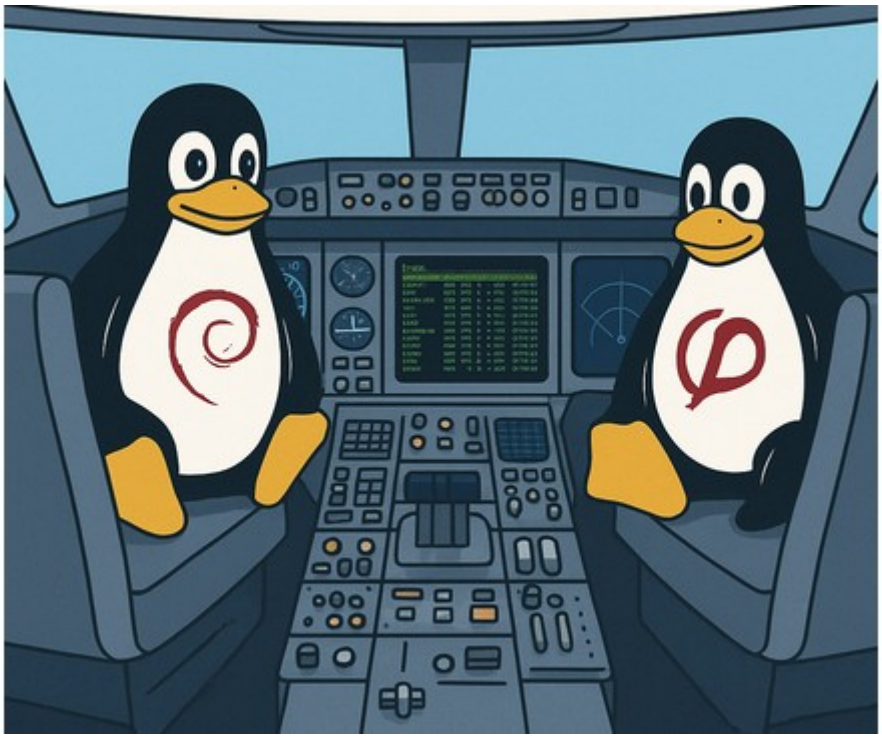
Cockpit como interfaz de gestión para entornos KVM/libvirt



Jose Antonio Canalo Gonzalez 2ASIR

IES GONZALO NAZARENO

Sumario

1. Objetivos del proyecto

El objetivo principal de este proyecto es demostrar cómo la herramienta web **Cockpit** puede facilitar la gestión de entornos de virtualización con **KVM** y **libvirt** en un sistema **Debian GNU/Linux**.
A continuación, se detallan los objetivos específicos:

- Instalar y configurar correctamente un entorno funcional con KVM y libvirt en Debian.
- Integrar la herramienta Cockpit y sus módulos de gestión de virtualización.
- Administrar máquinas virtuales desde la interfaz gráfica de Cockpit.
- Documentar detalladamente el proceso de instalación, configuración y administración.
- Comparar el uso de Cockpit con la administración tradicional mediante consola.
- Evaluar la utilidad real de Cockpit en un entorno técnico, especialmente para usuarios con menos experiencia en terminal.
- Demostrar el funcionamiento de la gestión de redes mediante la interfaz de Cockpit.
- Gestionar contenedores utilizando Podman integrado en Cockpit como ejemplo de ampliación práctica.
- Detectar posibles limitaciones o aspectos mejorables de la herramienta.

-

2. Escenario necesario para la realización del proyecto

2.1 Hardware del sistema

- **Procesador**: Intel Core i7-1165G7 (11ª generación), compatible con virtualización por hardware (VT-x).
- **Memoria RAM**: 16 GB (aproximadamente 10 GB disponibles durante las pruebas).
- **Almacenamiento**:
  - Disco NVMe principal de 477 GB, particionado con LVM (`/`, `/boot`, `/home`, `swap`).
  - Segundo disco NVMe adicional de 27 GB para pruebas adicionales.
- **Redes**:
  - Interfaces activas por cable (`enx000ec6512870`) y Wi-Fi (`wlo1`), ambas con direcciones dinámicas IPv4 (DHCP).

2.2 Sistema operativo

El sistema base utilizado ha sido **Debian GNU/Linux 12**, actualizado a su versión estable más reciente mediante los repositorios oficiales.
Esta distribución fue elegida por su **estabilidad, seguridad y compatibilidad** con tecnologías de virtualización como **KVM, libvirt y Cockpit**.
Antes de comenzar la instalación del entorno de pruebas, se aplicaron todas las actualizaciones mediante `apt`.

2.3 Tecnologías utilizadas

A lo largo del desarrollo se han empleado diversas tecnologías especializadas en virtualización y administración de sistemas:

2.3.1 KVM (Kernel-based Virtual Machine)

Tecnología de virtualización completa integrada en el kernel de Linux.
Permite ejecutar múltiples sistemas operativos como máquinas virtuales de forma eficiente y segura, aprovechando las capacidades del procesador (VT-x).

2.3.2 libvirt

Capa de abstracción para la gestión de tecnologías como KVM, Xen o QEMU.
Ofrece herramientas y una API para controlar máquinas virtuales de forma uniforme, facilitando su administración desde consola o interfaz gráfica.

2.3.3 QEMU (Quick Emulator)

Emulador de hardware que, junto con KVM, permite virtualización acelerada por hardware.
Emula dispositivos (discos, interfaces de red, pantallas virtuales) para conseguir un entorno de virtualización completo.

2.3.4 Cockpit

Herramienta web de administración de sistemas Linux accesible desde el navegador.
Permite controlar el sistema, usuarios, servicios, actualizaciones y redes, además de gestionar entornos virtualizados mediante módulos específicos.

2.3.5 Módulos adicionales de Cockpit

- **cockpit-machines**: gestión gráfica de máquinas virtuales (crear, arrancar, detener, eliminar, acceder a consola vía SPICE o VNC).
- **cockpit-networkmanager**: configuración de interfaces de red (IPs, bridges, VLANs) de forma visual.
- **cockpit-podman**: gestión de contenedores con Podman (crear, ejecutar, detener contenedores, revisar registros y acceder a terminales).

3. Fundamentos teóricos y conceptos

Para comprender el alcance y funcionamiento del proyecto, es necesario conocer algunos conceptos clave relacionados con la virtualización, la administración de sistemas y el uso de herramientas modernas como **Cockpit** y **Podman**.
A continuación, se explican los fundamentos teóricos más relevantes.

3.1 Virtualización

La **virtualización** es una tecnología que permite ejecutar múltiples sistemas operativos de forma simultánea sobre un mismo equipo físico.
Esto se logra mediante el uso de un **hipervisor**, un software capaz de crear, gestionar y ejecutar máquinas virtuales (VMs), cada una con sus propios recursos (CPU, RAM, disco, red, etc.).

Gracias a la virtualización, es posible simular entornos completos de forma aislada, lo cual resulta especialmente útil en contextos de **pruebas, formación, desarrollo y administración de servicios**.

3.2 KVM (Kernel-based Virtual Machine)

KVM es un módulo del núcleo de Linux que convierte el sistema operativo en un **hipervisor de tipo 1**, permitiendo ejecutar máquinas virtuales con acceso directo al hardware.
Al estar integrado en el kernel, proporciona un alto rendimiento y se utiliza ampliamente en entornos de servidores.

Para su correcto funcionamiento, el procesador debe ser compatible con tecnologías de virtualización como **Intel VT-x** o **AMD-V**.

3.3 libvirt

libvirt es una herramienta que proporciona una capa de gestión sobre motores de virtualización como KVM o QEMU.
A través de su **API o comandos** (`virsh`), permite crear, configurar y controlar máquinas virtuales.

Su principal ventaja es ofrecer una interfaz unificada para administrar distintos sistemas de virtualización desde una misma herramienta.

3.4 QEMU

QEMU es un emulador de hardware que permite virtualizar procesadores, dispositivos, discos duros, etc.
Combinado con KVM, permite ejecutar máquinas virtuales con **aceleración por hardware**.

En este contexto, QEMU actúa como el emulador de componentes virtuales de la máquina (tarjetas de red, pantallas, controladoras), mientras que KVM se encarga del acceso al procesador físico.

3.5 Cockpit

Cockpit es una interfaz web para la administración de sistemas GNU/Linux.
Permite controlar el sistema desde un navegador, evitando el uso directo de la terminal.

Entre sus funciones se incluyen: monitorización de recursos, gestión de servicios, usuarios, redes, actualizaciones y, mediante módulos adicionales, también máquinas virtuales y contenedores.

Su objetivo es **simplificar la gestión de sistemas** y facilitar el acceso a administradores sin experiencia en línea de comandos.

3.6 Redes virtuales (bridges)

En entornos virtualizados, es habitual crear **puentes de red** (**bridges**) que permiten a las máquinas virtuales comunicarse entre sí y con la red externa.

Estos bridges funcionan como un **switch virtual**.
En este proyecto se han utilizado bridges configurados desde Cockpit, lo que ha permitido conectar las VMs a la red local como si fueran dispositivos físicos independientes.

3.7 Contenedores y Podman

Los **contenedores** permiten ejecutar aplicaciones de forma aislada, ligera y rápida, sin necesidad de crear una máquina virtual completa.

A diferencia de las VMs, los contenedores comparten el núcleo del sistema anfitrión, lo que **reduce el consumo de recursos**.

En este proyecto se ha utilizado **Podman**, una herramienta compatible con Docker que no requiere daemon en segundo plano (**daemonless**) y que está bien integrada en entornos Linux modernos.

Cockpit permite **gestionar contenedores Podman desde la interfaz gráfica**, facilitando su uso sin necesidad de recurrir a comandos complejos.

4. Puesta en marcha del entorno de virtualización

4.1 Instalación del sistema base

4.1.1 Instalación de Cockpit

Una vez actualizado el sistema **Debian GNU/Linux 12** a su última versión estable mediante los repositorios oficiales, se procedió a instalar **Cockpit**, que será la interfaz principal de administración utilizada a lo largo del proyecto.

A continuación, se describen los pasos seguidos:

Actualización del sistema:
Se actualizó la lista de paquetes y se aplicaron todas las actualizaciones disponibles:

sudo apt update

sudo apt upgrade -y

Instalación de Cockpit:
Se instaló el paquete principal desde los repositorios oficiales:

sudo apt install cockpit -y

Activación del servicio:
Se habilitó y se inició el servicio para que Cockpit esté disponible desde el arranque del sistema

sudo systemctl enable --now cockpit.socket

Una vez completados estos pasos, Cockpit quedó accesible a través de un navegador web en el puerto 9090, usando la dirección IP del sistema anfitrión.
Desde ese momento fue posible acceder a su interfaz gráfica y comenzar con la gestión del entorno de virtualización.

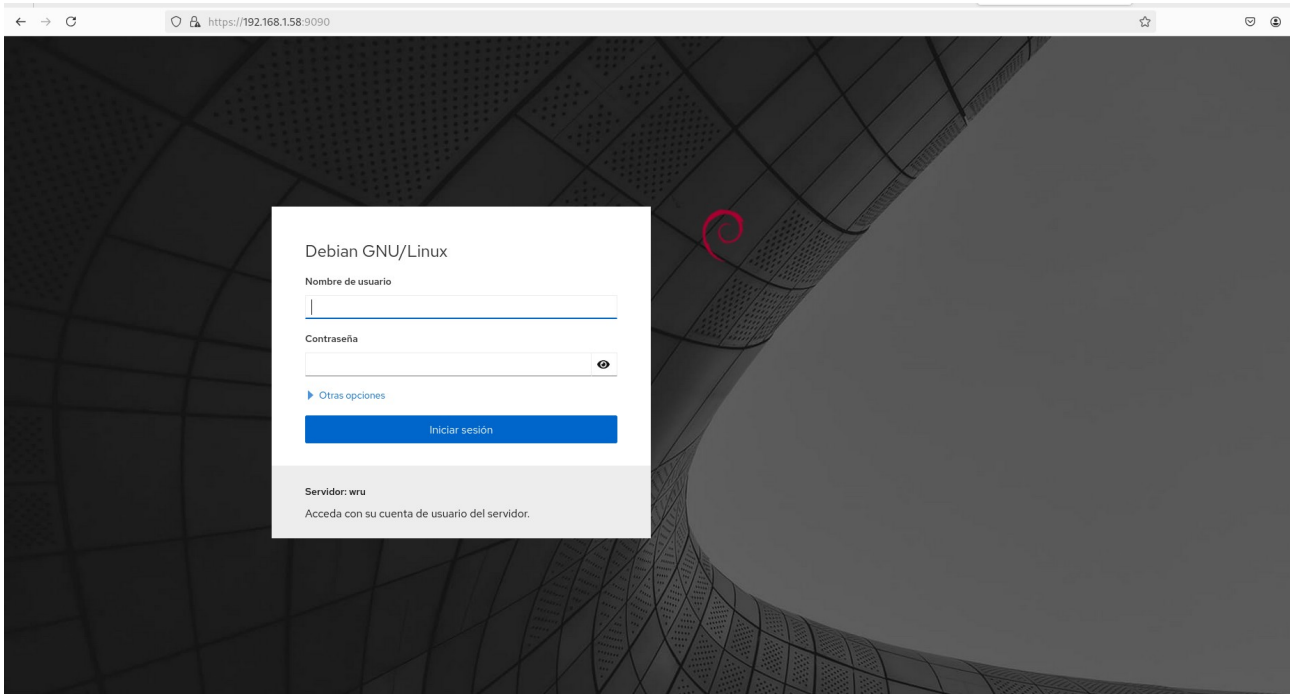


Figura 1: Pantalla de inicio de sesión de Cockpit accesible desde el navegador en el puerto 9090 del sistema anfitrión.

