info-file = /var/lib/mysql/mysql-relay-log.info  
log-bin = /var/lib/mysql/mysql-bin
```

Como ves, la configuración es muy parecida a la que hemos realizado en el servidor Maestro, sólo que en este caso le indicamos a nuestro esclavo cuál es la IP del servidor Maestro, cuántos segundos tardará en volver a intentar la conexión en caso de que falle el primer intento, y el usuario y contraseña que creamos en el servidor Maestro para permitir la replicación.

A continuación, importamos el backup que hicimos en el Maestro y reiniciamos el servidor MySQL:

```
# mysql -u root -p < /root/backup.db
```



```
# service mysqld restart
```

Ahora accedemos a MySQL como root, detenemos el esclavo, y le indicamos el nombre del archivo del Maestro (mysql-bin.000002) y la posición (1112) que obtuvimos durante la configuración del Maestro, con el comando «SHOW MASTER STATUS;». Recuerda que en el siguiente comando debes reemplazar la IP por la IP de tu Maestro, así como el usuario y contraseña que creaste para conectarte a él:

```
# mysql -u root -p
```



```
mysql> slave stop;

mysql>      CHANGE      MASTER      TO      MASTER_HOST='192.168.1.1',
MASTER_USER='esclavo',                      MASTER_PASSWORD='contraseña',
MASTER_LOG_FILE='mysql-bin.000002', MASTER_LOG_POS=1112;

mysql> slave start;

mysql> show slave status\G
***** 1. row *****

Slave_IO_State: Waiting for master to send event

Master_Host: 192.168.1.2

Master_User: esclavo

Master_Port: 3306

Connect_Retry: 60

Master_Log_File: mysql-bin.000002

Read_Master_Log_Pos: 12345100

Relay_Log_File: mysql-relay-bin.000006

Relay_Log_Pos: 11381900

Relay_Master_Log_File: mysql-bin.000002

Slave_IO_Running: Yes

Slave_SQL_Running: Yes

Replicate_Do_DB:

Replicate_Ignore_DB:

Replicate_Do_Table:

Replicate_Ignore_Table:

Replicate_Wild_Do_Table:

Replicate_Wild_Ignore_Table:

Last_Error:

Last_Errno: 0
```



Skip_Counter: 0

Exec_Master_Log_Pos: 12345100

Relay_Log_Space: 11382055

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_Error:

Last_IO_Error:

Last_SQL_Error:

Last_SQL_Error:

1 row in set (0.00 sec)

Si obtienes algo como esto, significa que tienes tu replicación Maestro/Esclavo funcionando y perfectamente sincronizada. Si deseas probarlo, sólo tienes que crear un registro nuevo en cualquier base de datos del servidor Maestro, o incluso una base de datos completa y verás que es inmediatamente replicada en el esclavo. En este reporte que obtenemos del estado del Esclavo, los campos más importantes son «Last_SQL_Error», que estará correcto mientras se encuentre en «0»; en caso contrario habrá que mirar qué ha sucedido y reiniciar el servicio Esclavo con los comandos «stop slave» y «start slave»; y el campo



«Seconds_Behind_Master», que indica cuántos segundos está retrasado el Esclavo respecto a su Maestro. Este valor nunca debe ser demasiado alto.

Reparando un error de replicación

En ocasiones, puede efectuarse una consulta errónea que resulte en un error de replicación. Aquí vamos a ver cómo poder resolverlo y continuar replicando los datos sin tener que configurar de nuevo todos los pasos anteriores.

Lo primero es detectar el error, de modo que lo buscaremos en primer lugar en el log de MySQL (dependiendo del Sistema Operativo que utilices, tendrás el log de MySQL en una ubicación u otra), en CentOS solemos encontrarlo en el directorio `/var/lib/mysql/error.log`.

```
# nano /var/lib/mysql/error.log

ABR 08 09:56:08 mysqld[1380]: 080529 9:56:08 [ERROR] Slave: Error 'Table
'mydb.taggregate_temp_1212047760' doesn't exist' on query. Default database: 'mydb'.
Query: 'UPDATE thread AS thread,taggregate_temp_1212047760 AS aggregate
ABR 08 09:56:08 mysqld[1380]: ^ISET thread.views = thread.views + aggregate.views
ABR 08 09:56:08 mysqld[1380]: ^IWHERE thread.threadid = aggregate.threadid',
Error_code: 1146
ABR 08 09:56:08 mysqld[1380]: 080529 9:56:08 [ERROR] Error running query, slave SQL
thread aborted. Fix the problem, and restart the slave SQL thread with «SLAVE START».
We stopped at log 'mysql-bin.001079' position 203015142
```

Como vemos, en la última línea nos indica en qué posición se detuvo la replicación.

Esto podremos comprobarlo accediendo a MySQL en el servidor Esclavo y ejecutando el comando mencionado anteriormente «SHOW SLAVE STATUS \G», que nos arrojará también la consulta que nos ha dado error.

Para reparar esto, primero detendremos el esclavo:



```
mysql> stop slave;
```

Y ejecutaremos el siguiente comando a continuación:

```
mysql> SET GLOBAL SQL_SLAVE_SKIP_COUNTER = 1;
```

Con esto le indicamos al esclavo que ignore la última consulta realizada (que es la que nos causó el error). Si quieras ignorar dos consultas, sólo tendrás que cambiar el valor de ese parámetro de 1 a 2, y así con cuantas consultas deseas ignorar.

Ahora ya podemos iniciar de nuevo el esclavo y consultar el estado de la replicación:

```
mysql> start slave;
```

```
mysql> SHOW SLAVE STATUS \G
```

Ya hemos aprendido cómo realizar una replicación de un servidor MySQL Maestro/Esclavo y a solucionar los posibles problemas que puedan surgir con la replicación MySQL. Ahora no tenemos excusa para tener siempre a buen recaudo nuestras bases de datos.

1-Servidor Maestro

SQL

PC Maestro

Log Binario:

insert

update

delete

alter

2-Servidor Esclavo

SQL



PC Esclavo

Archivos cuenta con rutinas de backup y sus correspondientes scripts para el administrador.

```
#!/bin/bash
fecha=$(date +%Y%d%H%M%S)

mysqldump -u root -rkatsuisbo1234 katsuenterprise >
/home/kenterprise/ScriptCrontab/Respaldo_${fecha}.sql
```



Configuración del firewall de Gnu/Linux.

```
# Verificar el estado del firewall  
sudo firewall-cmd --state
```

```
# Habilitar el firewall (si no está activo)  
sudo systemctl start firewalld  
sudo systemctl enable firewalld
```

```
# Verificar las zonas de red activas  
sudo firewall-cmd --get-active-zones
```

```
# Configurar reglas de firewall (ejemplo para SSH)  
I.S.B.O. 3BF 24
```

```
sudo firewall-cmd --zone=public --add-service=ssh --permanent
```

```
# Recargar el firewall para aplicar cambios  
sudo firewall-cmd --reload
```

```
# Verificar reglas configuradas para una zona específica (ejemplo para la zona  
public)  
sudo firewall-cmd --zone=public --list-all
```



Filtrado de Ips mediante Firewall.

