Monitorización básica en Linux

1) Conceptos rápidos (qué mirar y por qué)

- **CPU**: % de uso por proceso, carga (load average).
- **Memoria**: RAM usada, buffers, cache, uso de swap.
- **I/O disco**: procesos que leen/escriben mucho → impacto en latencia.
- **Disco/espacio**: tamaño usado, inodos, discos llenos.
- **Red**: conexiones abiertas, puertos escuchando, tráfico.
- **Procesos**: procesos "pesados", usuarios que consumen recursos.

Regla para la **load average**: son promedios (1m, 5m, 15m). Interpretación rápida: comparar con número de núcleos. Por ejemplo, carga 4.0 en CPU de 4 cores ≈ saturación.

Conceptos rápidos de monitorización

CPU

- **Qué mirar**:
  - % de uso por proceso → quién está consumiendo CPU.
  - load average → nivel de carga global del sistema.
- **Por qué**:
  - El % de CPU te dice si un proceso está acaparando recursos.
  - El *load average* mide cuántos procesos están esperando CPU o I/O.
- **Ejemplo**:

  top

  Supongamos que ves:

  %CPU COMMAND
  95.0 gzip

  → El proceso gzip está usando casi un núcleo completo.

  Y en la cabecera:

  load average: 4.00, 3.80, 2.50

Salida al iniciar la máquina:

top - 09:55:53 up 9 min, 1 user, load average: 8,28, 5,36, 2,83

El **Load Average (Carga Promedio)** mide el número de procesos que están esperando activamente para ser ejecutados. Se compara típicamente con el número de núcleos de la CPU.

Métrica	**Valor**
Carga de 1 minuto	8,28
Núcleos de CPU (asumido)	4

- Una carga promedio de **4.00** en un sistema de 4 núcleos significaría que todos los núcleos están completamente ocupados.
- Una carga promedio de **8.28** significa que, en promedio durante el último minuto, había **más del doble de procesos (8.28 / 4 $\approx$ 2.07)** esperando la CPU de las que podían ejecutar simultáneamente.

Conclusión

Si la carga promedio en un sistema de **4 cores** es de **8.28,** significa que su sistema está **severamente sobrecargado** (207% de capacidad de la CPU), lo que concuerda con que el sistema solo lleva **9 minutos** encendido y ya está experimentando un alto uso de CPU (39,9% us + 12,9% sy = $\approx 53\%$ de tiempo activo de CPU total).

Es **muy probable** que el consumo inicial alto de CPU, especialmente si involucra el comando gzip, se deba a un **script** que se ejecuta al inicio o de forma programada y que comprime archivos grandes.

La alta carga que vio en el top (Load Average: 8,28) podría haber sido el pico de actividad de este script de compresión. Una vez que el script termina o el proceso de compresión finaliza, el consumo de CPU y la carga promedio disminuyen drásticamente.

Cómo Confirmar si es un Script

Justo después de que el sistema se inicie y antes de que la carga baje (o al notar un nuevo pico de carga), puede usar ps para ver la línea de comandos completa del proceso.

- **Para ver la lista completa de procesos y la ruta del comando (incluyendo los argumentos y, a menudo, el script que lo lanzó):**

  ps aux | grep gzip

Si el comando gzip fue iniciado por un script (por ejemplo, /usr/local/bin/backup_diario.sh), lo verá en la salida de ps.

Salida de ps y grep:

```
ps aux | grep gzip

root 1334  0.0  0.0   2800  1688 ?      S    09:47 0:00 /bin/sh -c gzip

root 1335 65.1  0.0   3424  1600 ?      R    09:47 8:56 gzip

jose 7280  0.0  0.0   6648  2360 pts/0  S+   10:00 0:00 grep --color=auto gzip
```

Análisis de la Salida de ps

Línea	Usuario	PID	%CPU	Estado	Hora Inicio	Tiempo CPU	Comando
1.	root	1334	0.0	S (Dormido)	09:47	0:00	/bin/sh -c gzip
2.	root	**1335**	**65.1**	**R (Ejecutando)**	09:47	**8:56**	**gzip**
3.	jose	7280	0.0	S+	10:00	0:00	grep --color=auto gzip

Conclusiones Clave

1. **El Culpable Directo:** El proceso con **PID 1335** es el que está consumiendo una gran cantidad de CPU (65.1%). Este es el comando **gzip** en sí, y está en estado **R (Running)**, lo que significa que está ejecutando una compresión activamente.
2. **El Proceso Padre:** La línea superior (PID 1334) es el **shell (/bin/sh)** que inició el proceso gzip. El hecho de que este shell se esté ejecutando como **root** y use la opción -c (que a menudo indica que está ejecutando una cadena de comandos) sugiere fuertemente que fue lanzado por un **script automatizado** o un **trabajo de sistema**.
3. **Duración:** El proceso de compresión (gzip) lleva activo desde las **09:47** y ya ha acumulado **8 minutos y 56 segundos** de tiempo de CPU. Esto se alinea perfectamente con que la carga es alta al inicio (que fue a las 09:47 en el informe inicial de top).

Siguientes Pasos: Encontrar el Script

Dado que el proceso fue iniciado por **root** y se inició automáticamente poco después del arranque del sistema (o a una hora programada), el siguiente paso es averiguar qué script o servicio ejecutó el shell (/bin/sh) con ese comando.

Obtener Más Detalles del Proceso Padre

Podemos usar el comando ps para ver qué proceso es el padre del PID 1334 (el shell) y obtener la línea de comando completa de ese padre:

```
ps -fp 1334
```

Esta salida es crítica. Podría mostrar algo como:

- /usr/sbin/cron (si fue una tarea programada)
- /usr/bin/logrotate (si es la rotación de logs)
- Un script de inicio específico en /etc/init.d/ o /etc/systemd/

Salida de ps -fp:

```
ps -fp 1334

UID          PID    PPID  C STIME TTY          TIME CMD

root        1334    1329  0 09:47 ?        00:00:00 /bin/sh -c gzip
```

El comando **ps -fp 1334** nos ha revelado la última pieza del rompecabezas: el **proceso padre (PPID)** es el **1329.**

Campo	**Valor**	**Significado**
PID (Proceso Shell)	1334	El shell (/bin/sh) que lanzó el proceso gzip.
PPID (Proceso Padre)	**1329**	El proceso que lanzó el shell (1334).

Ahora solo necesitamos identificar el proceso con **PID 1329.** Este proceso es el **origen real** de la alta carga de CPU.

Identificando la Fuente Real del Problema

Ejecutemos el comando ps -fp 1329 para ver el proceso padre:

```
ps -fp 1329

salida:
UID          PID    PPID  C STIME TTY          TIME CMD
```

La ejecución del comando ps -fp 1329 no ha proporcionado la información que esperábamos para ese **PID** (Proceso Padre).

La salida incompleta:

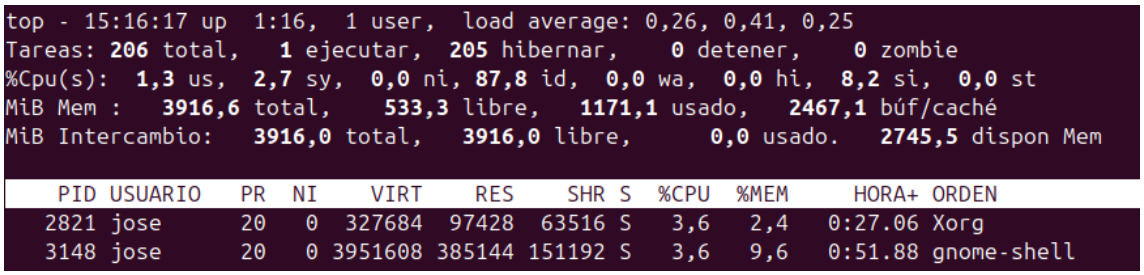
```
UID          PID    PPID  C STIME TTY          TIME CMD
```

...sin una línea de proceso debajo, significa que el proceso con **PID 1329 ya terminó** antes de que pudiera consultarlo con ps.