4.2 Exploración de la interfaz de Cockpit

Una vez instalado Cockpit y accedido desde el navegador mediante el puerto `9090`, se presenta una interfaz gráfica clara y moderna que permite comenzar con la administración del sistema de forma visual.
A continuación, se describe la pantalla principal que se muestra tras iniciar sesión correctamente.

4.2.1 Visión general del sistema

La primera vista, denominada **Visión global**, ofrece un resumen del estado general del sistema anfitrión. Esta pantalla centralizada permite consultar e intervenir en distintas áreas clave de administración.

Aviso de acceso limitado

En la parte superior se muestra un mensaje en color amarillo que indica que la consola web está en **modo de acceso limitado**.
Para disponer de todas las funcionalidades, es necesario activar el modo administrativo haciendo clic en el botón azul **“Turn on administrative access”** e introduciendo la contraseña del usuario. Esta acción es necesaria para gestionar servicios del sistema, redes, máquinas virtuales y contenedores.

Información del sistema y estado general

En el panel principal se presentan distintas secciones:

- **Salud del sistema**: se notifica si existen actualizaciones o problemas. En el ejemplo, se indica que hay **actualizaciones disponibles que corrigen errores**.
- **Uso de recursos**: muestra el uso actual de **CPU** (10% de 8 núcleos) y **RAM** (3,8 GiB de 15,3 GiB disponibles).
- **Información del sistema**: incluye detalles del equipo anfitrión como el modelo del hardware (ASUSTeK COMPUTER INC. VivoBook X515EA), el identificador de máquina y el tiempo de actividad desde el último arranque.
- **Configuración general**: presenta el nombre del host (`wru`), la **hora del sistema**, el estado de unión a dominio (actualmente no unido), y las huellas digitales de las **claves SSH** del sistema, relevantes en entornos con acceso remoto.

Menú de navegación lateral

A la izquierda se encuentra el menú de navegación, con acceso a diferentes apartados:

- **Registros** (logs del sistema)
- **Almacenamiento**
- **Redes**

- **Cuentas de usuario**
- **Servicios activos**
- **Herramientas**, como:
    - **Actualizaciones de software**
    - **Aplicaciones**
    - **Terminal** del sistema

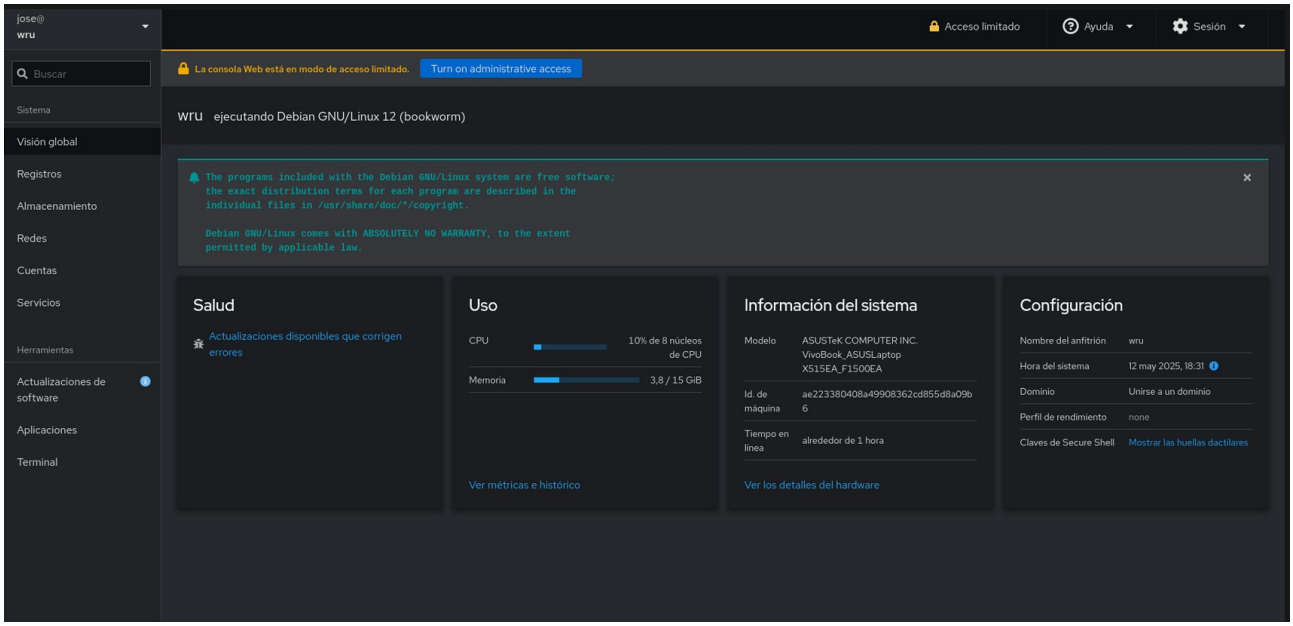


Figura 2: Vista principal de Cockpit tras iniciar sesión. Se muestra la información del sistema, el estado de los recursos y el acceso limitado.

4.2.2 Acceso administrativo en Cockpit y control total del sistema

Al iniciar sesión en Cockpit, la interfaz se carga por defecto en **modo de acceso limitado**, incluso si se ha accedido con un usuario con permisos administrativos.
Este comportamiento está diseñado para proteger el sistema frente a accesos no autorizados, sesiones compartidas o errores involuntarios, restringiendo inicialmente las funciones críticas del sistema.

Funciones limitadas por defecto

Mientras el acceso administrativo no esté habilitado, no es posible realizar acciones sensibles como:

- Crear o gestionar máquinas virtuales.
- Configurar interfaces de red o puentes virtuales.
- Administrar contenedores con Podman.
- Gestionar servicios del sistema mediante systemd.
- Aplicar actualizaciones de software.
- Acceder al terminal del sistema desde el navegador.

Activar el acceso completo

Para desbloquear estas funciones, es necesario activar el **modo administrativo** directamente desde la interfaz web. El proceso es el siguiente:

1. En la parte superior aparece un mensaje en color amarillo:

   La consola Web está en modo de acceso limitado

2. Se hace clic en el botón azul contiguo:

   Turn on administrative access

3. El sistema solicita la **contraseña del usuario** con el que se ha iniciado sesión.
4. Tras introducirla correctamente, la sesión se recarga con **permisos administrativos completos**.

Figura 3: Aviso de acceso limitado en Cockpit y opción para habilitar permisos administrativos.

Requisitos para que funcione correctamente

- El usuario debe pertenecer al grupo `sudo` en el sistema para poder escalar privilegios desde Cockpit.

¿Qué se desbloquea con el acceso administrativo?

Una vez activado el acceso administrativo, Cockpit habilita su funcionalidad completa. Esto incluye:

- Creación, configuración, arranque, eliminación y acceso por consola a máquinas virtuales.
- Administración completa de interfaces de red: asignación de direcciones IP, creación de bridges, configuración de VLANs, etc.
- Gestión avanzada de contenedores con **Podman**: crear, detener, revisar logs y acceder a terminales.
- Supervisión y control de **servicios del sistema** mediante `systemd` (iniciar, detener, reiniciar servicios).
- Visualización de **logs del sistema en tiempo real**.
- Acceso a un **terminal integrado** para ejecutar comandos directamente desde el navegador.
- Aplicación de **actualizaciones del sistema**.
- Administración de **usuarios y grupos** del sistema.

Gracias a este modo, **Cockpit se transforma en una herramienta completa de administración de sistemas GNU/Linux**, permitiendo centralizar tareas que normalmente requerirían conocimientos avanzados en terminal o acceso físico al servidor.

4.3 Creación de una máquina virtual desde Cockpit

Una vez habilitado el acceso administrativo y con el módulo `cockpit-machines` correctamente instalado, es posible crear una **máquina virtual (MV)** directamente desde la interfaz gráfica de Cockpit.
Todo el proceso se realiza sin necesidad de ejecutar comandos en consola, lo que facilita la gestión, especialmente en entornos educativos o de administración básica.

4.3.1 Acceso al módulo de máquinas virtuales

Desde el menú lateral izquierdo de Cockpit, se accede a la sección **“Máquinas virtuales”**. Esta pantalla muestra el listado de MVs definidas en el sistema (si las hubiera), junto con las acciones disponibles para cada una: crear, arrancar, detener, reiniciar o eliminar.

En el caso de un sistema recién configurado, la lista aparecerá vacía y se ofrecerán dos opciones:

- **Importar una MV** existente.
- **Crear una MV** nueva.

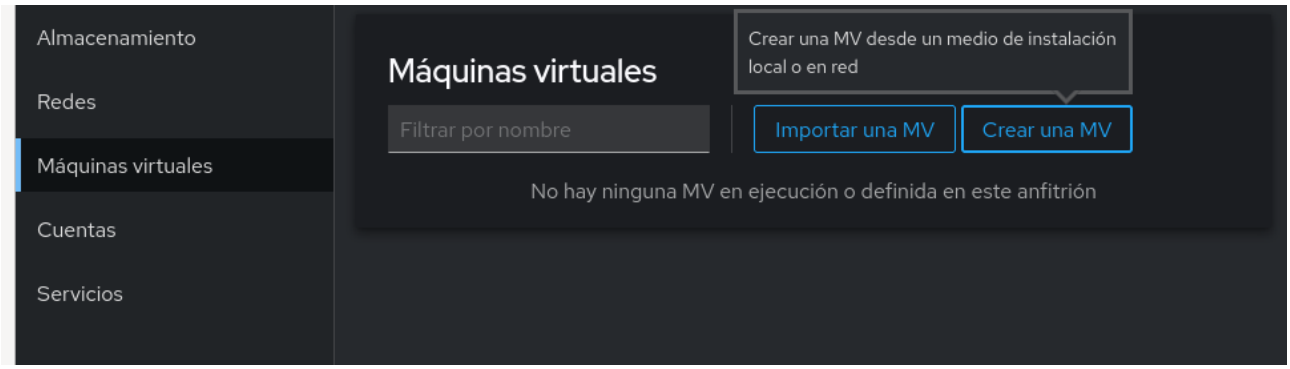


Figura 4: Pantalla inicial del módulo “Máquinas virtuales” en Cockpit. Se ofrecen opciones para crear o importar MVs.

4.3.2 Iniciar el asistente de creación

Para crear una nueva máquina virtual, se debe hacer clic en el botón **“Crear una MV”**.
Esto inicia un asistente que guiará paso a paso el proceso, solicitando información sobre:

- El nombre de la máquina.
- La fuente de instalación (imagen ISO local, red, etc.).
- La asignación de recursos (CPU, RAM, almacenamiento).
- Las opciones de red y configuración adicional.

Este asistente facilita enormemente la configuración inicial, eliminando la necesidad de editar archivos XML o utilizar comandos complejos en consola.4.3.2 Iniciar el asistente de creación

Al pulsar en el botón **“Crear una MV”**, se abre un asistente gráfico que guía al usuario en el proceso de configuración inicial de la máquina virtual. A continuación se describen los campos que deben completarse:

1. **Nombre**: se asigna un nombre identificativo para la máquina virtual, por ejemplo: `prueba1`.
2. **Tipo de instalación**: se selecciona la opción *“Medio de instalación local (imagen ISO o lista de instalación)”*, que permite instalar el sistema operativo a partir de una imagen ISO descargada previamente.
3. **Origen de instalación**: se indica la ruta en la que se encuentra la ISO. En este caso:

   /var/lib/libvirt/images/debian-12.10.0-amd64-netinst.iso

- **Sistema operativo**: Cockpit suele detectar automáticamente el sistema a instalar a partir del contenido de la ISO. En el ejemplo se propone *Debian testing*.
- **Almacenamiento**: se selecciona *“Crear un volumen”*, lo que genera un disco virtual nuevo para la máquina. Se establece un **límite de almacenamiento de 10 GiB**, que será reservado del espacio disponible en el anfitrión.
- **Memoria RAM**: se asignan **2 GiB de RAM**, teniendo en cuenta la cantidad total disponible (15,3 GiB en este caso).

