VifraSoft

S.I.G.P.D.

Administración de Sistemas Operativos

VifraSoft

Rol	Apellido	Nombre	C.I	Email
Coordinador	Reyes	Franco	5.676.219-7	franco07sierra@gmail.com
Sub-Coordinador	Bittencourt	Luis	5.710.007-1	santiagobittencourt17@gmail.com
Integrante 1	Larrosa	Maria	5.633.663-5	victolarrosa13@gmail.com

Docente: Martinez, Santiago.

Fecha de culminación

10/11/2025

TERCERA ENTREGA

I.S.B.O. **3MI**

Índice

Índice **2**
Proyecto **4**
Relevamiento del Sistema Operativo **5**
1. Identificación General 5
2. Entorno de Virtualización 5
3. Finalidad del Servidor 5
4. Requisitos del Sistema vs Recursos Disponibles 6
5. Características del Sistema Operativo 6
6. Compatibilidad y Seguridad 7
7. Servicios de Base de Datos 7
8. Ventajas y Desventajas de Fedora Server 42 para este proyecto 7
9. Conclusión del Relevamiento 8
Manual de instalación del S.O **9**
Introducción **9**
Conocimientos útiles 9
Requisitos **10**
Instalación de la máquina virtual 10
Instalación del Sistema Operativo 12
Conclusión **19**
Configuración de red e Instalación de paquetes **20**
NMTUI: 22
SSH: 24
COCKPIT: 26
COMUNICACIÓN SSH: 27
Paso a paso: 27
Conclusión: 31
Script para creación de grupos y usuarios **32**
Funcionalidades 32
Código 33
Interfaz de red externa **36**
Paso a paso: 36
Conclusión 38
Docker **39**
¿Qué es un contenedor? 39
¿Para qué nos sirve un contenedor en nuestro proyecto? 39
¿Cómo se emplea Docker? 40

¿Qué es Docker Hub?....40
Instalación de Docker....40
Conclusión....42
Imagenes de Docker Hub necesarios para nuestro proyecto....43
httpd (Apache HTTP Server):....43
MySQL:....43
Backups....44
Tipos de backups:....44
Frecuencias....46
Tipos de almacenamiento....46
NAS:....46
RAID:....47
Manera de hacer Backup....47
Manual....47
Automática....48
Simulación de backups en nuestro server....51
Tipos de Backup definidos....52
Herramientas usadas....52
Configuración en cron....52
Qué sucede en la práctica....53
Conclusión....53
Script de Firewalld....54
Funcionalidades....54
Código....55
Script de Backups....58
Funcionalidades....58
Código....58
Script principal....60
Funcionalidades....60
Código....60
Entorno de desarrollo en Linux para el proyecto....62
Paso a paso de la configuración:....63
Instalación de Lamp....63
Instalación de Composer....64
Instalación de Node.js y npm....64
Prueba de que todo funciona con una versión antigua del proyecto....65

Proyecto

Nuestro proyecto de egreso de tercer año de Tecnologías de la Información consta de hacer una aplicación web capaz de gestionar, seguir y contabilizar puntos en una partida del juego de mesa Draftosaurus. Para lograr el cometido contamos con una conformación de grupos para lograr la realización del proyecto y además contamos con una "letra de proyecto" que es un documento formal dado por el Instituto Superior Brazo Oriental, el cual tiene una explicación completa de que trata dicho proyecto y las pautas a seguir para lograr con satisfacción el resultado deseado y a la vez más completo que el equipo pueda lograr.

En la letra del proyecto marca como pauta necesaria lo siguiente:
"Es importante destacar que el servidor de bases de datos deberá ejecutarse en un entorno basado en **GNU/Linux**, garantizando estabilidad y compatibilidad con la infraestructura establecida."

En esta carpeta se mostrará la documentación completa del proceso de cómo el equipo de VifraSoft cumplira los punteos generales y las rúbricas de evaluación tomadas como criterios de logro para la materia Administración de Sistemas Operativos que a su vez es la materia encargada de acompañarnos, guiarnos y ayudarnos en el cumplimiento de la pauta anteriormente mencionada.

Relevamiento del Sistema Operativo

1. Identificación General

- Nombre del sistema operativo: Fedora Server
- Versión: 42
- Distribución base: Red Hat (familia RPM)
- Arquitectura: x86_64 (64 bits)
- Licencia: GNU GPL y otras licencias libres
- Modo de instalación: Consola (sin entorno gráfico)

2. Entorno de Virtualización

- Tipo de instalación: Máquina Virtual
- Software de virtualización: Oracle VirtualBox
- Recursos asignados:
  - CPU: 2 núcleos virtuales
  - RAM: 5 GB
  - Disco: 64 GB de almacenamiento virtual
- Objetivo del entorno: Pruebas y ejecución educativa como servidor de base de datos para un proyecto de egreso.

3. Finalidad del Servidor

- Rol principal: Servidor de base de datos
- Uso específico: Servir como Servidor alojador de base de datos de una aplicación web
- Tipo de entorno: Educativo y de desarrollo (no productivo)
- Acceso: Solo por terminal/SSH

4. Requisitos del Sistema vs Recursos Disponibles

Recurso	Requerido Mínimo	Recomendado	Asignado (proyecto)
CPU	1 núcleo	2 o más núcleos	2 núcleos
RAM	1-2 GB	4 GB o más para BD	5 GB
Disco	20-40 GB	40+ GB según servicios	64 GB
Interfaz gráfica	No necesaria	No necesaria	No instalada

5. Características del Sistema Operativo

- Gestor de paquetes: dnf
- Sistema de paquetes: .rpm
- SELinux: Activado (modo enforcing)
- Firewall: firewalld habilitado
- Soporte para contenedores: Podman incluido
- Actualizaciones: Automáticas posibles vía dnf-automatic
- Herramientas CLI incluidas: sshd, nano/vim, top, htop, dnf, firewall-cmd, cockpit (no habilitado por defecto)

6. Compatibilidad y Seguridad

- Compatibilidad: Excelente con VirtualBox y hardware virtualizado
- Seguridad:
  - Cortafuegos activo
  - SELinux proporciona control de acceso obligatorio
  - Repositorios oficiales firmados
  - SSH seguro (configurable para evitar acceso root directo)

7. Servicios de Base de Datos

- Soporte para bases de datos:
  - PostgreSQL (recomendado en Fedora)
  - MariaDB/MySQL

8. Ventajas y Desventajas de Fedora Server 42 para este proyecto

Ventajas:
- Sistema moderno y actualizado.
- Comunidad activa y documentación oficial completa.
- Seguridad sólida de base (SELinux, firewall).
- Ideal para entornos de desarrollo educativo.

Desventajas:
- Ciclo de vida corto (~13 meses).
- Requiere mantenimiento regular (actualizaciones).
- No trae entorno gráfico, lo cual exige mayor familiaridad con la terminal (aunque es ideal para aprender).

9. Conclusión del Relevamiento

Fedora Server 42 es una distribución robusta, moderna y segura, adecuada para entornos educativos y de desarrollo.

Su integración con VirtualBox es estable, y los recursos asignados son suficientes para cumplir con el rol de servidor

de base de datos en el proyecto de egreso. La ausencia de entorno gráfico refuerza el aprendizaje de administración

por consola, una habilidad clave en servidores reales. Se recomienda mantener actualizado el sistema para aprovechar sus mejoras y correcciones de seguridad.

Manual de instalación del S.O

Introducción

En este manual se mostrará como es la instalación de un sistema operativo linux dentro de una máquina virtual con el propósito de usar el sistema operativo como servidor para cumplir con las pautas del proyecto de egreso de los estudiantes de tercer año de Bachillerato en Tecnologías de la Información del Instituto Superior Brazo Oriental.
En este manual se explicará el paso a paso que utilizaron los estudiantes Franco Reyes, Santiago Bittencourt y Victoria Larrosa en la instalación del Sistema Operativo en una máquina virtual con los conocimientos adquiridos en las clases de Administración de Sistemas Operativos de la mano del Profesor Santiago Martinez

Conocimientos útiles

Sistema operativo: Un sistema operativo es el software esencial que permite que tu computadora funcione. Imagínalo como el cerebro que conecta el hardware (las partes físicas de la computadora) con el usuario y los programas.

ISO: Una imagen ISO es como una fotocopia digital perfecta de un CD, DVD o Blu-ray. Guarda todo lo que está en el disco (archivos, carpetas y cómo está organizado), pero en un solo archivo en tu computadora. Es la forma más común de descargar sistemas operativos

Máquina Virtual: Una máquina virtual (VM) es como una computadora simulada que funciona dentro de tu computadora real. Imagina que tu PC es una casa, y la máquina virtual es un apartamento completo y funcional dentro de ella.

Servidor Linux: Un servidor Linux es una computadora (física o virtual) que tiene instalado el sistema operativo Linux y está configurada para ofrecer servicios a otras computadoras o dispositivos (clientes) a través de una red.

Requisitos

Instalación de la máquina virtual

Para poder tener un sistema operativo dentro de una máquina virtual primero necesitaremos instalar un software de virtualización. ¿Que es un software de virtualización?
Un software de virtualización crea una capa que te permite usar una sola computadora física para ejecutar múltiples computadoras virtuales a la vez. Piensa que es como tener varias computadoras dentro de una sola.

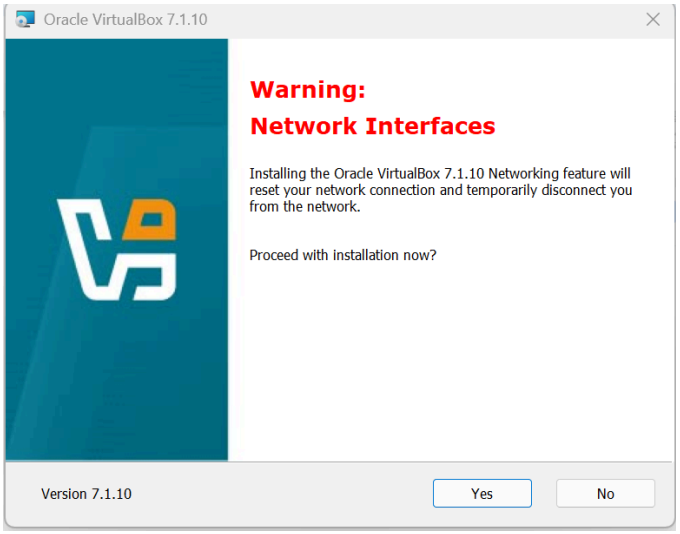
En nuestro caso utilizaremos Oracle VirtualBox que como su nombre lo dice es software de virtualización distribuido por el equipo de Oracle, se optó por este medio por su manejo intuitivo, gratis para uso educativo y profesional y su fácil instalación

Enlace de descarga: https://www.virtualbox.org/wiki/Downloads
Después de entrar al enlace de descarga en la parte inferior a la izquierda de la pestaña nos mostrará el siguiente cuadro:

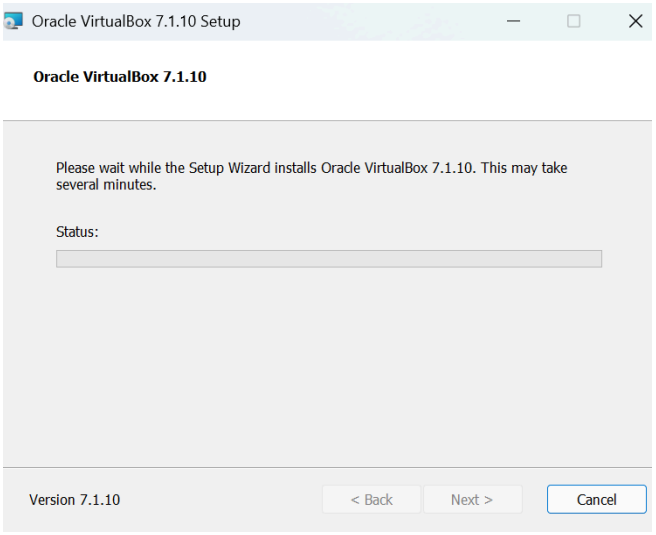
Aquí se debe seleccionar la plataforma donde quiere ser instalada la virtualbox, una vez clickeado la opción que sea necesaria la descarga del paquete de instalación del programa iniciará automáticamente.

Una vez descargado el paquete de instalación, debemos ejecutarlo para que comience la instalación.
A continuacion una guia rapida de como comenzar la instalación

El primer mensaje que aparecera sera el siguente, debemos presionar “yes”



A continuación empezará con la instalación de paquetes y únicamente hay que esperar que termine de cargar



Por último, después de que termine de cargar nos mostrará el cuadro para finalizar la instalación, presionamos en que se ejecute una vez instalada y en “finish”

Instalación del Sistema Operativo

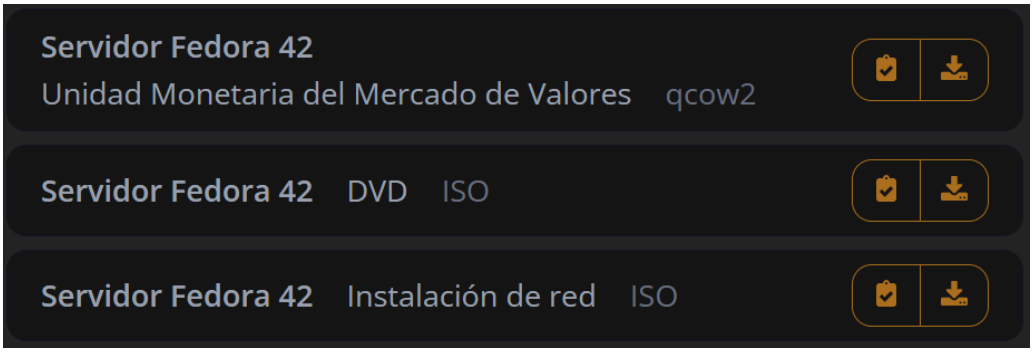
Una vez instalada la máquina virtual debemos elegir que sistema operativo utilizar para instalar su ISO, los criterios para elegir el SO son propios según respondan a las necesidades de cada usuario, en nuestro caso elegiremos un sistema Linux sin interfaz gráfica, es decir solo servidor ya que nuestro propósito será usar únicamente la terminal de linux para el manejo de un servidor.