```

#!/bin/bash
# Función para mostrar el menú principal
show_menu() {
choice=$(zenity --list --title="Filtrar direcciones IP" --column="Acción" "Bloquear IP"
"Permitir IP" "Listar Reglas" "Salir")
case "$choice" in
"bloquear IP")
block_ip
;;
"Permitir IP")
allow_ip
;;
"Listar Reglas")
list_rules
;;
"Salir")
exit 0
;;
*)
zenity --error --text="Opción no válida."
show_menu
;;
esac
}
# Función para bloquear una IP
block_ip() {

I.S.B.O. 3BF 25

ip_to_block=$(zenity --entry --title="Bloquear IP" --text="Introduce la dirección IP que deseas bloquear:")
if [ -n "$ip_to_block" ]; then
sudo firewall-cmd --permanent --add-rich-rule='rule source ipset("'"$ip_to_block"'"
drop'
sudo firewall-cmd --reload
zenity --info --text="IP $ip_to_block bloqueada con éxito."
fi
show_menu
}
# Función para permitir una IP
allow_ip() {
ip_to_allow=$(zenity --entry --title="Permitir IP" --text="Introduce la dirección IP que deseas permitir:")
if [ -n "$ip_to_allow" ]; then

```



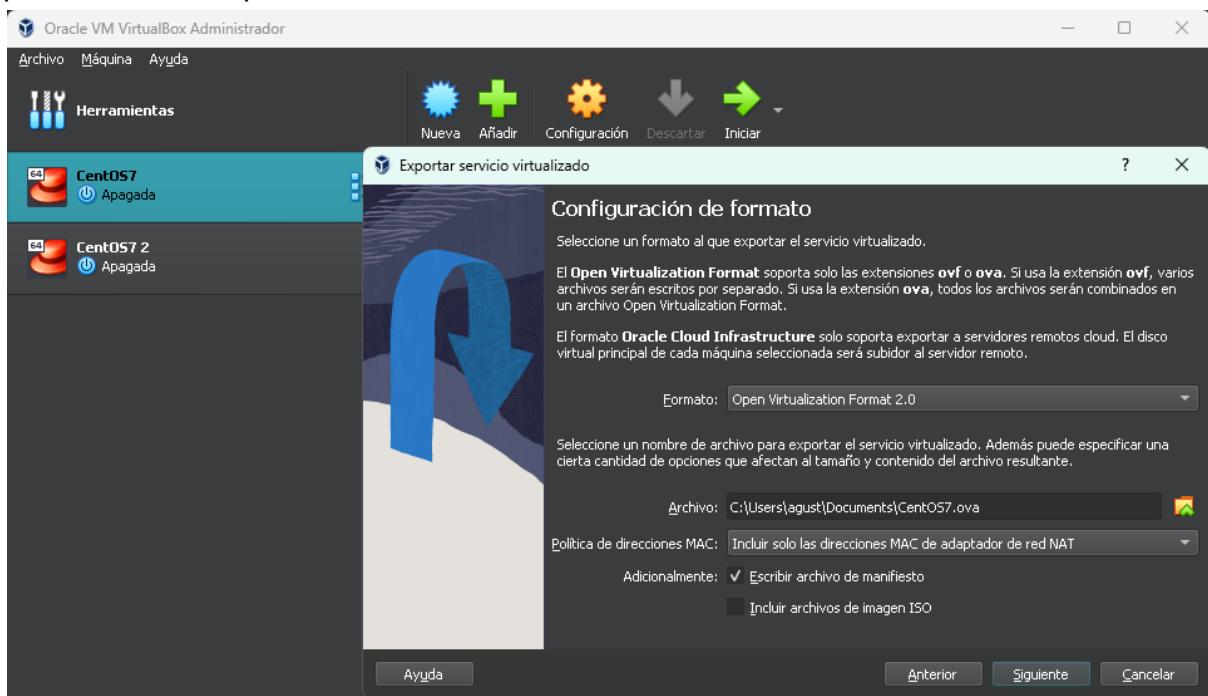
```
sudo firewall-cmd --permanent --add-rich-rule='rule source ipset="$ip_to_allow"
accept'
sudo firewall-cmd --reload
zenity --info --text="IP $ip_to_allow permitida con éxito."
fi
show_menu
}
# Función para listar las reglas de firewall
list_rules() {
firewall-cmd --list-all