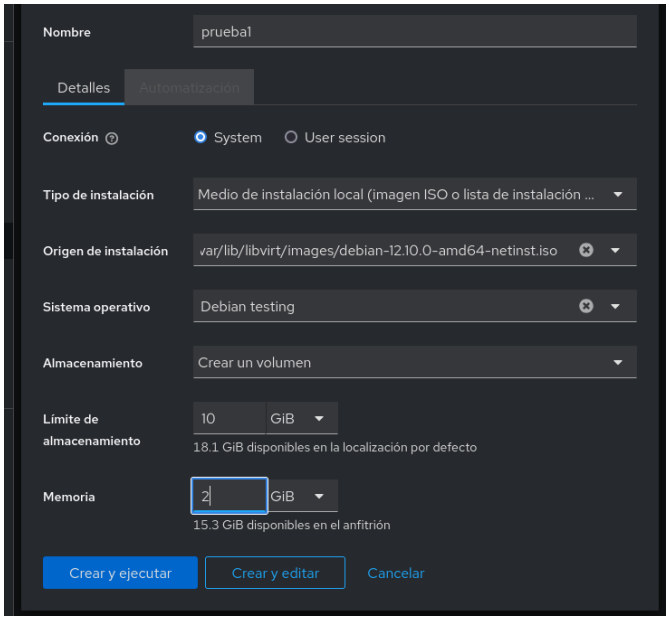


Figura 5: Asistente de creación de máquina virtual en Cockpit. Se configuran nombre, instalación desde ISO, almacenamiento y memoria.

Una vez completados todos los campos, es posible finalizar el proceso haciendo clic en el botón:

- **“Crear y ejecutar”**: la máquina se crea y se inicia de forma automática, comenzando la instalación del sistema operativo desde la imagen ISO indicada.4.3.3 Acceso a la consola de la máquina virtual

4.3.3 Acceso a la consola de la máquina virtual

Una vez creada la máquina virtual, **Cockpit** permite acceder directamente a su consola desde el navegador utilizando un visor integrado **VNC**.
Esto elimina la necesidad de emplear clientes externos o conexiones SSH, facilitando la interacción con la VM de forma inmediata y centralizada.

En la figura se observa el proceso de instalación del sistema operativo **Debian 12** en curso. La interfaz gráfica de instalación se carga desde la imagen ISO previamente definida y puede manejarse mediante teclado y ratón como si se tratara de una máquina física.

Información proporcionada por Cockpit en esta fase:

- **Estado de la MV**: indica que la máquina está en ejecución (*Running*).
- **Recursos asignados**: se muestran los **2 vCPUs** y los **2 GiB de RAM** configurados.
- **Orden de arranque**: definido como `disk,disk`, lo que garantiza que la VM inicie desde el disco tras la instalación.
- **Consola VNC**: activa y funcional, accesible desde la interfaz web para interactuar en tiempo real con el entorno instalado.
- **Consumo de recursos**: se visualiza el uso en tiempo real de **CPU** y **memoria** desde la sección "Uso".
- **Gestión de almacenamiento**: se muestran los discos virtuales asociados, con posibilidad de añadir nuevas unidades si es necesario.

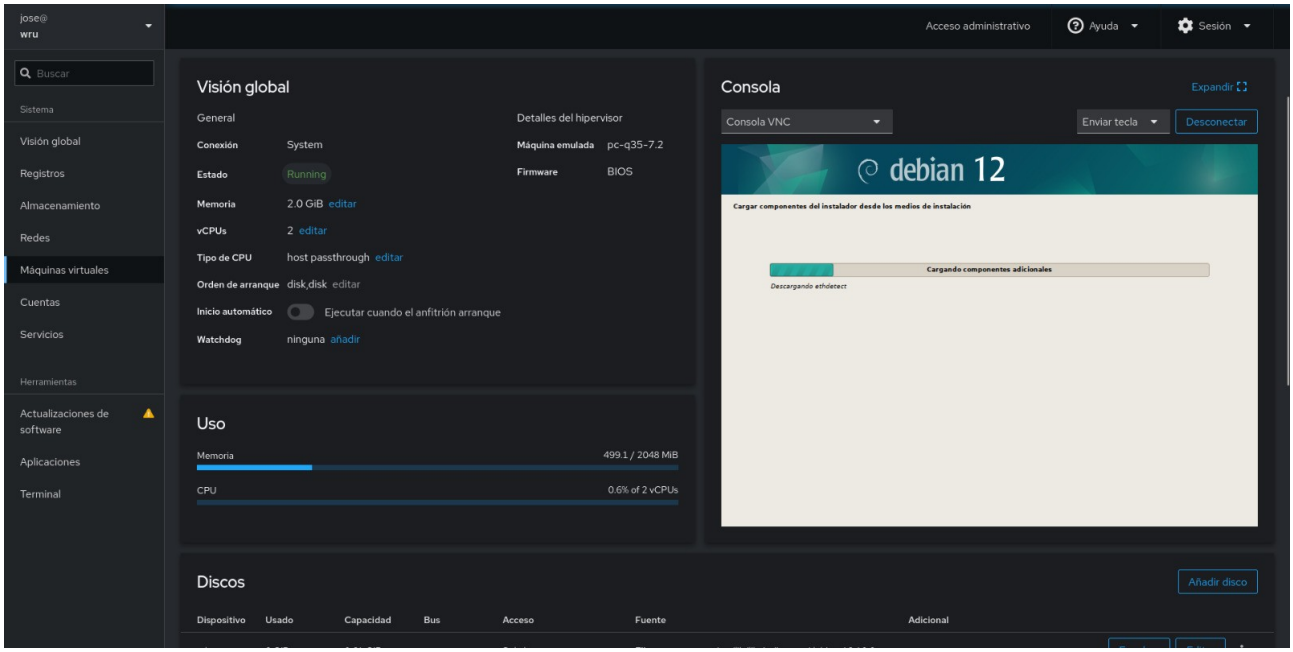


Figura 6: Consola de la máquina virtual en Cockpit durante la instalación de Debian 12 desde la ISO seleccionada.

Información proporcionada por Cockpit en esta fase:

- **Estado de la MV**: indica que la máquina está en ejecución (*Running*).
- **Recursos asignados**: se muestran los **2 vCPUs** y los **2 GiB de RAM** configurados.
- **Orden de arranque**: definido como `disk,disk`, lo que garantiza que la VM inicie desde el disco tras la instalación.
- **Consola VNC**: activa y funcional, accesible desde la interfaz web para interactuar en tiempo real con el entorno instalado.
- **Consumo de recursos**: se visualiza el uso en tiempo real de **CPU** y **memoria** desde la sección "Uso".
- **Gestión de almacenamiento**: se muestran los discos virtuales asociados, con posibilidad de añadir nuevas unidades si es necesario.

4.3.4 Primer arranque desde el disco virtual

Tras completar la instalación del sistema operativo **Debian 12** dentro de la máquina virtual, se procede a **expulsar la imagen ISO** utilizada como medio de instalación.
Una vez realizada esta operación desde la interfaz de Cockpit, se reinicia la MV para que **arranque desde el disco virtual** recién creado.

La consola integrada (VNC) muestra el **cargador de arranque GRUB**, lo que indica que el sistema ha detectado correctamente el entorno de arranque y está listo para iniciar.
Desde este menú, es posible seleccionar el núcleo predeterminado de **Debian GNU/Linux** y continuar con el primer arranque del sistema instalado.

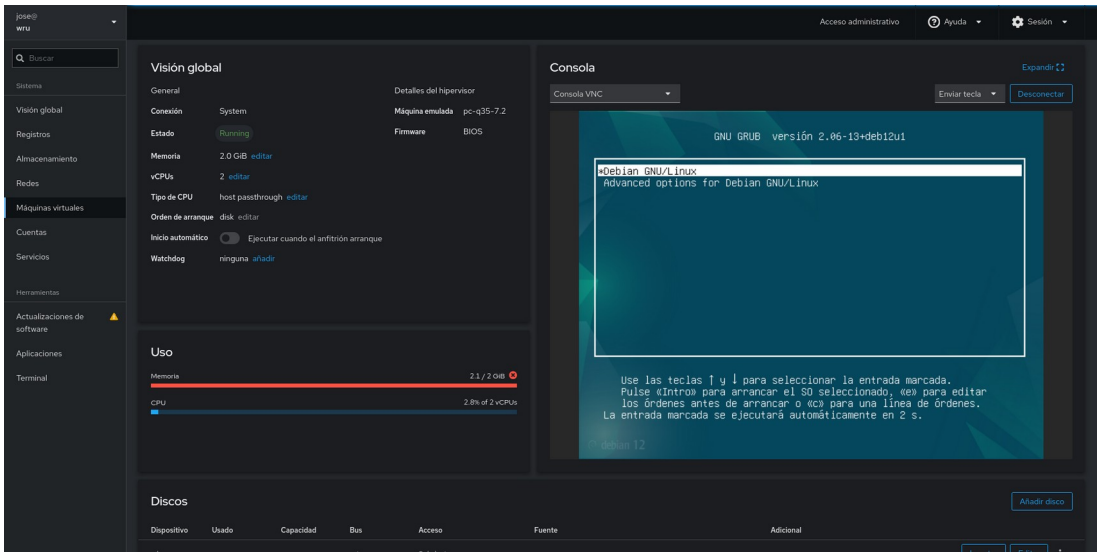


Figura 7: Primer arranque de la máquina virtual tras completar la instalación. Se muestra el cargador GRUB desde la consola integrada en Cockpit.

4.3.5 Clonación de una máquina virtual en Cockpit

Cockpit permite realizar la **clonación de máquinas virtuales** de forma sencilla desde su interfaz web. Esta funcionalidad es especialmente útil para replicar configuraciones, generar entornos de prueba idénticos o acelerar la creación de nuevas máquinas a partir de un sistema ya instalado y funcional.

El proceso se lleva a cabo desde la sección **“Máquinas virtuales”**:

- Se selecciona la máquina que se desea clonar.
- Se hace clic en el **botón de tres puntos (⋮)** situado a la derecha de la máquina.

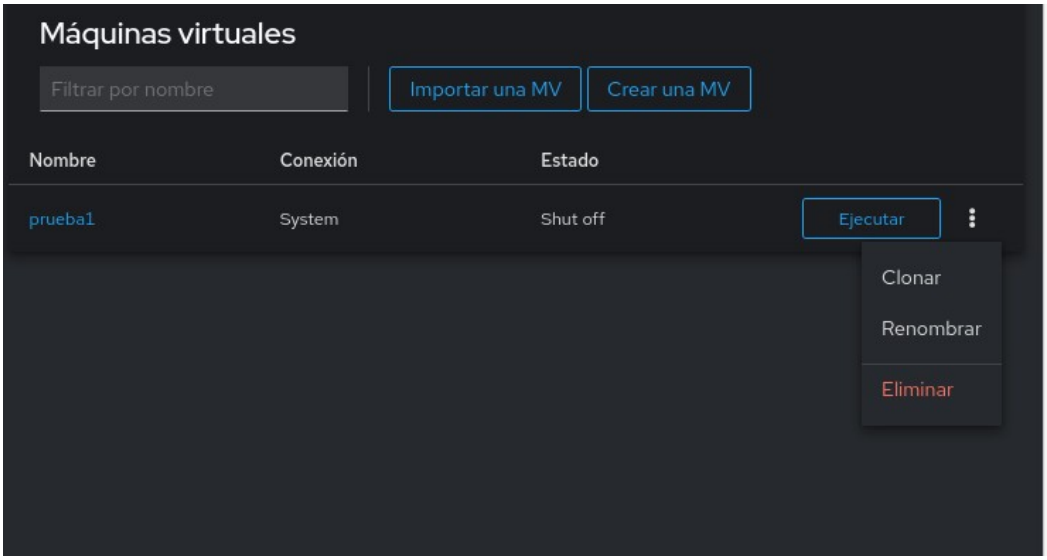


Figura 8: Opciones disponibles para una máquina virtual en Cockpit. Desde el menú ⋮ se puede clonar, renombrar o eliminar.

- En el menú desplegable, se elige la opción **“Clonar”**.
- Se abre un pequeño asistente donde se puede indicar:
- El **nombre** de la nueva máquina virtual.
- La **ruta del nuevo disco virtual** (opcional).

Una vez confirmada la operación, **Cockpit crea una copia completa** de la máquina seleccionada, incluyendo su configuración de hardware (CPU, memoria, almacenamiento), así como su estado actual (si estaba apagada o encendida).