Elegimos Fedora Server, ¿Qué es Fedora? Fedora es una distribución de GNU/Linux de propósito general, desarrollada por una comunidad internacional de ingenieros, diseñadores y usuarios, y patrocinada principalmente por Red Hat, ¿Qué es Fedora Server? Fedora Server es una de las ediciones oficiales del sistema operativo Fedora, diseñada específicamente para funcionar como un servidor. A diferencia de Fedora Workstation, que está optimizada para el uso en computadoras de escritorio y portátiles (con una interfaz gráfica completa y herramientas para el usuario final).

¿Por qué elegimos Fedora Server?

Elegir Fedora Server es una excelente opción si buscás un sistema operativo moderno, seguro y respaldado por una comunidad activa y Red Hat. Fedora Server ofrece acceso a las últimas tecnologías de Linux, como versiones recientes del kernel, systemd, firewalld y la herramienta de administración web Cockpit. A pesar de ser una distribución de vanguardia, mantiene una buena estabilidad, ideal para entornos de prueba, desarrollo y hasta producción ligera. Recibe actualizaciones frecuentes y parches de seguridad rápidamente, lo que la hace adecuada para quienes priorizan la innovación sin sacrificar confiabilidad.

Enlace de descarga: https://fedoraproject.org/server/download

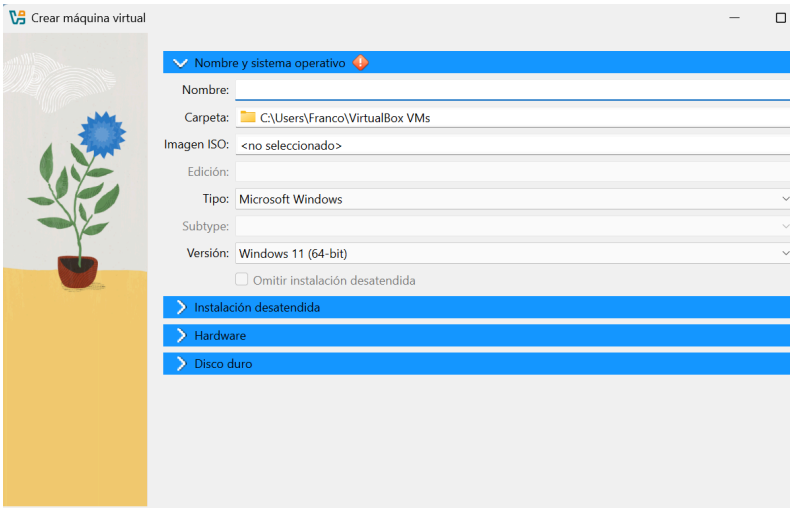


Seleccionamos la tercer opción que es la mas rapida y como dice el texto es la instalación de red y comenzará a descargarse automáticamente el paquete de instalación

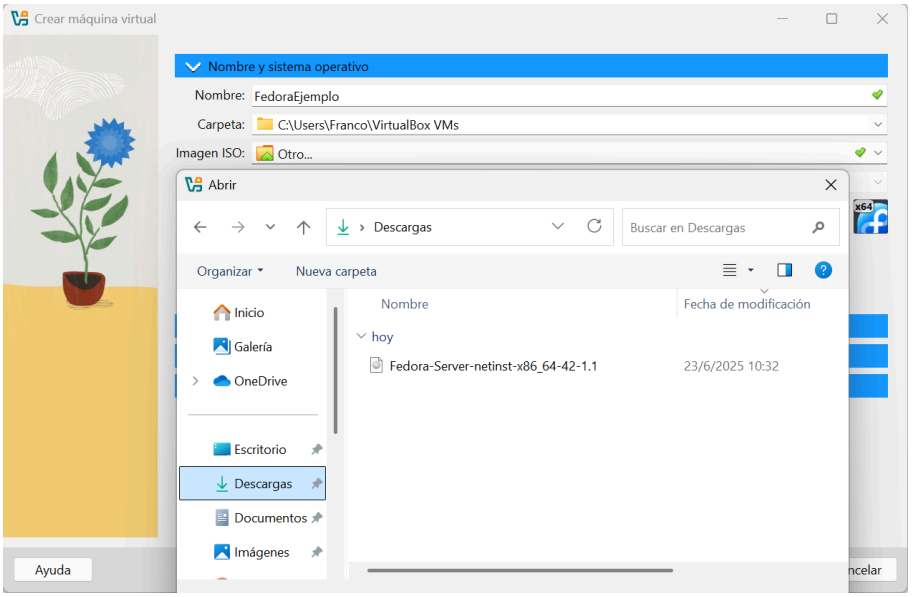
Como siguiente paso hay que dirigirse a la máquina virtual que previamente se había ejecutado y continuación en el paso a paso para poder ejecutar la ISO en la VM

Una vez estando en el programa de Virtualbox veremos dos botones en la parte superior de la ventana que dicen “Nueva” y “Añadir” aquí debemos presionar “Nueva”

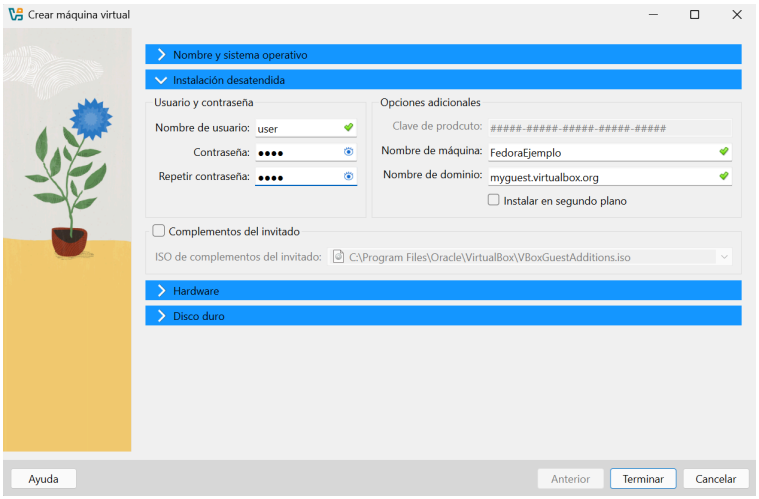
Una vez presionado el botón “Nueva” nos mostrará la siguiente ventana que es donde pondremos la ISO que elegimos y además nos dejará configurar el hardware virtual



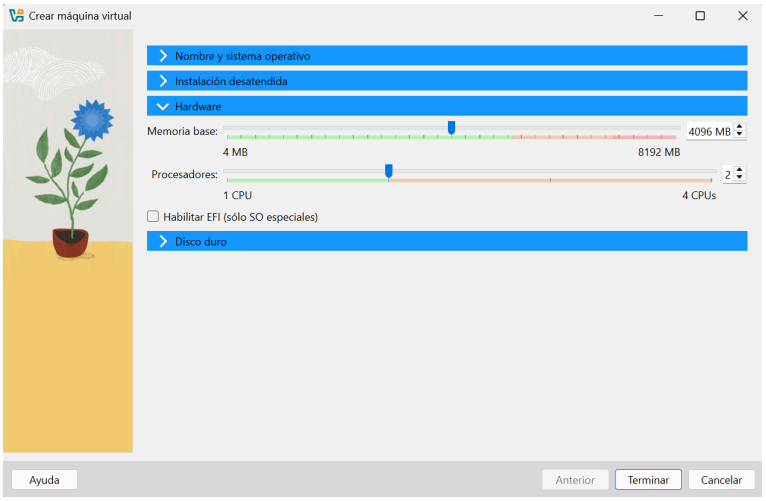
A continuación un ejemplo de como lo haría nuestro equipo



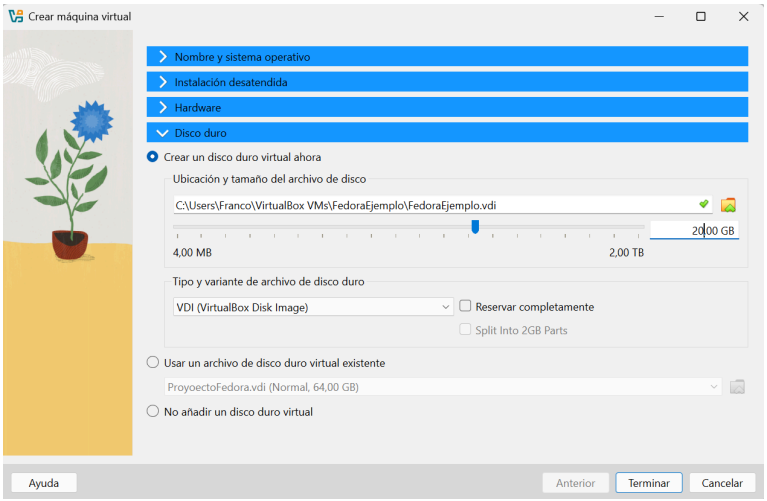
Aca mostramos como colocar el ISO: imagen ISO >> Otro… >> Archivo.iso



La función de la instalación desatendida es que el proceso de instalar la ISO sea de forma automática, es para simplificar el trabajo y ahorrar más tiempo.

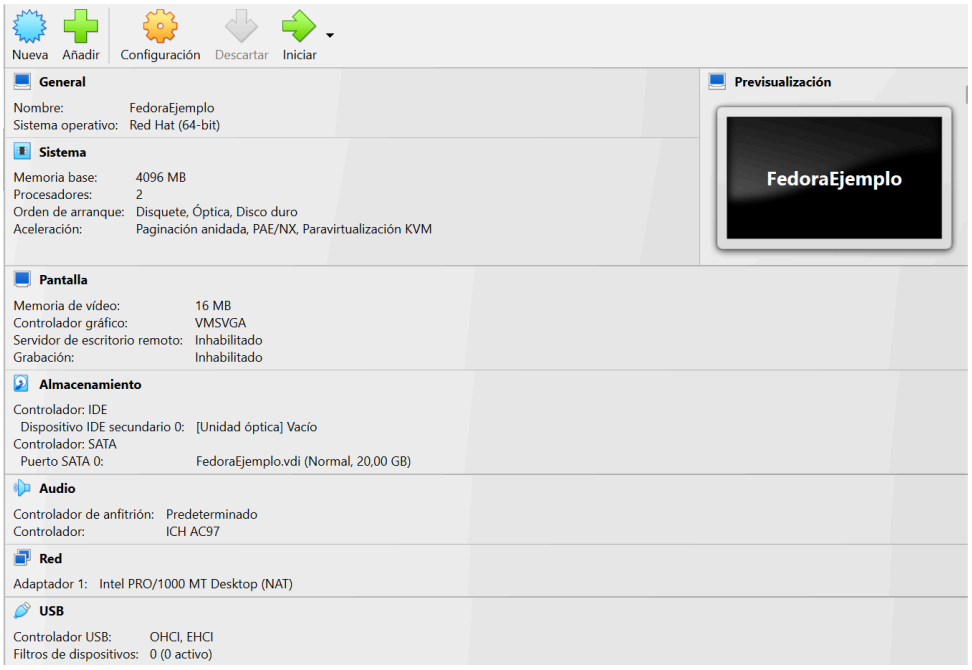


Dejamos en 4gb de RAM porque al ser usado solo en el contexto de servidor no es necesario más de eso y 2 CPUs para evitar cuellos de botella y mejorar el rendimiento.

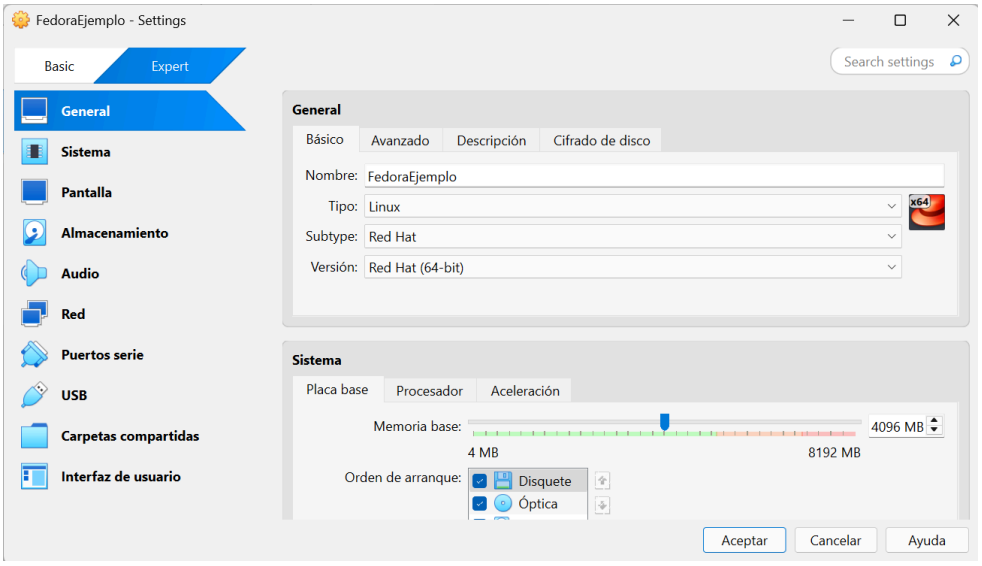


El tamaño de disco dependerá de la necesidad de cada usuario, lo dejamos en 20gb porque para nuestro uso es el correcto tamaño.

