6. Troubleshooting

1. Pruebas de stress

a. Nos cambiamos al entorno de trabajo **vagrant_perf** y arrancamos la máquina virtual. NOTA: en este caso arrancamos una ubuntu/bionic, con lo que es posible que se tenga que descargar de internet.
b. Nos conectamos a la máquina virtual como usuario root.
c. Instalamos la herramienta para realizar pruebas de stress con **apt install stress-ng**.
d. Ejecutamos varias pruebas de stress y analizamos con las herramientas explicadas en la documentación las posibles métricas: top, htop, free, vmstat,
    i. Operaciones matriciales en 1 CPU: stress-ng --matrix 1 -t 1m
    ii. Operaciones matriciales en todas las CPU: stress-ng --matrix 0 -t 1m
    iii. Transferencia de mensajes utilizando una cola de mensajes POSIX: stress-ng --mq 0 -t 30s --times --perf
    iv. Reserva de memoria: stress-ng --brk 2 --stack 2 --bigheap 2
    v. Diferente tipo de pruebas de stress: stress-ng --seq 4 -t 20
    vi. Más ejemplos en: https://wiki.ubuntu.com/Kernel/Reference/stress-ng

2. Instalación de PCP y Vector

a. Nos conectamos a la máquina como usuario root en el entorno vagrant_perf
b. Instalar los paquetes correspondientes a PCP: **apt-get install -y pcp pcp-webapi libpcp-pmda3-dev libpcp3-dev**.
c. Verificamos que los servicios pmcd, pmlogger y pmwebd han arrancado: **systemctl status pm***.
d. A continuación descargamos el código fuente de vector y lo descomprimimos: **mkdir vector && tar -zxvf vector.tar.gz -C vector**
e. A continuación le indicamos al demonio pmwebd que arranque la aplicación que hemos descomprimido en la carpeta vector en el puerto 8080: **/usr/lib/pcp/bin/pmwebd -R vector -p 8080**
f. Ya nos podremos conectar con un navegador a http://localhost:8080 y analizar métricas de nuestra máquina virtual.

3. Repetir las pruebas de stress y monitorizar con Vector

a. Con vector arrancado lanzar las mismas pruebas que en el ejercicio 6.1 y analizar las métricas.