zenity --info --text="Reglas de firewall listadas en la terminal."
show_menu
}
```

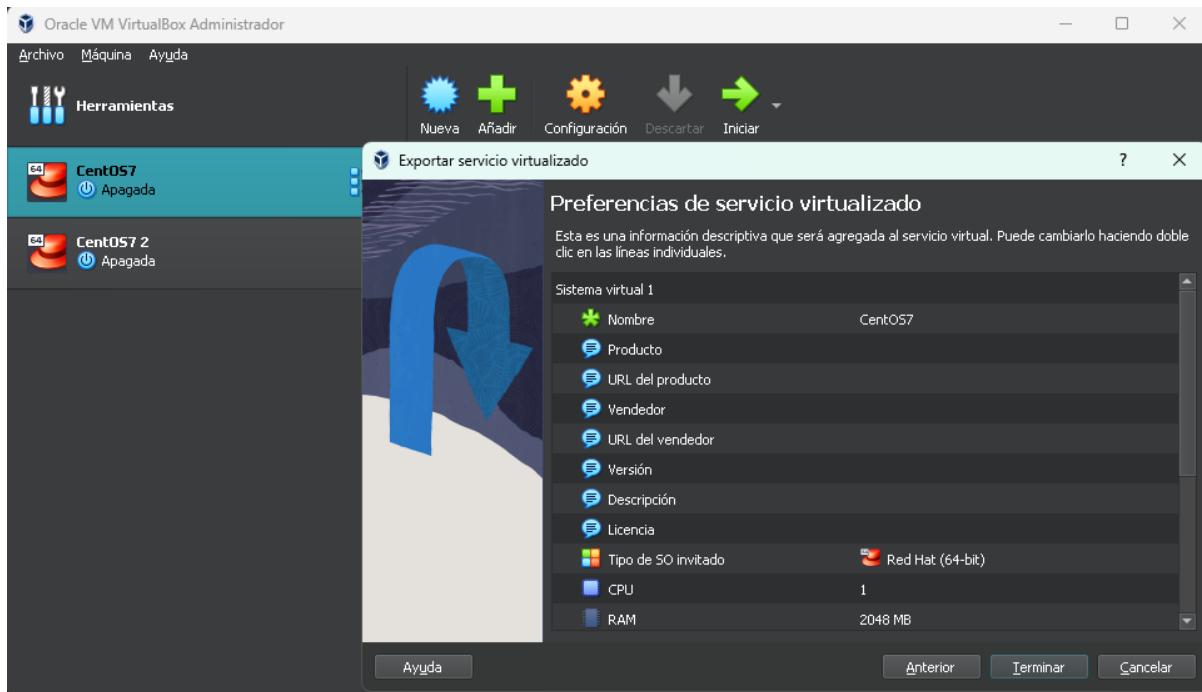


Generar un servidor de respaldo de datos

Para hacer el servidor de respaldo optamos por clonar el servidor principal, cambiando la ip del mismo para diferenciarlo, así obtendremos la misma configuración en ambos, contando con php, ssh, mysql y otros Sin la necesidad de volver a configurar un CentOS 7 desde cero.

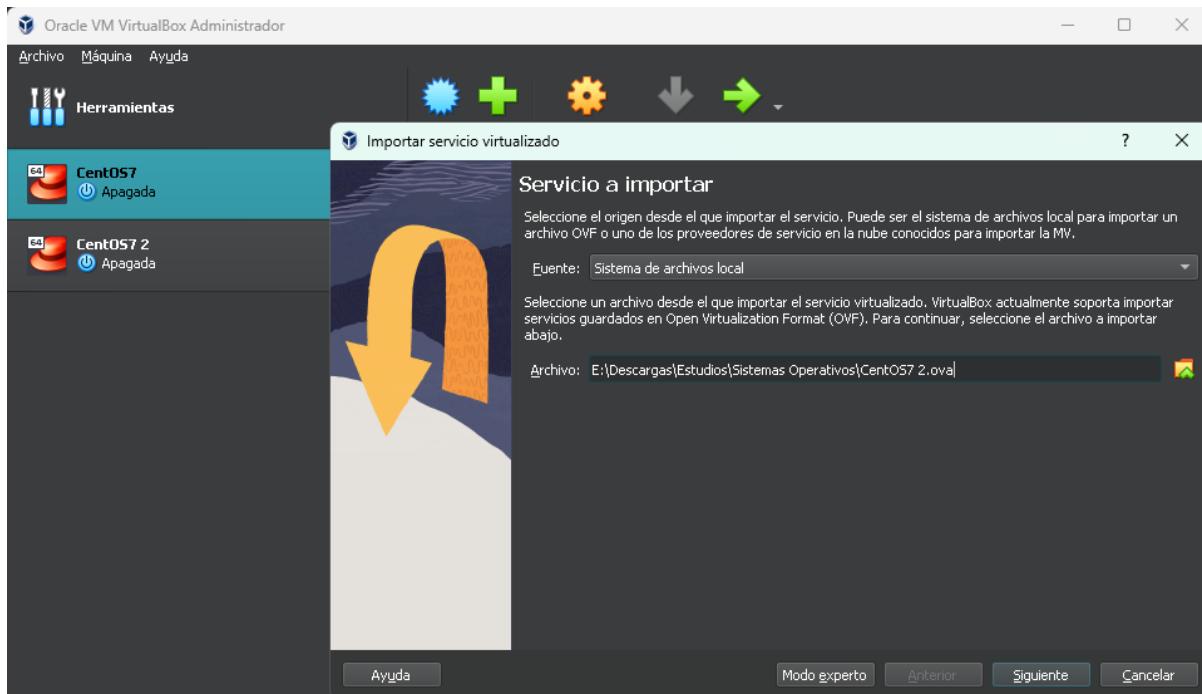
En este apartado le daremos a exportar en Virtual Box el centos que necesitamos para hacer el respaldo.

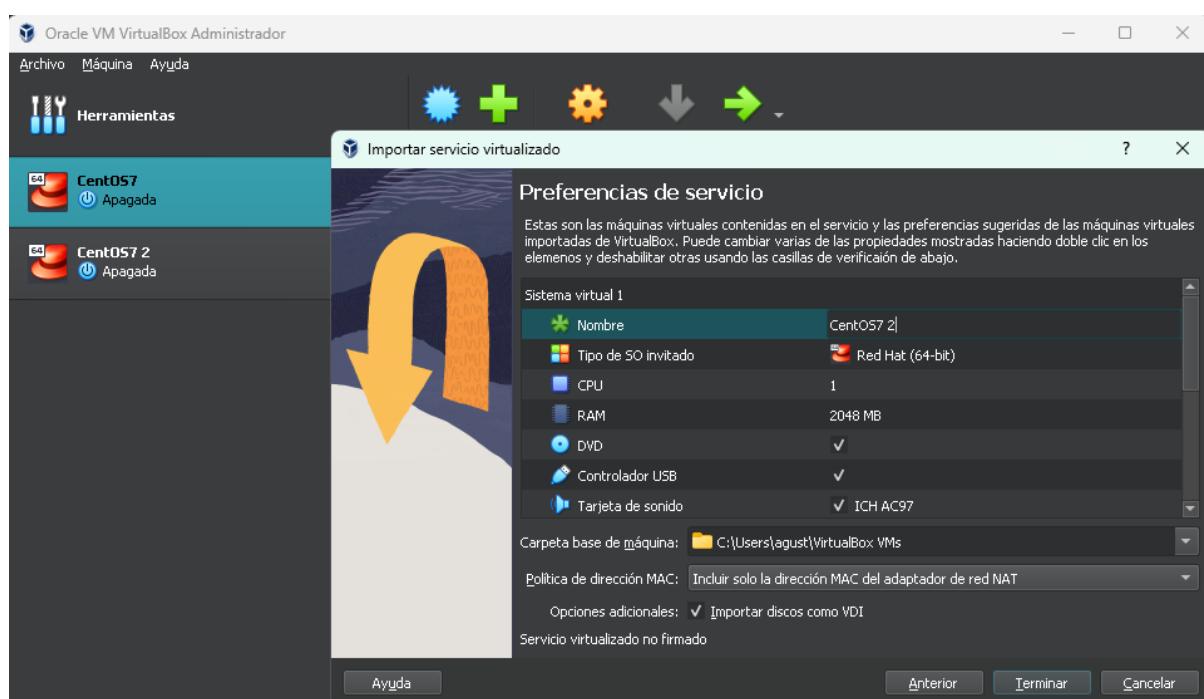




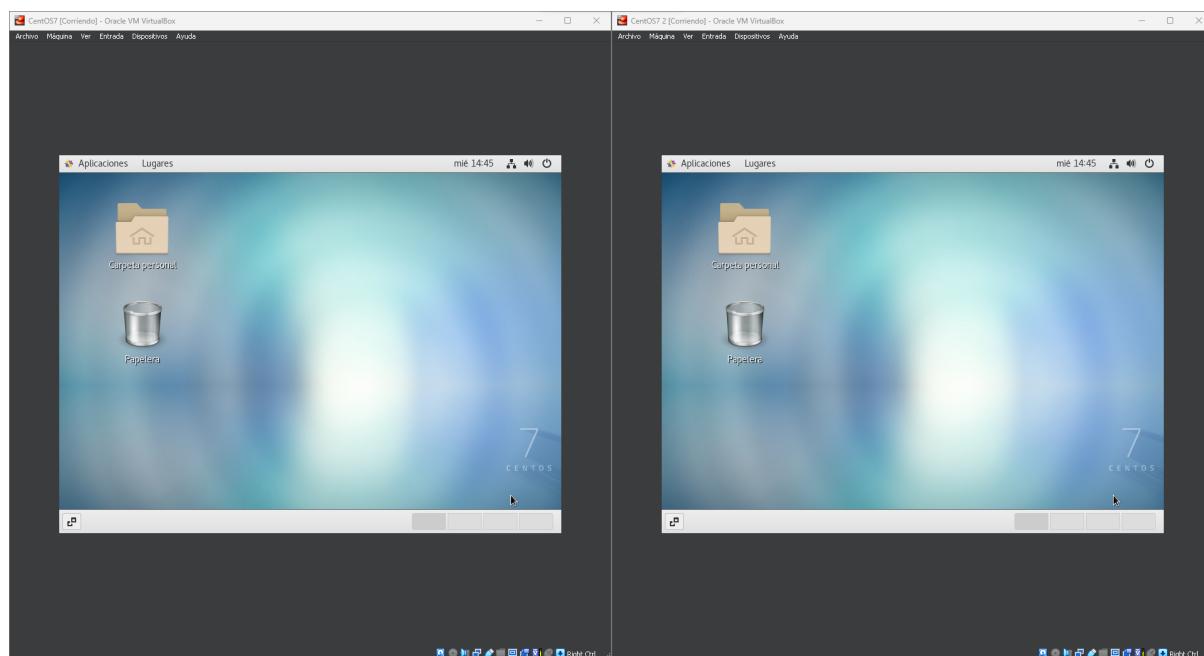
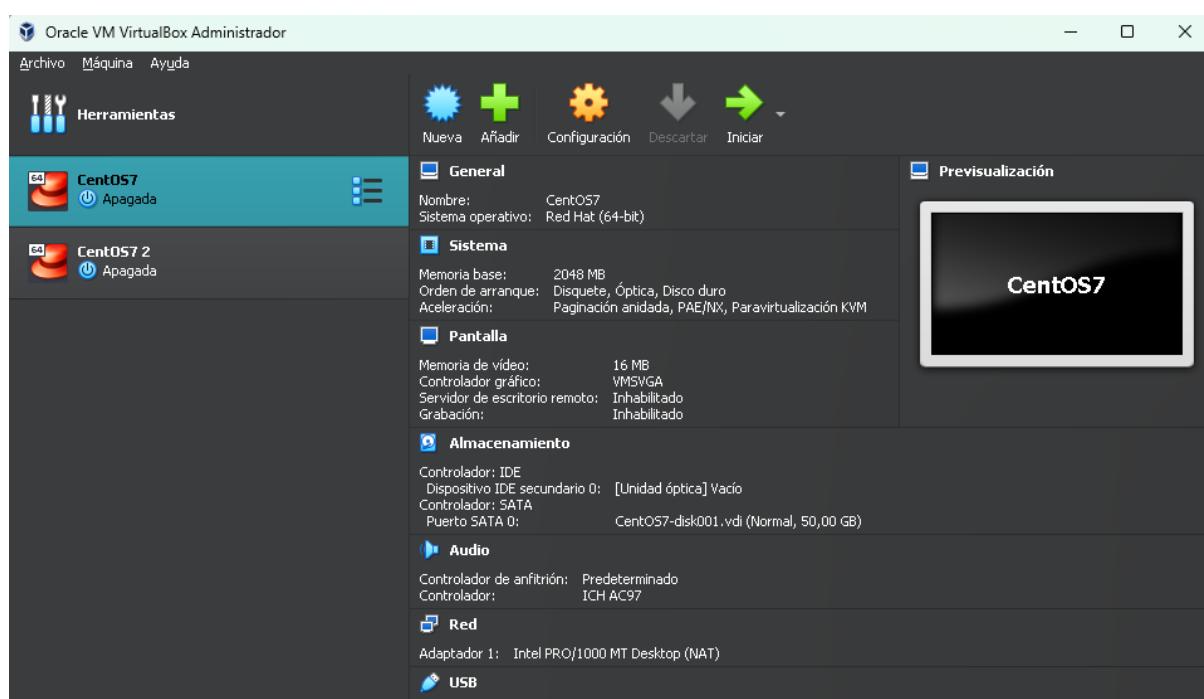
Luego de exportar la maquina virtual deseada, lo que haremos sera importarla.

Buscaremos la ruta donde se encuentra la maquina virtual y le daremos a siguiente.





Después de realizar la importación, ya tendremos la maquina virtual de respaldo, lo único que queda por hacer seria cambiar el ip de la misma y quedaría listo.





Menú para el Operador del Centro de Cómputos (Administrador del Sistema)

Script Gestión de Usuarios

- **Función menú:** la función expone un menú, en el cual, el usuario elige qué acción quiere realizar ingresando el número correspondiente.

