

Memoria - Monitorización Servidor Linux

Autor: Lucas Hidalgo Herrera

Asignatura: Ingeniería de Servidores

Grado: Doble Grado en Ingeniería Informática y Matemáticas



1. Introducción

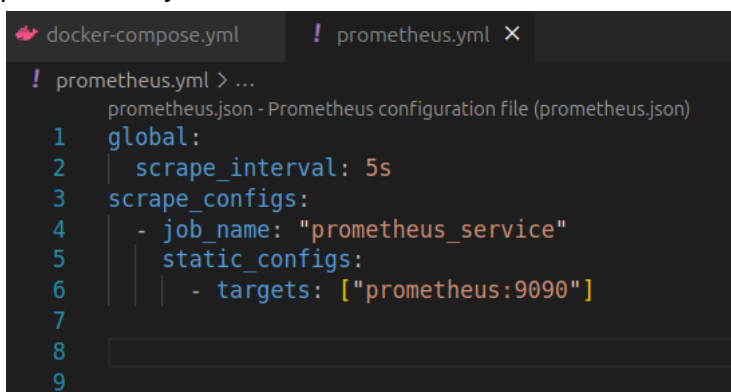
Tal y como aparece en el guión de prácticas, esta memoria contendrá el desarrollo de la primera parte del tercer ejercicio obligatorio. Es decir, contiene la configuración de *Prometheus+Grafana* junto con la integración de *Node Exporter* como servicio desde la máquina virtual *Rocky*.

Además, contiene una comprobación de funcionamiento de la monitorización de los servicios *Sshd* y *Apache Httpd*. De la misma forma, desarrolla la creación de una alerta en un panel de la *Dashboard* así como una comprobación de funcionamiento de la misma.

2. Configuración de la zona de trabajo

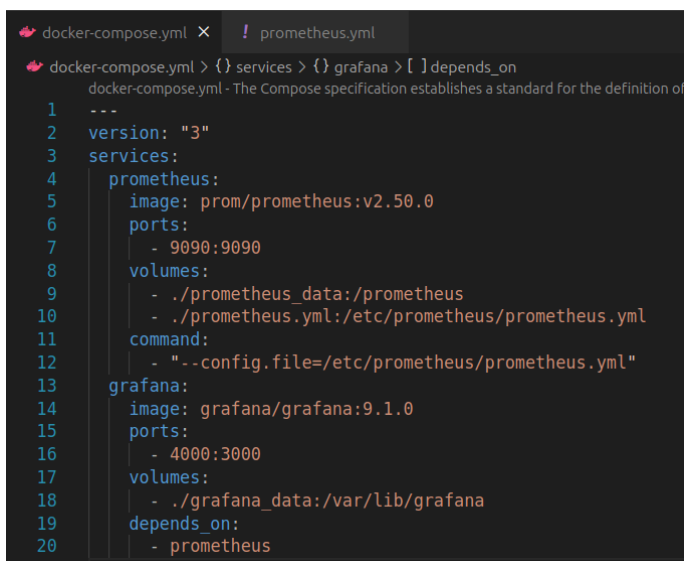
Como punto de partida, usaremos la configuración aportada en el guión con los archivos *docker_compose.yml* y *prometheus.yml*, que crean una instancia inicial de *Grafana+Prometheus* de manera que podemos ejecutar métricas como “*prometheus_engine_query_duration_seconds{slice="inner_eval",quantile="0.5"}"*”:

prometheus.yml:



```
! prometheus.yml > ...
prometheus.json - Prometheus configuration file (prometheus.json)
1 global:
2   scrape_interval: 5s
3 scrape_configs:
4   - job_name: "prometheus_service"
5     static_configs:
6       - targets: ["prometheus:9090"]
7
8
9
```

docker_compose.yml:



```
docker-compose.yml > {} services > {} grafana > [ ] depends_on
docker-compose.yml - The Compose specification establishes a standard for the definition of
1 ---
2 version: "3"
3 services:
4   prometheus:
5     image: prom/prometheus:v2.50.0
6     ports:
7       - 9090:9090
8     volumes:
9       - ./prometheus_data:/prometheus
10      - ./prometheus.yml:/etc/prometheus/prometheus.yml
11     command:
12       - "--config.file=/etc/prometheus/prometheus.yml"
13   grafana:
14     image: grafana/grafana:9.1.0
15     ports:
16       - 4000:3000
17     volumes:
18       - ./grafana_data:/var/lib/grafana
19     depends_on:
20       - prometheus
```

