

Reto Día 10: Supervisión de Procesos y Optimización de Rendimiento en Linux

Por Daniel Ariza

23/06/2025

Fase 1: Análisis en tiempo real del sistema

- Ejecutar y analizar las siguientes herramientas:
- **top** → ver procesos activos, carga y uso de CPU/RAM:

Ejecutando el comando **top** nos muestra información del sistema con vista dinámica que se actualiza cada pocos segundos.

En la parte superior podemos ver un resumen del sistema con datos como la carga del sistema, el uso de memoria ram y swap... En la parte inferior podemos ver una lista de los procesos.

Archivo Máquina Ver Entrada Dispositivos Ayuda										
Tareas: 77 total, 1 ejecutar, 76 hibernar, 0 detener, 0 zombie										
Cpu(s): 0,0 usuario, 0,3 sist, 0,0 adecuado, 99,7 inact, 0,0 en espera, 0,0 hardw int, 0,0										
KiB Mem : 2048168 total, 1551284 free, 51396 used, 445488 buff/cache										
KiB Swap: 998396 total, 998396 free, 0 used. 1841160 avail Mem										
PID	USUARIO	PR	NI	UIRT	RES	SHR	S	%CPU	%MEM	HORA+ ORDEN
2052	prueba1	20	0	46764	3924	3324	R	0,3	0,2	0:00.15 top
1	root	20	0	119184	5296	3924	S	0,0	0,3	0:00.94 systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00 kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00.04 ksoftirqd/0
5	root	0	-20	0	0	0	S	0,0	0,0	0:00.00 kworker/0:0H
7	root	20	0	0	0	0	S	0,0	0,0	0:01.50 rcu_sched
8	root	20	0	0	0	0	S	0,0	0,0	0:00.00 rcu_bh
9	root	rt	0	0	0	0	S	0,0	0,0	0:00.00 migration/0
10	root	rt	0	0	0	0	S	0,0	0,0	0:00.06 watchdog/0
11	root	20	0	0	0	0	S	0,0	0,0	0:00.00 kdevtmpfs
12	root	0	-20	0	0	0	S	0,0	0,0	0:00.00 netns
13	root	0	-20	0	0	0	S	0,0	0,0	0:00.00 perf
14	root	20	0	0	0	0	S	0,0	0,0	0:00.00 khungtaskd
15	root	0	-20	0	0	0	S	0,0	0,0	0:00.00 writeback
16	root	25	5	0	0	0	S	0,0	0,0	0:00.00 ksmd
17	root	39	19	0	0	0	S	0,0	0,0	0:00.00 khugepaged
18	root	0	-20	0	0	0	S	0,0	0,0	0:00.00 crypto
19	root	0	-20	0	0	0	S	0,0	0,0	0:00.00 kintegrityd
20	root	0	-20	0	0	0	S	0,0	0,0	0:00.00 bioset
21	root	0	-20	0	0	0	S	0,0	0,0	0:00.00 kblockd
22	root	0	-20	0	0	0	S	0,0	0,0	0:00.00 ata_sff
23	root	0	-20	0	0	0	S	0,0	0,0	0:00.00 md
24	root	0	-20	0	0	0	S	0,0	0,0	0:00.00 devfreq_wq
28	root	20	0	0	0	0	S	0,0	0,0	0:00.00 kswapd0
29	root	0	-20	0	0	0	S	0,0	0,0	0:00.00 vmstat
30	root	20	0	0	0	0	S	0,0	0,0	0:00.00 fsnotify_mark
31	root	20	0	0	0	0	S	0,0	0,0	0:00.00 ecryptfs-kthrea
47	root	0	-20	0	0	0	S	0,0	0,0	0:00.00 kthrotld
48	root	0	-20	0	0	0	S	0,0	0,0	0:00.00 acpi_thermal_pm
49	root	0	-20	0	0	0	S	0,0	0,0	0:00.00 bioset