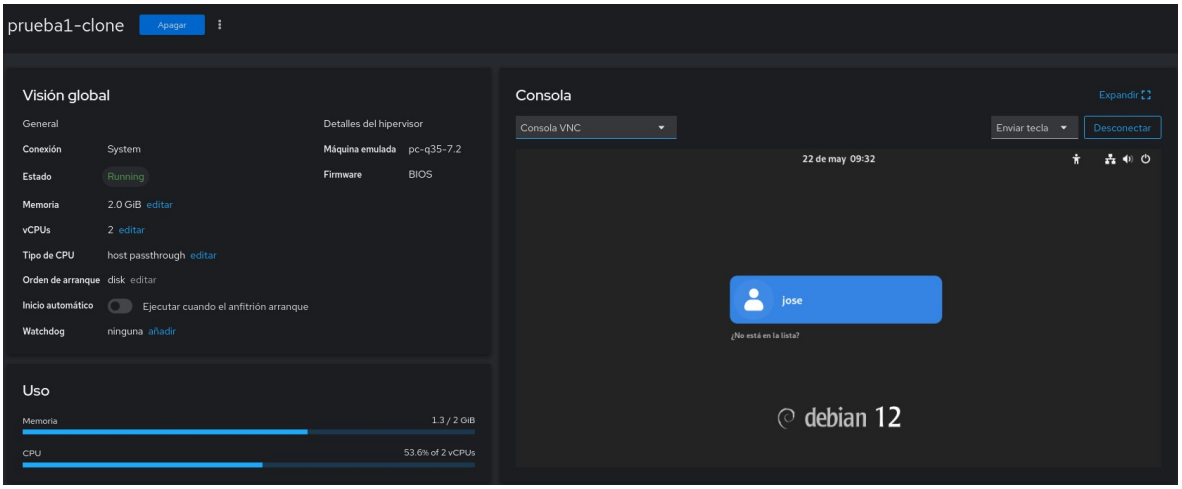


Figura 9: Primer arranque de la máquina virtual tras completar la instalación. Se muestra el cargador GRUB desde la consola integrada en Cockpit.

4.3.6 Eliminación de una máquina virtual en Cockpit

Cockpit permite **eliminar una máquina virtual** directamente desde su interfaz web, de forma rápida y segura. Esta operación resulta útil cuando una VM ya no se necesita o se desea **liberar espacio de almacenamiento** en el sistema anfitrión.

Proceso paso a paso:

• Desde la sección **"Máquinas virtuales"**, se localiza la máquina que se desea eliminar.
• Se hace clic en el **botón de tres puntos (⋮)** junto a la máquina en cuestión.
• En el menú desplegable, se selecciona la opción **"Eliminar"**.

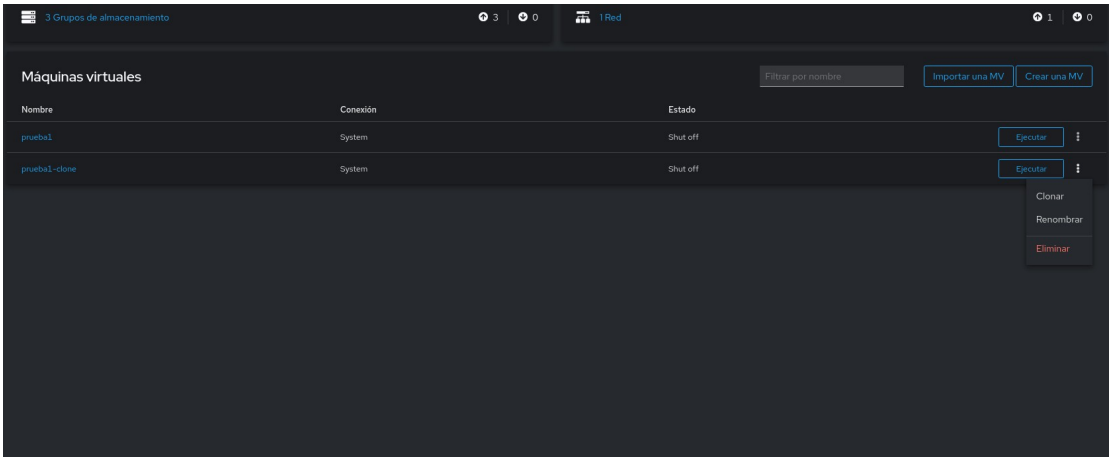


Figura 10: Opciones para una máquina virtual en Cockpit. El menú ⋮ permite clonar, renombrar o eliminar la MV seleccionada.

Una vez confirmada la operación, **Cockpit crea una copia completa** de la máquina seleccionada, incluyendo su configuración de hardware (CPU, memoria, almacenamiento), así como su estado actual (si estaba apagada o encendida).

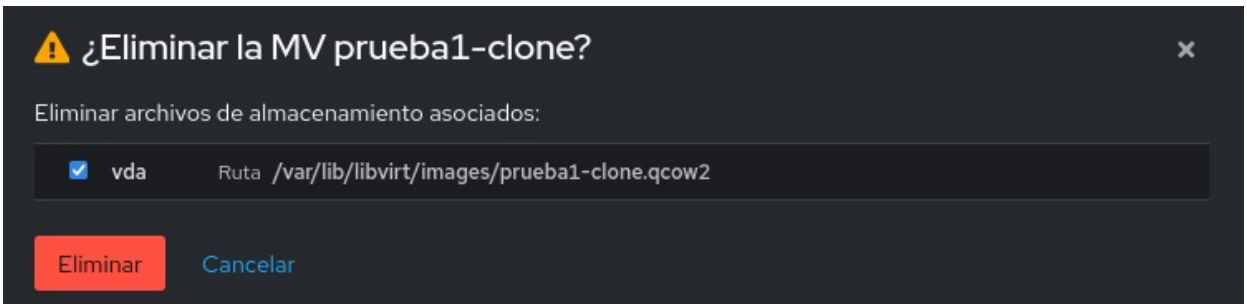


Figura 11: Confirmación de eliminación de la máquina virtual en Cockpit. Se puede optar por eliminar también el archivo de almacenamiento (.qcow2).

4.4 Gestión de contenedores con Cockpit (Podman)

Cockpit permite **gestionar contenedores de forma gráfica** mediante la integración con **Podman**, una alternativa moderna a Docker que no requiere un demonio en ejecución.
Gracias a esta funcionalidad, es posible **crear, ejecutar, supervisar y eliminar contenedores directamente desde el navegador**, sin necesidad de utilizar la línea de comandos.

Esta herramienta resulta especialmente útil para desplegar servicios ligeros, entornos de prueba o sistemas aislados de forma sencilla y accesible.

Instalación del módulo de contenedores

El módulo de gestión de contenedores **no se incluye por defecto** en la instalación base de Cockpit. Para activarlo, es necesario instalar el paquete adicional `cockpit-podman`.

Comando de instalación:

sudo apt install cockpit-podman -y

Una vez completada la instalación, se recomienda reiniciar el servicio para cargar correctamente el nuevo módulo:

sudo systemctl restart cockpit

Acceso al módulo

Tras recargar Cockpit en el navegador (normalmente en `https://localhost:9090` o la IP del anfitrión), aparecerá una nueva sección en el menú lateral llamada **“Contenedores”**.

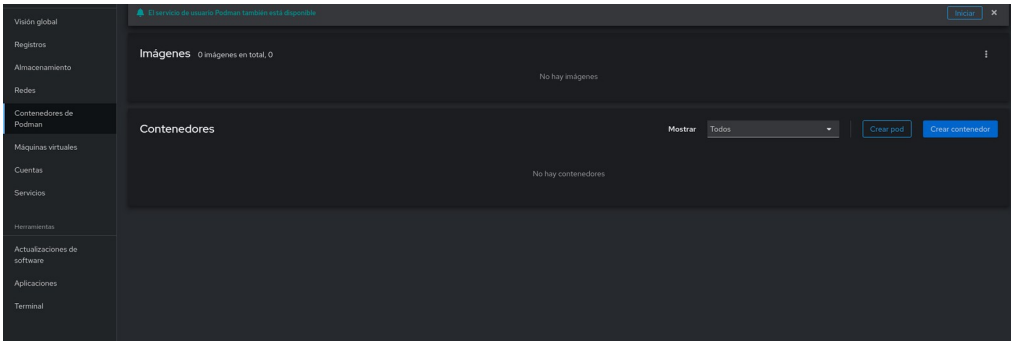


Figura 12: Vista inicial del módulo de contenedores en Cockpit. Desde aquí se puede crear pods, contenedores e inspeccionar imágenes locales.

Desde allí se podrá:

- Consultar contenedores en ejecución
- Crear nuevos contenedores
- Gestionar imágenes locales
- Supervisar el estado y recursos de cada contenedor

4.4 Gestión de contenedores y pods con Cockpit (Podman)

Cockpit permite gestionar contenedores de forma visual mediante su integración con **Podman**, una alternativa moderna a Docker que **no requiere demonio en ejecución**. Gracias al módulo `cockpit-podman`, es posible crear, iniciar, detener, eliminar y monitorizar contenedores directamente desde el navegador, sin necesidad de utilizar comandos en terminal.

Esta funcionalidad es ideal para desplegar servicios aislados, realizar pruebas en entornos ligeros o trabajar con múltiples contenedores sin abandonar la interfaz gráfica del sistema.

4.4.1 Instalación del módulo cockpit-podman

El módulo `cockpit-podman` no viene instalado por defecto. Para añadirlo

sudo apt install cockpit-podman -y

Después de la instalación, reinicia el servicio de Cockpit:

sudo systemctl restart cockpit

Al volver a acceder a la interfaz (normalmente en `https://localhost:9090`), aparecerá una nueva sección en el menú lateral llamada Contenedores de Podman.

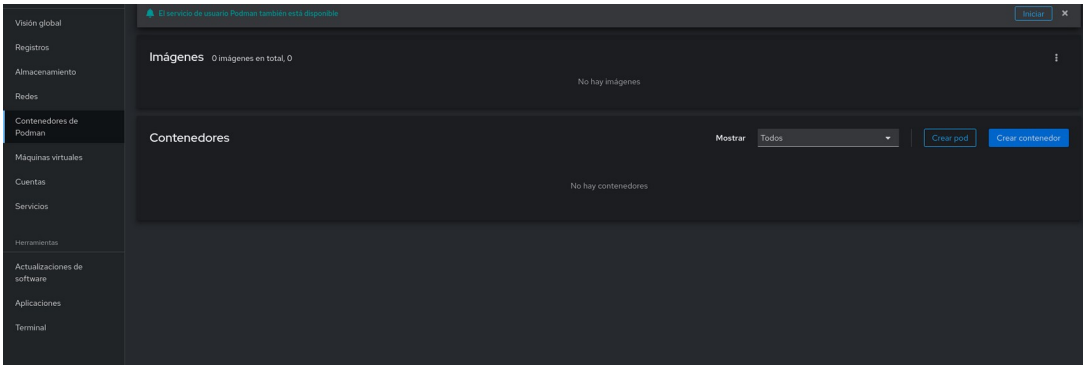


Figura 13: Vista inicial del módulo de contenedores. Muestra "Imágenes" y "Contenedores" vacíos

4.4.2 Creación de un contenedor

Para crear un nuevo contenedor:

• Ve a la sección **Contenedores de Podman**
• Haz clic en el botón **“Crear contenedor”**
• Se abrirá un asistente gráfico paso a paso

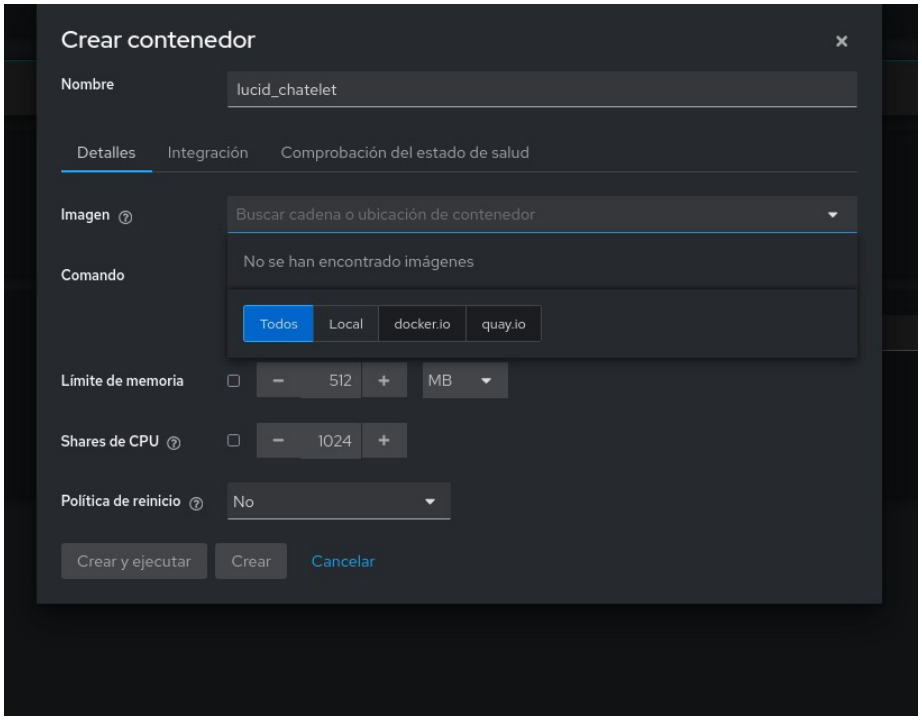


Figura 14: Captura del formulario de creación del contenedor, con los campos vacíos.

Campos principales:

• **Nombre**: identificador del contenedor. Puede ser autogenerado (como `lucid_chatelet`) o personalizado.
• **Imagen**: nombre de la imagen base, por ejemplo:

- `alpine`
- `nginx`
- `ubuntu`
- `debian`

  Cockpit buscará la imagen en registros como `docker.io`, `quay.io`, o el repositorio local. Si no encuentra coincidencias, mostrará un aviso.