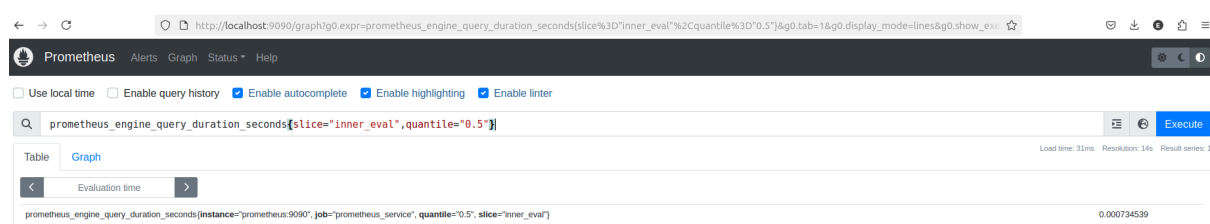
Ya solo nos queda establecer como *Data source* a *Prometheus*:

The image shows the Prometheus configuration interface in Grafana. At the top, the 'Name' is set to 'Prometheus' and the 'Default' toggle is turned on. Under the 'HTTP' section, the 'URL' is 'http://prometheus:9090', 'Access' is 'Server (default)', 'Allowed cookies' is 'New tag (enter key to add)', and 'Timeout' is 'Timeout in seconds'. The 'Auth' section has several toggles: 'Basic auth' (off), 'With Credentials' (off), 'TLS Client Auth' (off), 'With CA Cert' (off), 'Skip TLS Verify' (off), and 'Forward OAuth Identity' (off). There is a 'Custom HTTP Headers' section with an 'Add header' button. The 'Alerting' section has 'Manage alerts via Alerting UI' turned on and 'Alertmanager data source' set to 'Choose'.

The image shows the 'Data sources' list in Grafana. At the top, there is a search bar and an 'Add data source' button. Below, the 'Prometheus' data source is listed with its icon, name, URL 'http://prometheus:9090', and a 'default' status tag.

Podemos ya ver que todo funciona correctamente ejecutando la métrica comentada anteriormente.

Funcionamiento de la métrica:



2.1. Configuración de la máquina virtual.

No obstante, lo que nosotros buscamos es poder monitorizar alguna de las máquinas virtuales que tenemos creadas en *VirtualBox* con el sistema operativo *Rocky Linux*. Para ello, usaremos el exportador de métricas *Node_exporter* como servicio de la máquina virtual.

Para crear el servicio seguiremos las directrices de la referencia [\[51\]](#) del guión. Procedemos a realizar los pasos en la máquina virtual con dirección privada **192.168.57.2**:

```
[lucas@lhhMV01-22:33:49 ~]$ifconfig
emp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:feb7:dea2 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:b7:de:a2 txqueuelen 1000 (Ethernet)
    RX packets 33 bytes 3638 (3.5 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 49 bytes 4207 (4.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

emp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.57.2 netmask 255.255.255.255 broadcast 0.0.0.0
    inet6 fe80::6e6a:ac88:bae6:49f0 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:f7:2b:91 txqueuelen 1000 (Ethernet)
    RX packets 1 bytes 86 (86.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 17 bytes 1304 (1.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Nota: Por comodidad, a partir de aquí, trabajaremos en una sesión de SSH desde el dispositivo *host*, finalmente se mostrarán los resultados desde la máquina virtual.

Descargamos el paquete de node_exporter y lo desempaquetamos:

```
[lucas@lhhhost:22:42:44 programa]$ ssh lucas@192.168.57.2
Last login: Mon May 19 22:41:55 2025 from 192.168.57.1
[lucas@lhhMV01-22:42:46 ~]$ cd /tmp
-bash: cd: command not found
[lucas@lhhMV01-22:42:49 ~]$ cd /tmp
[lucas@lhhMV01-22:42:52 tmp]$ curl -LO https://github.com/prometheus/node_exporter/releases/download/v0.18.1/node_exporter-0.18.1.linux-amd64.tar.gz
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
0 0 0 0 0 0 0 0 0:00:01 0:00:01 0:00:01 28.2M
[lucas@lhhMV01-22:42:57 tmp]$ tar -xvf node_exporter-0.18.1.linux-amd64.tar.gz
node_exporter-0.18.1.linux-amd64/
node_exporter-0.18.1.linux-amd64/node_exporter
node_exporter-0.18.1.linux-amd64/NOTICE
node_exporter-0.18.1.linux-amd64/LICENSE
[lucas@lhhMV01-22:43:02 tmp]$ ls
node_exporter-0.18.1.linux-amd64
node_exporter-0.18.1.linux-amd64.tar.gz
systemd-private-0c69561baa31450fb54ae17d7ef0c1a3-chrond.service-yT0UC
systemd-private-0c69561baa31450fb54ae17d7ef0c1a3-dbus-broker.service-8NcMCR
systemd-private-0c69561baa31450fb54ae17d7ef0c1a3-httpd.service-8d0PF8
systemd-private-0c69561baa31450fb54ae17d7ef0c1a3-systemd-hostnamed.service-fk6bW2
systemd-private-0c69561baa31450fb54ae17d7ef0c1a3-systemd-logind.service-afvz09
[lucas@lhhMV01-22:43:04 tmp]$
```

Movemos el binario a /usr/local/bin:

```
[lucas@lhhMV01-22:43:04 tmp]$ sudo mv node_exporter-0.18.1.linux-amd64/node_exporter /usr/local/bin/
[sudo] password for lucas:
[lucas@lhhMV01-22:44:29 tmp]$ ls
node_exporter-0.18.1.linux-amd64
node_exporter-0.18.1.linux-amd64.tar.gz
systemd-private-0c69561baa31450fb54ae17d7ef0c1a3-chrond.service-yT0UC
systemd-private-0c69561baa31450fb54ae17d7ef0c1a3-dbus-broker.service-8NcMCR
systemd-private-0c69561baa31450fb54ae17d7ef0c1a3-httpd.service-8d0PF8
systemd-private-0c69561baa31450fb54ae17d7ef0c1a3-systemd-hostnamed.service-fk6bW2
systemd-private-0c69561baa31450fb54ae17d7ef0c1a3-systemd-logind.service-afvz09
[lucas@lhhMV01-22:44:31 tmp]$ cd /usr/local/bin
[lucas@lhhMV01-22:44:36 bin]$ ls
node_exporter
[lucas@lhhMV01-22:44:39 bin]$
```

Creamos un usuario que ejecutará el servicio, así como creamos el servicio:

```
[lucas@lhhMV01-22:46:46 bin]$sudo useradd -rs /bin/false node_exporter
useradd: user 'node_exporter' already exists
[lucas@lhhMV01-22:46:47 bin]$sudo userdel node_exporter
[lucas@lhhMV01-22:47:45 bin]$sudo useradd -rs /bin/false node_exporter
[lucas@lhhMV01-22:47:50 bin]$sudo vi /etc/systemd/system/node_exporter.service
[lucas@lhhMV01-22:48:58 bin]$cat /etc/systemd/system/node_exporter.service
[Unit]
Description=Node Exporter
After=network.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter

[Install]
WantedBy=multi-user.target
[lucas@lhhMV01-22:49:16 bin]$
```

Debido a que necesitaremos métricas que no se encuentran de forma natural en las librerías de *Node Exporter*, usaremos *collectors* relacionados con servicios del sistema como *systemd*; por tanto, en la última línea de la sección *[Service]* añadiremos *--collector.systemd*:

```
[lucas@lhhMV01-22:49:16 bin]$sudo vi /etc/systemd/system/node_exporter.service
[lucas@lhhMV01-22:53:07 bin]$cat /etc/systemd/system/node_exporter.service
[Unit]
Description=Node Exporter
After=network.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter --collector.systemd

[Install]
WantedBy=multi-user.target
[lucas@lhhMV01-22:53:11 bin]$
```

Una vez hecho esto, podemos ya habilitar el servicio, tras darle los permisos de ejecución necesarios.

```
[lucas@lhhMV01-22:59:04 bin]$systemctl status node_exporter
● node_exporter.service - Node Exporter
   Loaded: loaded (/etc/systemd/system/node_exporter.service; disabled; vendor preset: disabled)
   Active: active (running) since Mon 2025-05-19 22:58:58 CEST; 21s ago
     Main PID: 11367 (node_exporter)
        Tasks: 4 (limit: 11109)
       Memory: 2.4M
          CPU: 9ms
     CGroup: /system.slice/node_exporter.service
            └─11367 /usr/local/bin/node_exporter --collector.systemd

May 19 22:58:58 lhhMV01 node_exporter[11367]: time="2025-05-19T22:58:58+02:00" level=info msg=" - stat" source="node_exporter.go:104"
May 19 22:58:58 lhhMV01 node_exporter[11367]: time="2025-05-19T22:58:58+02:00" level=info msg=" - systemd" source="node_exporter.go:104"
May 19 22:58:58 lhhMV01 node_exporter[11367]: time="2025-05-19T22:58:58+02:00" level=info msg=" - textfile" source="node_exporter.go:104"
May 19 22:58:58 lhhMV01 node_exporter[11367]: time="2025-05-19T22:58:58+02:00" level=info msg=" - time" source="node_exporter.go:104"
May 19 22:58:58 lhhMV01 node_exporter[11367]: time="2025-05-19T22:58:58+02:00" level=info msg=" - timex" source="node_exporter.go:104"
May 19 22:58:58 lhhMV01 node_exporter[11367]: time="2025-05-19T22:58:58+02:00" level=info msg=" - uname" source="node_exporter.go:104"
May 19 22:58:58 lhhMV01 node_exporter[11367]: time="2025-05-19T22:58:58+02:00" level=info msg=" - vmstat" source="node_exporter.go:104"
May 19 22:58:58 lhhMV01 node_exporter[11367]: time="2025-05-19T22:58:58+02:00" level=info msg=" - xfs" source="node_exporter.go:104"
May 19 22:58:58 lhhMV01 node_exporter[11367]: time="2025-05-19T22:58:58+02:00" level=info msg=" - zfs" source="node_exporter.go:104"
May 19 22:58:58 lhhMV01 node_exporter[11367]: time="2025-05-19T22:58:58+02:00" level=info msg="Listening on :9100" source="node_exporter.go:170"
[lucas@lhhMV01-22:59:19 bin]$
```

Cabe destacar que, antes de que todo estuviera correcto, ocurrió un fallo debido a que *SELinux* se encontraba en modo *enforcing*, es decir, estaba impidiendo la ejecución del servicio. Esto ocurre porque el servicio no tiene los permisos de ejecución correctos; por tanto, sólo impuse permisos de ejecución :