```
● ● ●

function menu(){
    clear
    echo "MENÚ DE GESTIÓN DE USUARIOS"
    echo "1 - Agregar usuario"
    echo "2 - Borrar usuario"
    echo "3 - Listar usuarios del sistema"
    echo "4 - Buscar un usuario en el sistema"
    echo "5 - Cambiar contraseña de un usuario"
    echo "6 - Bloquear usuario"
    echo "7 - Desbloquear usuario"
    echo "0 - Salir"
}
```

- **Función agregar_usuario:** Esta función permite ingresar usuarios al sistema en un formato de apellidónombre, igualmente, en caso de agregarse un nombre con mayúsculas (ej.: MRTINEZRAUL), la función está programada para guardar todo en minúsculas.

```
● ● ●

function agregar_usuario(){
    clear
    #Nomeclatura del usuario apellidónombre
    echo "Ingrese el apellido y nombre del usuario en formato: apellidónombre: "
    read nombre
    usuario=$(echo $nombre | tr [:upper:] [:lower:])
    nomb=$(cat /etc/passwd | grep -c $usuario)
    if [ $nomb -eq 1 ]; then
        echo "El usuario ya existe"
        read pausa
    else
        echo "Ingrese el grupo: "
        read grupo
        user_group=$(echo $grupo | tr [:upper:] [:lower:])
        grup=$(cat /etc/group | grep -c $user_group)
        if [ $grup -eq 1 ]; then
            useradd -g $user_group -d /home/$usuario -s /bin/bash $usuario
            passwd -e -d $usuario
            echo "usuario dado de alta"
            read pausa
        else
            echo "El grupo no existe"
            read pausa
        fi
    fi
}
```



- **Función borrar_usuario:** Como deja a imaginar el nombre de la función, ésta permite borrar usuarios. Tiene la capacidad de entender el nombre de usuario aunque se le escriba en mayúsculas.

```
● ● ●

function borrar_usuario(){
    clear
    #Nombrellatura del usuario apellidonombre
    echo "Ingrese el apellido y nombre del usuario en formato: apellidonombre: "
    read nombre
    usuario=$(echo $nombre | tr [:upper:] [:lower:])
    existe=$(cat /etc/passwd | grep -c $usuario)
    if [ $existe -eq 1 ]; then
        echo "El usuario $usuario será eliminado del sistema, está seguro que desea eliminarlo S/N ?"
        read letra
        if [ $letra == 'S' -o == 's' ]; then
            echo "Usuario eliminado del sistema, presione enter para continuar"
            userdel $usuario
            read pausa
        else
            echo "Operación cancelada, presione enter para volver al menú principal"
            read pausa
        fi
    else
        echo "El usuario no existe en el sistema, presione enter para volver al menú principal"
        read pausa
    fi
}
}
```

- **Función listar_usuarios:** Posibilita verificar qué usuarios están ingresados.