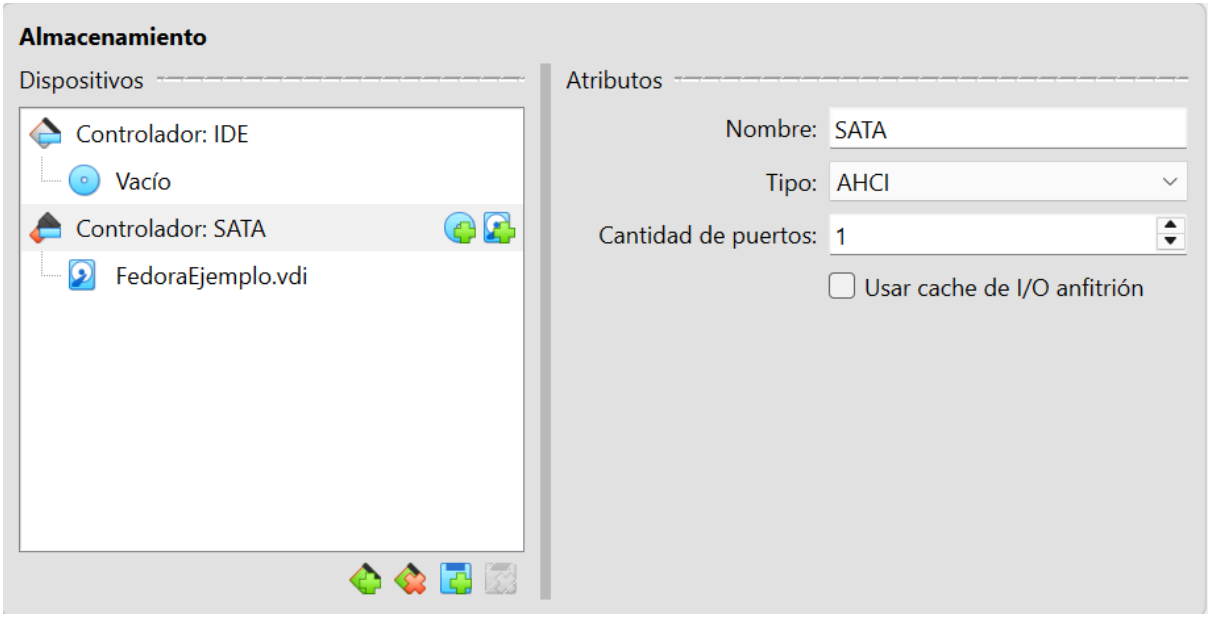
Una vez hecha la configuración inicial tendremos que hacer un paso muy importante para que la ISO se instale correctamente



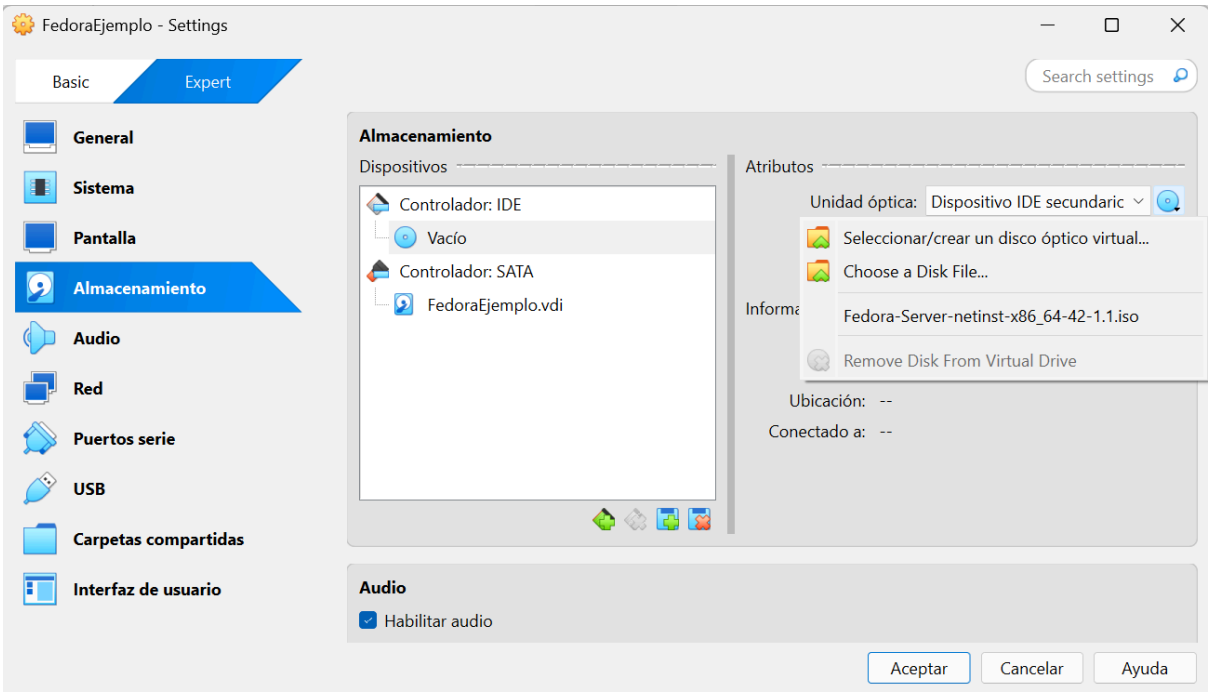
Esta es la información general de la maquina que creamos pero como se muestra en el apartado de almacenamiento muestra que el dispositivo IDE secundario esta vacio, para cambiarlo tenemos que presionar el boton de configuracion que se muestra en la parte superior



Una vez presionado el botón nos mostrará la siguiente ventana y tendremos que dirigirnos hacia Almacenamiento que nos muestra en la parte de la izquierda



Una vez aquí debemos dirigirnos a la parte donde dice “Controlador: IDE” y tocar la imagen del CD que dice vacío

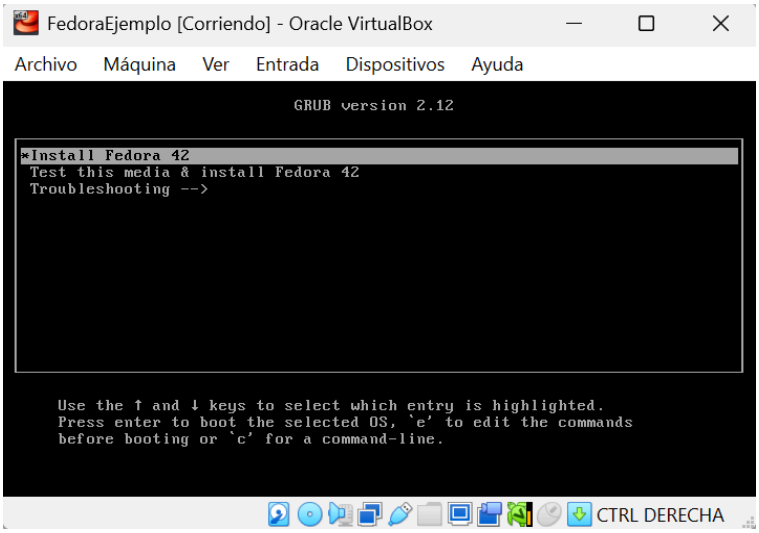


Estando aquí en la parte derecha de la ventana en donde dice “Unidad óptica” presionamos la imagen del CD con una flecha para elegir el dispositivo IDE y nos aparecerá el archivo ISO que usamos para crear la máquina virtual, debemos seleccionarlo.

¿Por qué hay que hacer esto?, se elige un dispositivo IDE para asegurar que la máquina virtual detecte bien el disco o el instalador del sistema. Algunos sistemas operativos, sobre todo los más básicos o antiguos, reconocen mejor los discos conectados por IDE. Además, la máquina virtual puede estar configurada para arrancar primero desde dispositivos IDE. Por eso, al usar esta opción se evitan errores durante la instalación.

Una vez teniendo todas las configuraciones estén prontas tendremos que ejecutar o iniciar la máquina virtual, el botón que la inicia esta cercano al boton de configuracion en la parte superior de la ventana

lo siguiente es esperar que cargue la máquina para poder comenzar con la instalación automática del sistema operativo, solamente tendremos que esperar un breve momento y nos saldrá el siguiente mensaje



Sin importar qué distribución de linux sea, siempre saldrán estas tres opciones que elegiremos la opción con las flechas y para seleccionar usamos el enter, teniendo en cuenta esto debemos seleccionar la opción de “Install Fedora 42”, depues de esto tenemos que esperar un momento y así finalmente tendremos nuestro sistema operativo linux instalado dentro de una máquina virtual.

Conclusión

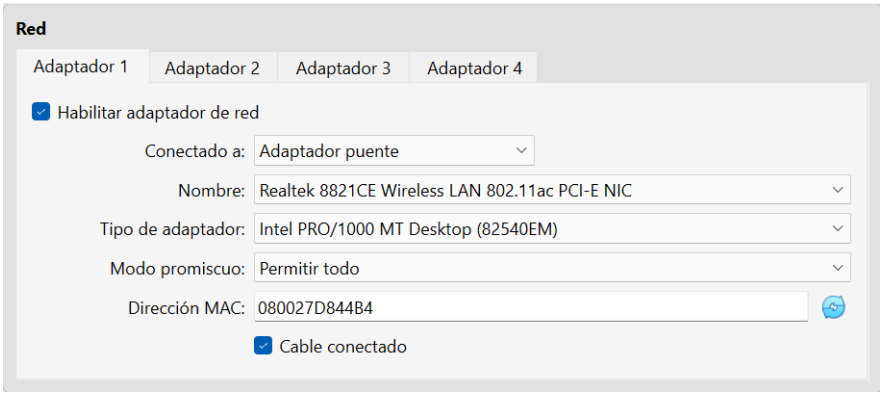
En este manual aprendimos a crear e instalar una máquina virtual con Fedora Server utilizando VirtualBox. A lo largo del proceso configuramos el hardware virtual, seleccionamos el tipo de almacenamiento adecuado y completamos la instalación del sistema operativo.

Elegimos Fedora Server porque es una distribución moderna, estable y pensada para uso en servidores, ideal para prácticas y entornos educativos. Usar VirtualBox nos permite simular un servidor real sin necesidad de modificar el equipo físico del estudiante.

Este entorno servirá como base para aprender a administrar servicios, manejar configuraciones de red, instalar software y aplicar comandos de administración de sistemas. Recomendamos mantener el sistema actualizado y hacer copias de seguridad antes de probar cambios importantes.

Configuración de red e Instalación de paquetes

El equipo de VifraSoft configuró la red de del servidor linux desde la máquina virtual que lo aloja. El objetivo de esta configuración es poder tener una comunicación más eficiente entre dispositivos para poder efectuar el correcto funcionamiento de la aplicación, a continuación se mostrará todo lo que el equipo hizo para tener una correcta y eficiente comunicación entre el dispositivo anfitrión y además tener la posibilidad de administrar el servidor linux fácilmente desde el navegador



Primeramente dejamos la configuración de red de la máquina virtual como se muestra en la imagen. Está configurada para que se pueda conectar a internet y asi poder descargar los servicios deseados como ssh, nmtui y cockpit y también para que pueda haber una conexión entre la pc anfitrión y la VM, la dirección MAC siempre sera unica, no habrán dos iguales, y es sumamente necesario que la casilla de cable conectado esté seleccionada con el tic.

Una vez la configuración esté lista deberemos iniciar la maquina virtual Linux
Los servicios que utilizaremos serán: ssh, nmtui y cockpit
Los comandos para instalarlos son los siguientes:

ssh:

dnf install openssh-server para instalar ssh

systemctl enable sshd para activarlo

systemctl start sshd para iniciarlo

firewall-cmd --add-service=ssh para permitir las comunicaciones

nmtui:

dnf install NetworkManager-tui para instalar nmtui

cockpit:

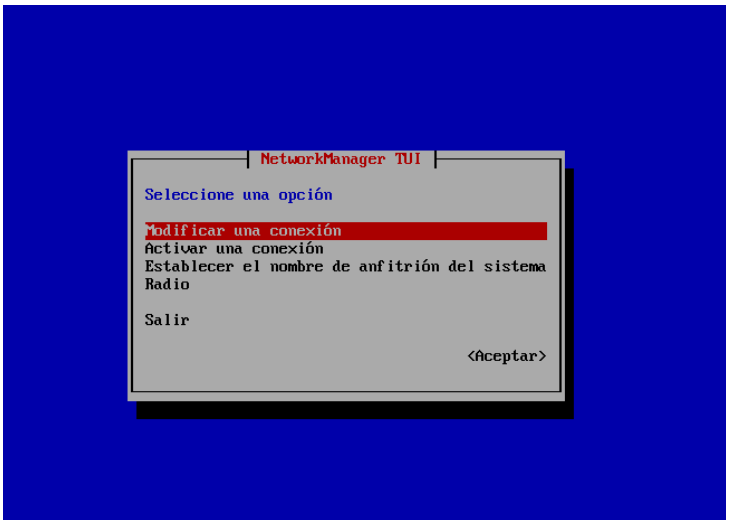
dnf install cockpit para instalar cockpit

systemctl enable cockpit.socket para activar el cockpit

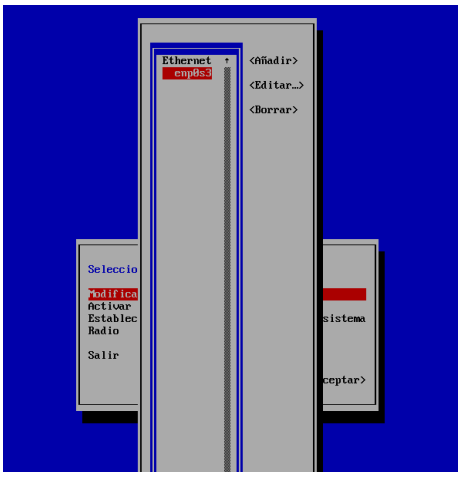
Una vez teniendo en cuenta que tenemos todos los servicios instalados en la máquina virtual linux pasaremos a la configuración y uso de cada uno

NMTUI:

Para poder utilizar el servicio de nmtui debemos ingresar este mismo en la terminal de linux y nos abrirá un menú tal que así:

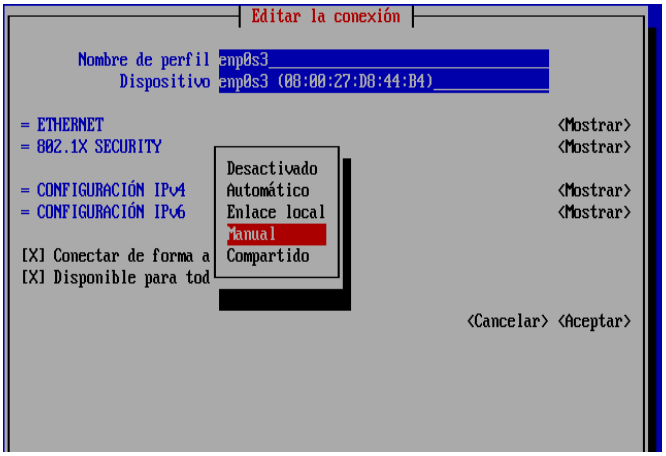


Debemos presionar **enter** en el mensaje “Modificar una conexión” como en la imagen.

