

LABORATORIO 3: MONITORIZACIÓN

Gestión de recursos del sistema

Esta parte del laboratorio propone unas tareas para trabajar con los comandos relacionados con la monitorización y gestión de los recursos del sistema.

1. Obtener el número de procesos en ejecución en el sistema.

```
maitane@as2-maitane:~$ ps al
F  UID      PID    PPID PRI  NI      VSZ    RSS WCHAN  STAT TTY      TIME COMMAND
4   0        655      1  20   0    7360   2404 -        Ss+  ttyS0    0:00
/sbin/agetty -o -p -- \u --keep
4   0        719      1  20   0    5836   1928 -        Ss+  tty1     0:00
/sbin/agetty -o -p -- \u --nocl
0  1001      1001     1000  20   0   10040   5112 do_wai Ss    pts/0    0:00 -bash
0  1001      1037     1001  20   0   10540   3048 -        R+   pts/0    0:00 ps al
```

O también:

- *ps aux*: muestra una lista de todos los procesos en ejecución en el sistema, junto con información detallada sobre cada uno.

```
maitane@as2-maitane:~$ ps aux | wc -l
102
```

2. Obtener el número de procesos en ejecución que pertenezcan a usuario root.

```
maitane@as2-maitane:~$ ps aux | grep root | wc -l
89
```

3. Instalar el paquete "stress-ng". Ejecutar el benchmark para 1 núcleo de CPU y 20 segundos.

```
maitane@as2-maitane:~$ sudo apt install stress-ng
maitane@as2-maitane:~$ stress-ng -c 1 -v --timeout 20s
stress-ng: debug: [1122] 2 processors online, 2 processors configured
stress-ng: info: [1122] dispatching hogs: 1 cpu
stress-ng: debug: [1122] cache allocate: default cache size: 56320K
stress-ng: debug: [1122] starting stressors
stress-ng: debug: [1122] 1 stressor spawned
stress-ng: debug: [1123] stress-ng-cpu: started [1123] (instance 0)
stress-ng: debug: [1123] stress-ng-cpu using method 'all'
stress-ng: debug: [1123] stress-ng-cpu: exited [1123] (instance 0)
```

```
stress-ng: debug: [1122] process [1123] terminated
stress-ng: info: [1122] successful run completed in 20.12s
```

4. Ejecutar de nuevo "stress-ng", esta vez sin límite de tiempo. Mientras esté en ejecución:

```
maitane@as2-maitane:~$ stress-ng -c 1
stress-ng: info: [1210] defaulting to a 86400 second (1 day, 0.00 secs) run per
stressor
stress-ng: info: [1210] dispatching hogs: 1 cpu
```

- Usar una señal para pausar su ejecución. En la terminal 2:

```
maitane@as2-maitane:~$ pgrep stress-ng
1210
1211

maitane@as2-maitane:~$ kill -SIGSTOP 1210

maitane@as2-maitane:~$ top
 1211 maitane   20   0  94228   7212   3728 R 100.0   0.4   2:29.70 stress-ng-
cpu
```

- Usar una señal para reanudar su ejecución.

```
maitane@as2-maitane:~$ kill -SIGCONT 1210
```

- Reducir la prioridad del proceso al mínimo. ¿Cambia algo? En la terminal 2:

```
maitane@as2-maitane:~$ sudo renice -n 19 -p 1211
1211 (process ID) old priority 0, new priority 19
```

No cambia gran cosa ya que no hay más procesos en ejecución, además de que disponemos de 2 núcleos de CPU. Pero si otros procesos entran en ejecución y se llenan ambos núcleos, se le daría prioridad a los demás procesos.

5. Detectar qué proceso tiene la mayor prioridad en el sistema. Buscar cuál es el propósito de ese proceso.

```
maitane@as2-maitane:~$ top
```

Una vez dentro de TOP:

- Presiona la tecla F (mayúscula) para acceder al menú de ordenación de campos.
- Utiliza las teclas de flecha para moverte hacia arriba o hacia abajo y resaltar el campo que deseas utilizar para la ordenación.
- En este caso, selecciona "PR" para ordenar por el valor priority
- Selecciona 's' para guardar la selección y luego 'q'

Una vez hecho esto el proceso con mayor prioridad es:

```
34 root      39  19          0          0          0 S    0.0    0.0    0:00.00 khugepaged
```

Se refiere a un componente del kernel de Linux llamado "khugepaged", responsable de la gestión de la memoria en sistemas Linux y se encarga de optimizar el uso de la memoria mediante el escaneo y la liberación de páginas de memoria que no se utilizan con frecuencia o que pueden ser fusionadas para reducir el uso de la memoria física.