- **htop (si no está, instalarlo).**

htop es una herramienta más visual y potente que **top** para monitorizar el rendimiento del sistema. Es más interactiva y nos muestra información con colores y barras gráficas, nos permite navegación con el teclado y el uso de filtros y acciones más intuitivas. lo instalamos con:

sudo apt update

sudo apt install htop

Y lo ejecutamos con: **htop**

PID	USER	PRI	NI	UIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
2640	daniel	20	0	30904	3868	3172	R	0.0	0.2	0:00.13	htop
1021	www-data	20	0	352M	4152	2644	S	0.0	0.2	0:04.07	/usr/sbin/apache2 -k start
995	www-data	20	0	352M	4152	2644	S	0.0	0.2	0:04.13	/usr/sbin/apache2 -k start
967	www-data	20	0	352M	4152	2644	S	0.0	0.2	0:04.13	/usr/sbin/apache2 -k start
966	www-data	20	0	352M	4152	2644	S	0.0	0.2	0:04.07	/usr/sbin/apache2 -k start
1	root	20	0	116M	5304	3924	S	0.0	0.3	0:00.95	/sbin/init splash
182	root	20	0	29640	3552	3240	S	0.0	0.2	0:00.16	/lib/systemd/systemd-journald
271	root	20	0	44800	4352	3060	S	0.0	0.2	0:00.27	/lib/systemd/systemd-udev
594	root	20	0	31780	2964	2692	S	0.0	0.1	0:00.01	/usr/sbin/cron -f
660	syslog	20	0	250M	3164	2580	S	0.0	0.2	0:00.00	/usr/sbin/rsyslogd -n
661	syslog	20	0	250M	3164	2580	S	0.0	0.2	0:00.03	/usr/sbin/rsyslogd -n
662	syslog	20	0	250M	3164	2580	S	0.0	0.2	0:00.04	/usr/sbin/rsyslogd -n
598	syslog	20	0	250M	3164	2580	S	0.0	0.2	0:00.10	/usr/sbin/rsyslogd -n
599	root	20	0	28676	3172	2844	S	0.0	0.2	0:00.02	/lib/systemd/systemd-logind
658	root	20	0	274M	6296	5528	S	0.0	0.3	0:00.28	/usr/lib/accountsservice/accounts-daemon
664	root	20	0	274M	6296	5528	S	0.0	0.3	0:00.00	/usr/lib/accountsservice/accounts-daemon
601	root	20	0	274M	6296	5528	S	0.0	0.3	0:00.33	/usr/lib/accountsservice/accounts-daemon
602	messagebus	20	0	42892	3776	3364	S	0.0	0.2	0:00.03	/usr/bin/dbus-daemon --system --add
757	root	20	0	65828	3508	3000	S	0.0	0.2	0:00.02	/bin/login --
931	root	20	0	373M	2372	2040	S	0.0	0.1	0:00.00	/usr/sbin/UBoxService --pidfile /var
932	root	20	0	373M	2372	2040	S	0.0	0.1	0:00.00	/usr/sbin/UBoxService --pidfile /var
933	root	20	0	373M	2372	2040	S	0.0	0.1	0:00.11	/usr/sbin/UBoxService --pidfile /var
934	root	20	0	373M	2372	2040	S	0.0	0.1	0:01.16	/usr/sbin/UBoxService --pidfile /var
935	root	20	0	373M	2372	2040	S	0.0	0.1	0:00.00	/usr/sbin/UBoxService --pidfile /var
936	root	20	0	373M	2372	2040	S	0.0	0.1	0:00.11	/usr/sbin/UBoxService --pidfile /var
937	root	20	0	373M	2372	2040	S	0.0	0.1	0:00.20	/usr/sbin/UBoxService --pidfile /var
938	root	20	0	373M	2372	2040	S	0.0	0.1	0:00.00	/usr/sbin/UBoxService --pidfile /var
930	root	20	0	373M	2372	2040	S	0.0	0.1	0:01.61	/usr/sbin/UBoxService --pidfile /var
963	root	20	0	71580	4260	3080	S	0.0	0.2	0:00.60	/usr/sbin/apache2 -k start
996	www-data	20	0	352M	4152	2644	S	0.0	0.2	0:00.00	/usr/sbin/apache2 -k start

Al igual que en **top** la parte superior es el resumen del sistema y la parte inferior es el listado de procesos. Y como podemos ver es una forma más visual e intuitiva de monitorizar el sistema.