Esto es habitual cuando los trabajos finalizan rápidamente después de lanzar su tarea. El proceso padre (1329) lanzó el shell (1334), el shell lanzó el gzip (1335), y luego el proceso padre (1329) se cerró inmediatamente. Sin embargo, el proceso hijo (gzip, PID **1335**) **sigue ejecutándose** porque está comprimiendo un archivo grande.

load average

En un servidor con **4 cores**, un load de 4.0 ≈ todos los núcleos ocupados.
Si tuvieras **2 cores**, un load de 4.0 = sobrecarga (cola de procesos esperando).



top - 15:15:59 up 1:16, 1 user, load average: 0,36, 0,44, 0,25

Interpretación línea por línea

- **Hora actual**: 15:15:59 → la hora del sistema.
- **Tiempo encendido**: up 1:16 → el servidor lleva 1 hora y 16 minutos funcionando.
- **Usuarios conectados**: 1 user → solo hay un usuario logueado.
- **Load average**: 0,36 0,44 0,25 → carga media en los últimos 1, 5 y 15 minutos.

Cómo interpretar la *load average*

- Son **promedios de procesos en cola** (esperando CPU o I/O).
- Se comparan con el número de **núcleos de CPU** que tiene tu máquina.

Ejemplo práctico:

- Si tu VM tiene **2 cores**:
  - Load 0.36 → menos de medio núcleo ocupado → sistema muy tranquilo.
  - Load 0.44 (5 min) y 0.25 (15 min) → también muy bajos.

    - Conclusión: la máquina está **sobrada de recursos**.
- Si tuvieras **4 cores**, todavía más claro: 0.36 sobre 4 = apenas un 9% de carga global.

Regla rápida

- **Load ≈ núm. de cores** → sistema al 100% de uso, pero no saturado.
- **Load > núm. de cores** → hay procesos esperando → posible cuello de botella.
- **Load < núm. de cores** → sistema relajado.

En nuestro caso:
Con 0,36 0,44 0,25, tu servidor está **muy desahogado**. No hay saturación ni procesos en cola.

Regla de oro para la Load Average

- Son **promedios de procesos en cola** en 1, 5 y 15 minutos.
- Se interpretan **comparando con el número de núcleos**:
    - 4 cores, load 2.0 → el sistema va sobrado.
    - 4 cores, load 4.0 → todos los núcleos ocupados (saturación).
    - 4 cores, load 8.0 → el doble de procesos esperando → cuello de botella.

2) Herramientas rápidas: instalación (Debian/Ubuntu y RHEL/CentOS)

```
# Debian/Ubuntu

sudo apt update
sudo apt install -y htop iotop net-tools procps sysstat

# RHEL/CentOS (con dnf/yum)

sudo dnf install -y htop iotop net-tools procps-ng sysstat
# o
```

sudo yum install -y htop iotop net-tools procps-ng sysstat

netstat viene en net-tools. En sistemas modernos ss (parte de iproute2) es la alternativa preferida.

3) top — monitorización básica (instalado por defecto)

Uso básico

top

Dentro de top (modo interactivo):

- P ordenar por CPU.
- M ordenar por memoria.
- 1 mostrar cada CPU por separado.
- u filtrar por usuario.
- k matar PID.

Comando no interactivo (snapshot):

top -b -n1 | head -n 20

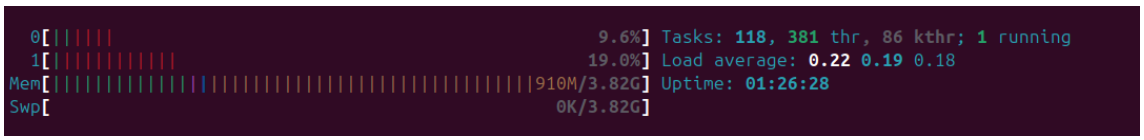
Ejemplo:

muestra top, presiona M y observa la columna %MEM, RES, VIRT, COMMAND.

4) htop — versión mejorada e interactiva

Arrancar

htop



1. Barras de CPU

- Cada núcleo aparece con su propia barra (CPU0, CPU1, etc.).
- Los colores indican el tipo de carga:
  - **Verde** → procesos de usuario.
  - **Rojo** → procesos del kernel.
  - **Azul** → procesos con baja prioridad (nice).
  - **Naranja/amarillo** → tiempo de espera de I/O (esperando disco).
- Ejemplo: si ves mucho rojo → el kernel está trabajando duro (drivers, interrupciones).

En nuestra captura: tenemos **2 cores** y ambos están con carga ligera, nada saturado.

2. **Memoria (Mem)**

- Barra horizontal que muestra la RAM usada frente al total.
- Ejemplo: 1940M/3.82G → estás usando ~1,9 GB de 3,8 GB disponibles.
- Colores:
  - **Verde** → memoria usada por procesos.
  - **Azul** → buffers.
  - **Amarillo** → caché.
- Importante: Linux usa RAM libre como caché, así que no te asustes si parece "llena".

3. **Swap**

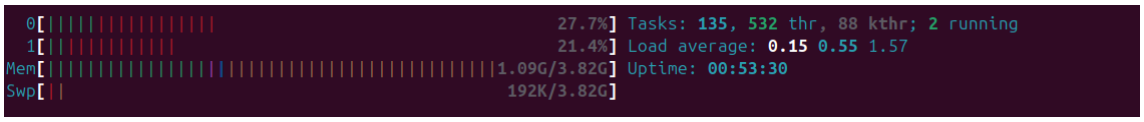
- Muestra el uso de la memoria de intercambio en disco.
- Ejemplo: 0K/3.82G → no estás usando nada de swap (perfecto).
- Si empieza a crecer mucho → la RAM se queda corta y el sistema puede volverse lento.

4. **Información del sistema (cabecera)**

- **Tasks**: número de procesos y threads activos.
  - Ejemplo: 118 total, 381 threads, 86 kernel threads, 1 running.
  - → Tienes 118 procesos, de los cuales solo 1 está ejecutándose en ese instante; el resto está en espera.

- **Load average**: 0.12, 0.49, 0.18 → carga en 1, 5 y 15 minutos.
    - Con 2 cores, un load de 0.12 es bajísimo → sistema relajado.
- **Uptime**: 01:26:43 → el sistema lleva 1h 26m encendido.

En este caso:



Resumen del Sistema (2 Cores)

Campo	Valor	Significado
Uptime	00:53:30	El sistema ha estado funcionando durante **53 minutos y 30 segundos.**
Tasks	135, 532 thr, 88 kthr, 2 running	Hay **135 procesos** totales, gestionando 532 hilos, con solo **2 procesos/hilos** activos o listos para ejecutarse (running).

Uso de CPU (2 Cores)

Los valores de la CPU (Q y L) confirman que se trata de un sistema de 2 núcleos (Core 0 y Core 1).

Core	Valor	Significado
Core 0	27.7%	El primer núcleo está al **27.7%** de uso.
Core 1	21.4%	El segundo núcleo está al **21.4%** de uso.
Total Estimado	$\approx 49.1\%$	El consumo total de CPU es **bajo** a **moderado.**

Carga Promedio (Load Average)

Período	Valor	Interpretación (para 2 Cores)
1 minuto	**0.15**	**Muy baja** (casi inactiva). Solo 0.15 procesos esperan la CPU.
5 minutos	**0.55**	**Baja.**
15 minutos	**1.57**	**Baja** a **Moderada.**

- **Punto de Referencia:** En un sistema de 2 núcleos, cualquier carga **por debajo de 2.00** indica que hay suficiente capacidad de procesamiento para manejar la demanda sin cuellos de botella.
- **Conclusión:** El sistema está actualmente **muy tranquilo** y tiene mucha capacidad sobrante.