6. Limitar el máximo tiempo de uso de CPU a 5 minutos para todos los usuarios.

```
maitane@as2-maitane:~$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority     (-e) 0
file size               (blocks, -f) unlimited
pending signals         (-i) 7808
max locked memory       (kbytes, -l) 65536
max memory size         (kbytes, -m) unlimited
open files              (-n) 1024
pipe size               (512 bytes, -p) 8
POSIX message queues    (bytes, -q) 819200
real-time priority      (-r) 0
stack size              (kbytes, -s) 8192
cpu time                (seconds, -t) unlimited
max user processes      (-u) 7808
virtual memory          (kbytes, -v) unlimited
file locks              (-x) unlimited
maitane@as2-maitane:~$ ulimit -t 300
maitane@as2-maitane:~$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority     (-e) 0
file size               (blocks, -f) unlimited
pending signals         (-i) 7808
max locked memory       (kbytes, -l) 65536
max memory size         (kbytes, -m) unlimited
open files              (-n) 1024
pipe size               (512 bytes, -p) 8
POSIX message queues    (bytes, -q) 819200
real-time priority      (-r) 0
stack size              (kbytes, -s) 8192
cpu time                (seconds, -t) 300
```

```
max user processes      (-u) 7808
virtual memory          (kbytes, -v) unlimited
file locks              (-x) unlimited
```

7. Crear un fichero crontab para el usuario root con las siguientes tareas:

```
maitane@as2-maitane:~$ sudo crontab -e
no crontab for root - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano          <---- easiest
 2. /usr/bin/vim.basic
 3. /usr/bin/vim.tiny
 4. /bin/ed

Choose 1-4 [1]: 1
crontab: installing new crontab
```

- Ejecutar el comando date cada minuto y escribir su salida estándar al fichero /tmp/date.log (se debe escribir al final del fichero cada vez). Dentro del fichero crontab:

```
* * * * * date >> /tmp/date.log
```

- Borrar el directorio /tmp los primeros 5 días de cada mes a las 17:00. Dentro del fichero crontab:

```
0 17 1-5 * * rm -r /tmp/*
```

8. Comprobar que las tareas de cron funcionan correctamente (ver el fichero /tmp/date.log).

```
maitane@as2-maitane:~$ cat /tmp/date.log
Fri Sep 29 11:07:01 UTC 2023
Fri Sep 29 11:08:01 UTC 2023
```

Las siguientes tareas del laboratorio son relativas a las conexiones de red y se propone trabajar en parejas. Seréis A y B, y cada uno utilizará su propia máquina virtual:

9. Usando netcat, A abre una conexión a la escucha en el puerto 3000. Si fuese necesario abrir puertos de Google Cloud, se recomienda este tutorial. B se conecta a ese puerto y escribe el mensaje "Hola A". Tras recibirlo, en la misma conexión, B escribe "Hola B" y cierra la conexión.

- nc: Es el comando Netcat.
- -l: Indica a Netcat que debe estar en modo de escucha.
- -p 3000: Especifica el número de puerto que deseas abrir.

A:

```
nc -l -p 3000
```

B:

```
maitane@as2-maitane:~$ nc 34.171.168.81 3000
```

10. El comando “dd if=/dev/urandom” sirve para generar números aleatorios. A abre una conexión a la escucha en el puerto 3000 y B envía números aleatorios a esa conexión de A. Sin cerrar la conexión, A y B utilizan nethogs para comprobar la tasa de bytes enviados y recibidos en cada parte. ¿Los valores coinciden? Después, A cierra la conexión.