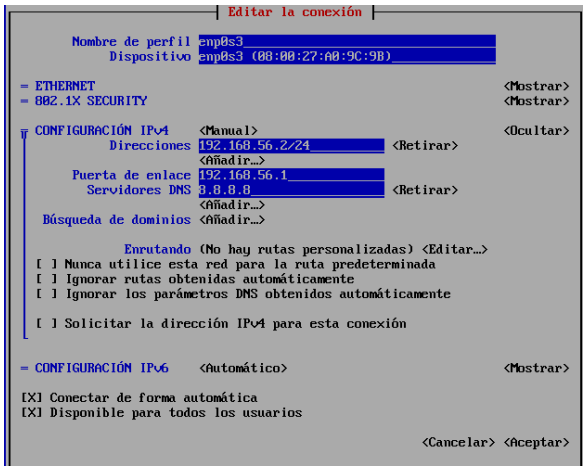


cuando se despliegue este menú lo que debemos hacer es seleccionar la única opción como se muestra en la imagen

Nos aparecerá el siguiente menú



Utilizando las flechas del teclado nos dirigiremos hacia configuración ipv4 para poner la opcion de configuracion manual como se muestra en la imagen, luego de esto deberás presionar el mensaje de "<Mostrar>" del lado derecho para ingresar las configuraciones deseadas



Para cada configuración aparecerá el mensaje de "<Añadir>" y solo configuraremos "Direcciones", "Puerta de enlace" y "DNS", la configuración de la imagen son únicamente ejemplos y cada uno debe ser único por servidor y a la vez coincidir en el rango de la ip de la pc anfitriona para poder tener una comunicación correcta entre ellas, el DNS recomendamos siempre dejarlo en "8.8.8.8" y la puerta de enlace siempre sera XXX.XXX.XXX.1 según la ip

El motivo por el cual se instala nmtui es para tener un panel para modificar la o las conexiones de nuestra máquina virtual, en pocas palabras se utiliza para evitar hacer las configuraciones de conexión evitando usar comandos en consola lo cual genera una mayor comodidad y eficiencia.

SSH:

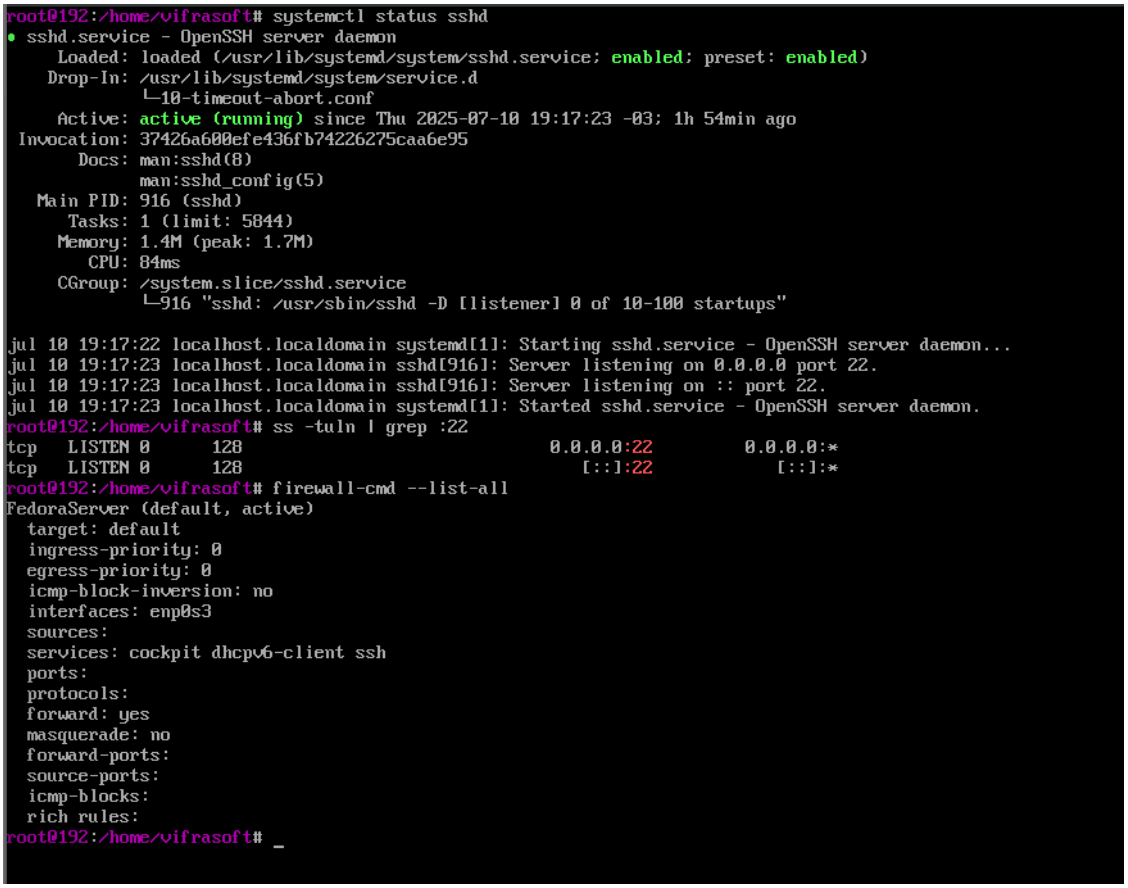
SSH, sigla de Secure Shell, es un protocolo de red que permite establecer una conexión segura y cifrada entre dos computadoras. En sistemas Linux, se utiliza principalmente para acceder y controlar remotamente otro equipo a través de la línea de comandos.

El uso de SSH es fundamental en la administración de servidores, especialmente aquellos que no cuentan con interfaz gráfica, como muchas versiones de Linux Server (por ejemplo, Fedora Server).

En la página 3 del documento se muestran los comandos de instalación de SSH pero de igual forma los reiteramos.
dnf install openssh-server para instalar ssh
systemctl enable sshd para activarlo
systemctl start sshd para iniciarlo
firewall-cmd --add-service=ssh para permitir las comunicaciones

SSH también permite otras funciones avanzadas, como la transferencia segura de archivos (mediante scp o sftp), la creación de túneles de red, y la autenticación mediante llaves criptográficas, lo que mejora aún más la seguridad.

Una vez instalado todo correctamente e inicializado el servicio veremos el estado del servicio



En la imagen se muestra el comando **systemctl status sshd** la información que otorga este comando es en qué estado se encuentra el servicio que como muestra está activado por el comando **systemctl enable sshd.**

Después el conjunto de comandos **ss -tuln | grep :22** que es para verificar que el puerto definido por defecto para establecer las conexiones

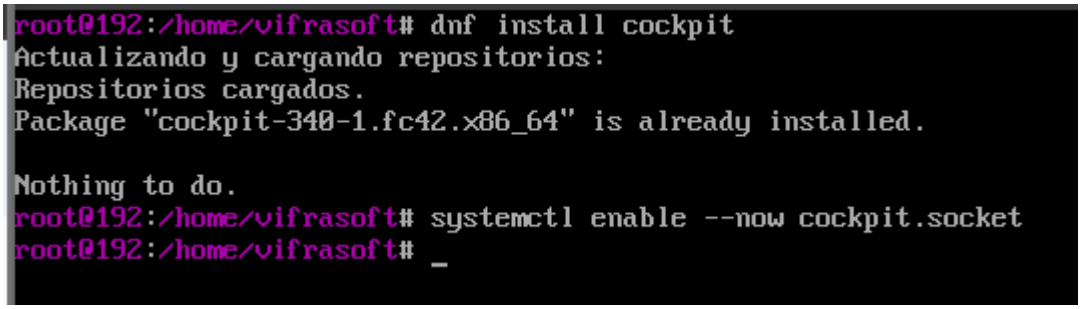
Por último usamos el comando **firewall-cmd --list-all** ya que este nos muestra la configuración actual del firewall en linux lo cual nos permite hacer diagnósticos de errores de conexión o seguridad y además ver que está permitido o bloqueado dentro del mismo, por ejemplo en la imagen muestra los servicios permitidos.

COCKPIT:

Cockpit es una herramienta de administración de servidores Linux que permite gestionar el sistema a través de una interfaz web moderna y fácil de usar. Está diseñada para facilitar tareas comunes sin necesidad de utilizar exclusivamente la línea de comandos, lo que la convierte en una opción ideal tanto para administradores experimentados como para principiantes.

A través de Cockpit es posible supervisar el estado del sistema en tiempo real, controlar el uso de CPU, memoria, red y disco, gestionar servicios, usuarios, paquetes, y configurar el firewall. También permite administrar discos y particiones, revisar los logs del sistema y acceder directamente a una terminal desde el navegador.

Su instalacion y activacion son muy sencillas y se mostrará en la siguiente imagen



Cockpit se accede ingresando la dirección IP del servidor en un navegador, seguida del puerto 9090 (por ejemplo, https://192.168.56.2:9090).

COMUNICACIÓN SSH:

A continuación haremos que la PC anfitriona se conecte con la máquina virtual linux mediante el SSH con autenticación de llave pública y privada

Las llaves pública y privada son un par de archivos criptográficos que se utilizan para autenticar de forma segura el acceso a un servidor. En lugar de usar una contraseña que podría ser interceptada o adivinada, SSH usa este par de llaves para verificar tu identidad.

La **llave pública** es un archivo que puedes compartir sin preocupaciones. Piensa en ella como un candado abierto que entregas al servidor al que deseas conectarte. Este candado puede ser visto por cualquiera, pero solo puede ser abierto por la llave privada correspondiente.

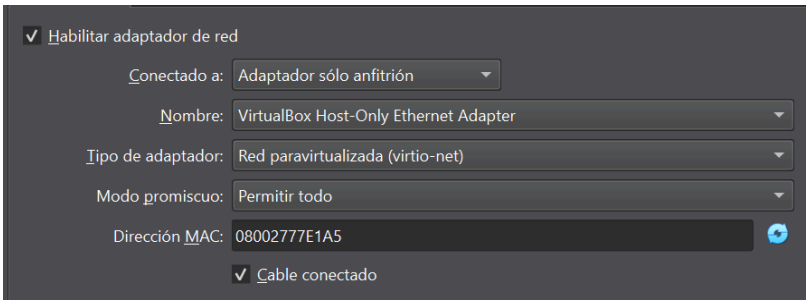
La **llave privada** es el componente secreto y crucial de este sistema. Esta llave debe permanecer siempre en tu posesión y nunca, bajo ninguna circunstancia, debe ser compartida con nadie. Es la llave que abre el candado creado por la llave pública.

En conjunto, este par de llaves permite una autenticación robusta: la llave pública cifra un mensaje que solo puede ser descifrado por la llave privada. Al descifrar y responder correctamente, demuestras que tienes la llave privada sin necesidad de enviarla por la red, lo que la convierte en un método de acceso mucho más seguro que las contraseñas.

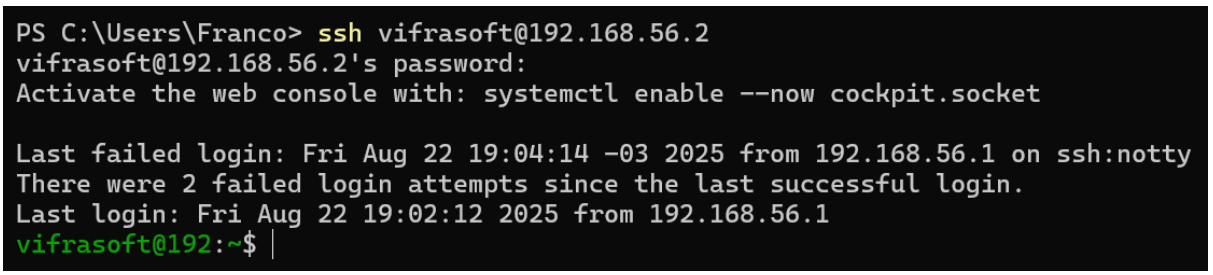
Paso a paso:

El primer paso para poder realizar la conexión SSH mediante llaves públicas y privadas es hacer la conexión SSH mediante contraseña para poder compartir las llaves entre ambas máquinas (PC anfitriona-VM), para realizar esto los pasos a seguir son simples, lo principal es tener la VM iniciada y tener el cmd abierto.

El primer paso que vamos a realizar es el cambio de adaptador 1 a sólo anfitrión para poder realizar correctamente la conexión ssh. Ejemplo:



Siguiente a esto hay que usar el comando de conexión SSH: **ssh usuario@IP**, en nuestro caso sería **ssh vifrasoft@192.168.56.2,** siguiente a este comando nos pedirá la contraseña para entrar al usuario, idéntico a cuando entramos a la terminal en el servidor, ejemplo:



Lo recomendable es tener dos pestañas en cmd, una con la terminal de la PC anfitriona y otra con la terminal del servidor, esto nos permitirá agilizar cualquier comunicación de datos entre ambas máquinas.