Uso de Memoria (RAM y Swap)

Recurso	Total	Usado (aproximado)	Porcentaje de Uso
RAM (Mem)	3.82G	1.09G	$\approx 28.5\%$
Swap	3.82G	192K	**Mínimo** (casi cero)

- **RAM:** Solo se está utilizando $\approx **1.09 GB**$ de los $3.82 GB$ disponibles, lo que es un uso **muy bajo**.
- **Swap:** El uso de **Swap** es **prácticamente nulo** ($192 KB$), lo que confirma que el sistema tiene suficiente RAM y no necesita usar el disco como memoria virtual.

Resumen General del Estado

El sistema se encuentra en un estado **excelente** y **saludable**:

- **Baja Carga:** La carga promedio es muy baja, lo que indica que el sistema no está estresado.
- **CPU Disponible:** Ambos núcleos tienen mucha capacidad libre (alrededor del 70%-78% libre).
- **Buena Memoria:** El sistema tiene mucha memoria RAM libre y casi no está usando el espacio de intercambio (Swap).

En contraste con el informe anterior donde la carga era 8.28, este sistema está actualmente **subutilizado** y funcionando de manera óptima.

5. **Lista de procesos (parte inferior)**

- Cada fila es un proceso.
- Columnas típicas:
  - **PID** → identificador del proceso.
  - **USER** → usuario dueño del proceso.
  - **PRI/NI** → prioridad y "nice" (ajuste de prioridad).
  - **VIRT/RES/SHR** → memoria virtual, residente y compartida.
  - **S** → estado (R = running, S = sleeping, Z = zombie).
  - **%CPU / %MEM** → consumo de CPU y RAM.
  - **TIME+** → tiempo total de CPU usado.
  - **COMMAND** → nombre del proceso.

Aquí es donde cazas a los "glotones" de CPU o RAM.

Resumen visual

- **Arriba** → estado global (CPU, RAM, Swap, carga, uptime).
- **Abajo** → detalle por proceso (quién consume qué).

Atajos útiles:

- F6 ordenar (cpu, mem, time).
- F3 buscar proceso.
- F9 matar proceso (seleccionar señal).
- F2 configurar columnas.

Ejercicio

1. sudo apt install htop (si no está).

   htop -u root (ver procesos de root).

2. Usa F5 para mostrar árbol y observa que los procesos hijos heredan recursos.

5) iotop — quién está haciendo I/O a disco

Instalación

```
sudo apt install iotop
# Debian/Ubuntu

sudo dnf install iotop
# RHEL/CentOS
```

Uso

```
sudo iotop
# interactivo

sudo iotop -o -P
# -o muestra solo procesos con I/O activo, -P muestra PIDs

sudo iotop -b -n5
#El comando sudo iotop -b -n5 generará una salida no interactiva (batch mode) que muestra las estadísticas de uso de Entrada/Salida de Disco (I/O) por proceso, y se actualizará 5 veces antes de terminar.
```

Componente	Significado	Explicación
sudo	**Superuser Do**	Se requiere para ejecutar iotop ya que necesita privilegios de *root* para acceder a la información de I/O del *kernel*.

Componente	Significado	Explicación
iotop	**I/O Top**	La utilidad que rastrea el uso de disco (I/O) por cada proceso o hilo en el sistema.
-b	**Batch Mode**	Le dice a iotop que se ejecute en modo *batch* (por lotes) en lugar de en modo interactivo. Esto significa que **no se refrescará en la pantalla continuamente** como lo hace top o htop, sino que imprimirá la información una vez por iteración y luego saldrá, lo cual es ideal para *scripts* o para recopilar datos de forma limpia.
-n5	**Iterations**	Le dice a iotop que tome **5 muestras** (o iteraciones) antes de salir. Por defecto, cada muestra suele tener un segundo de retraso, por lo que el comando tardará unos 5 segundos en completarse.

Salida Generada por iotop -b -n5

La salida será una tabla que se imprime 5 veces consecutivas (una tabla por iteración), cada una mostrando el consumo de I/O durante el intervalo de tiempo anterior. La tabla tendrá un formato similar a este:

```
Total DISK READ:       [Valor_Total_Lectura] KB/s | Total DISK
WRITE:      [Valor_Total_Escritura] KB/s
Actual DISK READ:      [Valor_Actual_Lectura] KB/s | Actual DISK
WRITE:     [Valor_Actual_Escritura] KB/s
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>
COMM
    1 be/4 root        0.00 K/s    0.00 K/s  0.00 %  0.00 % init
 1234 be/4 root      123.45 K/s  200.00 K/s  0.00 % 45.00 % gzip
archivo_grande.log
 5678 be/4 jose        0.00 K/s    0.00 K/s  0.00 %  0.00 %
chrome --type=renderer
...
```

1. **Totales Globales:**
    - **Total DISK READ/WRITE**: El ancho de banda total del disco leído y escrito por **todos** los procesos en el sistema.
    - **Actual DISK READ/WRITE**: El ancho de banda actual que se está utilizando, excluyendo las escrituras que son gestionadas por la caché (lo que refleja un uso más real del disco físico).

2. **Estadísticas por Proceso/Hilo:** La tabla principal desglosa el uso por proceso (o hilo).
    - **TID**: ID del Hilo (Thread ID) o del Proceso (si está en modo de procesos, -P).
    - **USER**: El usuario que posee el proceso.
    - **DISK READ / DISK WRITE**: La **velocidad** de lectura y escritura del disco del proceso, generalmente en KB/s o MB/s. **Esto es lo más importante** para identificar a los "culpables".
    - **IO>**: El **porcentaje de tiempo** que el proceso está esperando I/O (bloqueado en llamadas al disco). Un valor alto aquí (>50%) indica que el proceso está **limitado por la velocidad del disco.**

Aplicación:

Este comando es ideal para capturar exactamente el proceso **gzip** que identificamos previamente. Si lo ejecuta durante un pico de alta carga de disco:

1. Verá en las 5 iteraciones cómo **gzip** domina las columnas **DISK READ** y/o **DISK WRITE.**
2. Le dará valores cuantificables (KB/s o MB/s) para saber exactamente qué tan rápido está comprimiendo ese archivo.

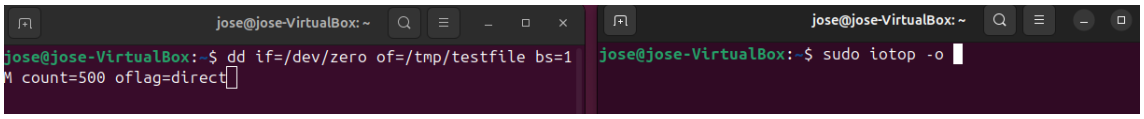
Demostración práctica: genera I/O de prueba y observa:

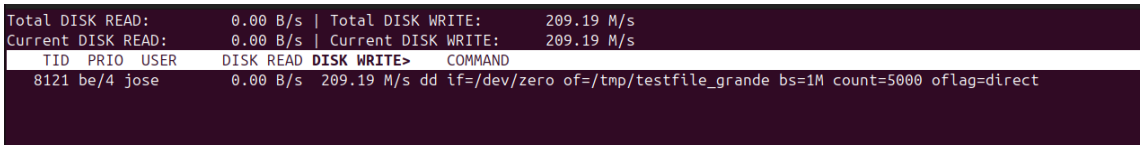
```
# prueba (en otra terminal)

dd if=/dev/zero of=/tmp/testfile bs=1M count=500 oflag=direct

o para tener más tiempo:
dd if=/dev/zero of=/tmp/testfile_grande bs=1M count=5000 oflag=direct

# observa en iotop qué PID escribe
```

Nota: iotop necesita contabilidad I/O activada en el kernel; si no funciona en una distro, usa iotop con permisos o usa pidstat -d.