A:

```
nc -l -p 3000  
sudo nethogs
```

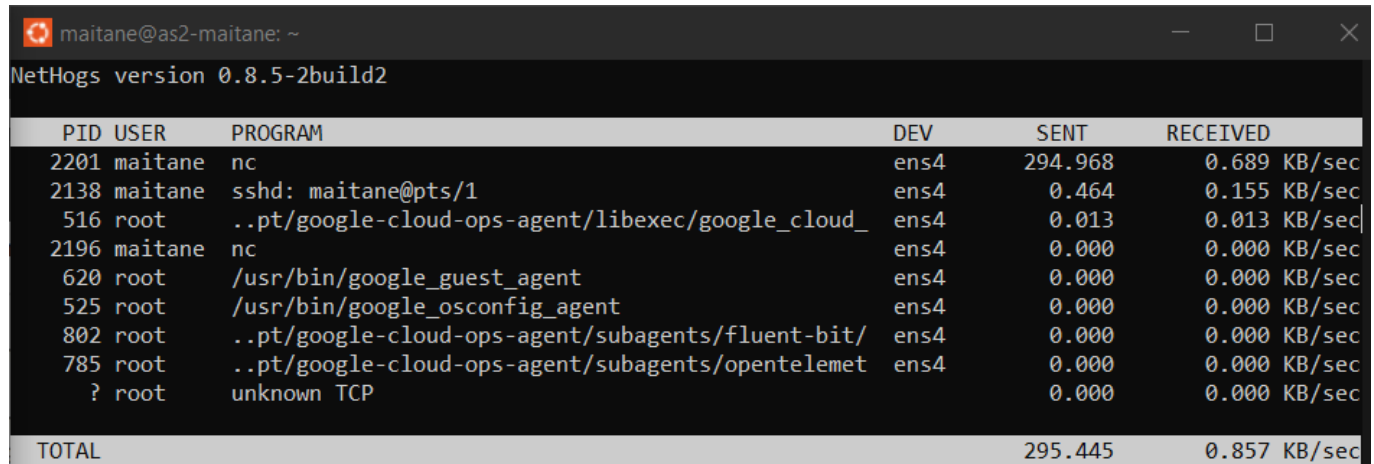


```
NetHogs version 0.8.6-3
```

PID	USER	PROGRAM	DEV	SENT	RECEIVED
2288	aingeru	nc	ens4	0.916	514.514 KB/sec
2260	aingeru	sshd: aingeru@pts/3	ens4	439.331	1.610 KB/sec
?	root	unknown TCP		0.000	0.000 KB/sec
TOTAL				440.247	516.125 KB/sec

B:

```
maitane@as2-maitane:~$ dd if=/dev/urandom | nc 34.171.168.81 3000
maitane@as2-maitane:~$ sudo nethogs
```



Nethogs version 0.8.5-2build2

PID	USER	PROGRAM	DEV	SENT	RECEIVED
2201	maitane	nc	ens4	294.968	0.689 KB/sec
2138	maitane	sshd: maitane@pts/1	ens4	0.464	0.155 KB/sec
516	root	..pt/google-cloud-ops-agent/libexec/google_cloud_	ens4	0.013	0.013 KB/sec
2196	maitane	nc	ens4	0.000	0.000 KB/sec
620	root	/usr/bin/google_guest_agent	ens4	0.000	0.000 KB/sec
525	root	/usr/bin/google_osconfig_agent	ens4	0.000	0.000 KB/sec
802	root	..pt/google-cloud-ops-agent/subagents/fluent-bit/	ens4	0.000	0.000 KB/sec
785	root	..pt/google-cloud-ops-agent/subagents/opentelemet	ens4	0.000	0.000 KB/sec
?	root	unknown TCP		0.000	0.000 KB/sec
TOTAL				295.445	0.857 KB/sec

11. A crea localmente un fichero con texto aleatorio. Utilizar netcat para que A envíe este fichero a B. ¿Qué diferencia hay entre utilizar netcat y scp para enviar ficheros entre diferentes máquinas?

B:

```
maitane@as2-maitane:~$ nc -l -p 3000 > archivoAingeru
```

A:

```
dd if=/dev/urandom of=/home/aingeru/random_data bs=1M count=2048
nc 34.173.208.253 3000 < /home/aingeru/random_data
```

Gestión de los registros del sistema

En esta parte del laboratorio se propondrá trabajar con los registros (logs) del sistema.

1. Leer la página de manual de logger. ¿Con qué parámetro se indica la prioridad de los mensajes?

```
maitane@as2-maitane:~$ man logger
```

...

-p, --priority priority

Enter the message into the **log** with the specified priority. The priority may be specified numerically or as a facility.level pair. For example, -p local3.info logs the message as informational **in** the local3 facility. The default is

```
user.notice.  
...
```

1. Utilizando la línea de comandos, enviar el mensaje "Hola Mundo de Logs" al fichero /var/log/syslog. Comprobar que se ha hecho correctamente.

```
maitane@as2-maitane:~$ sudo logger "Hola Mundo de Logs"  
  
maitane@as2-maitane:~$ tail -n 5 /var/log/syslog  
Sep 29 15:08:01 as2-maitane CRON[1203]: (root) CMD (date >> /tmp/date.log)  
Sep 29 15:09:01 as2-maitane CRON[1206]: (root) CMD (date >> /tmp/date.log)  
Sep 29 15:10:01 as2-maitane CRON[1209]: (root) CMD (date >> /tmp/date.log)  
Sep 29 15:10:44 as2-maitane maitane: Hola Mundo de Logs  
Sep 29 15:11:01 as2-maitane CRON[1214]: (root) CMD (date >> /tmp/date.log)
```