- **uptime y free -m.**

```

Archivo Máquina Ver Entrada Dispositivos Ayuda
daniel@ubuntucodearts:~$ uptime
13:27:01 up 5:02, 1 user, load average: 0,00, 0,02, 0,00
daniel@ubuntucodearts:~$ _

```

uptime nos muestra la siguiente información en este orden:

hora actual del sistema, tiempo que lleva encendido el sistema, usuarios conectados ahora y carga del sistema.

uptime nos sirve para saber hace cuánto está encendido el sistema y si está sobrecargado.

```
daniel@ubuntucodearts:~$ free -m
              total        used        free      shared  buff/cache   available
Memoria:      2000         52        1506           3         442        1795
Swap:          974           0         974
daniel@ubuntucodearts:~$ _
```

free -m para saber cuánta memoria RAM y swap estás usando realmente.

- **identificar:**
- **Proceso con mayor consumo de CPU.**

Utilizaremos el comando **top** y después pulsamos la tecla **P** (mayúscula) Esto ordena los procesos por **uso de CPU (de mayor a menor)**.

PID	USUARIO	PR	NI	UIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
2653	daniel	20	0	46768	3840	3240	R	0,7	0,2	0:00.11	top
1	root	20	0	119184	5304	3924	S	0,0	0,3	0:00.95	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00.09	ksoftirqd/0

PID: ID del proceso

USER: usuario que lo ejecuta

%CPU: uso actual de CPU

COMMAND: nombre del proceso.

- **Proceso con mayor uso de memoria.**

Volvemos a utilizar **top** pero ahora pulsamos la letra **M** (mayúscula) Esto ordena los procesos por **uso de memoria (de mayor a menor)**.

PID	USUARIO	PR	NI	UIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
601	root	20	0	281028	6296	5528	S	0,0	0,3	0:00.36	accounts-daemon
2001	root	20	0	65508	5376	4684	S	0,0	0,3	0:00.00	sshd
1	root	20	0	119184	5304	3924	S	0,0	0,3	0:00.95	systemd
1131	daniel	20	0	27380	5072	3396	S	0,0	0,2	0:00.15	bash

PID: ID del proceso

%MEM: porcentaje de RAM utilizada

COMMAND: nombre del proceso

- **Tiempo que lleva encendido el sistema y carga promedio.**

Ambas cosas se realizan con el comando **uptime** como explicamos anteriormente.

Fase 2: Gestión activa de procesos y prioridades

- **Finalizar un proceso inactivo o que no sea esencial (kill, killall o pkill).**

Primero veremos los procesos activos y su estado con **ps aux** y después filtraremos procesos inactivos (dormidos o sin uso de CPU) con **ps aux --sort=%cpu | grep 'S'** Y buscaremos un proceso que no sea del sistema ni crítico.

Elegiremos este proceso:

```
prueba1  2030  0.0  0.2 27372 5064 tty1  S 12:24  0:00 -su
```

Ya que este usuario ahora mismo no está en uso.

Lo finalizamos con **sudo kill 2030**

```
daniel@ubuntucodearts:~$ ps aux --sort=%cpu | grep ' S '
root      2  0.0  0.0   0   0 ?      S   08:53   0:00 [kthreadd]
root      3  0.0  0.0   0   0 ?      S   08:53   0:00 [ksoftirqd/0]
root      7  0.0  0.0   0   0 ?      S   08:53   0:01 [rcu_sched]
root      8  0.0  0.0   0   0 ?      S   08:53   0:00 [rcu_bh]
root      9  0.0  0.0   0   0 ?      S   08:53   0:00 [migration/0]
root     10  0.0  0.0   0   0 ?      S   08:53   0:00 [watchdog/0]
root     11  0.0  0.0   0   0 ?      S   08:53   0:00 [kdevtmpfs]
root     14  0.0  0.0   0   0 ?      S   08:53   0:00 [khungtaskd]
root     28  0.0  0.0   0   0 ?      S   08:53   0:00 [kswapd0]
root     30  0.0  0.0   0   0 ?      S   08:53   0:00 [fsnotify_mark]
root     31  0.0  0.0   0   0 ?      S   08:53   0:00 [ecryptfs-kthrea]
root     57  0.0  0.0   0   0 ?      S   08:53   0:00 [scsi_eh_0]
root     59  0.0  0.0   0   0 ?      S   08:53   0:00 [scsi_eh_1]
root    126  0.0  0.0   0   0 ?      S   08:53   0:00 [scsi_eh_2]
root    154  0.0  0.0   0   0 ?      S   08:53   0:00 [jbd2/sda1-8]
root    210  0.0  0.0   0   0 ?      S   08:53   0:00 [kauditd]
daniel   1128  0.0  0.0 60652 1440 ?      S   08:57   0:00 (sd-pam)
daniel   1131  0.0  0.2 27380 5072 tty1    S   08:57   0:00 -bash
root    2029  0.0  0.1 57288 3360 tty1    S   12:24   0:00 su - prueba1
prueba1  2030  0.0  0.2 27372 5064 tty1    S   12:24   0:00 -su
root    2049  0.0  0.0   0   0 ?      S   13:00   0:02 [kworker/0:2]
root    2061  0.0  0.1 57288 3568 tty1    S   13:48   0:00 su - daniel
daniel   2062  0.0  0.2 27380 4888 tty1    S   13:48   0:00 -su
root    2655  0.0  0.0   0   0 ?      S   14:14   0:00 [kworker/u2:1]
root    2659  0.0  0.0   0   0 ?      S   14:28   0:00 [kworker/u2:2]
root    2660  0.0  0.0   0   0 ?      S   14:30   0:00 [kworker/0:0]
root    2663  0.0  0.0   0   0 ?      S   14:39   0:00 [kworker/0:1]
daniel   2666  0.0  0.1 19104 2084 tty1    S+  14:39   0:00 grep --color=auto S
daniel@ubuntucodearts:~$ _

daniel@ubuntucodearts:~$ ps -p 2030
  PID TTY          TIME CMD
daniel@ubuntucodearts:~$ _
```