Lo siguiente a hacer es crear las llaves pública y privada, para hacerlo en la terminal de la PC anfitriona pondremos el comando **ssh-keygen** y lo siguiente a este paso es muy intuitivo, consta de presionar enter hasta que se complete, ya que el archivo de las llaves estarán dentro de la carpeta .ssh en que a su vez esta esa carpeta estará en el directorio del usuario y si la carpeta .ssh no existe la creara junto a las llaves, despues opcionalmente nos pedirá que dejemos escrito un passphrase que es como una contraseña adicional pero como el motivo del uso de las llaves es evitar la comunicación mediante contraseñas para que no sean descifrables lo dejaremos vacío presionando enter, ejemplo:


```
C:\Users\Franco>ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (C:\Users\Franco/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\Franco/.ssh/id_ed25519
Your public key has been saved in C:\Users\Franco/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:O7TdM8aY09oHEuIVpczOrez9rVKhJpYpJptK5kNUBrk franco@DESKTOP-K1M1DKE
The key's randomart image is:
+--[ED25519 256]--+
|     .o     ..    |
|    . o  o..      |
|     +    +.      |
|    E   .oo.  .   |
|   .   .Soo+.. .  |
|    . ..+=*B+ .   |
|   .o  =+oB+Bo    |
|   +. o  o *.o..  |
|    oo    o o+o..|
+----[SHA256]-----+
```

Una vez estén generadas las llaves deberemos copiar la llave publica para poder pegarla en el server, antes de esto debemos recordar que por defecto las llaves se guardan en la ruta **/home/usuario/.ssh/authorized_keys,** si no existe ni la carpeta .ssh ni el archivo authorized_keys los crearemos con mkdir para la carpeta y touch para el archivo, dicho esto continuamos.

deberemos ir a la carpeta .ssh de la pc anfitriona y el archivo que sea .pub lo abrimos con el bloc de notas y copiamos el contenido, siguiente a esto en la terminal del servidor abierta desde la PC anfitriona iremos al archivo authorized_keys con el comando **sudo nano /.ssh/authorized_keys** y copiaremos la llave con un ctrl+V.

Aunque ya esté copiada la llave nos hace falta un paso fundamental para poder dejar de lado las conexiones mediante contraseña y que sea solamente con la llave pública y privada es cambiar la configuración del SSH, es un paso simple pero fundamental.

Para cambiar la configuración tendremos que pararnos en la ruta raíz del servidor y usar el comando **sudo nano etc/ssh/sshd_config** y nos abrirá un archivo que es el que usa el servicio SSH para funcionar, la recomendación es inhibir las configuraciones que dictan que la comunicación debe ser mediante contraseña y dejar solo las configuración en la que use las llaves pública y privada.

A continuación cómo lo configuro el equipo de Vifrasoft:

```
PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

PubkeyAuthentication yes

# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
AuthorizedKeysFile      .ssh/authorized_keys

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no

# Change to no to disable s/key passwords
KbdInteractiveAuthentication no
```

PermitRootLogin prohibit-password: prohíbe las contraseñas

PubKeyAuthentication yes: permite la comunicación mediante llaves

PasswordAuthentication no: niega la autenticación con contraseñas

KdbInteractiveAuthentication no: esto quiere decir que el que se quiera comunicar recibirá uno o más desafíos para acceder pero la seguridad es mínima, por eso se cancela.

Se puede apreciar que las configuraciones marcadas en azul tienen un “#”, esto hace que el protocolo lo tomé como un comentario y no sea una configuración que tome en cuenta.

Para que funcionen los nuevos cambios debemos guardar con ctrl+O, enter y ctrl+X y con el comando **sudo systemctl restart sshd,** esto reiniciará el servicio SSH y tome los cambios hechos en el archivo.

Para confirmar que funcionó todo debemos salir del servidor con un **exit** y eso nos devolverá a la terminal de la PC anfitriona y tendremos que hacer de nuevo la conexión SSH y si todo salio con exito nos iniciara el servidor solo y sin ninguna autenticación, esto quiere decir que usó las llaves creadas e hizo la comunicación correctamente. ejemplo:


```
PS C:\Users\Franco> ssh vifrasoft@192.168.56.2
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Fri Aug 22 21:04:11 2025 from 192.168.56.1
vifrasoft@192:~$
```

Como se ve no pido ninguna autenticación manual, esto sucede porque se usaron las llaves, dando como resultado un rotundo éxito en la configuración para la comunicación mediante llaves.

Conclusión:

Lo que se ha visto en esta documentación es la forma en cómo el equipo de VifraSoft configuró la red del servidor para dejar una IP estática y tener los puertos necesarios abiertos para tener una correcta comunicación con otros equipos, principalmente el host.
Además se vio la instalación, gestión y configuración de los paquetes necesarios para que el servidor pueda alojar y ejecutar la aplicación deseada, reiterando los paquetes: **SSH** para el control remoto, **cockpit** para la administración web y **nmtui** para la configuración de red.

Script para creación de grupos y usuarios

Para el manejo óptimo en la organización de los usuarios y grupos de nuestro servidor se decidió hacer un Script de Shell. Un Script de Shell es un programa que está creado con instrucciones que son ejecutadas por un Shell en un entorno de líneas de comandos de Unix o Linux. El código no es compilado ni precompilado, se va ejecutando línea por línea efectuando lo que cada instrucción le indica.

Funcionalidades

En base a los requerimientos de nuestro script llegamos a la siguientes funcionalidades de comandos de la siguiente forma:

- Crear usuario
- Eliminar usuario
- Crear grupo
- Eliminar grupo
- Listar usuarios
- Listar grupos
- Mover usuario a grupo
- Salir

En base a estas instrucciones a seguir armamos el script de forma que cumpliera con todas las instrucciones, a continuación le vamos a mostrar la forma en la que aplicamos las funcionalidades a los comandos:

Código

```
1   #!/bin/bash
2
3   # Comprobar si el script se ejecuta como root
4   if [[ "$EUID" -ne 0 ]]; then
5     echo "Este script debe ejecutarse como root."
6     exit 1
7   fi
8
9   # Función para mostrar el menú
10  mostrar_menu() {
11    clear
12    echo "========= GESTIÓN DE USUARIOS Y GRUPOS ========="
13    echo "1. Crear usuario"
14    echo "2. Eliminar usuario"
15    echo "3. Crear grupo"
16    echo "4. Eliminar grupo"
17    echo "5. Listar usuarios existentes"
18    echo "6. Listar grupos existentes"
19    echo "7. Mover usuario a grupo"
20    echo "0. Salir"
21    echo "================================================"
22    echo -n "Seleccione una opción: "
23  }
```

Al principio colocamos #!/bin/bash para que el sistema operativo intérprete al archivo como Bash y así pueda ejecutar correctamente los comandos que están a continuación del archivo.

La línea 3 se percibe como un comentario por ende el sistema operativo no lo toma como ejecutable pero desde la linea 4 a la 7 se abre un if para que el sistema detecte si el que abre el archivo lo ejecuta el root o un usuario con permisos de superusuario, se hace así para evitar usar el **sudo** en el archivo.

De la línea 10 a la 23 se abre una función en la cual determina la opciones que hay para elegir y se usarán para tomar el dato de la opción que elija el usuario que ejecute el archivo.

```
25  while true; do
26      mostrar_menu
27      read opcion
28
29      case $opcion in
30          1)
31              read -p "Ingrese el nombre del nuevo usuario: " usuario
32              useradd "$usuario" && echo "Usuario '$usuario' creado."
33              sleep 2
34              ;;
35          2)
36              read -p "Ingrese el nombre del usuario a eliminar: " usuario
37              userdel -r "$usuario" && echo "Usuario '$usuario' eliminado."
38              sleep 2
39              ;;
40          3)
41              read -p "Ingrese el nombre del nuevo grupo: " grupo
42              groupadd "$grupo" && echo "Grupo '$grupo' creado."
43              sleep 2
44              ;;
45          4)
46              read -p "Ingrese el nombre del grupo a eliminar: " grupo
47              groupdel "$grupo" && echo "Grupo '$grupo' eliminado."
48              sleep 2
49              ;;
50          5)
51              echo "Usuarios del sistema:"
52              cut -d: -f1 /etc/passwd
53              sleep 3
```

Aquí vemos como hacemos uso de la estructura de control **while** para que cada vez que el usuario elija la opción y la use no se cierre el programa y pueda ejecutar otra opción si así lo desea.

A partir de la línea 29 usamos el **case** que usando la variable opción determina qué eligió el usuario y se dirige a los comandos de la opcion en cuestion, como se ve en la imagen los comandos son enteramente los que se pueden usar en la terminal de linux solamente que usa un poco de semántica de Bash.

Al final de cada opción se le agrega un sleep para que el usuario después de ejecutar la opción pueda leer el mensaje de respuesta en caso de haber alguno.


```
    6)
      echo "Grupos del sistema:"
      cut -d: -f1 /etc/group
      sleep 3
      ;;
    7)
      read -p "Ingrese el nombre del usuario: " usuario
      read -p "Ingrese el grupo al que desea moverlo: " grupo
      if getent group "$grupo" > /dev/null && id "$usuario" > /dev/null 2>&1; then
        usermod -g "$grupo" "$usuario" && echo "Usuario '$usuario' movido al grupo '$grupo'."
      else
        echo "El usuario o el grupo no existen."
      fi
      sleep 2
      ;;
    0)
      echo "Saliendo..."
      exit 0
      ;;
    *)
      echo "Opción inválida."
      sleep 1
      ;;
  esac
done
```

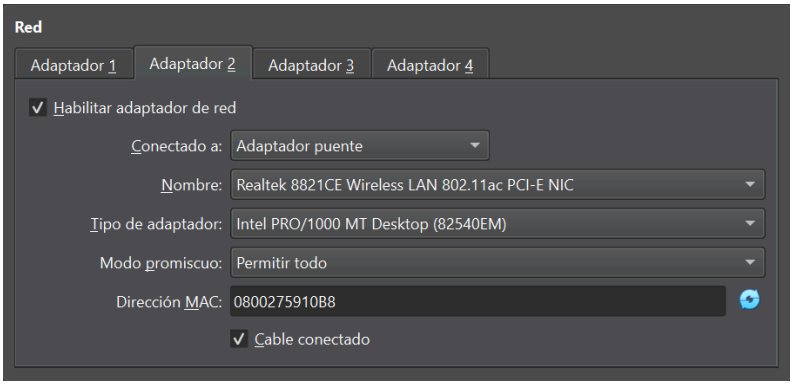
Aquí en la parte final de nuestro código se aprecian las opciones que no eran vistas en la imagen anterior. Sin nada destacable que comentar ya que el código se da a entender fácilmente y no tiene algo que no se haya explicado, se da por concluido la explicación del código

Interfaz de red externa

Hasta el momento el servidor solo tenia configurado un adaptador de conexión y estaba configurada para que solo el anfitrión tuviera la conexión con el servidor y lo teníamos configurado de esta forma para poder tener la conexión mediante SSH, esto nos limita a poder descargar paquetes o servicios mediante red WIFI ya que el servidor va a buscar la comunicación con el adaptador que tiene la configuracion “Solo Anfitrión” y claramente no va a recibir ninguna respuesta, y si se modifica este adaptador podremos perder la conexión SSH por ende lo más factible a hacer es configurar un nuevo adaptador solo para la conexión WIFI, esto nos garantiza no perder la comunicación mediante SSH y tener WIFI para cualquier descarga que lo necesite

Paso a paso:

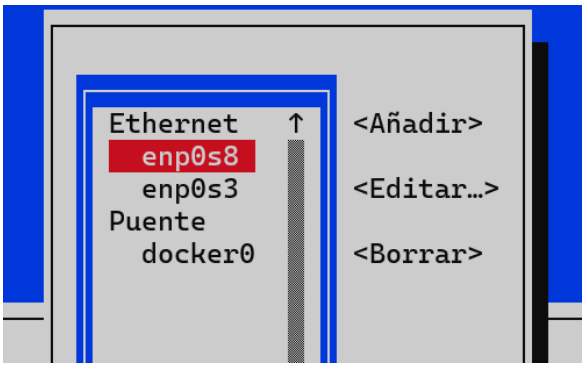
Lo primero que debemos hacer es abrir la configuración de la máquina virtual en nuestro software de virtualización y entrar a la configuración de red y presionaremos en donde dice “Adaptador 2” y habrá una casilla llamada “**Habilitar adaptador de red**” y procederemos a la configuración:



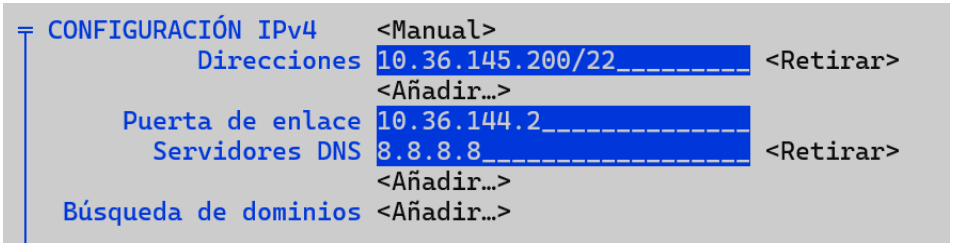
Lo único distinto a la configuración del adaptador 1 es que el 2 está conectado a un adaptador puente, lo demás es la configuración que viene por defecto al elegir adaptador puente y la dejaremos así porque es la más eficiente para cubrir nuestra necesidad de tener conexión a WIFI.