3. Enviar todos los mensajes de nivel "debug" generados por el servicio sshd al fichero /var/log/ssh.log. Este fichero debe haber sido creado anteriormente y estar vacío. Configurar el servicio sshd para que funcione en modo "debug" y comprobar el efecto que tiene en el fichero ssh.log. **Enviar todos los mensajes de nivel "debug" generados por el servicio sshd al fichero /var/log/ssh.log:**

```
# Creamos el fichero que se nos pide  
maitane@as2-maitane:~$ sudo touch /var/log/ssh.log  
  
# Abirmos el fichero de configuración para indicar las acciones  
maitane@as2-maitane:~$ sudo nano /etc/rsyslog.d/50-default.conf
```

Añadimos en el fichero:

```
auth.debug                                -/var/log/ssh.log
```

Cambiamos los permisos y los grupos para que coincidan con los de syslog (el fichero por defecto para escribir los log)

```
maitane@as2-maitane:~$ ls -l /var/log  
maitane@as2-maitane:~$ sudo chown syslog:adm /var/log/ssh.log
```

Reiniciamos para que se apliquen cambios y CONFIGURAMOS EL MODO DEBUG:

```
# Reiniciamos el servicio rsyslog  
maitane@as2-maitane:~$ sudo service rsyslog restart
```

```
# CONFIGURAMOS EL MODO DEBUG
# Editamos el archivo de configuración ssh
maitane@as2-maitane:~$ sudo vim /etc/ssh/sshd_config
# Buscamos la línea 'LogLevel INFO' y la sustituimos por 'LogLevel DEBUG'

# Reiniciamos el sistema sshd
maitane@as2-maitane:~$ sudo systemctl restart sshd

# Comprobamos que se esté escribiendo el fichero
maitane@as2-maitane:~$ cat /var/log/ssh.log
```

Link: https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/6/html/deployment_guide/s1-basic_configuration_of_rsyslog

4. Configurar la rotación de logs (fichero /var/log/syslog) para que se guarden de manera mensual y comprimida, y para que todos los logs generados en un año se guarden en un directorio llamado /var/log/syslog.old.

```
maitane@as2-maitane:~$ sudo nano /etc/logrotate.d/rsyslog
```

Dentro del archivo escribir lo siguiente:

```
/var/log/syslog
{
    rotate 12
    monthly
    compress
    olddir /var/log/syslog.old
    createolddir 0640 syslog adm
}
```

Y luego:????????????????????????????????

```
maitane@as2-maitane:~$ sudo cat /etc/cron.daily/logrotate
#!/bin/sh

# skip in favour of systemd timer
if [ -d /run/systemd/system ]; then
    exit 0
fi

# this cronjob persists removals (but not purges)
if [ ! -x /usr/sbin/logrotate ]; then
    exit 0
fi
```



```

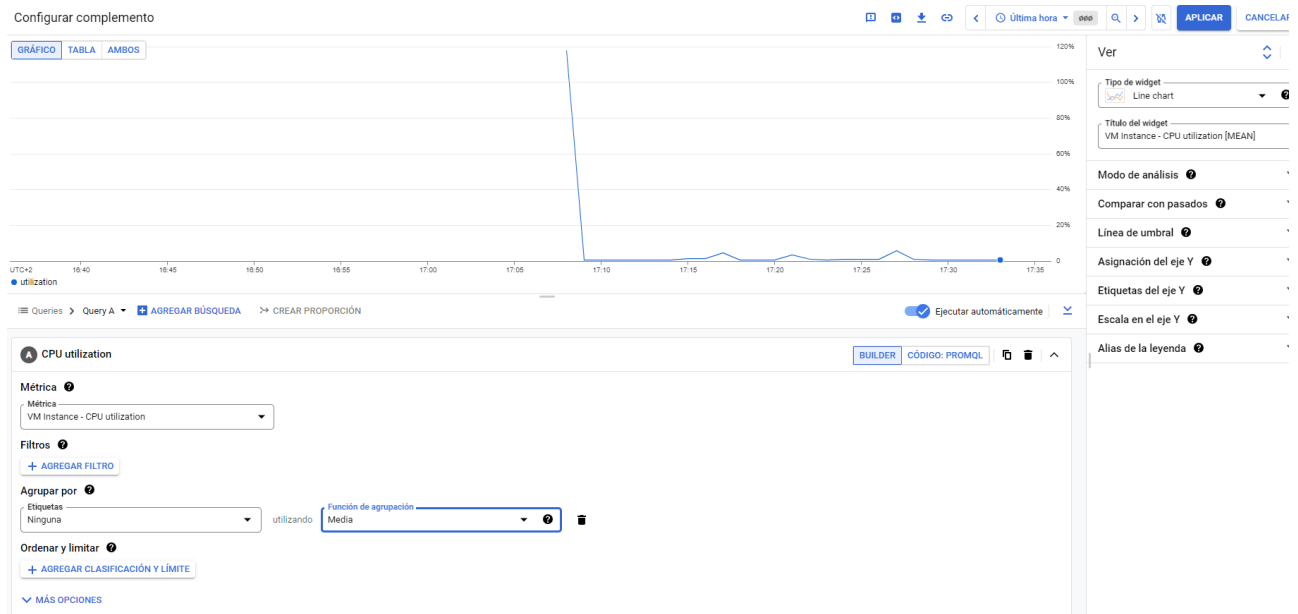
/usr/sbin/logrotate /etc/logrotate.conf
EXITVALUE=$?
if [ $EXITVALUE != 0 ]; then
    /usr/bin/logger -t logrotate "ALERT exited abnormally with [$EXITVALUE]"
fi
exit $EXITVALUE
maitane@as2-maitane:~$ sudo /etc/cron.daily/logrotate