Ejercicio

1. Ejecuta sudo iotop -o y, en otra consola, lanza el dd anterior.
2. Identifica el PID que genera la escritura.
   (Respuesta esperada: el PID del proceso dd aparece con alto WRITES.)





Campo	Valor	Interpretación
Total DISK WRITE	**209.19 M/s**	Es la velocidad total de escritura que el sistema está experimentando. Este es un valor **muy alto**, típico de una máquina virtual usando un SSD o una configuración optimizada.
Current DISK WRITE	**209.19 M/s**	Muestra que toda la actividad de escritura actual está yendo directamente al disco físico, sin depender de la caché.
TID	**8121**	El ID del Hilo/Proceso (dd) que está causando la actividad.

Campo	Valor	Interpretación
DISK WRITE	**209.19 M/s**	**El proceso dd es el único responsable** de toda la escritura del disco, usando 209.19 M/s.
COMMAND	dd if=/dev/zero of=/tmp/testfile_grande bs=1M count=5000 oflag=direct	Esta es la línea de comando que está generando la escritura.

1. **Proceso Identificado:** El proceso con **TID 8121** (dd) es el que está generando la actividad de I/O de disco.
2. **Actividad Comprobada:** Se confirma que el comando **dd** está escribiendo a una velocidad de **más de 200 MB/s** en el disco.
3. **Objetivo Cumplido:** La salida demuestra cómo la herramienta iotop permite identificar, de manera precisa y cuantitativa, qué proceso (PID/TID) es el causante de una alta carga de I/O de disco (wa o io> alto).

Esto valida la técnica que usaríamos para identificar al culpable del anterior pico de carga, que era el proceso **gzip**.

6) du y df — uso de disco

df — resumen de sistemas de ficheros

```
df -hT
# -h human readable, -T mostrar tipo de FS

df -h /var
# espacio usado en /var

df -i
# uso de inodos
```

du — cuánto ocupa cada directorio

```
sudo du -sh /var/log
# tamaño de /var/log

sudo du -ah /var | sort -rh | head -n 20
# 20 archivos/directorios más grandes (human)

sudo du --max-depth=1 -h /home
# resumen por subdirectorios (nivel 1)
```

Herramienta interactiva:

ncdu (muy útil) — instala ncdu y ejecuta ncdu /.

```
sudo apt install ncdu

sudo ncdu /
```

ncdu (NCurses Disk Usage) es una herramienta excelente e interactiva para navegar y analizar el uso del disco de forma rápida, lo cual es perfecto para un escenario de liberar espacio.

Para escanear **todo el sistema de archivos** (iniciando desde la raíz /), debes usar sudo porque necesita permisos para acceder a todos los directorios (incluidos /root, /var/log, etc.):

```
sudo ncdu /
```

Consejos de Uso de ncdu

1. **Escaneo Inicial:** La primera vez que lo ejecutes en /, tardará unos segundos (o minutos, dependiendo del tamaño de su disco) en escanear todo y calcular los tamaños.
2. **Navegación:**
    - Usa las **flechas de dirección (arriba/abajo)** para moverte entre directorios y archivos.
    - Presiona **Enter** para entrar en un directorio.
    - Presiona **Punto y coma** (.. o la flecha izquierda) para retroceder.
3. **Identificación de Archivos Grandes:**
    - Los directorios y archivos se **ordenan automáticamente por tamaño** (los más grandes arriba).
    - Busca cualquier directorio gigante que consuma la mayor parte del espacio.
4. **Eliminar Archivos:**

- Selecciona un archivo que quieras borrar (¡ten cuidado!).
- Presions la tecla **d** (de *delete*). ncdu te pedirá confirmación. **Solo borra archivos que sepas que son seguros de eliminar.**

5. **Ayuda:**
   - Presiona **?** para ver la lista completa de comandos y atajos de teclado.

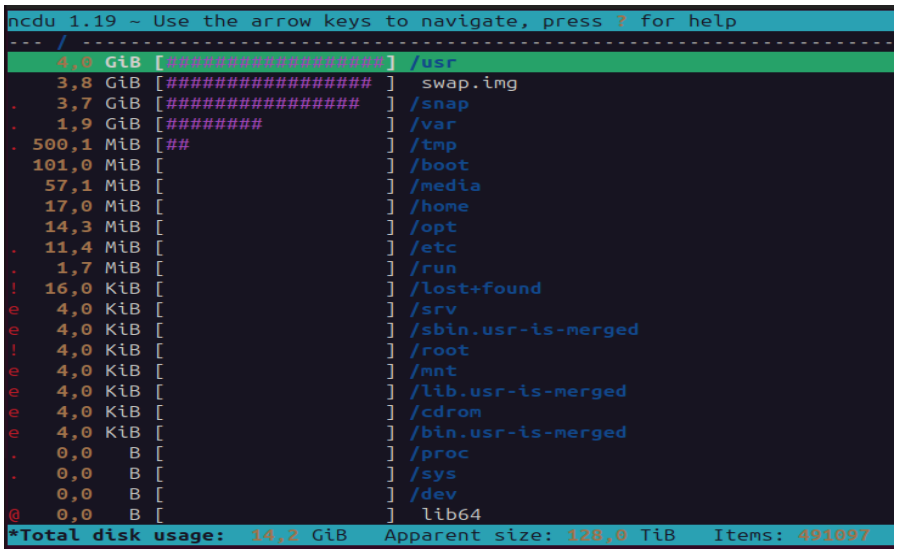
ncdu te permite visualizar gráficamente (con la barra de porcentaje a la izquierda) dónde se está yendo su espacio libre de 6.0 GB.

Ejercicio

1. Encuentra las 10 rutas que más ocupan bajo /var:

```
sudo du -ah /var | sort -rh | head -n 10
```

2. Anota el mayor culpable



7) netstat y ss — conexiones y puertos

Instalación (si hace falta)

```
sudo apt install net-tools
# para netstat
```

netstat ejemplos

```
# conexiones TCP activas
sudo netstat -antp
```

```
jose@jose-VirtualBox:~$ sudo netstat -antp
Conexiones activas de Internet (servidores y establecidos)
Proto  Recib Enviad Dirección local         Dirección remota       Estado       PID/Program name
tcp        0      0 127.0.0.1:631           0.0.0.0:*              ESCUCHAR     1160/cupsd
tcp        0      0 127.0.0.54:53           0.0.0.0:*              ESCUCHAR     373/systemd-resolve
tcp        0      0 127.0.0.53:53           0.0.0.0:*              ESCUCHAR     373/systemd-resolve
tcp6       0      0 ::1:631                 :::*                   ESCUCHAR     1160/cupsd
```

```
# puertos escuchando y el proceso
sudo netstat -tulnp
```

```
jose@jose-VirtualBox:~$ sudo netstat -tulnp
Conexiones activas de Internet (solo servidores)
Proto  Recib Enviad Dirección local         Dirección remota       Estado       PID/Program name
tcp        0      0 127.0.0.1:631           0.0.0.0:*              ESCUCHAR     1160/cupsd
tcp        0      0 127.0.0.54:53           0.0.0.0:*              ESCUCHAR     373/systemd-resolve
tcp        0      0 127.0.0.53:53           0.0.0.0:*              ESCUCHAR     373/systemd-resolve
tcp6       0      0 ::1:631                 :::*                   ESCUCHAR     1160/cupsd
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           618/avahi-daemon: r
udp        0      0 127.0.0.54:53           0.0.0.0:*                           373/systemd-resolve
udp        0      0 127.0.0.53:53           0.0.0.0:*                           373/systemd-resolve
udp        0      0 0.0.0.0:45370           0.0.0.0:*                           618/avahi-daemon: r
udp6       0      0 :::60460                :::*                                618/avahi-daemon: r
udp6       0      0 :::5353                 :::*                                618/avahi-daemon: r
jose@jose-VirtualBox:~$
```

```
# estadísticas de protocolo
netstat -s
```