A veces con **sudo kill 2030** no se termina de cerrar el proceso y es necesario aplicar **sudo kill -9 2030** para forzar la finalización definitiva. Comprobamos con **ps -p 2030** y como vemos en la captura se ha finalizado correctamente. Tenemos también estos comandos:

killall: Finaliza **todos los procesos** con un **nombre específico**. Por ejemplo, si utilizamos el nombre Firefox **sudo killall firefox** cierra **todas las instancias del navegador Firefox** abiertas. Usa el **nombre exacto** del proceso.

pkill: Finaliza procesos **por nombre o patrón**, como **killall**, pero es **más flexible**. **sudo pkill nano** esto mata cualquier proceso cuyo nombre **contenga** nano.

- **Cambiar la prioridad de un proceso en ejecución con renice.**

El comando **renice** le dice al **scheduler** de Linux: "Este proceso es más importante (o menos), así que dale más (o menos) tiempo de CPU". No mata ni reinicia el proceso y es útil en servidores si un proceso crítico necesita más CPU.

Valores posibles (nice): **-20** (más prioridad) hasta **19** (menos prioridad).

Por defecto, los procesos arrancan con nice **0**.

Sólo el **superusuario (root)** puede usar valores negativos (más prioridad).

Primero identificamos el PID del proceso que queremos modificar y le añadimos un nice, a menor nice más prioridad. **sudo renice [nuevo_nice] -p [PID]**

Haremos la prueba con el siguiente proceso:

PID: 1138

USUARIO: daniel

COMANDO: (sd-pam)

Aunque **sd-pam** está relacionado con la sesión, **es seguro cambiarle la prioridad** sólo como práctica porque no lo mataremos, solo modificaremos el uso de CPU (que es 0%).