```

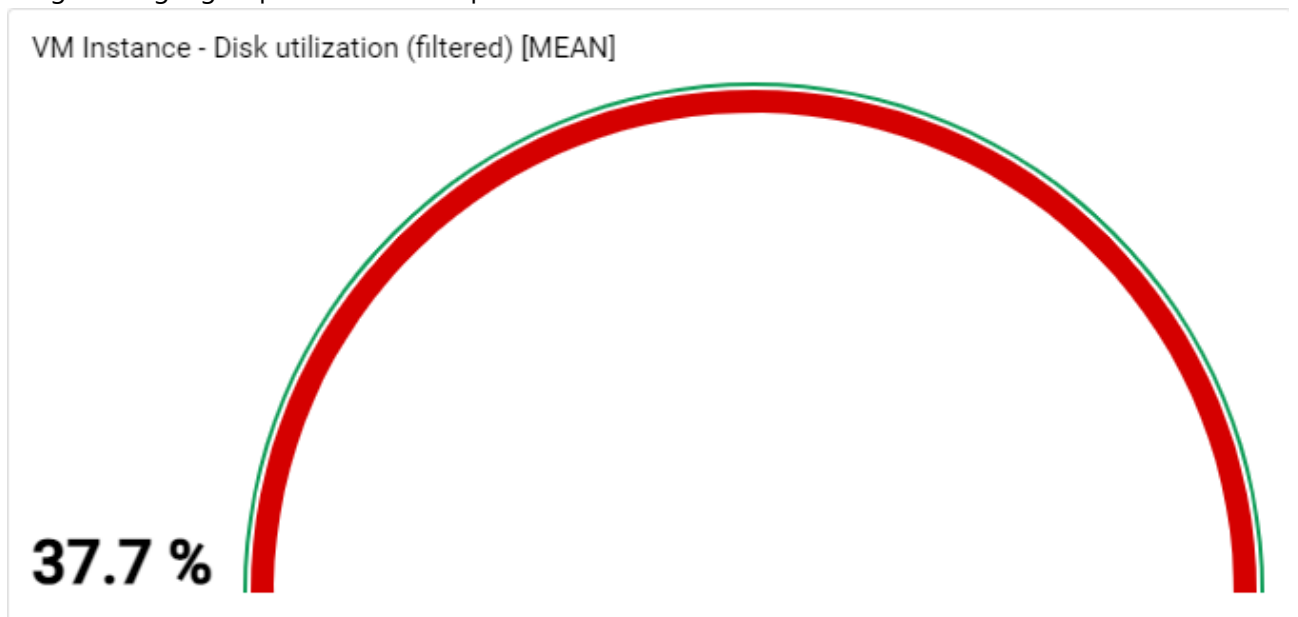
Monitorización en Google Cloud Platform

La primera tarea es crear un Dashboard personalizado que monitorice vuestra máquina virtual. Este deberá contener (al menos) los siguientes elementos:

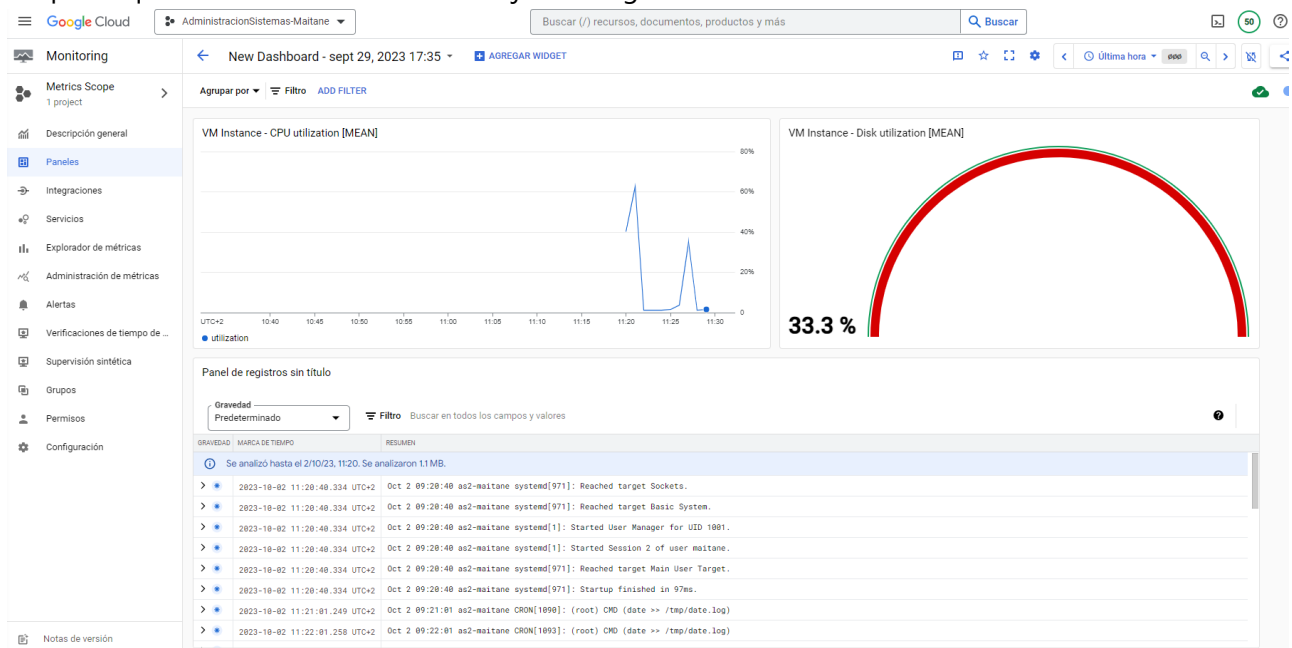
- Un gráfico de líneas que visualice el promedio de consumo de CPU.



- Un gráfico "gauge" que visualice el espacio utilizado en disco.



- Un panel que visualice los últimos mensajes de Log.



Una vez creado, enviad un mensaje al log del sistema desde la Shell y verificar que se visualiza en el panel de este nuevo Dashboard.

Para enviar mensajes log al panel que los recoge:

```
maitane@as2-maitane:~$ logger -p error "Error para GCC"
maitane@as2-maitane:~$ logger "Notificacion para GCC"
```

> *	2023-10-02 11:24:16.874 UTC+2	Oct 2 09:24:16 as2-maitane maitane: Error para GCC
> *	2023-10-02 11:24:28.796 UTC+2	Oct 2 09:24:28 as2-maitane maitane: Notificacion para GCC

La segunda tarea es configurar una alerta personalizada, para que Google Cloud nos avise cuando el espacio libre de una partición se reduzca:

1. Comprobar cuánto espacio libre hay en la partición /dev/sda1 de vuestra máquina virtual.

```
maitane@as2-maitane:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        20G   7.0G   13G   36% /
```

2. Configurar una alerta en Google Cloud para que se os envíe un e-mail cuando el espacio en la partición /dev/sda1 de vuestra máquina virtual sea un 5% mayor del actual (p.e. si ahora mismo está usado un 15%, cuando supere el 20%).

- Accede a Google Cloud bash: <https://bash.cloud.google.com/>
- Navega a la página "Monitoring".
- Selecciona "Alertas" para configurar una nueva alerta.
- Haz clic en "Crear política de alerta".

- Añade la métrica y sigue los pasos
 - Haz clic en "Guardar condición" para guardar la condición de alerta.
3. Utilizando el comando `dd` cread uno (o varios) ficheros en la carpeta `/tmp` de vuestra máquina virtual con datos aleatorios. El tamaño de estos ficheros debe hacer que el uso de partición se incremente hasta superar el umbral establecido en el paso anterior.