```
● ● ●

function listar_usuarios(){
    echo "USUARIOS DEL SISTEMA"
    cut -d ":" -f 1 /etc/passwd | sort | more
    echo "Presione enter para volver al menú principal"
    read pausa
}
```

- **Función buscar_usuario:** Si se desea buscar un usuario en específico, podemos hacerlo con la función buscar usuario.

```
● ● ●

function buscar_usuario(){
    clear
    #Nombrellatura del usuario apellidonombre
    echo "Ingrese el apellido y nombre del usuario en formato: apellidonombre: "
    read nombre
    usuario=$(echo $nombre | tr [:upper:] [:lower:])
    existe=$(cat /etc/passwd | grep -c $usuario)
    if [ $existe -eq 1 ]; then
        echo "El usuario: $usuario existe en el sistema"
        cat /etc/passwd | grep $usuario
        echo "presione enter para continuar"
        read pausa
    else
        echo "El usuario: $usuario no existe en el sistema, presione enter para continuar"
        read pausa
    fi
}
```



- **Función cambiar contra usuario:** Permite cambiar la contraseña del usuario.

```
● ● ●

function cambiar_contra_usuario(){
    clear
    #Nomenclatura del usuario apellidonombre
    echo "Ingrese el apellido y nombre del usuario en formato: apellidonombre: "
    read nombre
    usuario=$(echo $nombre | tr [:upper:] [:lower:])
    existe=$(cat /etc/passwd | grep -c $usuario)
    if [ $existe -eq 1 ]; then
        echo "Se procede a cambiar la contraseña al usuario $usuario"
        passwd $usuario
        read pausa
    else
        echo "El usuario: $usuario no existe en el sistema, presione enter para continuar"
        read pausa
    fi
}
```

- **Función bloquear usuario:** Autoriza el bloqueo de los usuarios.

```
● ● ●

function bloquear_usuario(){
    clear
    #Nomenclatura del usuario apellidonombre
    echo "Ingrese el apellido y nombre del usuario en formato: apellidonombre: "
    read nombre
    usuario=$(echo $nombre | tr [:upper:] [:lower:])
    existe=$(cat /etc/passwd | grep -c $usuario)
    if [ $existe -eq 1 ]; then
        echo "Se procede a bloquear la cuenta del usuario $usuario"
        usermod -L $usuario
        read pausa
    else
        echo "El usuario: $usuario no existe en el sistema, presione enter para continuar"
        read pausa
    fi
}
```

- **Función desbloquear usuario:** Ante usuarios bloqueados, podemos desbloquearlos con esta función.

```
● ● ●

function desbloquear_usuario(){
    clear
    #Nomenclatura del usuario apellidonombre
    echo "Ingrese el apellido y nombre del usuario en formato: apellidonombre: "
    read nombre
    usuario=$(echo $nombre | tr [:upper:] [:lower:])
    existe=$(cat /etc/passwd | grep -c $usuario)
    if [ $existe -eq 1 ]; then
        echo "Se procede a desbloquear la cuenta del usuario $usuario"
        usermod -U $usuario
        read pausa
    else
        echo "El usuario: $usuario no existe en el sistema, presione enter para continuar"
        read pausa
    fi
}
```



- Main Script Principal:



```
while [ $opc -ne 0 ]
do
    clear
    menu
    read -p "Ingrese la opción correspondiente: " opc
    case $opc in
        1)
            agregar_usuario;;
        2)
            borrar_usuario;;
        3)
            listar_usuarios;;
        4)
            buscar_usuario;;
        5)
            cambiar_contra_usuario;;
        6)
            bloquear_usuario;;
        7)
            desbloquear_usuario;;
        0)
            echo "Volviendo al menú principal"; break ;;
        *)
            echo "Seleccionó una opción incorrecta";;
    esac
done
```



Script grupos.sh

- **Funciones:** Las funciones de este script son prácticamente idénticas a las de usuarios.sh, con la diferencia de que éste script tiene sus propias variables.

```
● ● ●

function menu( ){
    clear
    echo "MENÚ DE GESTIÓN DE GRUPOS"
    echo "1 - Agregar grupo"
    echo "2 - Borrar grupo"
    echo "3 - Listar grupo del sistema"
    echo "4 - Buscar un grupo en el sistema"
    echo "0 - Salir"
}
```

```
● ● ●

function agregar_grupo(){
    clear
    echo "Ingrese el grupo: "
    read grupo
    user_group=$(echo $grupo | tr [:upper:] [:lower:])
    existe=$(cat /etc/group | grep -c $user_group)
    if [ $existe -eq 0 ]; then
        groupadd $user_group
        echo "grupo creado, presione enter para continuar"
        read pausa
    else
        echo "El grupo ya existe"
        read pausa
    fi
}
```



```
● ● ●

function borrar_grupo(){
    clear
    echo "Ingrese el grupo: "
    read grupo
    user_group=$(echo $grupo | tr [:upper:] [:lower:])
    existe=$(cat /etc/group | grep -c $user_group)
    if [ $existe -eq 1 ]; then
        groupdel $user_group
        echo "grupo borrado, presione enter para continuar"
        read pausa
    else
        echo "El grupo no existe"
        read pausa
    fi
}
}
```

```
● ● ●

function buscar_grupo() {
    clear
    echo "Ingrese el grupo: "
    read grupo
    user_group=$(echo $grupo | tr [:upper:] [:lower:])
    existe=$(cat /etc/group | grep -c $user_group)
    if [ $existe -eq 1 ]; then
        echo "el grupo existe"
        cat /etc/group | grep $user_group
        echo "presione enter para continuar."
        read pausa
    else
        echo "El grupo no existe"
        read pausa
    fi
}
}
```



```
● ● ●

function buscar_grupo() {
    clear
    echo "Ingrese el grupo: "
    read grupo
    user_group=$(echo $grupo | tr [:upper:] [:lower:])
    existe=$(cat /etc/group | grep -c $user_group)
    if [ $existe -eq 1 ]; then
        echo "el grupo existe"
        cat /etc/group | grep $user_group
        echo "presione enter para continuar."
        read pausa
    else
        echo "El grupo no existe"
        read pausa
    fi
fi

}
```

```
● ● ●

while [ $opc -ne 0 ]
do
    clear
    menu
    read -p "Ingrese la opción correspondiente: " opc
    case $opc in
    1)
        agregar_grupo;;
    2)
        borrar_grupo;;
    3)
        listar_grupos;;
    4)
        buscar_grupo;;
    0)
        echo "Volviendo al menú principal"; break ;;
    *)
        echo "Seleccionó una opción incorrecta";;
    esac
done
```



```
● ● ●

function buscar_grupo() {
    clear
    echo "Ingrese el grupo: "
    read grupo
    user_group=$(echo $grupo | tr [:upper:] [:lower:])
    existe=$(cat /etc/group | grep -c $user_group)
    if [ $existe -eq 1 ]; then
        echo "el grupo existe"
        cat /etc/group | grep $user_group
        echo "presione enter para continuar."
        read pausa
    else
        echo "El grupo no existe"
        read pausa
    fi
fi