Archivo Máquina Ver Entrada Dispositivos Ayuda											
top - 07:19:55 up 15 min, 1 user, load average: 0,00, 0,00, 0,00											
Tareas: 78 total, 1 ejecutar, 77 hibernar, 0 detener, 0 zombie											
%Cpu(s): 0,0 usuario, 0,0 sist, 0,0 adecuado, 99,7 inact, 0,3 en espera, 0,0 hard											
KiB Mem : 2048168 total, 1849968 free, 48952 used, 149248 buff/cache											
KiB Swap: 998396 total, 998396 free, 0 used. 1848020 avail Mem											
PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
133	root	20	0	0	0	0	S	0,0	0,0	0:00.42	kworker/0:3
135	root	20	0	0	0	0	S	0,0	0,0	0:00.02	scsi_eh_2
136	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	scsi_tmf_2
137	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	ttm_swap
138	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	bioaset
161	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/0:1H
163	root	20	0	0	0	0	S	0,0	0,0	0:00.00	jbd2/sda1-8
164	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	ext4-rsv-conver
210	root	20	0	29640	3176	2872	S	0,0	0,2	0:00.05	systemd-journal
214	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kauditd
269	root	20	0	44800	4360	3072	S	0,0	0,2	0:00.26	systemd-udev
466	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	iprt-VBoxWQueue
596	root	20	0	28676	3104	2788	S	0,0	0,2	0:00.01	systemd-logind
599	root	20	0	31780	2884	2624	S	0,0	0,1	0:00.00	cron
600	message+	20	0	42892	3804	3396	S	0,0	0,2	0:00.02	dbus-daemon
606	syslog	20	0	256388	3248	2660	S	0,0	0,2	0:00.01	rsyslogd
611	root	20	0	281032	6404	5636	S	0,0	0,3	0:00.03	accounts-daemon
757	root	20	0	65828	3396	2888	S	0,0	0,2	0:00.02	login
792	root	20	0	65508	5588	4900	S	0,0	0,3	0:00.00	sshd
943	root	20	0	71580	4524	3344	S	0,0	0,2	0:00.02	apache2
946	www-data	20	0	360744	3980	2472	S	0,0	0,2	0:00.17	apache2
947	www-data	20	0	360744	3980	2472	S	0,0	0,2	0:00.18	apache2
1019	root	20	0	316884	2484	2148	S	0,0	0,1	0:00.08	VBoxService
1114	root	20	0	0	0	0	S	0,0	0,0	0:00.05	kworker/u2:1
1115	root	20	0	0	0	0	S	0,0	0,0	0:00.23	kworker/0:0
1133	daniel	20	0	45120	4936	4184	S	0,0	0,2	0:00.00	systemd
1136	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kworker/0:2
1138	daniel	20	0	60700	1492	0	S	0,0	0,1	0:00.00	(sd-pam)
1141	daniel	20	0	27380	4828	3284	S	0,0	0,2	0:00.03	bash
1165	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kworker/u2:2

```

Archivo Máquina Ver Entrada Dispositivos Ayuda
daniel@ubuntucodearts:~$ sudo renice 10 -p 1138
1138 (process ID) prioridad antigua 0, prioridad nueva 10
daniel@ubuntucodearts:~$ ps -p 1138 -o pid,comm,nice
  PID COMMAND      NI
  1138 (sd-pam)      10
daniel@ubuntucodearts:~$

```

Aquí vemos el proceso realizado ejecutando **sudo renice 10 -p 1138** y verificando con **ps -p 1136 -o pid,comm,nice**

- Lanzar un proceso en segundo plano (&) y enviarlo al primer plano con fg.

Primero lanzamos un proceso en segundo plano con **sleep 300 &** Esto crea un proceso simple, seguro y visible con **top**.

Confirmamos que está corriendo con **jobs**

Y lo traemos al primer plano con **fg**

```
daniel@ubuntucodearts:~$ sleep 300 &
[1] 1186
daniel@ubuntucodearts:~$ jobs
[1]+  Ejecutando          sleep 300 &
daniel@ubuntucodearts:~$ fg
sleep 300
```

Esto lanzó un proceso **sleep** (que espera 300 segundos) en **segundo plano**.

La shell respondió con: **[1] 1186**

[1] : es el **job ID** (número del trabajo en segundo plano).

1186 : es el **PID real del proceso** en el sistema.

jobs mostró: **[1]+ Ejecutando sleep 300 &**

Confirma que el **trabajo número 1** está activo.

El símbolo **+** indica que es el **trabajo más reciente**, es el que se usará si llamas a **fg** sin número.

fg Esto trajo el proceso **sleep 300** al primer plano.

Lo ves claramente porque la terminal queda ocupada y muestra: **sleep 300** Ahora el sistema está esperando que se cumplan los 300 segundos o que tú lo interrumpas con **Ctrl + C**.

- Usar **nice** para iniciar un proceso con prioridad baja (por ejemplo, una copia pesada con **cp**).

El comando **nice** permite **asignar una prioridad inicial** a un proceso.

Cuanto **mayor** es el valor **nice**, **menos prioridad** tiene el proceso (menos uso de CPU).

Realizaremos una simulación rápida creando un archivo pesado de prueba:

dd if=/dev/zero of=archivo_grande bs=1M count=100 Esto crea un archivo de 100mb llamado `archivo_grande`.