Comparativa rápida de registros:

Registro	Propietario	Uso principal	Seguridad
docker.io	Docker Inc.	Proyectos personales y test	Moderada
quay.io	Red Hat / IBM	Producción, entornos serios	Alta (con escaneos)
Local	Sistema anfitrión	Pruebas offline	Variable

4.4.3 Ejemplo práctico: crear el contenedor alpine_test

Para esta prueba se utiliza la imagen `alpine`, muy ligera y común en entornos de contenedores.

Parámetros utilizados:

• **Nombre**: `alpine_test`
• **Imagen**: `alpine`
• **Comando de inicio**: `sh`
• **RAM y CPU**: valores por defecto (512 MB RAM)
• **Política de reinicio**: `No`

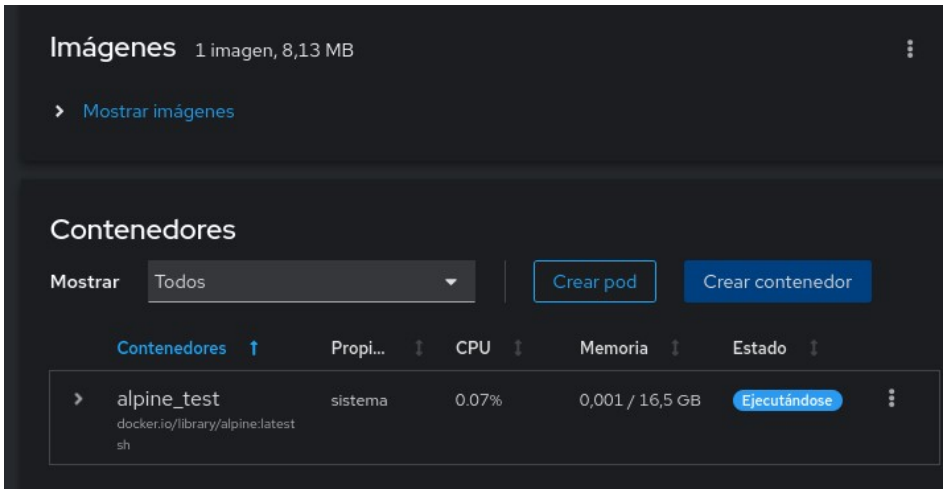


Figura 15: Contenedor "alpine_test" creado y en ejecución.]

Una vez creado, el contenedor aparece en la tabla de contenedores con información como:

• Nombre
• Imagen utilizada
• Estado (Ejecutándose)
• Uso de CPU y RAM
• Acceso a consola, logs y acciones (reiniciar, eliminar, etc.)

4.4.4 Gestión del contenedor

Cockpit ofrece varias acciones:

• **Ver detalles**: pestaña que muestra configuración y estado del contenedor
• **Consola**: terminal interactiva integrada en el navegador
• **Detener / Reiniciar / Eliminar**: accesible desde el menú contextual (⋮)
• **Ver registros**: logs del contenedor en tiempo real

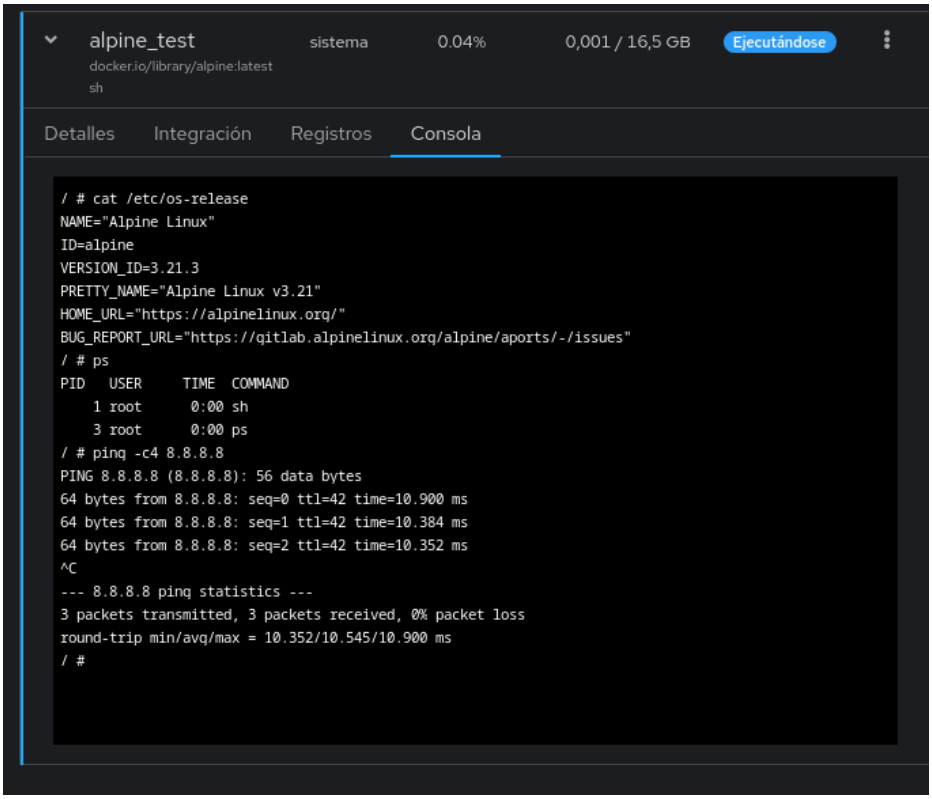


Figura 16: Consola de Cockpit mostrando la shell de alpine_test

4.4.5 Interacción desde el sistema anfitrión (host)

Si necesitas más control, puedes usar comandos desde Debian:

• **Listar contenedores**:

```
sudo podman ps -a
```

• **Iniciar contenedor**:

```
sudo podman start alpine_test
```

• **Acceder a su shell**:

sudo podman exec -it alpine_test sh

```
jose@wru:~$ sudo podman start alpine_test
sudo podman exec -it alpine_test sh
alpine_test
/ #
```

Figura 17: Inicio del contenedor desde la terminal

4.5 **Gestión de pods con Cockpit y Podman**

Cockpit permite no solo crear y gestionar contenedores individuales, sino también agruparlos en **pods**, una funcionalidad que toma como modelo el enfoque de Kubernetes. Los pods permiten que varios contenedores compartan red, almacenamiento y ciertas configuraciones, funcionando como una unidad lógica.

4.5.1 Creación de un pod vacío en Cockpit

Para comenzar, accedemos a la sección **Contenedores de Podman** desde el menú lateral de Cockpit.

Figura 18: Formulario de creación de pod vacío (nombre: gracious_meitner) donde todavía no se han definido puertos ni volúmenes.

Podemos asignar:

- Nombre del pod (identificativo).
- Mapeo de puertos (opcional).
- Volúmenes (opcional).

-

4.5.2 Añadir contenedores a un pod

Una vez creado el pod, aparece listado como un **Grupo de pod**. A la derecha, se muestra el botón “Crear un contenedor en el pod”.

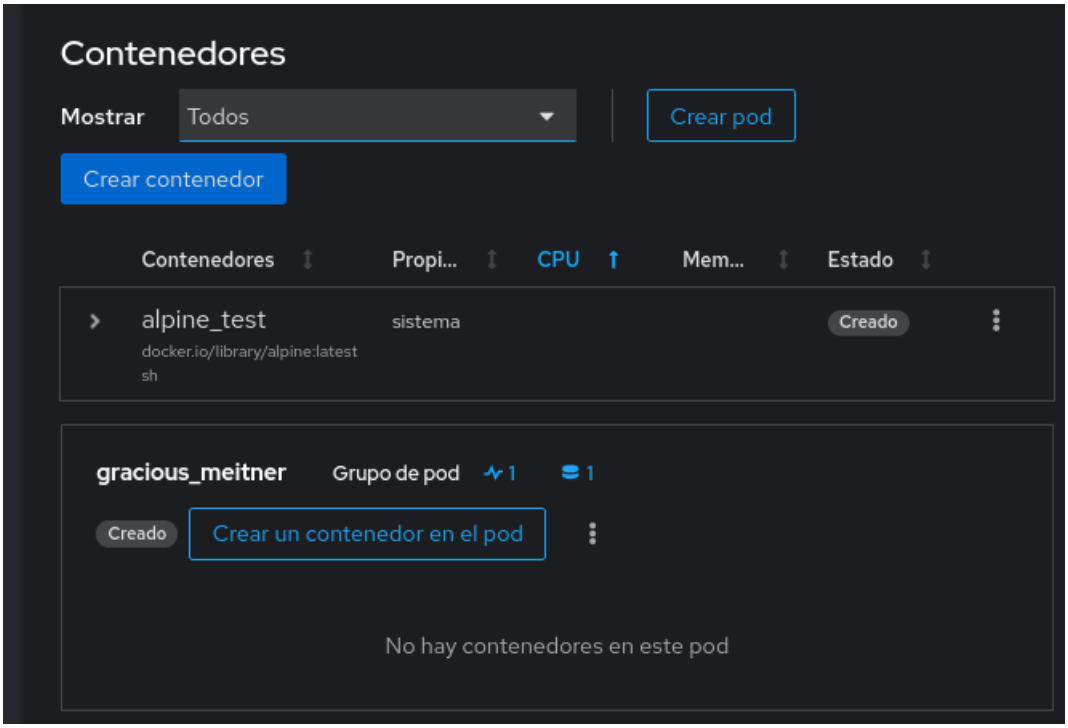


Figura 19: Lista de contenedores y pods, donde gracious_meitner aparece como pod sin contenedores.

Al pulsar el botón, se abre un asistente con los mismos campos que en los contenedores normales, pero todos los recursos definidos se aplicarán dentro del pod:

- Nombre del contenedor
- Imagen base (por ejemplo, `ubuntu` o `alpine`)
- Comando de inicio (por ejemplo, `sh` o `top`)
- Terminal interactiva (opcional)
- Recursos asignados (RAM, CPU)
- Política de reinicio (No, On failure, Always)´

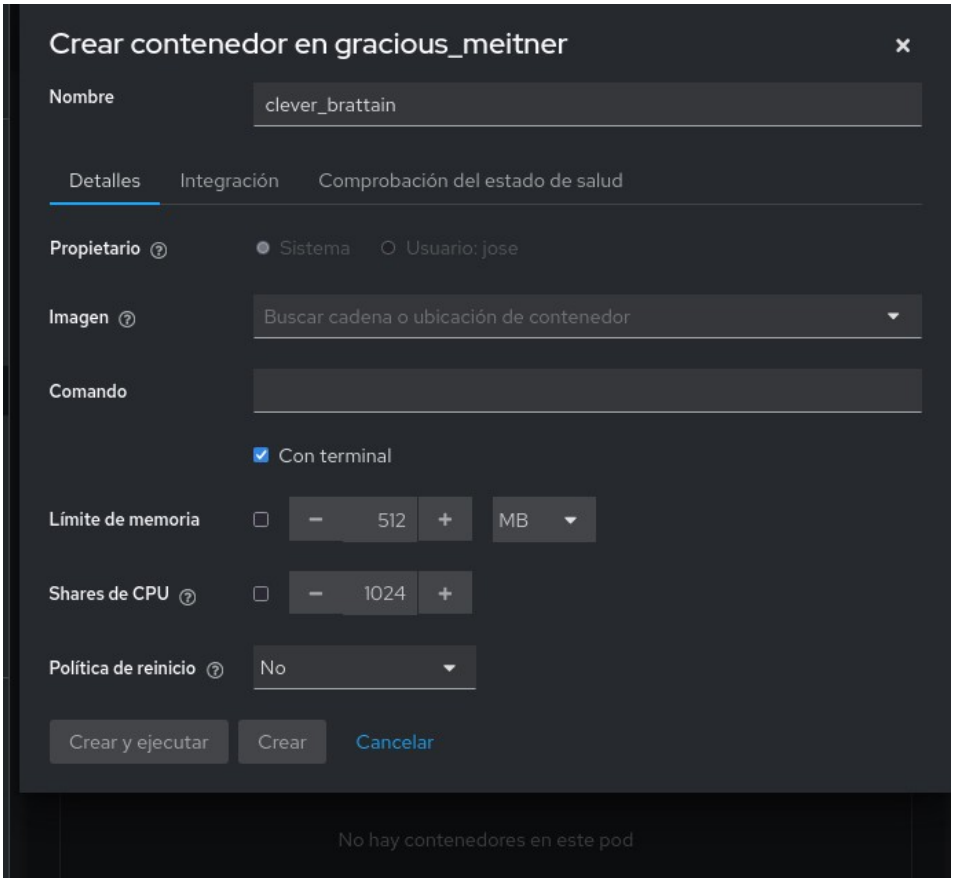


Figura 20: Formulario de creación de contenedor clever_brattain dentro del pod gracious_meitner, con opción de terminal, configuración de CPU, RAM y política de reinicio.

Figura 19: Formulario de creación de contenedor `clever_brattain` dentro del pod `gracious_meitner`, con opción de terminal, configuración de CPU, RAM y política de reinicio.