ss (más moderno, recomendado)

```
# listar puertos TCP/UDP escuchando con programa
sudo ss -tulpen
```

```
jose@jose-VirtualBox:~$ sudo ss -tulpen
Netid         State          Recv-Q         Send-Q                 Local Address:Port                 Peer Address:Port
Process
udp           UNCONN         0              0                          0.0.0.0:5353                      0.0.0.0:*
 users:(("avahi-daemon",pid=618,fd=12)) uid:108 ino:7820 sk:1 cgroup:/system.slice/avahi-daemon.service <->
udp           UNCONN         0              0                        127.0.0.54:53                       0.0.0.0:*
 users:(("systemd-resolve",pid=373,fd=16)) uid:991 ino:6555 sk:2 cgroup:/system.slice/systemd-resolved.service <->
udp           UNCONN         0              0                     127.0.0.53%lo:53                       0.0.0.0:*
 users:(("systemd-resolve",pid=373,fd=14)) uid:991 ino:6553 sk:3 cgroup:/system.slice/systemd-resolved.service <->
udp           UNCONN         0              0                          0.0.0.0:45370                     0.0.0.0:*
 users:(("avahi-daemon",pid=618,fd=14)) uid:108 ino:7822 sk:4 cgroup:/system.slice/avahi-daemon.service <->
udp           UNCONN         0              0                             [::]:60460                        [::]:*
 users:(("avahi-daemon",pid=618,fd=15)) uid:108 ino:7823 sk:1001 cgroup:/system.slice/avahi-daemon.service v6only:1 <->
udp           UNCONN         0              0                             [::]:5353                         [::]:*
 users:(("avahi-daemon",pid=618,fd=13)) uid:108 ino:7821 sk:1002 cgroup:/system.slice/avahi-daemon.service v6only:1 <->
tcp           LISTEN         0              4096                     127.0.0.1:631                       0.0.0.0:*
 users:(("cupsd",pid=1160,fd=7)) ino:9833 sk:1003 cgroup:/system.slice/cups.service <->
tcp           LISTEN         0              4096                   127.0.0.54:53                         0.0.0.0:*
 users:(("systemd-resolve",pid=373,fd=17)) uid:991 ino:6556 sk:1004 cgroup:/system.slice/systemd-resolved.service <->
tcp           LISTEN         0              4096                127.0.0.53%lo:53                         0.0.0.0:*
 users:(("systemd-resolve",pid=373,fd=15)) uid:991 ino:6554 sk:1005 cgroup:/system.slice/systemd-resolved.service <->
tcp           LISTEN         0              4096                    [::1]:631                         [::]:*
 users:(("cupsd",pid=1160,fd=6)) ino:9832 sk:1006 cgroup:/system.slice/cups.service v6only:1 <->
jose@jose-VirtualBox:~$
```

```
# conexiones establecidas (tcp)
ss -s
# resumen

ss -tn state established
```

```
jose@jose-VirtualBox:~$ ss -s
Total: 741
TCP:   4 (estab 0, closed 0, orphaned 0, timewait 0)

Transport Total     IP        IPv6
RAW       0         0         0
UDP       7         5         2
TCP       4         3         1
INET      11        8         3
FRAG      0         0         0
```

Diferencia entre *LISTEN* y *ESTABLISHED* y cómo identificar procesos que exponen puertos (PID/Programa).

Estados de red en TCP

LISTEN

- Significa que un proceso está **esperando conexiones entrantes** en un puerto.
- Ejemplo típico:
  - sshd en el puerto 22 → está en estado **LISTEN** esperando que alguien intente conectarse por SSH.
- Es como un teléfono sonando en espera de que alguien descuelgue.

ESTABLISHED

- Indica que ya existe una **conexión activa** entre cliente y servidor.
- Ejemplo:
  - Si te conectas por SSH a tu servidor, verás una entrada en **ESTABLISHED** entre tu IP y el puerto 22 del servidor.
- Es como cuando ya contestaste la llamada y estás hablando.

Resumen rápido:

- **LISTEN** = esperando conexiones.
- **ESTABLISHED** = conexión activa en curso.

Cómo identificar procesos que exponen puertos

1. Con ss (reemplazo moderno de netstat)

```
sudo ss -ltnp
```

- -l → solo sockets en escucha (LISTEN).
- -t → TCP.
- -n → mostrar números (no resolver nombres).
- -p → mostrar PID y programa.

Ejemplo de salida:

```
LISTEN 0 128 *:22   *:*   users:(("sshd",pid=1234,fd=3))
```

→ El proceso sshd (PID 1234) está escuchando en el puerto 22.

2. Con lsof

```
sudo lsof -i -P -n | grep LISTEN
```

Ejemplo:

```
sshd   1234 root   3u  IPv4  12345  0t0  TCP *:22 (LISTEN)
```

→ Igual: sshd está en LISTEN en el puerto 22.

3. Con netstat (si lo tienes instalado)

```
sudo netstat -tulnp
```

Ejemplo:

```
tcp   0   0 0.0.0.0:80   0.0.0.0:*   LISTEN   5678/nginx
```

→ El proceso nginx (PID 5678) escucha en el puerto 80.

Ejemplo

1. **LISTEN**:

   tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 1234/sshd

   → sshd está esperando conexiones SSH.

2. **ESTABLISHED**:

   tcp 0 0 192.168.1.10:22 192.168.1.50:54321 ESTABLISHED 1234/sshd

   → Un cliente en 192.168.1.50 ya está conectado por SSH.

Con esto puedes:

- Ver **qué servicios están expuestos** (LISTEN).
- Ver **qué conexiones están activas** (ESTABLISHED).
- Identificar **qué proceso/PID** está detrás de cada puerto.

Ejercicio

sudo ss -tulpen | less

identifica el proceso que escucha en el puerto 80/443 y su usuario.

identifica el proceso que escucha en el puerto 8125 y su usuario.

Puerto	Protocolo	Dirección	Proceso	UID (Usuario)
8125	UDP y TCP	127.0.0.1	**netdata**	uid:122
53	UDP y TCP	127.0.0.54 o 127.0.0.53%lo	**systemd-resolve**	uid:991
41136 / 5353	UDP	0.0.0.0 / [::]	**avahi-daemon**	uid:108
19999	TCP	0.0.0.0	**netdata**	(el mismo que 8125)

Puerto	Protocolo	Dirección	Proceso	UID (Usuario)
631	TCP	127.0.0.1	**cupsd**	(sin UID, pero es el proceso del servicio cups.service)

- **Puertos 8125 y 19999:** Utilizados por **netdata** (un sistema de monitorización de rendimiento en tiempo real).
- **Puerto 53:** Es el puerto estándar para **DNS** (Sistema de Nombres de Dominio), utilizado aquí por systemd-resolve (el resolvedor de nombres local). Las direcciones son de *loopback* (127.0.0.x), por lo que solo son accesibles localmente.
- **Puertos 41136 y 5353:** Utilizados por **avahi-daemon** para el descubrimiento de servicios de red locales (mDNS/DNS-SD).
- **Puerto 631:** Utilizado por **cupsd** (Common Unix Printing System), el servicio de impresión.

8) Integrando herramientas — ejemplos

Caso: El servicio va lento

1. top → ¿cpu alta?
2. iotop → ¿I/O alto?
3. df -h / du → ¿disco lleno?
4. ss -tn → ¿conexiones establecidas muy altas?
5. htop → identificar proceso culpable, strace si necesario (avanzado).