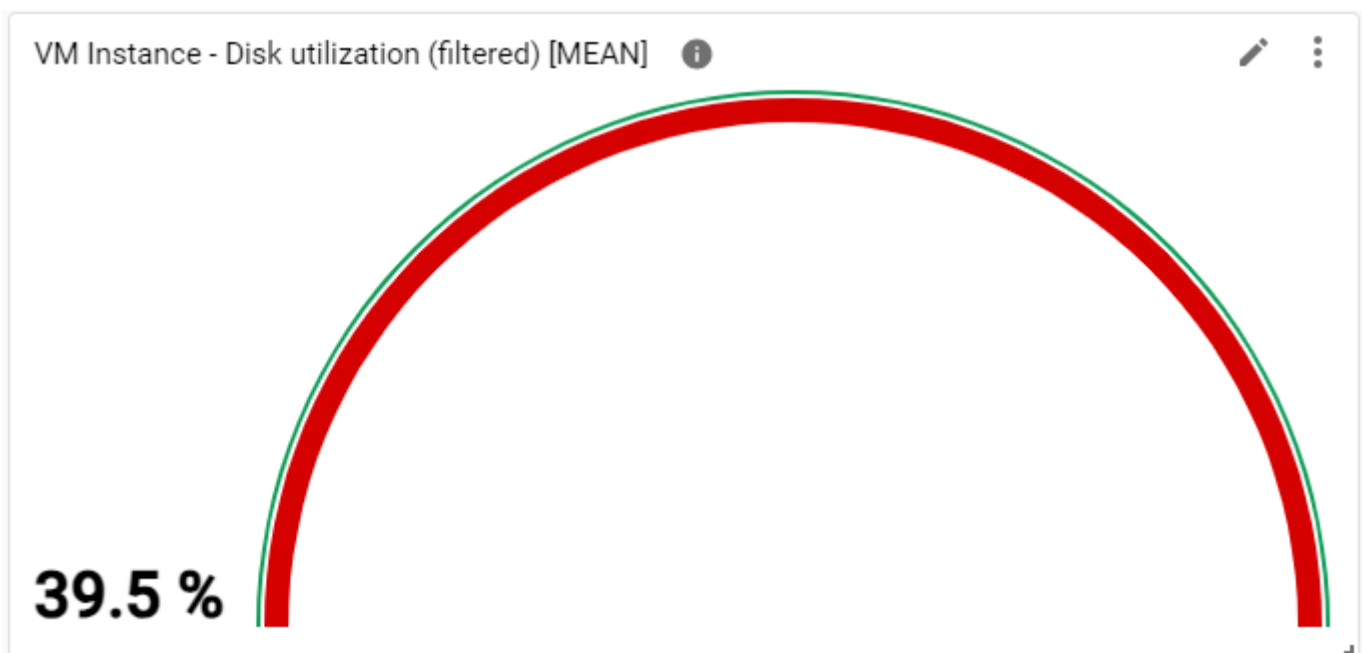
```
# Supongamos que deseamos crear un archivo de 2 GB en /tmp

# Utiliza dd para crear un archivo de 2 GB lleno de datos aleatorios
maitane@as2-maitane:~$ dd if=/dev/urandom of=/tmp/archivo_aleatorio bs=1M
count=2048

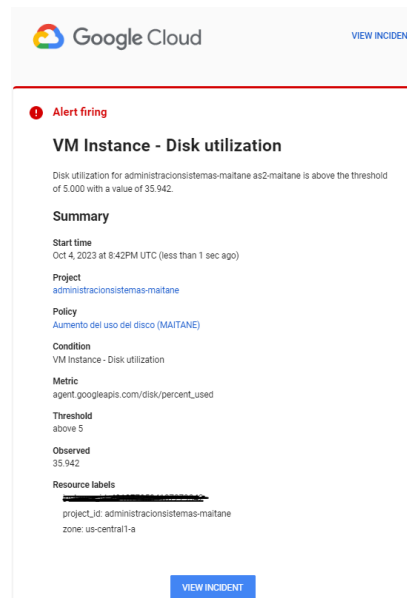
# Esto creará un archivo llamado "archivo_aleatorio" en /tmp
# El parámetro "if" especifica la fuente de datos aleatorios (urandom).
# "of" especifica la ubicación y el nombre del archivo.
# "bs" especifica el tamaño del bloque (en este caso, 1M = 1 megabyte).
# "count" especifica cuántos bloques se deben copiar (en este caso, 2048 bloques
de 1 megabyte cada uno, lo que da como resultado un archivo de 2 GB).
```

4. Verificad que el incremento de consumo en disco se refleja en el Dashboard creado en el paso anterior.

```
maitane@as2-maitane:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        20G   12G   7.5G  62% /
```



5. Verificad que habéis recibido uno (o varios) correos de Google Cloud con la alerta.



6. Elimina los ficheros creados en el paso 3 y la alerta creada en el paso 2.

Evaluación del rendimiento

En esta parte del laboratorio se propone para medir el rendimiento de la CPU del sistema. Si dispones de un equipo personal con Linux, se recomienda hacer las pruebas en tu propio equipo en lugar de utilizar la máquina virtual de GCP para obtener resultados relativos a un hardware no virtualizado.