4.5.3 Configuración avanzada de puertos y volúmenes

Durante la creación del pod, se puede hacer clic en “Añadir mapeo de puertos” para configurar el acceso externo:

- Dirección IP del anfitrión (opcional)
- Puerto del anfitrión (por ejemplo, `8080`)
- Puerto del contenedor (por ejemplo, `80`)
- Protocolo (`TCP` o `UDP`)

También es posible añadir volúmenes:

- Ruta del anfitrión (por ejemplo, `/tmp`)
- Ruta del contenedor (por ejemplo, `/mnt`)
- Modo de acceso (solo lectura o escritura habilitada)

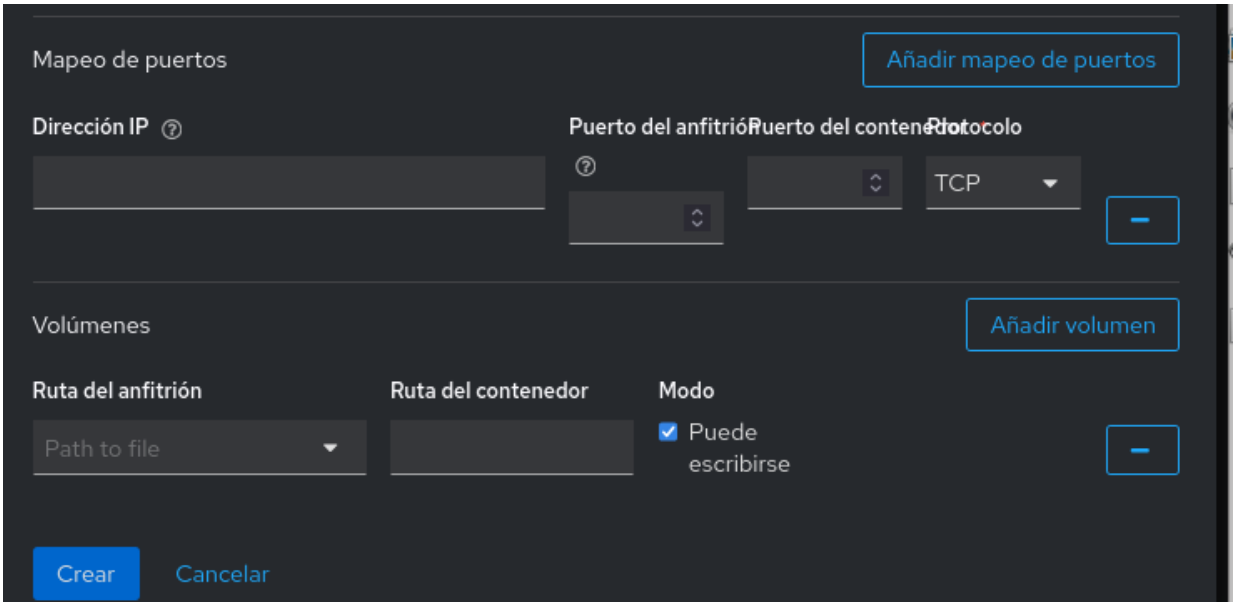


Figura 21: Sección de configuración de puertos y volúmenes para el pod

4.5.4 Consola y ejecución dentro del pod

Una vez creado y ejecutado el contenedor, Cockpit permite acceder a su consola desde la pestaña “Consola”.

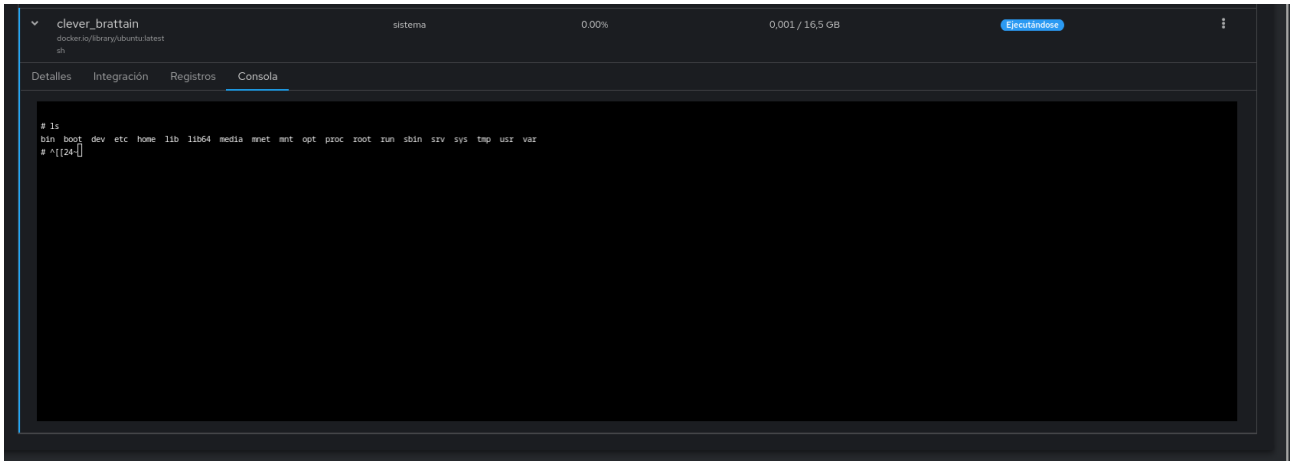


Figura 22: Consola interactiva del contenedor clever_brattain, mostrando el comando ls ejecutado dentro del contenedor.

Además, se puede interactuar con el contenedor directamente desde el sistema anfitrión con Podman:

```
sudo podman exec -it clever_brattain sh
```

```
jose@wru:~$ sudo podman exec -it clever_brattain sh
[sudo] contraseña para jose:
# cat /etc/os-release
PRETTY_NAME="Ubuntu 24.04.2 LTS"
NAME="Ubuntu"
VERSION_ID="24.04"
VERSION="24.04.2 LTS (Noble Numbat)"
VERSION_CODENAME=noble
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=noble
LOGO=ubuntu-logo
# 
```

Figura 23: Consola del sistema anfitrión con acceso al contenedor, mostrando el contenido de /etc/os-release y confirmando que se trata de Ubuntu 24.04.

Esta combinación de Cockpit + Podman proporciona una forma visual, segura y potente de trabajar con contenedores y pods sin depender completamente de la línea de comandos.

4.4 Gestión de redes con Cockpit

Cockpit también permite administrar la red del sistema de forma visual y sencilla, sin necesidad de editar archivos de configuración ni utilizar comandos en la terminal. Desde el módulo de red es posible consultar el estado de las interfaces, cambiar configuraciones, gestionar direcciones IP, activar o desactivar adaptadores, y mucho más.

Esta funcionalidad es especialmente útil en entornos virtualizados o en servidores donde se gestionan varias interfaces (físicas o virtuales), ya que ofrece una vista clara y centralizada del estado de la conectividad.

4.4.1 Estado de las interfaces de red

La vista principal del módulo de red en Cockpit muestra:

- Gráficas en tiempo real del tráfico transmitido y recibido por todas las interfaces del sistema.
- Una tabla con las interfaces disponibles, su dirección IP, estado (activo o inactivo), y el volumen de datos enviados y recibidos.

Figura 24: Vista principal del módulo de red mostrando las interfaces detectadas, su IP y el tráfico en tiempo real.

En el ejemplo mostrado se puede observar:

- `enx000ec6512870` es la interfaz Ethernet principal, conectada con la IP `192.168.1.58`.
- `wlo1` es una interfaz inalámbrica con la IP `192.168.1.228`, pero no está transfiriendo datos.
- `virbr0` es una interfaz virtual típica en entornos con KVM/libvirt, con la IP `192.168.122.1`.
- También aparecen interfaces internas como `lo` (loopback) y `p2p-dev-wlo1`.

4.4.2 Detalle de una interfaz de red

Al hacer clic sobre una de las interfaces listadas (por ejemplo, `enx000ec6512870`), se muestra una vista detallada con la siguiente información:

- Nombre y descripción del dispositivo: incluyendo fabricante, modelo y dirección MAC.
- Estado actual: IP asignada (IPv4 e IPv6), máscara de red y si está activa.
- Velocidad de enlace: por ejemplo, 1 Gbps.
- Conectividad automática: opción habilitada para que se conecte al arrancar el sistema.
- Configuración IP:
    - IPv4: actualmente configurada como automática (DHCP).
    - IPv6: automática.
    - MTU: en modo automático.

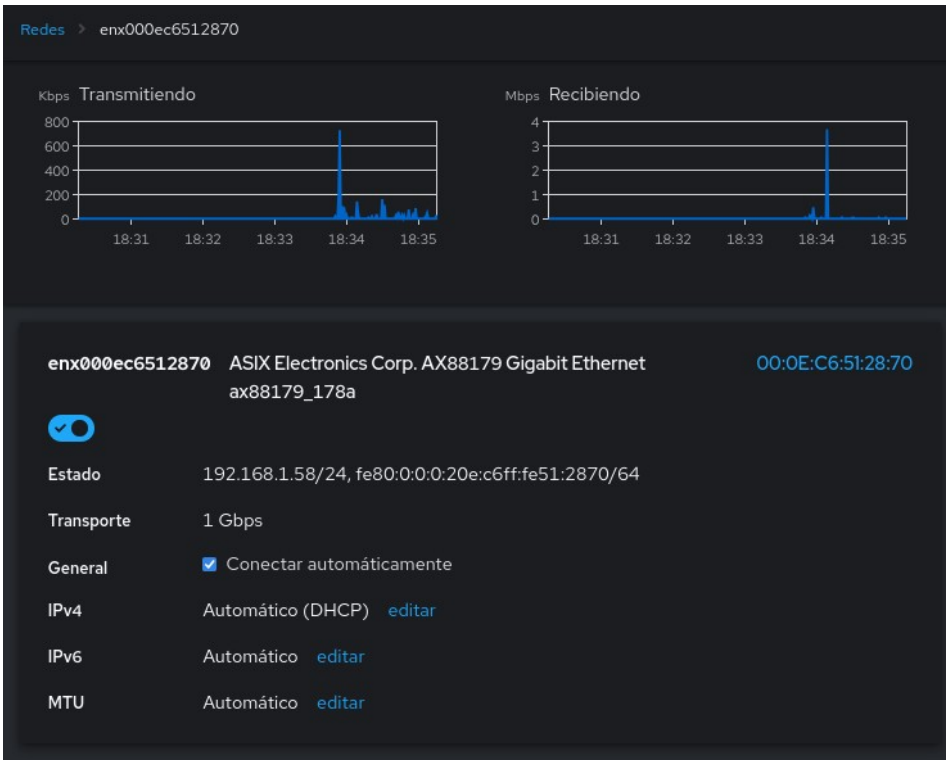


Figura 25: Detalle de configuración de la interfaz enx000ec6512870, incluyendo estado, direcciones IP y configuración avanzada.

Figura 15: Detalle de configuración de la interfaz `enx000ec6512870`, incluyendo estado, direcciones IP y configuración avanzada.

En la parte superior se incluyen gráficas de tráfico específicas para esa interfaz.

4.4.3 Creación de un puente de red (bridge)

Cockpit permite configurar redes virtuales avanzadas para conectar máquinas virtuales con la red física del host. Una de las opciones más habituales es la creación de un puente de red (bridge), que actúa como un conmutador virtual al que se pueden conectar tanto las interfaces físicas del sistema como las interfaces de las máquinas virtuales.

Para crear un puente:

1. Accedemos a la pestaña **Redes** dentro de Cockpit y pulsamos en el botón **Añadir puente**.
2. Se abrirá una ventana de configuración con los siguientes campos:
    - **Nombre**: se asigna un identificador para el puente. Por defecto se propone `bridge0`, aunque puede modificarse si se desea gestionar múltiples bridges.
    - **Puertos**: se muestran todas las interfaces de red disponibles en el sistema. En este caso, seleccionamos la interfaz física principal del host, `enx000ec6512870`, que tiene asignada la dirección IP `192.168.1.58/24`. Esto conectará el puente con la red real del entorno.
    - **Spanning Tree Protocol (STP)**: es una opción avanzada que permite evitar bucles de red. No es necesaria en entornos simples, por lo que puede dejarse desactivada.