Con esto haremos una copia de prioridad baja **nice -n 15 cp archivo_grande copia_grande**

```
daniel@ubuntucodearts:~$ dd if=/dev/zero of=archivo_grande bs=1M count=100
100+0 registros leídos
100+0 registros escritos
104857600 bytes (105 MB, 100 MiB) copied, 0,106394 s, 986 MB/s
daniel@ubuntucodearts:~$ nice -n 15 cp archivo_grande copia_grande
daniel@ubuntucodearts:~$
```

Verificamos el proceso en `top` o `htop`:

Puedes ver el valor de **NI (nice)** aplicado al proceso `cp`.

nice no acelera el proceso, solo le dice al sistema: "si estás ocupado, a este proceso dále menos prioridad".

Fase 3: Monitorización y registro del uso de recursos

- Usar el comando `vmstat` y guardar su salida en un archivo `/srv/logs/vmstat.log`.

vmstat (virtual memory statistics) muestra información sobre: Uso de **memoria**, **Swap**, **Procesos**, **CPU**, **Entrada/salida del disco**, Es muy útil para **diagnóstico de rendimiento**.

Ejecutando **vmstat** esto mostrará una línea resumen del estado del sistema.

```

Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
daniel@ubuntucodearts:~$ vmstat
procs -----memory----- ---swap-- ----io---- -system-- -----cpu-----
 r b   swpd  libre  búfer  caché    si   so    bi   bo   in   cs us sy id wa st
 0 0     0 1635700 16088 347784    0    0    22   37   38  100  0  0 99  1  0
daniel@ubuntucodearts:~$

```

Ahora creamos un directorio para guardar la anterior salida **sudo mkdir -p /srv/logs** y guardamos la salida en un archivo de log con **sudo bash -c 'vmstat > /srv/logs/vmstat.log'** esto redirige la salida estándar del comando al archivo vmstat.log.

con **cat /srv/logs/vmstat.log** verificamos el estado del archivo

```

daniel@ubuntucodearts:~$ sudo bash -c 'vmstat > /srv/logs/vmstat.log'
daniel@ubuntucodearts:~$ sudo cat /srv/logs/vmstat.log
procs -----memory----- ---swap-- ----io---- -system-- -----cpu-----
 r b   swpd  libre  búfer  caché    si   so    bi   bo   in   cs us sy id wa st
 0 0     0 1634564 16308 347828    0    0    20   33   39  100  0  0 99  1  0
daniel@ubuntucodearts:~$

```

- **Configurar una tarea en crontab que guarde el uso de recursos (top -b -n 1) cada 5 minutos en /srv/logs/top.log.**

Ejecutamos **crontab -e** y seleccionamos el editor **nano** añadimos al final del editor la siguiente línea ***/5 * * * * top -b -n 1 >> /srv/logs/top.log**

```

# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
*/5 * * * * top -b -n 1 >> /srv/logs/top.log

```

Después de 5 minutos ejecutaremos `tail /srv/logs/top.log` para verificarlo veremos el final del archivo con un resumen de procesos y uso de CPU/memoria.

```
daniel@ubuntucodearts:~$ tail /srv/logs/top.log
1133 daniel    20   0  45120  4936  4184 S  0,0  0,2   0:00.00 systemd
1138 daniel    30  10  60700  1492    0 S  0,0  0,1   0:00.00 (sd-pam)
1141 daniel    20   0  27380  5024  3348 S  0,0  0,2   0:00.10 bash
1255 root       20   0      0      0    0 S  0,0  0,0   0:00.04 kworker/u2:1
1264 root       20   0      0      0    0 S  0,0  0,0   0:00.00 kworker/0:1
1265 root       20   0      0      0    0 S  0,0  0,0   0:00.02 kworker/u2:0
1272 root       20   0      0      0    0 S  0,0  0,0   0:00.00 kworker/0:2
1273 root       20   0  57732  3940  3460 S  0,0  0,2   0:00.00 sudo
1274 root       20   0  17376  2984  2784 S  0,0  0,1   0:00.00 bash
1275 root       20   0  46608  3756  3240 R  0,0  0,2   0:00.00 top
```

Vemos que funcionó correctamente.