9) Instalar y configurar Netdata — monitorización en tiempo real (UI web)

Netdata es un monitor ligero con UI web en tiempo real (gráficas por segundo) — muy útil para demos.

Instalación recomendada (un-liner oficial)

Nota: este es el método más sencillo para entornos de demostración. En producción, sigue la guía oficial de tu distro.

```
# Método rápido (script de instalación oficial)
bash <(curl -Ss https://get.netdata.cloud/kickstart.sh -o kickstart.sh)
```

Opcional: instalar desde paquetes si tu repo lo provee:

```
sudo apt install curl -y
```

Descarga el script primero y ejecútalo

En lugar de usar <(curl …), hazlo en dos pasos para evitar problemas:

```
curl -sSL https://get.netdata.cloud/kickstart.sh -o kickstart.sh

sudo bash kickstart.sh
```

Iniciar / habilitar

```
sudo systemctl enable --now netdata

# comprobar estado

sudo systemctl status netdata
```

Acceso

- Abre en el navegador: http://localhost:19999
- Por defecto Netdata sirve en el puerto **19999**.

Configuración básica

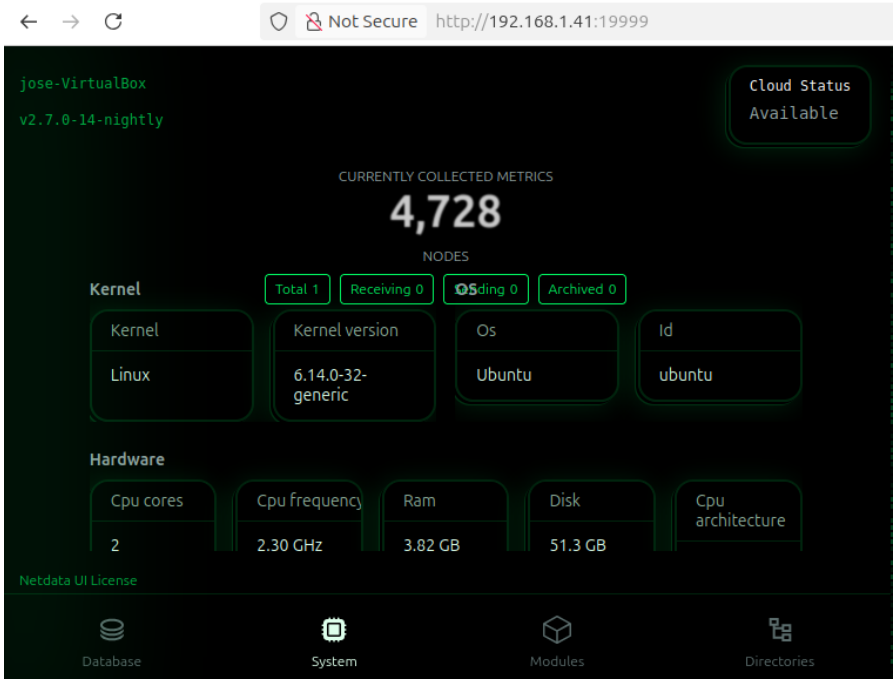
```
sudo nano /etc/netdata/netdata.conf

# busca la sección [web] y ajusta bind/puerto si necesitas limitar acceso
# tras cambios:

sudo systemctl restart netdata
```

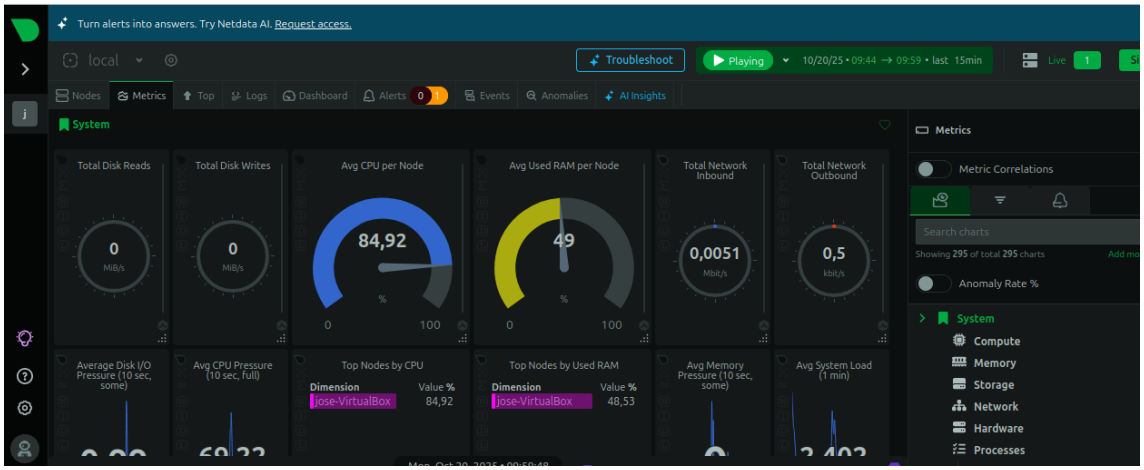
Seguridad / buenas prácticas

- Para exposiciones públicas: usa un proxy inverso (nginx) con HTTPS y autenticación, o limita acceso por firewall / VPN.
- Para demo: puedes tunelar con ssh -L 19999:localhost:19999 user@server y abrir http://localhost:19999.

Ejercicio Netdata

1. Instala Netdata en una VM.
2. Genera carga CPU (stress o dd) y observa la gráfica de CPU en Netdata.
3. Explica lo que muestra (picos, CPU user vs system).





Al cargar http://localhost:19999/ (que es la interfaz web por defecto de **Netdata**), deberías centrarte en los paneles que proporcionan una visión general inmediata de la salud y el rendimiento del sistema.

System Overview (Vista General del Sistema)

Este es el primer panel que aparece y ofrece un resumen vital:

- **CPU:** Observa la gráfica de uso general, especialmente la distinción entre el uso del sistema (system), del usuario (user) y el tiempo de espera de I/O (**iowait**). Un **iowait** alto indica que la CPU está esperando datos del disco (problema de disco o I/O).
- **Memoria:** Revisa el uso de RAM. Es crucial vigilar la memoria **libre** y, más importante, la memoria **usada** y el **swap** (intercambio). Si el uso de **swap** aumenta constantemente, tu sistema se está quedando sin RAM física.
- **Carga del Sistema (Load Average):** Comprueba los valores de 1, 5 y 15 minutos. Si son consistentemente más altos que el número de núcleos de tu CPU, el sistema está sobrecargado.
- **Espacio en Disco (Disks):** Asegúrate de que no haya particiones (especialmente la raíz /) cerca de su capacidad máxima.

Networking (Red)

Si tu servidor maneja tráfico de red, revisa:

- **Ancho de Banda (Bandwidth):** Mira las tasas de tráfico de red de entrada (in) y salida (out) para ver si hay picos inesperados.
- **Errores y Descartes de Red (Errors/Drops):** Una alta cantidad de paquetes descartados (drops) o errores puede indicar problemas con la tarjeta de red, el cableado o la configuración.

Disks (Discos)

Esta sección es fundamental para el rendimiento de I/O:

- **Latencia de I/O (I/O Time, o r/s y w/s):** Observa la cantidad de lecturas y escrituras por segundo. Es más importante revisar el **tiempo de I/O** (cuánto tiempo está ocupado el disco). Valores altos (>50ms) pueden causar cuellos de botella.
- **Utilización (Utilization):** Indica el porcentaje de tiempo que el disco ha estado ocupado sirviendo solicitudes. Un 100% de utilización significa que el disco está trabajando al máximo de su capacidad.