Las primeras pruebas serán para medir el rendimiento de la CPU con el benchmark Firestarter:

1. Descargar el benchmark desde la web oficial: <https://tu-dresden.de/zih/firestarter>

Descargar el .tar.gz del link que aparece en el apartado de descargas y (en el mismo equipo):

```
maitane@DESKTOP-02F7CUF:~$ tar -xzf ./FIRESTARTER_2-0-tar.gz
```

2. Ejecutarlo durante 30 segundos, haciendo que muestre información adicional (un "report" de rendimiento). Buscar los parámetros necesarios para ello. ¿Qué valor de GFLOP/s obtienes? Este valor es una métrica de la capacidad de cómputo de tu CPU.

```
maitane@DESKTOP-02F7CUF:~$ ./FIRESTARTER -r -t 30
```

...

performance report:

```
Thread 0: 38981825 iterations, tsc_delta: 84182558134
Thread 1: 39285287 iterations, tsc_delta: 84431592831
Thread 2: 38800020 iterations, tsc_delta: 84335714468
Thread 3: 38946067 iterations, tsc_delta: 84390635089
Thread 4: 38896785 iterations, tsc_delta: 84493481137
Thread 5: 38940220 iterations, tsc_delta: 84494196325
Thread 6: 39390025 iterations, tsc_delta: 84608969369
Thread 7: 39371664 iterations, tsc_delta: 84498526140
```

```
total iterations: 312611893
runtime: 30.19 seconds (84609268733 cycles)

estimated floating point performance: 169.02 GFLOPS
estimated memory bandwidth*: 9.28 GB/s
```

* this estimate is highly unreliable if `--function` is used in order to select a `function` that is not optimized for your architecture, or if FIRESTARTER is executed on an unsupported architecture!

Salen **169.02 GFLOPS**.

- La mayoría de CPUs modernas tienen varios conjuntos de instrucciones que permiten obtener diferentes niveles de rendimiento. Firestarter es capaz de detectar los conjuntos de instrucciones disponibles en la CPU y usarlos para evaluar su capacidad. Encuentra el parámetro que muestra los tipos disponibles en tu CPU.

```
maitane@DESKTOP-02F7CUF:~$ ./FIRESTARTER --list-instruction-groups
available instruction-groups for payload AVX512:

L1_BROADCAST,L1_L,L1_LS,L1_S,L2_L,L2_LS,L2_S,L3_L,L3_LS,L3_P,L3_S,RAM_L,RAM_LS,RAM_P,RAM_S,REG
```

- Crea un script que ejecute Firestarter con cada tipo de instrucción disponible en tu CPU durante 30 segundos y reporte el mayor valor de GFLOP/s obtenido. Este será el mayor rendimiento que el benchmark es capaz de obtener en tu CPU. El script en cuestión (4.sh):

```
#!/bin/bash

max=0.0
valor=0.0

list=$(./FIRESTARTER --list-instruction-groups | grep -A1 "available instruction-groups for payload AVX512:" | awk 'NR==2')
IFS=', ' read -ra instructions <<< "$list"

for instruction in "${instructions[@]}; do
    svalor=$(./FIRESTARTER --run-instruction-groups $instruction:1 -r -t 30 | grep "estimated floating point performance" | cut -d ' ' -f5)
    valor=$(echo "$svalor" | bc -l)
    resultado=$(echo "$valor > $max" | bc -l)

    if [ "$resultado" -eq 1 ]; then
        max="$valor"
    fi
done

echo "El valor máximo de GFLOPS es: $max"
```

Una vez creado lo ejecutamos de la siguiente manera y el resultado que nos da es:

```
maitane@DESKTOP-02F7CUF:~$ chmod +x ./4.sh
maitane@DESKTOP-02F7CUF:~$ ./4.sh
...
El valor máximo de GFLOPS es: 141.08
```

5. Crea un script similar al anterior, pero que devuelva el máximo valor obtenido para el ancho de banda de memoria (cuántos GB/s).

El script (5.sh):

```
#!/bin/bash
list=$(./FIRESTARTER --list-instruction-groups | grep -A1 "available instruction-
groups for payload AVX>max=0.0
valor=0.0
IFS=' ' read -ra instructions <<< "$list"
for instruction in "${instructions[@]"; do
    svalor=$(./FIRESTARTER --run-instruction-groups $instruction:1 -r -t 30 |
grep "estimated memory bandwidth" | cut -d ' ' -f4)
    valor=$( echo "$svalor" | bc -l)
    resultado=$(echo "$valor > $max" | bc -l)
    if [ "$resultado" -eq 1 ]; then
        max="$valor"
    fi
    echo "$instruction"
done
echo "El valor máximo para el ancho de banda de memoria es: $max"
```

Para ejecutarlo:

```
maitane@DESKTOP-02F7CUF:~$ chmod +x 5.sh
maitane@DESKTOP-02F7CUF:~$ ./5.sh
...
El valor máximo para el ancho de banda de memoria es: 46.46
```