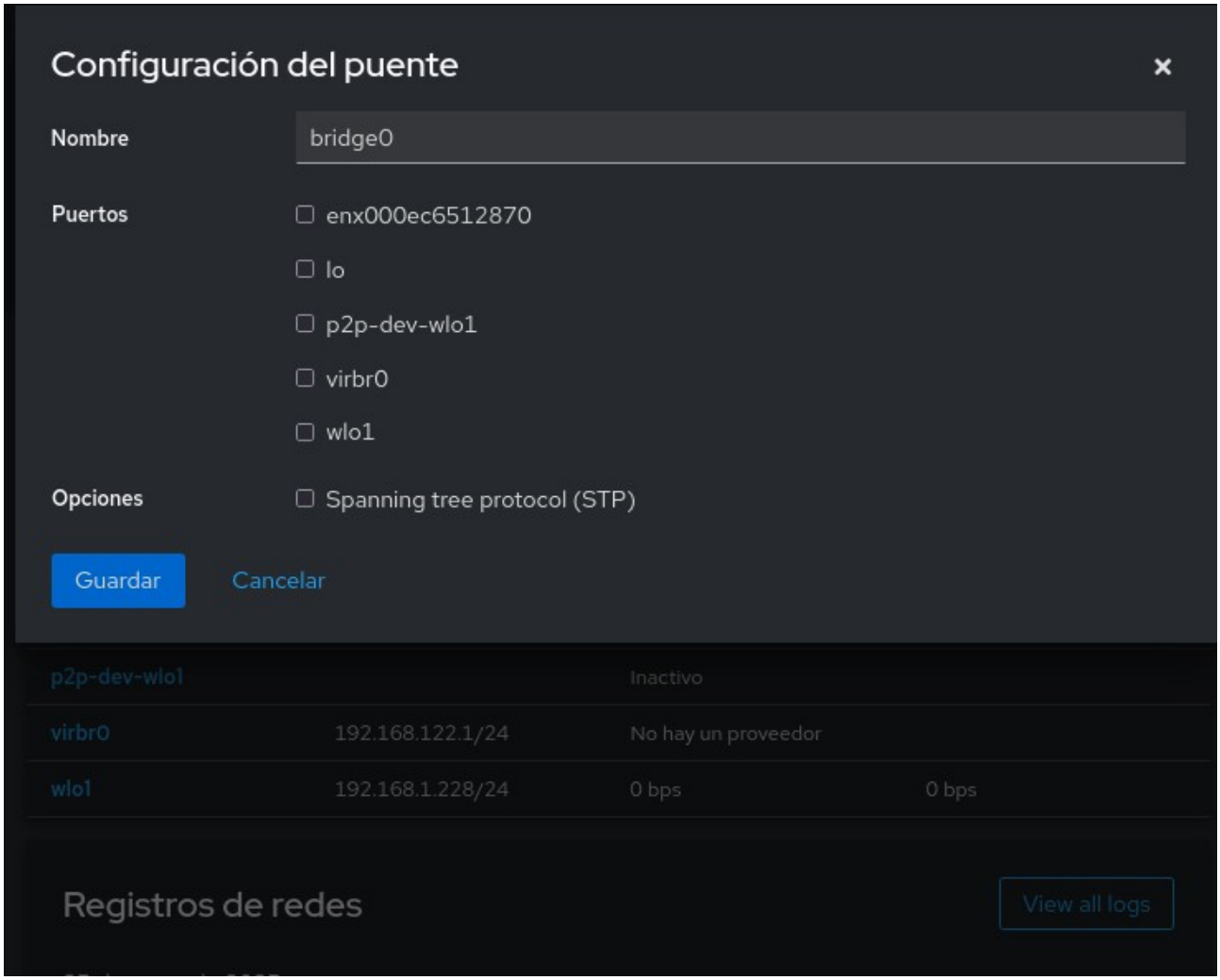


Figura 26: Ventana de configuración de un puente de red en Cockpit, seleccionando la interfaz física principal como puerto del bridge.

Una vez completados los campos, pulsamos **Guardar** para crear el puente.

Este puente podrá ser utilizado más adelante como interfaz de red al crear o modificar máquinas virtuales desde la sección correspondiente. De esta forma, las VMs recibirán una dirección IP desde el mismo router que el sistema anfitrión y podrán comunicarse con otros dispositivos de la red local.

4.4.4 Creación de una red virtual aislada

Además del uso de puentes, Cockpit permite crear redes virtuales completamente aisladas, lo que resulta útil para entornos de pruebas o simulaciones de red.

1. Accedemos a la sección **Máquinas virtuales** y seleccionamos **Redes**.
2. Hacemos clic en **Crear una red virtual**.
3. Rellenamos el formulario con los siguientes datos:
    - **Nombre**: por ejemplo, `redprivadadejose`.
    - **Modo de redireccionamiento**: seleccionamos `None (isolated network)` para que sea una red interna.
    - **Configuración de IP**: elegimos `Solo IPv4`, estableciendo una dirección base como `10.0.0.2` y una máscara `/24`.
    - (Opcional) Activar DHCP si queremos asignación automática de IPs a las Vms.

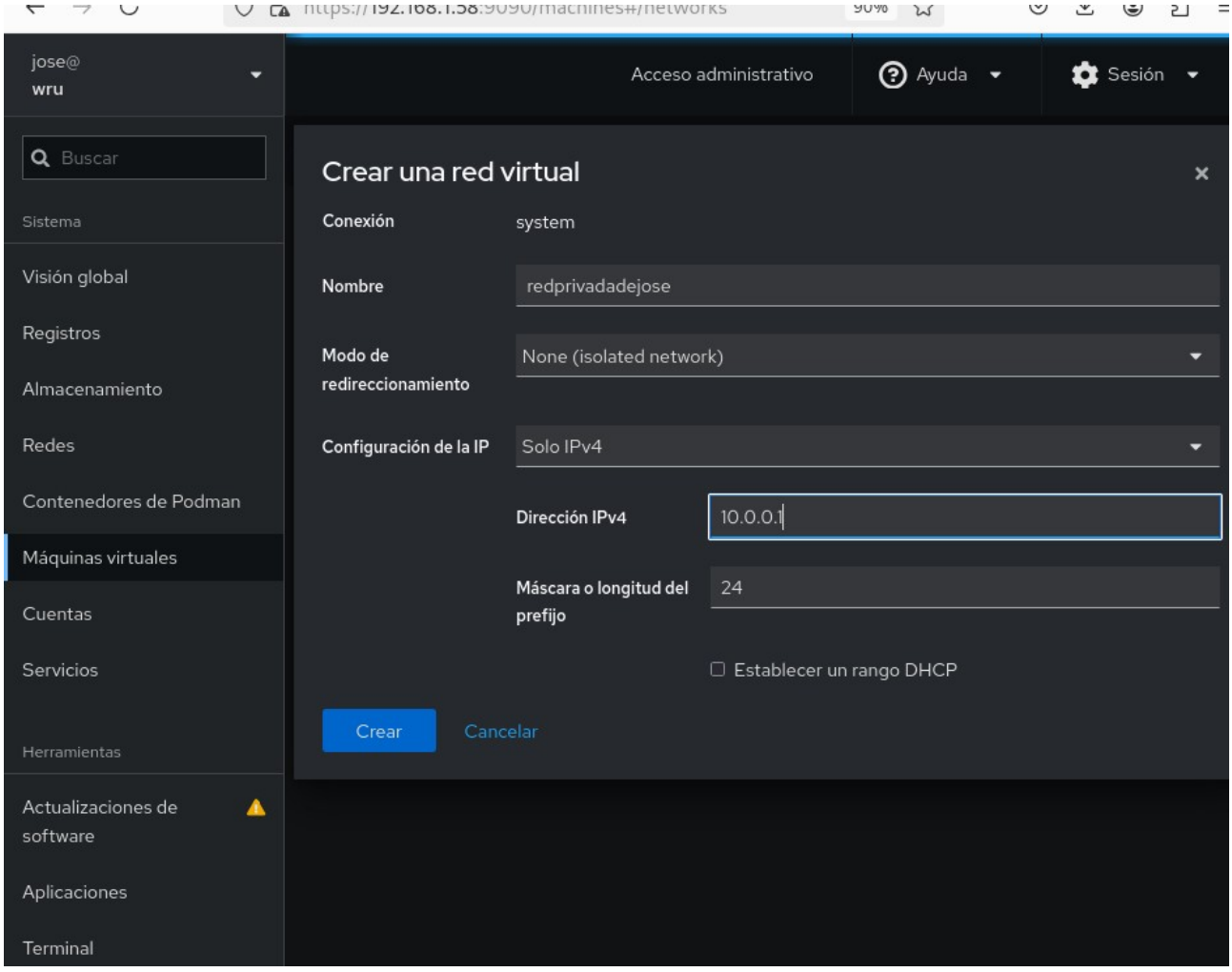


Figura 27: Formulario de creación de red virtual aislada en Cockpit, con dirección IP 10.0.0.1/24.

4.4.5 Asignar red virtual a una máquina

Una vez creada la red, podemos asociarla a una máquina virtual existente:

1. Editamos la máquina desde la sección **Máquinas virtuales**.
2. Pulsamos **Añadir una interfaz de red** o editamos una existente.
3. Seleccionamos como fuente la red `redprivadadejose`, el modelo `virtio` y verificamos que se asigna una dirección MAC.

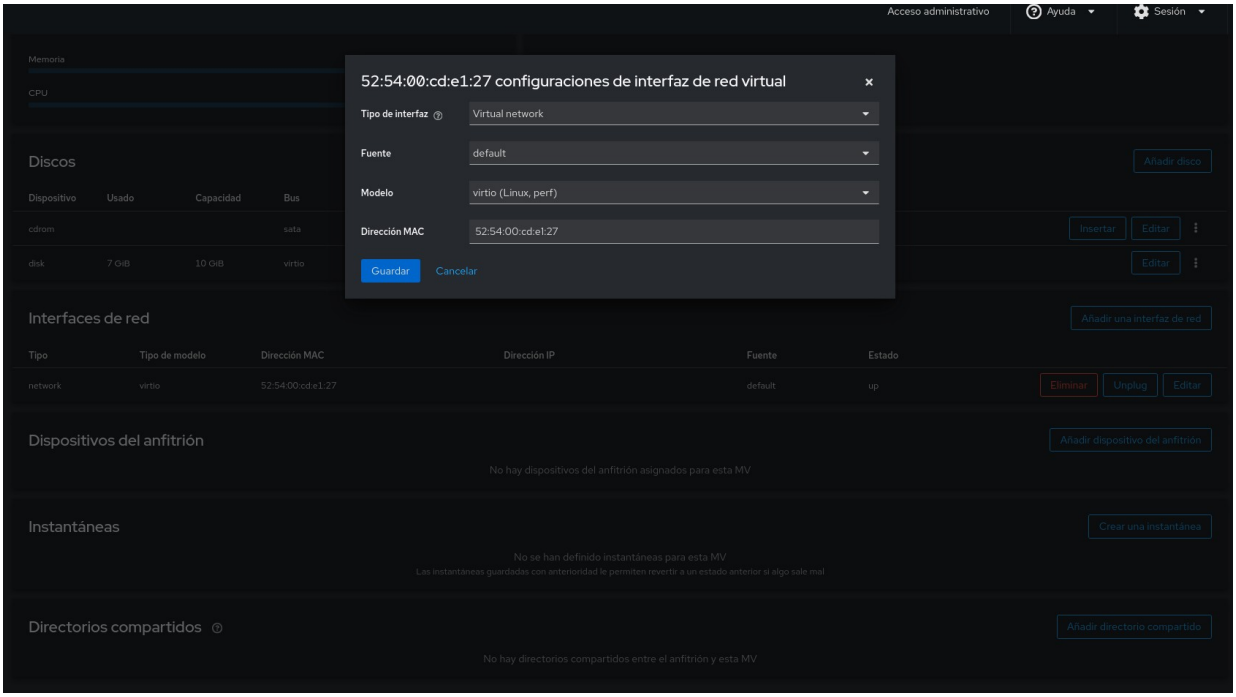


Figura 28: Configuración de interfaz de red virtual para una VM, asignando la red aislada creada.

Una vez iniciada la VM, al ejecutar `ip a` desde su terminal se puede comprobar que ha recibido una dirección dentro del rango definido, como `10.0.0.2`.

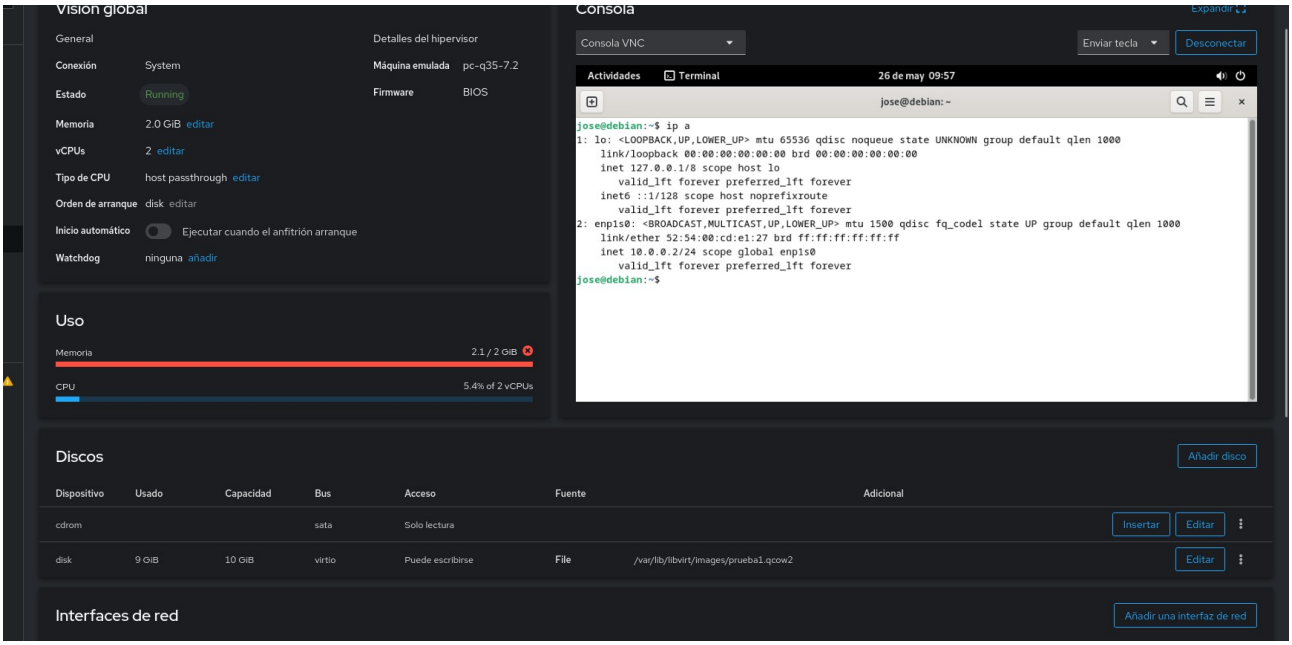


Figura 29: Consola de la máquina virtual mostrando la interfaz de red con IP 10.0.0.2, confirmando conectividad en la red aislada.