Processes (Procesos)

Netdata agrupa métricas por procesos. Esto es lo que debes buscar:

- **Top CPU Users:** Identifica qué procesos están consumiendo la mayor parte de la CPU. Si un proceso inesperado (como un php-fpm o un apache2 con uso anómalamente alto) aparece aquí, es una señal de problema.
- **Top Memory Users:** Identifica qué procesos están usando más memoria. Esto es útil para detectar pérdidas de memoria (memory leaks).

Applications (Aplicaciones)

Si tienes instaladas y configuradas las integraciones (como Apache, Nginx, MySQL/PostgreSQL, etc.), mira:

- **Servidor Web (Apache/Nginx):**
  - **Peticiones por Segundo (Requests/s):** Para medir la carga de trabajo.
  - **Workers Activos/Inactivos:** Para asegurar que tu servidor web tiene suficientes *threads* para manejar el tráfico.
  - **Códigos de Estado (Status Codes):** Revisa las peticiones con código **4xx** (errores del cliente) y **5xx** (errores del servidor). Un aumento en los errores **5xx** indica un problema grave en el servidor o la aplicación.
- **Bases de Datos (MySQL/PostgreSQL):**
  - **Conexiones:** Asegúrate de que no estás llegando al límite de conexiones permitidas.
  - **Consultas Lentas (Slow Queries):** Identifica si la base de datos está atascada procesando consultas ineficientes.

En resumen:

La clave de Netdata es la **monitorización en tiempo real.** Busca anomalías o desviaciones en los patrones normales: picos en el uso de CPU/Memoria/Iowait, aumento de errores de red o errores 5xx en el servidor web.

¿Qué es stress?

Es un programa que **lanza procesos artificiales** que consumen CPU, memoria, disco o I/O, para poner el sistema bajo carga. Se usa en pruebas de rendimiento, detección de cuellos de botella o para comprobar la estabilidad de un servidor.

Opciones principales

- **-c, --cpu N** → genera **N procesos que consumen CPU** (haciendo cálculos matemáticos).
  Ejemplo:

  stress --cpu 2 --timeout 30s

  → 2 procesos saturando CPU durante 30 segundos.

- **-i, --io N** → genera **N procesos de I/O** (operaciones de entrada/salida).
  Ejemplo:

  stress --io 4 --timeout 20s

  → 4 procesos haciendo operaciones de disco/memoria.

- **-m, --vm N** → genera **N procesos que reservan memoria** (malloc/free).
  - Con --vm-bytes B defines cuánto ocupa cada uno.
    Ejemplo:

  stress --vm 2 --vm-bytes 512M --timeout 15s

  → 2 procesos, cada uno reservando 512 MB de RAM durante 15 segundos.

- **-d, --hdd N** → genera **N procesos que escriben y borran archivos** (simula carga de disco).
  - Con --hdd-bytes B defines el tamaño de cada archivo.
    Ejemplo:

  stress --hdd 1 --hdd-bytes 2G --timeout 10s

  → 1 proceso escribiendo/borrando archivos de 2 GB durante 10 segundos.

- **-t, --timeout N** → duración de la prueba (ej. 10s, 2m, 1h).

Ejemplo completo

Consola 1:
stress --cpu 4 --io 2 --vm 2 --vm-bytes 256M --timeout 30s

Consola 2:
htop

- 4 procesos de CPU.
- 2 procesos de I/O.
- 2 procesos de memoria (256 MB cada uno).
- Todo durante 30 segundos.

Esto simula un escenario de carga mixta (CPU + I/O + RAM).

Precauciones

- stress **no es un benchmark**, solo genera carga.
- Puede ralentizar mucho tu VM o incluso colgarla si pides más recursos de los que tiene.
- Úsalo en entornos de prueba, no en producción.

Resumen rápido:

- **CPU** → --cpu N
- **I/O** → --io N
- **Memoria** → --vm N --vm-bytes X
- **Disco** → --hdd N --hdd-bytes X
- **Tiempo** → --timeout N

10) Instalar y configurar Glances — monitor más compacto, CLI + web

Qué es:

Glances es un monitor en consola con modo web (glances -w) útil para demos rápidas.

Instalación

Problema	Comando	Solución
Error de Sintaxis	glances -w 0.0.0.0	Se corrigió el comando para usar la opción de *Bind* correcta: **glances -w -B 0.0.0.0.**
Bloqueo de Red	sudo ufw allow 61208/tcp	**Se abrió el puerto 61208** en el firewall (UFW) para permitir el acceso externo.
Glances No Accesible	`sudo ss -tuln	grep 61208`	

Resolución de Dependencias Faltantes (Página en Blanco)

La página en blanco fue causada por la falta del archivo glances.js, debido a una instalación incompleta. Para corregirlo, se pasó de apt a una instalación moderna y aislada con pipx.

Problema	Comando de Instalación	Dependencia Faltante
Preparación	sudo apt install python3-pip pipx	Se instaló pip y el gestor de entornos aislados pipx.
Instalación Aislada	pipx install glances	Se instaló Glances en un entorno virtual dedicado.
Error de PATH	pipx ensurepath + source ~/.bashrc + hash -r	Se arregló la ruta de comandos para que el sistema pudiera encontrar el nuevo binario de glances.
Error FastAPI	pipx inject glances fastapi	Se instaló el *framework* web principal.
Error Jinja2	pipx inject glances jinja2	**Se instaló la última dependencia** necesaria para el renderizado de la página.

Comando de Ejecución Final

Una vez que todos los pasos fueron completados, Glances se pudo iniciar correctamente:

glances -w -B 0.0.0.0

Acceso Web

La interfaz web de monitoreo está disponible en:

http://192.168.1.44:61208/

Acceso desde fuera de la VM

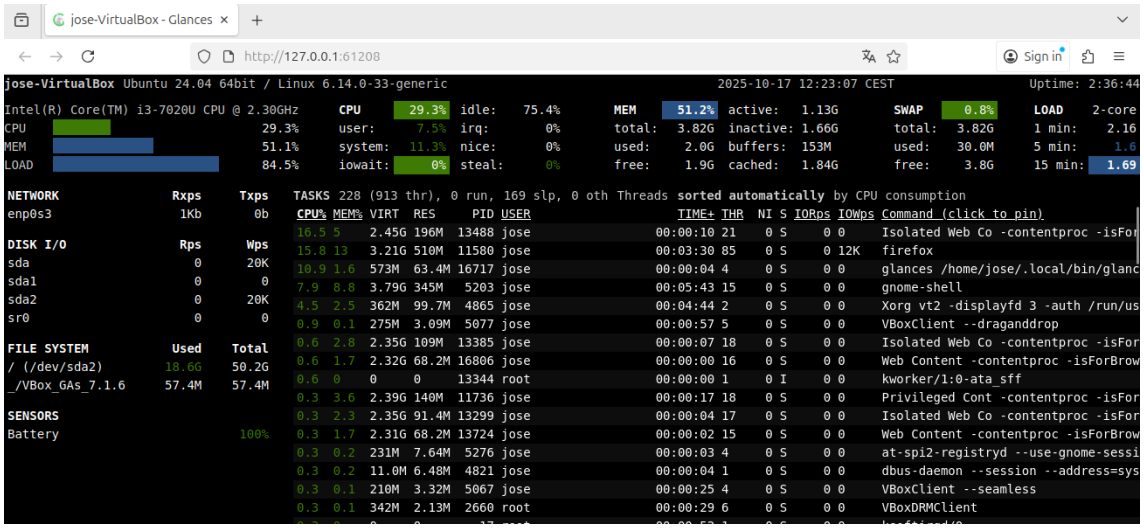
Por defecto, glances -w escucha en 127.0.0.1:61208.