}
```

```
● ● ●

while [ $opc -ne 0 ]
do
    clear
    menu
    read -p "Ingrese la opción correspondiente: " opc
    case $opc in
        1)
            agregar_grupo;;
        2)
            borrar_grupo;;
        3)
            listar_grupos;;
        4)
            buscar_grupo;;
        0)
            echo "Volviendo al menú principal"; break ;;
        *)
            echo "Seleccionó una opción incorrecta";;
    esac
done
```



Script de Redes

```
#!/bin/bash
# Función para mostrar el menú principal
show_menu() {
choice=$(zenity --list --title="Administrador de Redes" --column="Acción" "Crear
Red" "Eliminar Red" "Editar Red" "Salir")
case "$choice" in
"Crear Red")
create_network
;;
"Eliminar Red")
delete_network
;;
"Editar Red")
edit_network
;;
"Salir")
exit 0
;;
*)
zenity --error --text="Opción no válida."
show_menu
;;
esac
}
```



```
#!/bin/bash

# Función para mostrar el menú principal
show_menu() {
choice=$(zenity --list --title="Administrador de Redes" --column="Acción" "Crear Red" "Eliminar Red" "Editar Red" "Salir")

case "$choice" in
"Crear Red")
create_network
;;
"Eliminar Red")
delete_network
;;
"Editar Red")
edit_network
;;
"Salir")
exit 0
;;
*)
zenity --error --text="Opción no válida."
show_menu
;;
esac
}
```



Función para crear una nueva red

```
create_network() {  
    network_name=$(zenity --entry --title="Crear Red" --text="Introduce el nombre de  
la nueva red:")  
    if [ -n "$network_name" ]; then  
        sudo nmcli connection add con-name "$network_name" ifname "*" type  
        ethernet  
        zenity --info --text="Red '$network_name' creada con éxito."  
    fi  
    show_menu  
}
```



```
create_network() {  
    network_name=$(zenity --entry --title="Crear Red" --text="Introduce el nombre de  
la nueva red:")  
    if [ -n "$network_name" ]; then  
        sudo nmcli connection add con-name "$network_name" ifname "*" type  
        ethernet  
        zenity --info --text="Red '$network_name' creada con éxito."  
    fi  
    show_menu  
}
```

|



Función para eliminar una red existente

```
delete_network() {  
    network_name=$(zenity --entry --title="Eliminar Red" --text="Introduce el nombre  
de la red que deseas eliminar:")  
    if [ -n "$network_name" ]; then  
        sudo nmcli connection delete "$network_name"  
        zenity --info --text="Red '$network_name' eliminada con éxito."  
    fi  
    show_menu  
}
```

```
delete_network() {  
    network_name=$(zenity --entry --title="Eliminar Red" --text="Introduce el nombre  
de la red que deseas eliminar:")  
    if [ -n "$network_name" ]; then  
        sudo nmcli connection delete "$network_name"  
        zenity --info --text="Red '$network_name' eliminada con éxito."  
    fi  
    show_menu  
}
```

**# Función para editar una red existente**

```
edit_network() {  
    network_name=$(zenity --entry --title="Editar Red" --text="Introduce el nombre de  
    la red que deseas editar:")  
    if [ -n "$network_name" ]; then  
        nmcli connection edit "$network_name"  
        zenity --info --text="Editando la red '$network_name'."  
    fi  
    show_menu}
```

```
● ● ●  
  
edit_network() {  
    network_name=$(zenity --entry --title="Editar Red" --text="Introduce el nombre de  
    la red que deseas editar:")  
    if [ -n "$network_name" ]; then  
        nmcli connection edit "$network_name"  
        zenity --info --text="Editando la red '$network_name'."  
    fi  
    show_menu}
```



Script de Servicios

```
#!/bin/bash
```

```
# Función para mostrar el menú principal
show_menu() {
choice=$(zenity --list --title="Administrador de Servicios" --column="Servicio" "PHP"
"MySQL" "SSH" "Salir")

case "$choice" in
"PHP")
toggle_service "php7.0-fpm" # Cambia php7.0-fpm al nombre del servicio de
PHP en tu sistema
;;
"MySQL")
toggle_service "mysql" # Cambia mysql al nombre del servicio de MySQL en
tu sistema
;;
"SSH")
toggle_service "ssh" # Cambia ssh al nombre del servicio de SSH en tu
sistema
;;
"Salir")
exit 0
;;
*)
zenity --error --text="Servicio no válido."
show_menu
;;
esac
}
```



```
#!/bin/bash

# Función para mostrar el menú principal
show_menu() {
choice=$(zenity --list --title="Administrador de Servicios" --column="Servicio" "PHP"
"MySQL" "SSH" "Salir")

case "$choice" in
"PHP")
toggle_service "php7.0-fpm" # Cambia php7.0-fpm al nombre del servicio de
PHP en tu sistema
;;
"MySQL")
toggle_service "mysql" # Cambia mysql al nombre del servicio de MySQL en
tu sistema
;;
"SSH")
toggle_service "ssh" # Cambia ssh al nombre del servicio de SSH en tu
sistema
;;
"Salir")
exit 0
;;
*)
zenity --error --text="Servicio no válido."
show_menu
;;
esac
}
```

**# Función para activar o desactivar un servicio**

```
toggle_service() {
service_name="$1"
if systemctl is-active --quiet "$service_name"; then
systemctl stop "$service_name"
zenity --info --text="Servicio $service_name desactivado con éxito."
else
systemctl start "$service_name"
zenity --info --text="Servicio $service_name activado con éxito."
fi
show_menu
}
```

```
toggle_service() {
service_name="$1"
if systemctl is-active --quiet "$service_name"; then
systemctl stop "$service_name"
zenity --info --text="Servicio $service_name desactivado con éxito."
else
systemctl start "$service_name"
zenity --info --text="Servicio $service_name activado con éxito."
fi
show_menu
}
```



Script de Firewall

```
#!/bin/bash
```

```
# Función para mostrar el menú principal
show_menu() {
choice=$(zenity --list --title="Administrador de Firewall" --column="Acción" "Activar
Firewall" "Desactivar Firewall" "Aregar Regla" "Eliminar Regla" "Salir")

case "$choice" in
"Activar Firewall")
sudo iptables -F # Limpia todas las reglas existentes
sudo iptables -P INPUT DROP # Establece la política de entrada en DROP
(denegar todo)
sudo iptables -P FORWARD DROP # Establece la política de reenvío en
DROP
sudo iptables -P OUTPUT ACCEPT # Establece la política de salida en

ACCEPT (permitir todo)
zenity --info --text="Firewall activado con éxito."
show_menu
;;

"Desactivar Firewall")
sudo iptables -F # Limpia todas las reglas existentes
sudo iptables -P INPUT ACCEPT # Establece la política de entrada en
ACCEPT (permitir todo)
sudo iptables -P FORWARD ACCEPT # Establece la política de reenvío en
ACCEPT
sudo iptables -P OUTPUT ACCEPT # Establece la política de salida en
ACCEPT (permitir todo)
zenity --info --text="Firewall desactivado con éxito."
show_menu
;;