5. Dificultades encontradas y resolución de problemas

Durante el desarrollo de este proyecto se han presentado diversas dificultades técnicas que han requerido análisis, búsqueda de soluciones y aplicación práctica de conocimientos relacionados con la virtualización, el sistema operativo Linux y el uso de la herramienta Cockpit.

Aunque la mayor parte de las tareas han podido llevarse a cabo con normalidad, ha sido necesario resolver varios problemas reales que surgieron en diferentes fases, desde la instalación inicial hasta la puesta en marcha y gestión de máquinas virtuales.
Estos errores, lejos de suponer un obstáculo, han aportado una experiencia enriquecedora al proyecto, ya que han permitido afianzar conceptos clave y mejorar la comprensión del funcionamiento interno de los componentes utilizados, especialmente libvirt, QEMU, la gestión de redes virtuales, y la integración con Cockpit.

A continuación, se detallan las principales incidencias encontradas, su causa probable y la solución adoptada en cada caso.

5.1 Cockpit en modo de acceso limitado

- **Síntoma:**
  Al acceder por primera vez a la interfaz web de Cockpit, muchas funcionalidades aparecían deshabilitadas (máquinas virtuales, redes, servicios del sistema).
  Mensaje visible: `La consola web está en modo de acceso limitado`.
- **Causa:**
  Cockpit no habilita privilegios administrativos por defecto, aunque el usuario tenga permisos `sudo`.
- **Solución adoptada:**
  Se activó el modo administrativo desde la interfaz, pulsando “Turn on administrative access” e introduciendo la contraseña del usuario.

5.2 Falta del módulo de máquinas virtuales en Cockpit

- **Síntoma:**
  Aunque se había activado el acceso administrativo, no aparecía la opción “Máquinas virtuales” en el menú lateral.
- **Causa:**
  El módulo `cockpit-machines` no se instala automáticamente en algunas distribuciones.
- **Solución adoptada:**
  Se instaló el módulo manualmente ejecutando:
  `sudo apt install cockpit-machines -y`.

5.3 La red 'default' de libvirt no está activa

- **Síntoma:**
  Al intentar crear una nueva máquina virtual, Cockpit muestra un error que indica que la red `default` está inactiva, impidiendo la conectividad de la VM.
- **Causa:**
  Por defecto, libvirt no activa automáticamente la red virtual `default` al instalarse.
- **Solución adoptada:**
  Se activó manualmente la red ejecutando:
  `sudo virsh net-start default`
  y para que se active automáticamente al arrancar el sistema:
  `sudo virsh net-autostart default`.

5.4 La máquina virtual no arranca tras la instalación

- **Síntoma:**
  Al iniciar la máquina virtual, aparece el mensaje `Booting from Hard Disk...` pero no arranca el sistema.
- **Causa:**
  La instalación del sistema operativo no se completó correctamente, o la ISO se expulsó antes de finalizar.
- **Solución adoptada:**
  Se repitió el proceso asegurando que la instalación finalizara correctamente y se revisó el orden de arranque (`disk,disk`).

5.5 Dificultad al configurar puentes de red

- **Síntoma:**
  Surgieron dudas al configurar un puente de red (bridge), especialmente sobre qué interfaz física seleccionar y si se perdería la conectividad del sistema anfitrión.
- **Causa:**
  Falta de experiencia con la configuración de bridges y miedo a dejar el sistema sin conexión.
- **Solución adoptada:**
  Se seleccionó la interfaz `enx000ec6512870` como puerto del bridge, se dejó desactivado STP, y se confirmó que tanto el host como las máquinas virtuales mantenían la conectividad en la red local.

5.6 Dificultad al continuar tras crear un pod vacío

- **Síntoma:**
  Después de crear el pod `gracious_meitner`, no estaba claro cómo continuar, ya que el pod aparecía vacío.

- **Causa:**
  Desconocimiento del flujo de trabajo: crear un pod no crea automáticamente contenedores.
- **Solución adoptada:**
  Se usó la opción “Crear contenedor en el pod” desde Cockpit, seleccionando una imagen base (`alpine`) y el comando de inicio `sh`, habilitando así la consola y el uso del pod.

5.7 Consola inactiva tras crear un contenedor

- **Síntoma:**
  La consola del contenedor aparecía deshabilitada tras su creación.
- **Causa:**
  El contenedor se creó sin un comando interactivo (por ejemplo, `sh` o `bash`), por lo que finalizaba inmediatamente.
- **Solución adoptada:**
  Se recreó el contenedor añadiendo el comando `sh` y activando la opción de terminal. También se ajustó la política de reinicio para mantenerlo en ejecución.

6. Conclusiones y propuestas de mejora

La realización de este proyecto ha permitido explorar de forma práctica el uso de Cockpit como interfaz gráfica de administración de sistemas virtualizados con KVM y libvirt en entornos GNU/Linux. A lo largo del desarrollo se ha comprobado que esta herramienta ofrece una solución accesible, visual y funcional para la gestión de máquinas virtuales, contenedores, redes y recursos del sistema, sin necesidad de recurrir constantemente a la línea de comandos.

Entre los principales logros alcanzados destacan:

- La instalación y configuración completa de un entorno Debian con soporte para virtualización.
- La integración exitosa de los módulos `cockpit-machines`, `cockpit-podman` y `cockpit-networkmanager`.
- La creación y gestión de múltiples máquinas virtuales desde la interfaz web.
- La configuración de redes virtuales avanzadas, incluyendo bridges y redes aisladas.
- La implementación de pods y contenedores con Podman desde Cockpit, sin necesidad de comandos en terminal.
- La documentación detallada de cada fase del proceso, incluyendo errores reales y las soluciones aplicadas.

Este proyecto ha contribuido a consolidar conocimientos en administración de sistemas, virtualización, redes y servicios en entornos Linux, aplicándolos en un escenario realista que simula el uso de servidores locales con acceso remoto.

Propuestas de mejora

Aunque Cockpit ha demostrado ser una herramienta potente y flexible, también se han identificado algunas áreas susceptibles de mejora que podrían desarrollarse en trabajos futuros:

- **Seguridad de acceso web:**
  Cockpit utiliza por defecto el puerto 9090 sin cifrado. En entornos reales, sería recomendable implementar HTTPS con certificados válidos, autenticación multifactor y restricciones de acceso mediante firewall o listas de control de IP.
- **Supervisión remota y alertas:**
  Aunque Cockpit muestra información en tiempo real, no genera alertas automáticas. Integrarlo con sistemas como Prometheus, Zabbix o Grafana permitiría establecer notificaciones ante fallos o sobrecargas.
- **Automatización de despliegues:**
  En escenarios más complejos, Cockpit podría complementarse con herramientas como

Ansible o Terraform para facilitar el aprovisionamiento automatizado de máquinas, redes y servicios.

- **Gestión avanzada de snapshots y backups:**
  Si bien permite clonar máquinas, no ofrece una gestión completa de instantáneas ni copias de seguridad programadas. La inclusión de estas funciones facilitaría el trabajo en entornos de pruebas o recuperación ante fallos.
- **Mejor soporte para usuarios no técnicos:**
  Aunque simplifica notablemente muchas tareas, ciertos errores (como la red `default` inactiva o módulos no instalados) pueden resultar confusos. Una guía interactiva o un asistente inicial mejorarían la experiencia para usuarios sin experiencia previa.

Cierre

En conclusión, Cockpit se presenta como una alternativa eficaz para la gestión de sistemas virtualizados en entornos locales o educativos, especialmente útil para quienes desean evitar una dependencia constante de la consola. La experiencia obtenida ha sido altamente formativa, y demuestra que una interfaz bien diseñada puede simplificar tareas complejas sin renunciar al control ni a la eficiencia.

7. Bibliografía y enlaces de interés

- **Cockpit Project**
  https://cockpit-project.org/
  Sitio oficial de la herramienta Cockpit, con documentación sobre instalación, módulos y uso general.
- **Podman Documentation**
  https://docs.podman.io/
  Manual oficial de Podman, utilizado para crear y gestionar contenedores sin daemon.
- **Libvirt Wiki**
  https://wiki.libvirt.org/
  Recurso fundamental para comprender la arquitectura de libvirt y su integración con KVM y QEMU.
- **QEMU Wiki**
  https://wiki.qemu.org/
  Documentación técnica de QEMU, especialmente útil para entender la emulación de hardware.
- **Debian GNU/Linux**
  https://www.debian.org/
  Página oficial de la distribución empleada como sistema anfitrión en el proyecto.
- **Stack Overflow**
  https://stackoverflow.com/
  Plataforma de preguntas y respuestas técnicas. Se consultó para resolver errores relacionados con libvirt, Podman y la red `default`.
- **man virsh**
  Ayuda local disponible en terminal Linux sobre el comando `virsh`, útil para gestionar VMs desde consola.
- **Blog: TechOverflow – "How to fix 'default network is not active' in libvirt"**
  https://techoverflow.net/2020/06/28/how-to-fix-default-network-is-not-active-in-libvirt/
  Solución detallada a un error frecuente relacionado con redes inactivas en libvirt.
- **Blog: Red Hat Developer – “Getting started with Cockpit”**
  https://developers.redhat.com/articles/2021/12/13/getting-started-cockpit
  Guía práctica paso a paso para comenzar con Cockpit en entornos reales.
- **YouTube: Learn Linux TV – “Introduction to Cockpit - A Web-Based Interface for Linux Servers”**
  https://www.youtube.com/watch?v=vQvH0K4-E5U
  Vídeo con demostración real del uso de Cockpit, visualizando opciones y casos de uso.

- **YouTube: Christian Lempa – “Manage your Linux Server with Cockpit”**
  https://www.youtube.com/watch?v=GspvWBa8c9U
  Canal técnico con enfoque en administradores de sistemas. Vídeo muy útil para visualizar la gestión remota con Cockpit en Debian y derivados.
- **GitHub: cockpit-project/cockpit**
  https://github.com/cockpit-project/cockpit
  Repositorio oficial con el código fuente, documentación para desarrolladores y seguimiento de errores.

Agradecimientos

Quiero expresar mi agradecimiento a todas las personas que han hecho posible la realización de este proyecto, tanto en lo técnico como en lo personal. Este trabajo no solo representa el final de un ciclo, sino también un camino de esfuerzo, dudas y superación.

Gracias al departamento de Informática del IES Gonzalo Nazareno, y especialmente a **Conchi, Javi, José Domingo, Rafa y Raúl**.
Gracias por vuestra paciencia, por vuestra pasión y, sobre todo, por creer que dentro de mí había un administrador de redes.
Gracias por enseñarnos desde la cercanía, por estar cuando os necesitábamos y por recordarnos que somos capaces de mucho más de lo que creemos.

A los que me sufren en corto: a mi **familia y amigos**, por ser ese aliento cuando no me quedaba ninguno.
Por tener paciencia conmigo, por animarme a seguir cuando quería rendirme, por consolarme cuando llegaba roto a casa, por sujetar mi mundo cuando se derrumbaba.
Por empujarme contra mis miedos, por no dejarme tirar la toalla, por esos “seguro que lo logras” cuando más los necesitaba.

A los **compañeros**,
a los que me ayudaron y compartieron su camino conmigo, a los que no, a los que terminaron y a los que se quedaron por el camino.
Gracias por formar parte de este viaje, de una manera u otra.

Y por último, quiero agradecérmelo a mí.
A mi **yo del pasado**, porque un día decidió que merecíamos algo mejor que trabajar en hostelería.
Porque se apuntó a ASIR y, aunque el primer mes llegaba a casa llorando sin entender nada, no se rindió.
Porque siguió adelante, aunque el camino estuviera lleno de lágrimas de frustración, de dudas, de noches sin dormir.
Gracias por no soltarlo, por confiar, aunque todo estuviera nublado