- Si lo abres desde **la misma VM** → http://127.0.0.1:61208 funciona.
- Si lo intentas desde tu **host** en VirtualBox, no responde porque solo escucha en localhost. Solución:

  glances -w --bind 0.0.0.0

Y luego accedes desde tu host con la IP de la VM:

http://192.168.1.44:61208/

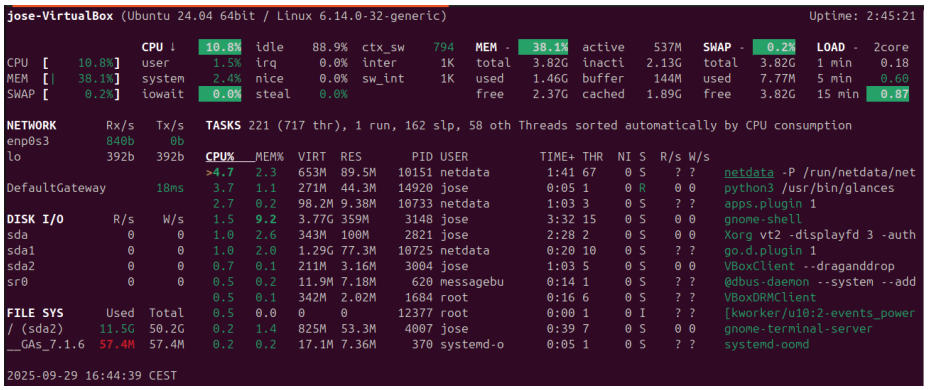


Firewall o red de VirtualBox

- Si usas NAT, el puerto 61208 no se expone al host.
- Para verlo desde fuera, necesitas **Bridge** o **Adaptador solo-anfitrión**, o configurar un **port forwarding** en NAT.

Modo terminal

```
glances
# teclas: h (help), q (quit), o (show/hide columns)
```



Modo web

```
# inicia el servidor web (puerto por defecto 61208)

glances -w

# abre en el navegador: http://<ip>:61208
```

Poner como servicio systemd (ejemplo básico)
Crea /etc/systemd/system/glances.service con contenido (ejemplo sencillo). Luego:

```
sudo systemctl daemon-reload
sudo systemctl enable --now glances
```

(Para entornos de producción, configura usuario, opciones y seguridad.)

Seguridad: por defecto el modo web no está asegurado. Usa **ssh tunnel**:

```
ssh -L 61208:localhost:61208 usuario@server
# luego abrir http://localhost:61208
```

O protege con nginx/HTTPS.

Cuando accedes a **Glances** a través de su interfaz web http://127.0.0.1:61208/, debes buscar la misma información clave que en la consola, ya que está diseñada para ser un panel de control rápido y completo.

Glances utiliza colores para indicar el estado de salud (verde: OK, azul: Cuidado, magenta: Alerta, rojo: Crítico), lo cual es lo primero que debes buscar.

1. **CPU (Central Processing Unit)**

- **Uso General:** Comprueba el porcentaje de uso total.
- **Distribución:** Observa qué porcentaje se destina a **System** (procesos del kernel), **User** (aplicaciones) e **I/O Wait**. Un **I/O Wait** alto es una señal de que la CPU está inactiva esperando que el disco termine una operación, indicando un cuello de botella de I/O.
- **Frecuencia (Freq):** Muestra la velocidad actual de los núcleos de la CPU.

2. **MEM (Memoria) y SWAP**

- **Uso de RAM:** Mira el porcentaje de memoria **Used** (usada) y **Free** (libre).
- **Caché/Búfer:** Esta memoria se usa activamente, pero se puede liberar rápidamente, por lo que no es tan crítica como la memoria **Used** pura.
- **SWAP:** Si el uso de **Swap** aumenta, significa que el sistema está moviendo datos de la RAM al disco duro (paginación), lo cual degrada seriamente el rendimiento. **Un uso constante de Swap es una señal de falta de RAM.**

3. **Load (Carga del Sistema)**

- **Valores (1m, 5m, 15m):** Estos números representan el promedio de procesos esperando por la CPU en los últimos 1, 5 y 15 minutos.
- **Análisis:** Compara estos valores con el número de núcleos de tu CPU. Si los números son consistentemente **mayores que el número de núcleos** disponibles, tu sistema está **sobrecargado**.

4. **Network I/O (Entrada/Salida de Red)**

- **Tráfico:** Revisa las tasas de transferencia de red (Rx para recibir, Tx para transmitir) en KB/s o MB/s.
- **Anomalías:** Busca picos inesperados que puedan indicar actividad maliciosa o una aplicación que consume demasiado ancho de banda.
- **Errores/Descartes:** Si ves que los contadores de errores o paquetes descartados aumentan, hay un problema en la capa física o en la configuración de red.

5. DISK I/O (Entrada/Salida de Disco)

- **Lectura/Escritura (R/W):** Muestra la velocidad de lectura (R) y escritura (W) en disco (KB/s o MB/s).
- **Utilización:** La clave es ver si el disco está alcanzando su límite de rendimiento (p. ej., si el disco está al 100% de utilización, pero solo transfiere 1 MB/s, la latencia será alta).

6. PROCESOS (Process List)

Esta es la sección más detallada, donde puedes identificar la causa raíz de un problema:

- **Ordenación:** Por defecto, Glances ordena por consumo de **CPU** o **MEM**.
- **Identificación:** Busca el **nombre del proceso (Command)**, su **PID**, el **usuario** que lo ejecuta y el porcentaje de **CPU** y **MEM** que consume.
- **Problemas Comunes:**
  - Un proceso que consume cerca del 100% de la CPU de un solo núcleo.
  - Un proceso que consume una cantidad excesiva de memoria, indicando un posible *memory leak*.

En resumen:

Debes buscar rápidamente cualquier métrica que esté marcada en **magenta** (alerta) o **rojo** (crítico) para identificar cuellos de botella en la **CPU** (si hay I/O Wait alto), **Memoria** (si hay SWAP activo), o **Carga** (si es mayor que el número de núcleos).

Ejercicio Glances

1. glances -w en una VM.
2. Accede a la interfaz web vía SSH tunnel.
3. Observa métricas y describe CPU, memoria y procesos top.

11) Prácticas sugeridas

1. **Diagnóstico de saturación CPU**
   - Ejecuta top y htop. Identifica proceso top CPU. Confirma con ps.
   - Acciones: renice o investigar tu cmd.

2. **Detectar problema de I/O**
   - Usa iotop y du para localizar proceso/archivo. Simula con dd.

3. **Limpieza de disco**
   - Encuentra archivos >100MB:

```
sudo find / -type f -size +100M -exec ls -lh {} \; 2>/dev/null
```

   - Propón política: rotación de logs (logrotate), compresión, mover a otro disco.

4. **Red y puertos**

   ss -tulpen

   detectar servicios expuestos. Inicia

   python3 -m http.server 8080

   y verifica.

5. **Monitorización en tiempo real con Netdata/Glances**
   - Instala Netdata o Glances, genera carga y documenta las métricas.

12) Cheatsheet rápido (comandos útiles)

```
# procesos / CPU
top
htop
ps aux | sort -nrk 3,3 | head -n 10   # top 10 por CPU
ps aux | sort -nrk 4,4 | head -n 10   # top 10 por MEM

# disco
df -hT
du -sh /path/*
du -ah /path | sort -rh | head -n 20
ncdu /path

# I/O
sudo iotop -o -P
iostat -xz 1   # requiere sysstat

# red / sockets
sudo ss -tulpen
sudo netstat -tulnp

# netdata / glances
sudo systemctl enable --now netdata
glances -w
```

13) Consejos

- **Cuidado en máquinas compartidas**: no matar procesos críticos en ejercicios. Usa VMs o contenedores.
- **Seguridad**: siempre proteger netdata/glances si se exponen a la red.