"Agregar Regla")
rule=$(zenity --entry --title="Aregar Regla" --text="Introduce una regla
iptables:")
if [ -n "$rule" ]; then
sudo iptables -A INPUT $rule
zenity --info --text="Regla agregada con éxito: $rule"
fi
show_menu
;;

"Eliminar Regla")
```



```
rule=$(zenity --entry --title="Eliminar Regla" --text="Introduce la regla iptables que deseas eliminar:")
if [ -n "$rule" ]; then
sudo iptables -D INPUT -j $rule
zenity --info --text="Regla eliminada con éxito: $rule"
fi
show_menu
;;
"Salir")
exit 0
;;
*)
zenity --error --text="Opción no válida."
show_menu
;;
esac
}
```



```
● ● ●

Script de Firewall

#!/bin/bash

# Función para mostrar el menú principal
show_menu() {
choice=$(zenity --list --title="Administrador de Firewall" --column="Acción" "Activar Firewall" "Desactivar Firewall" "Agregar Regla" "Eliminar Regla" "Salir")

case "$choice" in
"Activar Firewall")
sudo iptables -F # Limpia todas las reglas existentes
sudo iptables -P INPUT DROP # Establece la política de entrada en DROP
(denegar todo)
sudo iptables -P FORWARD DROP # Establece la política de reenvío en
DROP
sudo iptables -P OUTPUT ACCEPT # Establece la política de salida en

ACCEPT (permitir todo)
zenity --info --text="Firewall activado con éxito."
show_menu
;;
"Desactivar Firewall")
sudo iptables -F # Limpia todas las reglas existentes
sudo iptables -P INPUT ACCEPT # Establece la política de entrada en
ACCEPT (permitir todo)
sudo iptables -P FORWARD ACCEPT # Establece la política de reenvío en
ACCEPT
sudo iptables -P OUTPUT ACCEPT # Establece la política de salida en
ACCEPT (permitir todo)
zenity --info --text="Firewall desactivado con éxito."
show_menu
;;
"Agregar Regla")
rule=$(zenity --entry --title="Agregar Regla" --text="Introduce una regla
iptables:")
if [ -n "$rule" ]; then
sudo iptables -A INPUT $rule
zenity --info --text="Regla agregada con éxito: $rule"
fi
show_menu
;;
"Eliminar Regla")
rule=$(zenity --entry --title="Eliminar Regla" --text="Introduce la regla iptables
que deseas eliminar:")
if [ -n "$rule" ]; then
sudo iptables -D INPUT -j $rule
zenity --info --text="Regla eliminada con éxito: $rule"
fi
show_menu
;;
"Salir")
exit 0
;;
*)
zenity --error --text="Opción no válida."
show_menu
;;
esac
}
```



Script de respaldo total

```
#!/bin/bash
opc=0
fecha=$(date +%Y%m%d-%H-%M-%S-backup_bd)

function menu() {
echo "MENÚ DE GESTIÓN DE RESPALDOS"
echo "1 - Realizar un respaldo completo"
echo "2 - Respaldar la estructura de la Base de datos"
echo "3 - Restaurar base de datos"
echo "4 - Realizar consulta personalizada"
echo "5 - Salir"
}

function realizar_respaldo_completo(){
echo "Ingrese el nombre de la base de datos"
read nomb
nombre=$(echo $nomb | tr [:upper:] [:lower:])

echo "show databases" | mysql -u root -p > bd.sql
existe=$(cat bd.sql | grep -c $nombre)
if [ $existe -ge 1 ]
then
mysqldump --opt --events --routines --triggers --default-character-set=utf8 -u root
-p $nombre > $nombre.sql
tar -czvf $fecha.tar.gz $nombre.sql
mv $fecha.tar.gz /root/respaldos/bd/completa/
rm $nombre.sql
rm bd.sql
else
echo "no existe la base de datos, presione enter para continuar.."
read enter
rm bd.sql
fi
}

function realizar_respaldo_estructura(){
echo "Ingrese el nombre de la base de datos"
read nomb
nombre=$(echo $nomb | tr [:upper:] [:lower:])
echo "show databases" | mysql -u root -p > bd.sql
existe=$(cat bd.sql | grep -c $nombre)
```



```
if [ $existe -ge 1 ]
then
mysqldump -v --opt --no-data --default-character-set=utf8 -u root -p $nombre >
$nombre.sql
tar -czvf $fecha.tar.gz $nombre.sql
mv $fecha.tar.gz /root/respaldos/bd/estructura
rm $nombre.sql
rm bd.sql
else

echo "no existe la base de datos, presione enter para continuar.."
read enter
rm bd.sql
fi
}

function restaurar_basedatos(){

echo "Ingrese el nombre de la base de datos"
read nomb
nombre=$(echo $nomb | tr '[:upper:]' '[:lower:]')
echo "show databases" | mysql -u root -p > bd.sql
existe=$(cat bd.sql | grep -c $nombre)
if [ $existe -ge 1 ]
then
mysqldump -v --opt --no-data --default-character-set=utf8 -u root -p $nombre >
$nombre.sql
tar -czvf $fecha.tar.gz $nombre.sql
mv $fecha.tar.gz /root/respaldos/bd/estructura
rm $nombre.sql
rm bd.sql
else
echo "no existe la base de datos, presione enter para continuar.."
read enter
rm bd.sql
fi
}

function pregunta_personalizada(){
echo "Ingrese el nombre de la base de datos"
read nomb
nombre=$(echo $nomb | tr '[:upper:]' '[:lower:]')
```



```
echo "Ingrese la sintaxis correspondiente"
read $sentencia
echo "$sentencia" | mysql -u root -p > bd.sql
existe=$(cat bd.sql | grep -c $nombre)
if [ $existe -ge 1 ]
then $sentencia
else
echo "no existe la base de datos, presione enter para continuar.."
read enter
rm bd.sql
fi
}

while [ $opc -ne 6 ]
do
menu
echo "Ingrese una opción: "
read opc
case $opc in
1) echo "realizar_respaldo_completo" ;;
2) echo "realizar_respaldo_estructura" ;;
3) echo "restaurar_basedatos" ;;
4) echo "realizar_respaldo_estructura" ;;
5) echo "Adios" ; break ;;
*) echo "Opción no valida" ; read pausa ;;
esac
done
```



```
● ● ●

Script de respaldo total

#!/bin/bash
opc=0
fecha=$(date +%Y%m%d-%H-%M-%S-backup_bd)

function menu() {
echo "MENÚ DE GESTIÓN DE RESPALDOS"
echo "1 - Realizar un respaldo completo"
echo "2 - Respaldar la estructura de la Base de datos"
echo "3 - Restaurar base de datos"
echo "4 - Realizar consulta personalizada"
echo "5 - Salir"
}

function realizar_respaldo_completo(){
echo "Ingrese el nombre de la base de datos"
read nombr
nombre=$(echo $nombr | tr '[[:upper:]]' '[[:lower:]]')

echo "show databases" | mysql -u root -p > bd.sql
existe=$(cat bd.sql | grep -c $nombre)
if [ $existe -ge 1 ]
then
mysqldump --opt --events --routines --triggers --default-character-set=utf8 -u root
-p $nombre > $nombre.sql
tar -czvf $fecha.tar.gz $nombre.sql
mv $fecha.tar.gz /root/respaldos/bd/completa/
rm $nombre.sql
rm bd.sql
else
echo "no existe la base de datos, presione enter para continuar.."
read enter
rm bd.sql
fi
}

function realizar_respaldo_estructura(){
echo "Ingrese el nombre de la base de datos"
read nombr
nombre=$(echo $nombr | tr '[[:upper:]]' '[[:lower:]]')
echo "show databases" | mysql -u root -p > bd.sql
existe=$(cat bd.sql | grep -c $nombre)
if [ $existe -ge 1 ]
```