Una vez este paso esté listo los siguientes pasos los haremos dentro del servidor mediante la terminal Linux.
Guardamos la configuración e iniciaremos el servidor con normalidad y para tener una mayor comodidad haremos conexión SSH en el powershell de Windows.

Una vez iniciado el servidor usaremos la aplicación nmtui que ya teníamos instalada y, recordar que si no somos root debemos usar sudo para que no nos rechace las configuraciones de red, seguido e ejecutar nmtui solamente escribiendo "**nmtui**" o "**sudo nmtui**" y usaremos la opción de "**Modificar una conexión**" y elegiremos modificar la conexion "**enp0s8**" ejemplo:



Lo que editaremos de esta conexión será su ip, se sabe que el router que da WIFI asigna las ip pero dejaremos una manualmente para evitar cualquier tipo de error de conexión, por ejemplo un error de asignación ip, a continuación un ejemplo de una configuración:



Elegimos esta IP (10.36.145.200/22) porque es una **dirección privada** que permite que la máquina virtual se integre a la perfección en una red local como un dispositivo independiente.

Hecho este cambio guardaremos la configuración asignada y desde el menú principal y saldremos hacia la terminal de nuevo.

Desde la terminal usaremos primero el comando **sudo ip link set enp0s8 down** primero y luego **sudo ip link set enp0s8 up,** esto hará que se "reinicie" la interfaz de red y los siguientes comandos son cruciales, como por defecto al momento de hacer una descarga la ruta por defecto era la de enp0s, por ende tendremos que darla de baja y dejar la de enp0s8 como la por defecto, primero usaremos el comando **sudo ip route del default dev enp0s3** para dar de baja y luego el comando **sudo ip route add default via gateway dev enp0s8,** donde en vez de poner gateway pondremos la dirección del gateway que en nuestro caso sería: **sudo ip route add default via 192.168.1.1 dev enp0s8**

Para confirmar que todo lo que hicimos esta correcto le haremos un ping a google y si se realiza correctamente las descargas se realizan mediante la interfaz **enp0s8** que esta conectada a inter,el comando es **ping -c 4 google.com** y si salebien deberia aparecer:

```
vifrasoft@fedora-server:~$ ping -c 4 google.com
PING google.com (2800:3f0:4002:809::200e) 56 bytes de datos
64 bytes desde 2800:3f0:4002:809::200e: icmp_seq=1 ttl=117 tiempo=17.0 ms
64 bytes desde 2800:3f0:4002:809::200e: icmp_seq=2 ttl=117 tiempo=16.7 ms
64 bytes desde 2800:3f0:4002:809::200e: icmp_seq=3 ttl=117 tiempo=18.1 ms
64 bytes desde 2800:3f0:4002:809::200e: icmp_seq=4 ttl=117 tiempo=21.3 ms

--- google.com estadísticas ping ---
4 paquetes transmitidos, 4 recibidos, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 16.658/18.260/21.277/1.818 ms
```

Así confirmamos que cumplimos con todos los pasos

Conclusión

Vimos cómo configurar un segundo adaptador de red en la máquina virtual para separar la conexión SSH de la conectividad a Internet, asignando una IP estática a enp0s8 y ajustando la ruta por defecto para que todo el tráfico de red salga por esta interfaz. Esto permitió mantener la comunicación con el anfitrión sin interrupciones mientras se habilitaba la descarga de paquetes y servicios mediante Wi-Fi

Docker

De acuerdo con la página principal del sitio oficial de Docker, Docker es:

“una plataforma diseñada para ayudar a los desarrolladores a construir, compartir y ejecutar aplicaciones en contenedores, sin las tediosas configuraciones del entorno” Docker.

En otras palabras, Docker simplifica la creación, distribución y ejecución de aplicaciones encapsulándolas junto con todas sus dependencias en contenedores que se comportan de forma idéntica en cualquier entorno.

¿Qué es un contenedor?

Los contenedores son una forma de empaquetar una aplicación junto con todo lo que necesita para funcionar (código, librerías, dependencias, configuraciones), de manera que siempre se ejecute igual sin importar el sistema operativo o el servidor donde esté. A diferencia de una máquina virtual, un contenedor no necesita un sistema operativo completo, sino que comparte el del anfitrión, lo que los hace más ligeros, rápidos y eficientes.

¿Para qué nos sirve un contenedor en nuestro proyecto?

En términos generales, Docker serviría como una herramienta que facilita la implementación y administración de los servicios que conforman tu sistema. Al utilizar contenedores, podrías asegurar que cada componente funcione de manera aislada, portable y reproducible, sin depender de la configuración específica del sistema operativo. Esto no solo simplifica la instalación y el mantenimiento, sino que también garantiza que tu proyecto pueda ejecutarse de forma consistente en distintos entornos, desde tu máquina virtual hasta otros servidores o incluso en la nube.

¿Cómo se emplea Docker?

Docker se emplea mediante el uso de **imágenes** y **contenedores**. Las imágenes funcionan como plantillas que incluyen todo lo necesario para ejecutar una aplicación, desde el sistema base hasta sus librerías y configuraciones. A partir de estas imágenes se crean los contenedores, que son las instancias en ejecución donde realmente corre la aplicación. El proceso suele comenzar descargando imágenes ya preparadas desde repositorios como **Docker Hub**, para luego ponerlas en marcha como contenedores mediante simples comandos.

¿Qué es Docker Hub?

Docker Hub es una plataforma en línea oficial de Docker que funciona como un repositorio central de imágenes de contenedores. Allí los desarrolladores pueden encontrar, descargar y compartir imágenes ya preparadas de diferentes aplicaciones, sistemas y servicios, lo que facilita enormemente el uso de Docker sin necesidad de configurar todo desde cero. En este espacio se alojan imágenes oficiales mantenidas por Docker y por los propios proveedores de software, lo que garantiza su confiabilidad y actualización constante. Al mismo tiempo, los usuarios pueden subir sus propias imágenes para distribuirlas a otros, convirtiendo a Docker Hub en una herramienta colaborativa y fundamental para la portabilidad y reutilización de aplicaciones en entornos de desarrollo y producción.

Instalación de Docker

Antes de instalar Docker desde el repositorio oficial, es importante limpiar cualquier instalación previa que pueda estar en el sistema. Esto incluye tanto paquetes antiguos de

Docker como versiones no oficiales que hayas podido instalar desde repositorios externos. Si se dejan, pueden causar conflictos porque usan rutas y dependencias diferentes, lo que muchas veces provoca errores al iniciar el servicio de Docker o al ejecutar contenedores.

Para realizar esta limpieza, usá el siguiente comando:

**sudo dnf remove docker **
**docker-client **
**docker-client-latest **
**docker-common **
**docker-latest **
**docker-latest-logrotate **
**docker-logrotate **
docker-engine

El siguiente paso es preparar tu Fedora para que pueda descargar Docker desde su fuente oficial. Fedora no incluye los paquetes de Docker en sus repositorios por defecto, por eso es necesario agregar el repositorio oficial de Docker. De esta manera, te aseguras de instalar la versión más actualizada, estable y mantenida directamente por Docker.

Primero, instala las herramientas necesarias para manejar repositorios adicionales con el comando:
sudo dnf -y install dnf-plugins-core

Luego, agregá el repositorio de Docker con el comando:

sudo dnf config-manager --add-repo
https://download.docker.com/linux/fedora/docker-ce.repo

Una vez configurado el repositorio, ya podés instalar Docker y todas las herramientas necesarias para trabajar con contenedores con el siguiente comando:

sudo dnf install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

Por último iniciaremos el servicio y haremos un contenedor de prueba con los siguientes comandos:

sudo systemctl enable --now docker para habilitar el servicio de Docker

sudo systemctl enable --now docker para crear un contenedor de prueba y si queremos verlo tendremos que usar el comando: **sudo docker images.**
Opcionalmente si queremos usar los comandos de Docker sin tener que hacer uso de **sudo** y sin tener que cambiar a root usamos el comando **sudo usermod -aG docker $USER** y así no tendremos que usar **sudo** para cada vez que usemos un comando de Docker

Conclusión

En conclusión, Docker se presenta como una herramienta esencial para el desarrollo y despliegue de aplicaciones modernas, ya que permite empaquetar cada servicio en contenedores ligeros, portables y reproducibles, garantizando que se ejecuten de manera uniforme en cualquier entorno. Gracias a su integración con repositorios como Docker Hub, se simplifica enormemente la obtención y distribución de imágenes, reduciendo tiempos de configuración y asegurando la disponibilidad de versiones actualizadas y confiables. De esta manera, Docker no solo optimiza la administración de aplicaciones, sino que también fortalece la escalabilidad, portabilidad y consistencia de los proyectos.

Imagenes de Docker Hub necesarios para nuestro proyecto

Sabiendo que nuestro proyecto es crear una aplicación de seguimiento para una partida del juego de mesa **Draftosaurus** y que el propósito de nuestro servidor linux es alojar la base de datos de dicha aplicación, decidimos que los contenedores necesarios para administrar correctamente todo son los siguientes:

httpd (Apache HTTP Server):

La imagen de **httpd (Apache)** es necesaria en el proyecto porque actúa como servidor web, permitiendo atender las peticiones de los usuarios y entregar la aplicación de forma consistente. Al ejecutarse en un contenedor, asegura portabilidad, evita conflictos de configuración y facilita tanto el despliegue como el mantenimiento del sistema.

Link donde se puede encontrar la imagen de httpd: https://hub.docker.com/_/httpd
Comando para instalar la imagen: **docker pull httpd:trixie**

MySQL:

La imagen de **MySQL** nos sirve para desplegar la base de datos de nuestra aplicación dentro de un contenedor de forma rápida y confiable. Al usarla, evitamos instalar MySQL directamente en el servidor, ya que la imagen oficial incluye todo lo necesario para su funcionamiento, lo que nos asegura estabilidad, portabilidad y un entorno aislado. De esta manera, nuestro servidor puede alojar la base de datos de manera más sencilla, con la posibilidad de mantenerla, actualizarla o migrarla sin complicaciones.

Link donde se puede encontrar la imagen de httpd: https://hub.docker.com/_/mysql
Comando para instalar la imagen: **docker pull mysql:oraclelinux9**

Backups

Los Backups(la traducción al español es “respaldos”) son por así decirlo un tipo de protocolo que realizamos para el guardado de datos, básicamente son copias de los datos que se realizan como medida de prevención para cualquier tipo de pérdida de estos mismos datos o algún tipo de error no deseado. En nuestro caso que nuestro servidor está puesto en una máquina virtual sería demasiado útil hacer backups de nuestros datos más importantes, como por ejemplo la base de datos de la aplicación, los scripts, las claves ssh o la configuración ssh, entre otras cosas. ¿Hacia dónde lo haríamos? Muy bien si tenemos en cuenta que lo principal a respaldar son los archivos importantes de nuestro servidor tenemos las opciones de hacer un backup guardando los datos en una máquina distinta como por ejemplo nuestra pc anfitriona u otra máquina virtual, pero también existe la opción de hacer un backup en la nube haciendo uso de algún servicio que nos otorgue esta posibilidad de hacer un respaldo en la nube como por ejemplo **AWS**(Amazon Web Services) o **Azure** (Microsoft Azure), ninguno de los dos es gratis pero cuentan con pruebas gratis para aprender o para realizar proyectos pequeños.

Tipos de backups:

Full Backup:

Qué es: Una copia de seguridad de todos los datos seleccionados.
Ventajas:
Copia íntegra y fácil de restaurar (solo necesitas la última copia completa).
Desventajas:
Ocupa mucho espacio de almacenamiento.
Tarda más tiempo en realizarse.
Ejemplo: Si tienes 100 GB de datos, el backup completo copiará los 100 GB siempre, aunque solo hayas modificado 1 GB.

Backup incremental:

Qué es: Solo copia los cambios realizados desde el último backup (sea completo o incremental).

Ventajas:

Rápido de ejecutar y ocupa poco espacio.

Desventajas:

La restauración puede ser más lenta, porque necesitas el último backup completo + todos los incrementales hasta la fecha de recuperación.

Ejemplo:

Día 1: Full (100 GB).

Día 2: Incremental (2 GB de cambios desde el día 1).

Día 3: Incremental (1 GB de cambios desde el día 2). Para restaurar al día 3, necesitas los 3 backups (Full + Incremental Día 2 + Incremental Día 3).

Backup Diferencial

Qué es: Copia los **cambios desde el último backup completo**, sin importar cuántos diferenciales existan.

Ventajas:

Restauración más sencilla que la incremental (solo necesitas el último Full + el último Diferencial).

Desventajas:

Con el tiempo, el tamaño de cada diferencial crece, ya que siempre incluye todos los cambios desde el Full.

Ejemplo:

Día 1: Full (100 GB).

Día 2: Diferencial (2 GB de cambios desde el día 1).

Día 3: Diferencial (3 GB de cambios desde el día 1, incluyendo lo del día 2).

Para restaurar al día 3, solo necesitas el Full + Diferenciado Día 3.

Frecuencias

Al momento de organizar los backups se debe elegir con qué frecuencia se harán dichos backups, las frecuencias son estándar y se determinan en diaria, semanal, mensual y anual, el uso de cada uno siempre dependerá de los intereses del backups, como se clasifican por importancia dentro del proyecto que eso es más que nada a criterio propio y la decisión de las frecuencias cuenta como parte de una estrategia que se determina acorde a la elección de qué tipo de backup se hará, por ejemplo, diariamente un backup incremental y mensualmente un full backup, la idea es siempre combinar según los intereses propios

Tipos de almacenamiento

Si bien los backups son muy importantes para garantizar no tener pérdida de datos sabemos que los respaldos ocupan espacio en la memoria y sin importar la elección de dónde hacemos el backup no existe espacio ilimitado en nuestra computadora, por ende hay opciones para guardar alta cantidad de datos en forma de "servidor", existe el formato **NAS** (Network Attached Storage) y el formato **RAID** (Redundant Array of Independent Disks).