- **Explorar iotop (si el sistema lo permite) para monitorizar I/O de disco.**

iotop muestra qué procesos están **leyendo o escribiendo en disco**, cuánto están usando y qué porcentaje del tiempo de I/O ocupan.

Es como un **top**, pero enfocado al **uso del disco**, no del CPU o RAM.

Primero lo instalamos con:

`sudo apt update`

`sudo apt install iotop`

Lo ejecutamos con: `sudo iotop`

Esto abrirá una interfaz tipo **top** donde verás columnas como:

Columna

Qué indica

PID	ID del proceso
DISK READ	lectora de disco en tiempo real
DISK WRITE	escritura en disco
SWAPIN	uso de swap
IO	porcentaje de tiempo esperando I/O
COMMAND	nombre del comando o proceso

¿Cuándo usar **iostat**?

Si notas el disco muy ocupado o lento, para saber qué proceso está leyendo/escribiendo mucho (por ejemplo, copias grandes, **rsync**, procesos mal optimizados).

Para analizar cuellos de botella en servidores o sistemas con alto uso de almacenamiento.

Archivo	Máquina	Ver	Entrada	Dispositivos	Ayuda		
Total DISK READ :			0.00 B/s	Total DISK WRITE :	0.00 B/s		
Actual DISK READ:			0.00 B/s	Actual DISK WRITE:	0.00 B/s		
TID	PRIO	USER	DISK READ	DISK WRITE	SWAPIN	IO>	COMMAND
2461	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.02 %	[kworker/u2:1]
1024	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	VBoxService --pidfi~n/vboxadd-servi
1	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	init splash
2	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[kthreadd]
3	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[ksoftirqd/0]
5	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[kworker/0:0H]
7	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[rcu_sched]
8	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[rcu_bh]
9	rt/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[migration/0]
10	rt/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[watchdog/0]
11	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[kdevtmpfs]
12	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[netns]
13	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[perf]
14	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[khungtaskd]
15	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[writeback]
16	be/5	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[ksmd]
17	be/7	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[khugepaged]
18	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[crypto]
19	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[kintegrityd]
20	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[bioset]
21	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[kblockd]
22	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[ata_sff]
23	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[md]
24	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[devfreq_wq]
28	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[kswapd0]
29	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[vmstat]
30	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[fsnotify_mark]
31	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[ecryptfs-kthrea]
47	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[kthrotld]
48	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[acpi_thermal_pml]
49	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[bioset]
50	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[bioset]
51	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[bioset]
52	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[bioset]

Fase 4: Simulación de sobrecarga controlada

● Instalar el paquete stress o stress-ng.

stress: Herramienta ligera para generar carga en CPU, RAM, disco y procesos.

stress-ng: Versión más moderna y potente, con más opciones y tests específicos.

Lo instalamos con:

```
sudo apt update
```

```
sudo apt install stress
```

● Ejecutar una prueba con carga simulada de CPU, memoria o disco durante 1 minuto.

Ahora con **stress-ng --cpu 2 --timeout 10s** hacemos que se utilicen 2 núcleos al 100% durante 10 segundos.

```
daniel@ubuntucodearts:~$ sudo stress-ng --cpu 2 --timeout 10s
stress-ng: info:  [37271] dispatching hogs: 2 cpu
stress-ng: info:  [37271] cache allocate: default cache size: 2048K
```

dispatching hogs → significa que lanza 2 procesos que consumen CPU al máximo.

[37271] → es el PID del proceso stress-ng.

Y ahora lo haremos durante un minuto con carga en CPU, memoria y disco con el comando:

sudo stress-ng --cpu 2 --vm 2 --vm-bytes 256M --hdd 1 --timeout 60s

```
daniel@ubuntucodearts:~$ sudo stress-ng --cpu 2 --vm 2 --vm-bytes 256M --hdd 1 --timeout 60s
stress-ng: info: [3745] dispatching hogs: 2 cpu, 1 hdd, 2 vm
stress-ng: info: [3745] cache allocate: default cache size: 2048K
```

- **Observar el comportamiento del sistema con htop y anotar el resultado.**

Al no disponer de interfaz gráfica no me ha sido posible observar el comportamiento del sistema mientras se ejecutaba la prueba de carga en el sistema ya que las opciones por las que he optado y he probado no me daban datos lo suficientemente concluyentes como para plasmarlos en este documento.