```
then
mysqldump --opt --events --routines --triggers --default-character-set=utf8 -u root
-p $nombre > $nombre.sql
tar -czvf $fecha.tar.gz $nombre.sql
mv $fecha.tar.gz /root/respaldos/bd/completa/
rm $nombre.sql
rm bd.sql
else
echo "no existe la base de datos, presione enter para continuar.."
read enter
rm bd.sql
fi
}

function realizar_respaldo_estructura(){
echo "Ingrese el nombre de la base de datos"
read nombr
nombre=$(echo $nombr | tr '[:upper:]' '[:lower:]')
echo "show databases" | mysql -u root -p > bd.sql
existe=$(cat bd.sql | grep -c $nombre)
if [ $existe -ge 1 ]
then
mysqldump -v --opt --no-data --default-character-set=utf8 -u root -p $nombre >
$nombre.sql
tar -czvf $fecha.tar.gz $nombre.sql
mv $fecha.tar.gz /root/respaldos/bd/estructura
rm $nombre.sql
rm bd.sql
else

echo "no existe la base de datos, presione enter para continuar.."
read enter
rm bd.sql
fi
}
```



```
function restaurar_basedatos(){

echo "Ingrese el nombre de la base de datos"
read nombr
nombre=$(echo $nombr | tr '[:upper:]' '[:lower:]')
echo "show databases" | mysql -u root -p > bd.sql
existe=$(cat bd.sql | grep -c $nombre)
if [ $existe -ge 1 ]
then
mysqldump -v --opt --no-data --default-character-set=utf8 -u root -p $nombre >
$nombre.sql
tar -czvf $fecha.tar.gz $nombre.sql
mv $fecha.tar.gz /root/respaldos/bd/estructura
rm $nombre.sql
rm bd.sql
else
echo "no existe la base de datos, presione enter para continuar.."
read enter
rm bd.sql
fi
}

function pregunta_personalizada(){
echo "Ingrese el nombre de la base de datos"
read nombr
nombre=$(echo $nombr | tr '[:upper:]' '[:lower:]')

echo "Ingrese la sintaxis correspondiente"
read $sentencia
echo "$sentencia" | mysql -u root -p > bd.sql
existe=$(cat bd.sql | grep -c $nombre)
if [ $existe -ge 1 ]
then $sentencia
else
echo "no existe la base de datos, presione enter para continuar.."
read enter
rm bd.sql
fi
}
```

```
while [ $opc -ne 6 ]
do
menu
echo "Ingrese una opción: "
read opc
case $opc in
1) echo "realizar_respaldo_completo" ;;
2) echo "realizar_respaldo_estructura" ;;
3) echo "restaurar_basedatos" ;;
4) echo "realizar_respaldo_estructura" ;;
5) echo "Adios" ; break ;;
*) echo "Opción no valida" ; read pausa ;;
esac
done
```



Configuraciones de red en las terminales y el servidor

```
# Verificar la configuración actual  
ip addr show
```

```
# Editar el archivo de configuración de red para tu interfaz (por ejemplo, eth0)  
sudo nano /etc/sysconfig/network-scripts/ifcfg-eth0
```

```
# Dentro del archivo, configura la dirección IP estática, la máscara de subred, la  
puerta de enlace y los servidores DNS:
```

```
DEVICE=eth0  
BOOTPROTO=static  
ONBOOT=yes  
IPADDR=192.168.1.2  
NETMASK=255.255.255.0  
GATEWAY=192.168.1.1  
DNS1=8.8.8.8
```

```
# Reiniciar el servicio de red  
sudo systemctl restart network
```

```
# Verificar la nueva configuración  
ip addr show
```



Configuracion SSH

1: Instalar el paquete de software del servidor OpenSSH Ingresá el siguiente comando desde tu terminal para iniciar el proceso de instalación:

sudo yum install openssh-server openssh-clients

2: Iniciar el servicio SSH Para iniciar el servicio SSH en el servidor OpenSSH:

sudo systemctl start sshd

Asegúrate de que el puerto 22 esté abierto: **netstat -tulpn | grep :22**

3: Comprobar el estado de sshd Verifica el estado del daemon SSH con el siguiente comando:**sudo systemctl status sshd**

Para detener el daemon SSH, ingresa: **systemctl stop sshd**

4: Configuración del servidor OpenSSH Configurar adecuadamente el archivo de configuración sshd refuerza la seguridad del servidor. Los ajustes más comunes para mejorar la seguridad son cambiar el número de puerto, deshabilitar las conexiones de root y limitar el acceso solo a ciertos usuarios. Para editar estos ajustes, accede al archivo /etc/ssh/sshd_config:

sudo vim /etc/ssh/sshd_config

sudo nano /etc/ssh/sshd_config

5: Configuración del cortafuegos Añade el servicio SSH al cortafuegos usando uno de estos comandos:

firewall-cmd --zone=public --add-service=ssh --permanent

or

sudo semanage port -a -t ssh_port_t -p tcp 8089 firewall-cmd --add-port=8089/tcp --permanent firewall-cmd --reload

6: Habilitar el servicio OpenSSH Habilita el inicio automático del servicio SSH después de cada reinicio del sistema utilizando el comando systemctl:

sudo systemctl enable sshd

sudo systemctl restart sshd

service iptables restart



Bibliografía

Iptables centos: <https://linuxize.com/post/how-to-install-iptables-on-centos-7/>

Master slave: <https://www.proxadmin.es/blog/replicar-mysql-maestro-esclavo-en-centos/>

Configuracion ssh: <https://youtu.be/6sCY90PpRmk?si=JepZqH37p2PZMPpw>

Menú Crontab para administración de sistemas:

Cómo usar Cron y Crontab en Linux para programar tareas en servidores (redeszone.net)

Página para instalar iso de CentOS 7: www.centos.org

Referencias para la instalación de Apache, MYSQL y PHP:

Apache: <https://www.digitalocean.com/>

mysql: Código de instalación otorgador por Prof. Walter Dominguez.

PHP: <https://comoinstalar.me/>

Definición de Apache: <https://axarnet.es/>

<https://dinahosting.com/>

Definición de MYSQL: <https://www.hostinger.es/>

Definición de PHP: <https://rockcontent.com/>

Script de alta y baja de usuarios otorgado por: Prof. Luis Fagundez.

Mayor parte de las imágenes del código fueron hechas en: <https://carbon.now.sh/>

Configuracion SSH: <https://youtu.be/6sCY90PpRmk?si=JepZqH37p2PZMPpw>