NAS:

Es un dispositivo de almacenamiento conectado a la red, diseñado específicamente para centralizar archivos y respaldos. Funciona como un pequeño servidor que puede ser accedido por varios usuarios y equipos al mismo tiempo a través de la red local o incluso de manera remota. Generalmente está compuesto por varios discos duros y un sistema operativo propio que facilita la gestión de carpetas compartidas, usuarios, permisos y tareas de respaldo automático. En la práctica, un NAS se utiliza como repositorio central para guardar documentos, copias de seguridad de computadoras y servidores, o incluso como nube privada para una organización. Su principal ventaja es que permite un acceso centralizado y seguro a la información, evitando depender del almacenamiento disperso en cada equipo.

RAID:

No es un dispositivo, sino una tecnología que organiza y combina varios discos duros para mejorar la tolerancia a fallos o el rendimiento del sistema. Al configurar un conjunto de discos en RAID, estos trabajan de manera coordinada, ya sea duplicando la información para mayor seguridad (RAID 1), dividiéndola para aumentar la velocidad (RAID 0) o combinando ambas estrategias mediante diferentes niveles de RAID como 5, 6 o 10. Aunque ofrece protección frente a la falla de uno o varios discos, el RAID no sustituye a un respaldo real, porque si los datos son borrados o dañados, el error se replica en todos los discos. Su función principal es garantizar la disponibilidad y continuidad de los datos, no reemplazar una copia de seguridad externa.

Manera de hacer Backup

Para hacer los backups existen dos formas de realizarlos, de forma manual y de forma automática, que esto también va a eleccion propia de cada usuario o equipo según le parezca más cómodo, a continuación explicamos cada una:

Manual

La forma más común de hacer un respaldo manual es de una máquina a otra, que si contamos con una conexión SSH entre esas máquinas el proceso es mucho más eficiente y además que es la forma por excelencia para realizar respaldos manualmente

Para hacer un respaldo de forma manual primero seleccionamos qué archivos queremos respaldar y usando el comando **tar** los uniremos, algo parecido a lo que es un comprimido, para otra computadora como una de sistema operativo Windows será un comprimido, pero en linux solo seran la unión de los archivos

Como cualquier comando tendrá opciones que determinaremos con un guión (**-**) seguido de una letra que determinará la opción. Dicho todo esto aquí un ejemplo util de como usar (y usaremos) **tar**:

tar -cvzf comprimido$(date -I).tar.gz archivo1 archivo2

Opciones -cvzf

c → crear un archivo nuevo.

v → verbose, muestra los archivos procesados en la terminal.

z → comprime usando gzip, por eso la salida es .tar.gz.

f → indica el nombre del archivo de salida que sigue a continuación.

comprimido$(date -I).tar.gz

Acá definimos el nombre del archivo resultante.

$(date -I) es una sustitución de comandos en bash → se ejecuta date -I, que devuelve la fecha actual en formato ISO (YYYY-MM-DD).

Ejemplo: si hoy es 2025-09-11, el nombre del archivo será: comprimido2025-09-11.tar.gz

archivo1 archivo2

Son los archivos que se van a unir o empaquetar, puede ser tanto archivo como un directorio

Por último podremos ver la unión hecha por el tar con el comando **ls** y nos aparecerá algo como: **comprimido2025-09-11.tar.gz**

Para poder enviar el "comprimido" hacia otra máquina usaremos el comando **scp**, comando el cual podremos usar siempre y cuando exista la conexión SSH entre ambas máquinas.

Sintaxis: scp |Nombreorigen| |destino|: |donde quiero dejarlo en el otro equipo|

En nuestro caso por si queremos hacerlo hacia nuestra pc anfitriona:

scp comprimido2025-09-11.tar.gz vifra@192.168.56.1:/Users/vifra/Backups/

Automática

Para automatizar los backups podremos usar **cron**, ¿Qué es cron? **cron** es un **servicio de programación automática de tareas,** instalado por defecto en linux. Permite que un comando o script se ejecute en momentos específicos: a cierta hora, en ciertos días, de manera repetida (cada minuto, cada semana, etc.). El servicio que lo gestiona se llama **crond**, que corre en segundo plano revisando cada minuto si hay tareas pendientes.

Para verificar que el servicio esté funcionando haremos uso del comando **sudo systemctl status crond,** resultado:

```
vifrasoft@fedora-server:~$ sudo systemctl status crond
● crond.service - Command Scheduler
     Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; preset: enabled)
    Drop-In: /usr/lib/systemd/system/service.d
             └─10-timeout-abort.conf
     Active: active (running) since Thu 2025-09-11 20:41:39 -03; 16min ago
 Invocation: dfcd62ae9a3f4c04bcee1be8063afb7c
   Main PID: 1067 (crond)
      Tasks: 1 (limit: 6110)
     Memory: 1M (peak: 1.2M)
        CPU: 51ms
     CGroup: /system.slice/crond.service
             └─1067 /usr/sbin/crond -n

set 11 20:41:39 fedora-server systemd[1]: Started crond.service - Command Scheduler.
set 11 20:41:39 fedora-server crond[1067]: (CRON) STARTUP (1.7.2)
set 11 20:41:39 fedora-server crond[1067]: (CRON) INFO (Syslog will be used instead of sendmail.)
set 11 20:41:39 fedora-server crond[1067]: (CRON) INFO (RANDOM_DELAY will be scaled with factor 50% if used.)
set 11 20:41:39 fedora-server crond[1067]: (CRON) INFO (running with inotify support)
```

Cron funciona como un ejecutante de tareas en segundo plano, tareas que nosotros diremos en qué momento se harán y la configuración es bastante sencilla, pero para hacerla tendremos que entrar a un archivo de dicha configuración, si somos root nos posicionamos en la raiz del servidor y escribimos **nano /etc/crontab**, cabe aclarar que si lo hacemos desde root se nos creará una columna para especificar con qué usuario se hará la tarea. Si somos un usuario con permisos de superusuario escribimos **crontab -e**, pero como ya lo haremos desde un usuario la columna para especificar con qué usuario se hará la tarea.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

En esta imagen vemos un ejemplo de la configuración de cron desde root, como se aprecia en la imagen hay un orden para dejar una tarea mediante un parámetro de fecha y hora, donde dependiendo del interés, si dejamos solo el * la opción queda nula, es decir, si solo queremos que se haga en cierta hora dejamos el numero de la hora en el segundo espacio y lo demás solo con asteriscos, igual explicamos el orden para dejar una tarea y es el siguiente:

MINUTO HORA DÍA_DEL_MES MES DÍA_DE_LA_SEMANA COMANDO

Sabiendo todo esto podemos usar el servicio de cron para unirlo con los comandos **tar** y **scp** para que haga los backups de forma automática con la frecuencia que nosotros querramos
Para crear una tarea en cron debemos escribirla en el mismo archivo de **crontab** sin borrar el contenido ya que es el que cuenta como guía y si queremos dejar una regla escrita pero sin que funcione tendremos que comentar la línea con un numeral (#) a continuación dos ejemplos:

```
# Backup semanal de archivo1 y archivo2, todos los domingos a las 3:00 AM
0 3 * * 0 tar -czf /backups/respaldo_semana_$(date +\%F).tar.gz /ruta/archivo1 /ruta/archivo2 &&

#backup de archivo3 y archivo4 los sábados a las 2:00 AM
# 0 2 * * 6 tar -czf /backups/respaldo_alt_$(date +\%F).tar.gz /ruta/archivo3 /ruta/archivo4 &&
```

```
 && scp /backups/respaldo_semana_$(date +\%F).tar.gz usuario@IP_HOST_WINDOWS:/c/Users/usuario/backups/


&& scp /backups/respaldo_alt_$(date +\%F).tar.gz usuario@IP_HOST_WINDOWS:/c/Users/usuario/backups/
```

En el ejemplo se muestra un ejemplo de un backup semanal activo y un un backup semanal comentado
Guardamos el archivo con ctrl+O

Simulación de backups en nuestro server

En nuestro servidor Linux con Fedora Server, necesitamos garantizar que los datos más importantes estén respaldados para evitar pérdidas ante fallos o errores humanos. En este caso, los archivos críticos son:

La base de datos MySQL, ubicada en /var/lib/mysql/.
Cuatro scripts de shell que forman parte de la lógica del sistema:
/home/proyecto/scripts/scriptPrincipal.sh
/home/proyecto/scripts/scriptUG.sh
/home/proyecto/scripts/scriptFirewalld.sh
/home/proyecto/scripts/scriptBackup.sh

El destino principal de los respaldos será un NAS montado en el directorio /mnt/nas/backups/. Este dispositivo de almacenamiento en red centraliza los archivos respaldados y permite mayor seguridad. Además, como copia secundaria, el backup será transferido automáticamente a la PC anfitriona con Windows, en la ruta C:/Users/usuario/backups/, utilizando SCP.

Tipos de Backup definidos

La estrategia combina dos enfoques:

Full Backup (Mensual):

Una vez al mes se hace una copia completa de la base de datos y los scripts.

Pros: Restauración sencilla, ya que es un paquete íntegro.

Contras: Requiere más tiempo y espacio de almacenamiento.

Backup Diferencial (Semanal):

Una vez por semana se hace un respaldo de todo lo que haya cambiado desde el último Full Backup.

Pros: Más rápido y ocupa menos espacio que un full.

Contras: A medida que pasa el mes, los diferenciales se hacen más grandes.

Herramientas usadas

cron: Para la automatización de tareas, ejecutando los respaldos de forma periódica.

tar: Para empaquetar y comprimir los directorios y scripts en un solo archivo .tar.gz.

scp: Para copiar automáticamente el backup desde el servidor Linux al host Windows.

NAS: Como almacenamiento principal de los respaldos en la red.

Configuración en cron

En el archivo de configuración de cron (crontab -e desde root) dejamos programadas las dos tareas:

Full Backup mensual (1° de cada mes a las 2:00 AM)

0 2 1 * * tar -czf /mnt/nas/backups/full_$(date +\%F).tar.gz /var/lib/mysql//home/proyecto/scripts/ && scp /mnt/nas/backups/full_$(date +\%F).tar.gz usuario@192.168.56.1:/c/Users/usuario/backups/

Backup diferencial semanal (todos los domingos a las 3:00 AM)
0 3 * * 0 tar -czf /mnt/nas/backups/diff_$(date +\%F).tar.gz --newer-mtime="$(date -d 'last month' +\%F)" /var/lib/mysql/ /home/proyecto/scripts/ && scp /mnt/nas/backups/diff_$(date +\%F).tar.gz usuario@192.168.56.1:/c/Users/usuario/backups/

Qué sucede en la práctica

El servicio crond se mantiene ejecutándose en segundo plano y revisa cada minuto si hay tareas programadas.
El primer día del mes a las 02:00, cron ejecuta el comando tar que empaqueta y comprime la base de datos y los scripts en un archivo llamado full_AAAA-MM-DD.tar.gz.

El archivo primero se guarda en el NAS.
Luego, con scp, se transfiere al host Windows en la carpeta de backups.
Cada domingo a las 03:00, cron ejecuta un backup diferencial.
Este backup solo incluye archivos que hayan cambiado desde la fecha del último full.
Se comprime en un archivo.
Se guarda en el NAS y luego se envía a Windows mediante scp.
En ambos casos, el backup queda duplicado: uno en el NAS y otro en la PC anfitriona.

Conclusión

En conclusión, los backups representan una práctica fundamental dentro de la administración de servidores, ya que permiten proteger los datos críticos frente a pérdidas accidentales, fallos técnicos o errores humanos. Vimos cómo, mediante el uso combinado de herramientas como

tar, scp y cron, junto con soluciones de almacenamiento como NAS y RAID, es posible implementar una estrategia de respaldo automatizada, eficiente y confiable. En nuestro caso particular, definimos una política que combina full backups mensuales y diferenciales semanales, garantizando así un equilibrio entre seguridad, tiempo de ejecución y espacio de almacenamiento. De esta manera, aseguramos que tanto la base de datos MySQL como los scripts esenciales del proyecto estén siempre resguardados en dos ubicaciones distintas, logrando redundancia y disponibilidad ante cualquier eventualidad.

Script de Firewalld

Debido a que muchas veces tenemos que realizar algo con o sobre el firewall creamos un script de shell para evitar el tedioso uso de comandos, a continuación toda su información:

Funcionalidades

- Activar o desactivar el firewall
- Activar o desactivar un servicio por nombre
- Activar o desactivar un servicio por puerto
- Agregar una rich rule
- Listar las rules

Código

```bash
if [[ "$EUID" -ne 0 ]]; then
  echo "Este script debe ejecutarse como root."
  exit 1
fi

pausar() {
  read -p "Presione Enter para continuar..."
}

activar_firewall() {
  systemctl start firewalld
  systemctl enable firewalld
  echo "Firewall activado y configurado para iniciar con el sistema."
}

desactivar_firewall() {
  systemctl stop firewalld
  systemctl disable firewalld
  echo "Firewall detenido y deshabilitado."
}
```

Lo primero que vemos es un mensaje de validación para que se recuerde que se debe acceder desde root y una función llamada **pausar()** para que despues de cada opcion hecha se presione enter para salir,
y vemos cómo las opciones son usando los comandos antes dados comunes del manejo del firewall

```bash
activar_servicio() {
  read -p "Ingrese el nombre del servicio : " servicio
  firewall-cmd --permanent --add-service=$servicio
  firewall-cmd --reload
  echo "Servicio '$servicio' habilitado."
}

desactivar_servicio() {
  read -p "Ingrese el nombre del servicio : " servicio
  firewall-cmd --permanent --remove-service=$servicio
  firewall-cmd --reload
  echo "Servicio '$servicio' deshabilitado."
}

activar_puerto() {
  read -p "Ingrese el puerto : " puerto
  firewall-cmd --permanent --add-port=$puerto
  firewall-cmd --reload
  echo "Puerto '$puerto' habilitado."
}

desactivar_puerto() {
  read -p "Ingrese el puerto : " puerto
  firewall-cmd --permanent --remove-port=$puerto
  firewall-cmd --reload
  echo "Puerto '$puerto' deshabilitado."
}
```

Aquí vemos la activación o desactivación de los servicios por nombre o por puerto

Lo siguiente que veremos es la configuración de una rich rule, na rich rule en es una regla avanzada de firewalld que te da un control muy preciso sobre el tráfico de red, permitiendo crear políticas complejas

```
agregar_rich_rule() {
  read -p "Ingrese la regla completa : " regla
  firewall-cmd --permanent --add-rich-rule="$regla"
  firewall-cmd --reload
  echo "Rich Rule agregada: $regla"
}

listar_reglas() {
  echo "=== Reglas actuales en el firewall ==="
  firewall-cmd --list-all
}
```

Por último el menú de usuario y case de las opciones

VifraSoft

```
mostrar_menu() {
  clear
  echo "========= MENÚ FIREWALLD ========="
  echo "1) Activar firewall"
  echo "2) Desactivar firewall"
  echo "3) Activar servicio por nombre"
  echo "4) Desactivar servicio por nombre"
  echo "5) Activar puerto"
  echo "6) Desactivar puerto"
  echo "7) Agregar Rich Rule"
  echo "8) Listar reglas actuales"
  echo "0) Salir"
  echo "================================="
}

opcion=-1
while [[ $opcion -ne 0 ]]; do
  mostrar_menu
  read -p "Seleccione una opción: " opcion
  case $opcion in
    1) activar_firewall; pausar ;;
    2) desactivar_firewall; pausar ;;
    3) activar_servicio; pausar ;;
    4) desactivar_servicio; pausar ;;
    5) activar_puerto; pausar ;;
    6) desactivar_puerto; pausar ;;
    7) agregar_rich_rule; pausar ;;
    8) listar_reglas; pausar ;;
    0) echo "Saliendo..." ;;
    *) echo "Opción no válida."; pausar ;;
  esac
done
```

I.S.B.O. **3MI**

Script de Backups

Para un manejo más óptimo y eficiente de los backups (respaldos) creamos el siguiente script para para facilitar las cosas, se basa en el manejo manual de backups

Funcionalidades

- Hacer full backup
- Hacer backup incremental
- Hacer backup diferencial

Código

```
if [[ "$EUID" -ne 0 ]]; then
    echo "Este script debe ejecutarse como root."
    exit 1
fi
ORIGEN="/etc"
DESTINO="/var/backups"
FECHA=$(date +"%Y-%m-%d_%H-%M-%S")

mkdir -p "$DESTINO"
```

Lo primero que veremos es esto, el aviso de que hay que ser root para hacer uso del script y como se marcan las rutas para los backups y si no existe el destino de los backup lo crea

Lo siguiente que veremos es las funcionalidades de los backups dados

```
backup_full() {
  ARCHIVO="$DESTINO/backup_full_$FECHA.tar.gz"
  tar -czvf "$ARCHIVO" "$ORIGEN"
  echo "Backup FULL creado en: $ARCHIVO"
}

backup_incremental() {
  ARCHIVO="$DESTINO/backup_incremental_$FECHA.tar.gz"

  tar --listed-incremental="$DESTINO/snapshot.snar" -czvf "$ARCHIVO" "$ORIGEN"
  echo "Backup INCREMENTAL creado en: $ARCHIVO"
}

backup_diferencial() {
  ARCHIVO="$DESTINO/backup_diferencial_$FECHA.tar.gz"

  tar --newer-mtime="$(date -d 'last sunday' +%Y-%m-%d)" -czvf "$ARCHIVO" "$ORIGEN"
  echo "Backup DIFERENCIAL creado en: $ARCHIVO"
}
```

Y por último el menú de usuario y el case de las opciones

```
mostrar_menu() {
  clear
  echo "========= MENÚ BACKUPS ========="
  echo "1) Crear backup FULL"
  echo "2) Crear backup INCREMENTAL"
  echo "3) Crear backup DIFERENCIAL"
  echo "0) Salir"
  echo "================================"
}

pausar() {
  read -p "Presione Enter para continuar..."
}

opcion=-1
while [[ $opcion -ne 0 ]]; do
  mostrar_menu
  read -p "Seleccione una opción: " opcion
  case $opcion in
    1) backup_full; pausar ;;
    2) backup_incremental; pausar ;;
    3) backup_diferencial; pausar ;;
    0) echo "Saliendo..." ;;
    *) echo "Opción no válida."; pausar ;;
  esac
done
```

Script principal

Para no tener que acceder manualmente a cada script que queramos usar, creamos uno principal donde estarán todos los script creados

Funcionalidades

- Entrar al script de usuarios y grupos
- Entrar al script de backups
- Entrar al script de firewalld

Código

Al ser solo una forma de enrutar los scripts que tenemos su código es muy corto en donde esta la validación del root, la función de pausa, el menu y el case, dicho menú es muy intuitivo y muy fácil de usar

```
if [[ "$EUID" -ne 0 ]]; then
  echo "Este script debe ejecutarse como root."
  exit 1
fi

pausar() {
  read -p "Presione Enter para continuar..."
}

mostrar_menu() {
  clear
  echo "========= MENÚ PRINCIPAL ========="
  echo "1) Gestión de Usuarios y Grupos"
  echo "2) Backups Manuales"
  echo "3) Configuración de Firewalld"
  echo "0) Salir"
  echo "================================="
}

opcion=-1
while [[ $opcion -ne 0 ]]; do
  mostrar_menu
  read -p "Seleccione una opción: " opcion
  case $opcion in
    1) bash /home/vifrasoft/scripts/UG.sh; pausar ;;
    2) bash /home/vifrasoft/scripts/backups.sh; pausar ;;
    3) bash /home/vifrasoft/scripts/Firewalld.sh; pausar ;;
    0) echo "Saliendo..." ;;
    *) echo "Opción no válida."; pausar ;;
  esac
done
```


```
vifrasoft@fedora-server:~$ sudo firewall-cmd --get-active-zones
FedoraServer (default)
  interfaces: enp0s3 enp0s8
docker
  interfaces: docker0
```

docker cp index.html miServidor:/var/www/html/

docker cp test.html VifraSoft@10.36.145.200:/var/www/html/

Entorno de desarrollo en Linux para el proyecto

Debido a ciertas circunstancias como complicaciones externas al proyecto y causas de mala gestión del equipo se retrasaron los tiempos y mediante acuerdo con el coordinador de informática de ISBO y el profesor de la materia,decidimos que el uso de docker para el funcionamiento de la aplicación en el servidor linux no era la mejor opción por temas de tiempo y pruebas, llegamos a la solución de crear un entorno dentro de nuestro Fedora Server 42 para que pueda correr nuestro proyecto de Laravel en el mismo.

Esquema de configuración del entorno:

- Instalar LAMP con las dependencias PHP
- Instalar y configurar MySQL o MariaDB
- Cargar la base de datos
- Instalar Composer, node.js con npm y Vite
- Instalar Laravel
- Cargar el proyecto

Después de crear el entorno haremos una prueba con una versión antigua del proyecto para ver que funcione todo para la entrega y para la configuración de LAMP usaremos la documentación que nos dio el coordinador Luis Fagundez, link: Configuración de un servidor LAMP

Paso a paso de la configuración:

Instalación de Lamp

- ¿Qué es LAMP? LAMP es un acrónimo que describe una pila de software de código abierto muy popular, utilizada para construir y alojar aplicaciones y sitios web dinámicos. El término representa la combinación de cuatro tecnologías que operan conjuntamente: **Linux**, que es el sistema operativo que sirve como base para todo el entorno; **Apache**, el servidor web HTTP encargado de recibir las solicitudes de los visitantes y entregarles el contenido; **MySQL** (o su derivado **M**ariaDB), que es el sistema gestor de bases de datos relacional donde se almacena y organiza toda la información persistente; y **PHP** (aunque a veces también **P**erl o **P**ython), que es el lenguaje de programación del lado del servidor. Este lenguaje actúa como el "motor" de la aplicación, procesando la lógica, interactuando con la base de datos y generando la página antes de que Apache la envíe al usuario.

El primer paso es actualizar todos los paquetes de Fedora con el comando: **yum update** y lo siguiente que hicimos fue instalar las herramientas básicas para desarrollo con el comando: **yum install dnf gcc binutils automake perl python**
el siguiente paso a realizar es la instalación de apache con **dnf install httpd**

Con el comando **systemctl** haremos que inicie el servidor y lo configure para iniciar el arranque del sistema primero **systemctl start httpd** y segundo **systemctl enable httpd**

Ahora instalamos el motor de base de datos MariaDB-Server con el comando **dnf install mariadb mariadb-server** y volveremos a usar el comando **systemctl** para proceder de la misma forma que apache **systemctl start mariadb.service** y **systemctl enable mariadb.service**

Lo siguiente que hicimos fue realizar las configuraciones del usuario root de MariaDB con el comando **mysql_secure_installation**

Y el último paso que hicimos es instalar PHP con el comando **dnf install php php-mysqlnd php-pdo php-gd php-mbstring**
Y hasta aquí usamos el material del coordinador Luis

Instalación de Composer

Composer es nuestro gestor de dependencias en nuestro proyecto de laravel

instalamos Composer en el server utilizando la documentación oficial de composer para su instalación link: [Instalacion de composer]

Según la documentación tendremos que copiar y pegar los siguientes comandos:
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') === 'c8b085408188070d5f52bcfe4ecfbee5f727afa458b2573b8eaaf77b3419b0bf2768dc67c86944da1544f06fa544fd47') { echo 'Installer verified'.PHP_EOL; } else { echo 'Installer corrupt'.PHP_EOL; unlink('composer-setup.php'); exit(1); }"
php composer-setup.php
php -r "unlink('composer-setup.php');"

segun la documentacion oficial las líneas harán lo siguiente en orden:

- Descargar el instalador al directorio actual
- Verificar el SHA-384 del instalador
- Ejecutar el instalador
- Eliminar el instalador

Y en la misma documentación nos recomienda renombrar composer.phar por composer con el comando **mv composer.phar /usr/local/bin/composer** por motivos de comodidad.

Instalación de Node.js y npm

Esto lo haremos porque laravel los necesita para procesar los archivos frontend y su instalación es muy simple, usaremos el comando **dnf install -y nodejs npm**

*Aclaramos que todos los comandos los hicimos en usuario root

Prueba de que todo funciona con una versión antigua del proyecto

Ahora para probar que el entorno funciona para correr la aplicación usaremos una versión antigua del proyecto a modo de test y estarnos seguros de que no habrá ningún error al correr el producto final

Para hacer la prueba iremos al directorio **var/www** que se creó cuando instalamos apache y dentro de la misma crearemos un directorio llamado test, para hacer las pruebas, como hicimos la instalación con usuario root nuestro usuario no tendrá los permisos para manejar dentro de ese directorio, la solución es o usar sudo para los comandos o integrar al usuario dentro del grupo de apache con **sudo usermod -a -G apache vifrasoft** (vifrasoft es nuestro usuario) y con eso ya podremos manejarnos dentro del directorio.
El siguiente paso que hicimos fue crear el directorio var/www/test para trabajar en él y dentro de test creamos un directorio con el nombre original del proyecto para transferir los archivos importantes

Con el comando **scp** pasamos los archivos de mayor importancia (cuando pasemos el proyecto completado haremos un scp completo): **scp -r app bootstrap config database public resources routes storage artisan composer.json package.json .env vifrasoft@192.168.56.2:/var/www/html/test/DraftoStation**

Lo siguiente fue pasar la base de datos, que la exportamos desde phpMyAdmin hacia la pc anfitriona y usando **scp** la moveremos al directorio donde tenemos el proyecto en linux así: **scp -r draftostation.sql vifrasoft@192.168.56.2:/var/www/test/DraftoStation**

Después de transferir el archivo debemos editarlo para que cree y use la base de datos ya que XAMPP lo hace automáticamente, lo haremos con **sudo vim draftostation.sql** y agregamos al inicio 2 lineas mas: **CREATE DATABASE draftostation;** y abajo **USE draftostation;** y para que ejecute todas las sentencias dentro del archivo usaremos el comando **sudo mysql -u root -p < draftostation.sql** nos pedirá la contraseña de root que la editamos anteriormente, posteriormente tendremos que hacer las migraciones de la base de datos con las del proyecto en laravel y para esto mismo usaremos el comando **php artisan migrate:fresh,** este comando

borrará y creará de nuevo las tablas que están dentro del proyecto en laravel y con todo esto hecho hasta aqui estaria listo la base de datos

El siguiente paso es editar el archivo .env para que utilice la base de datos alojada en el proyecto y es tan fácil como hacer **sudo vim .env** y la configuración que usamos fue:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=draftostation
DB_USERNAME=root
DB_PASSWORD=root
```

después de editarlo apretamos esc y escribimos :wq para guardar y salir

Preparado todo esto debemos instalar todas las dependencias necesarias para poder correr nuestro proyecto y lo haremos con el comando **composer install** posicionados en el directorio del proyecto
Después de instalar las dependencias necesitamos añadir el puerto al firewall con el comando **sudo firewall-cmd --add-port=8000/tcp --permanent** y luego **sudo firewall-cmd --reload**

Y para correr el proyecto y ver si funciona, posicionados en el directorio del proyecto usaremos el comando **php artisan serve --host=0.0.0.0 --port=8000** nos debería mostrar el mensaje de Server running y después en el navegador debemos poner **http://192.168.56.2:8000** y nos debería mostrar la aplicación web funcionando

Para la entrega final seguiremos todos los mismos pasos solamente que cambiaremos de ruta y será en /var/www/proyecto/DraftoStation, además que eliminaremos el test para evitar cualquier tipo de conflicto que pueda haber entre los archivos, por ejemplo la base de datos