```
sudo restorecon -v /usr/local/bin/node_exporter
```

```
[root@lhhMU01-13:29:16 ~]# restorecon -v /usr/local/bin/node_exporter
Relabeled /usr/local/bin/node_exporter from unconfined_u:object_r:httpd_sys_content_t:s0 to unconfined_u:object_r:bin_t:s0
```

Ahora, comprobamos el correcto funcionamiento desde la máquina virtual para corroborar que el proceso que se ha seguido es correcto:

```
[lucas@lhhMU01-23:05:42 tmp]$systemctl status node_exporter
● node_exporter.service - Node Exporter
   Loaded: loaded (/etc/systemd/system/node_exporter.service; disabled; vendor preset: disabled)
   Active: active (running) since Mon 2025-05-19 22:58:58 CEST; 6min ago
     Main PID: 11367 (node_exporter)
        Tasks: 4 (Limit: 11109)
       Memory: 2.4M
          CPU: 10ms
    CGroup: /system.slice/node_exporter.service
            └─11367 /usr/local/bin/node_exporter --collector.systemd

May 19 22:58:58 lhhMU01 node_exporter[11367]: time="2025-05-19T22:58:02:00" level=info msg=" - s"
May 19 22:58:58 lhhMU01 node_exporter[11367]: time="2025-05-19T22:58:02:00" level=info msg=" - s"
May 19 22:58:58 lhhMU01 node_exporter[11367]: time="2025-05-19T22:58:02:00" level=info msg=" - t"
May 19 22:58:58 lhhMU01 node_exporter[11367]: time="2025-05-19T22:58:02:00" level=info msg=" - t"
May 19 22:58:58 lhhMU01 node_exporter[11367]: time="2025-05-19T22:58:02:00" level=info msg=" - t"
May 19 22:58:58 lhhMU01 node_exporter[11367]: time="2025-05-19T22:58:02:00" level=info msg=" - u"
May 19 22:58:58 lhhMU01 node_exporter[11367]: time="2025-05-19T22:58:02:00" level=info msg=" - u"
May 19 22:58:58 lhhMU01 node_exporter[11367]: time="2025-05-19T22:58:02:00" level=info msg=" - x"
May 19 22:58:58 lhhMU01 node_exporter[11367]: time="2025-05-19T22:58:02:00" level=info msg=" - x"
May 19 22:58:58 lhhMU01 node_exporter[11367]: time="2025-05-19T22:58:02:00" level=info msg=" - z"
May 19 22:58:58 lhhMU01 node_exporter[11367]: time="2025-05-19T22:58:02:00" level=info msg="List"
lines 1-20/20 (END)
```

Ya solo nos queda abrir el puerto **9100**, es decir, el puerto por defecto de exportación de métricas de *Node Exporter*.

```
[root@lhhMU01-23:20:45 tmp]# firewall-cmd --remove-port=9100/tcp
success
[root@lhhMU01-23:21:18 tmp]# firewall-cmd --permanent --add-port=9100/tcp
success
[root@lhhMU01-23:21:36 tmp]# firewall-cmd --list-ports
80/tcp
[root@lhhMU01-23:21:56 tmp]# firewall-cmd --reload
success
[root@lhhMU01-23:22:57 tmp]# firewall-cmd --list-ports
80/tcp 9100/tcp
[root@lhhMU01-23:22:59 tmp]# _
```

2.2. Configuración del host y comprobación de funcionamiento

Por último, solo queda establecer una regla en el archivo *prometheus.yml* para determinar que *Prometheus* debe monitorizar nuestra máquina virtual, que llamaremos *node_exporter_vm*.

Por defecto, *Node Exporter* expone sus métricas en el puerto **9100**; luego, el archivo *prometheus.yml* queda como sigue:

```
docker-compose.yml  ! prometheus.yml X

! prometheus.yml > [ ] scrape_configs > { } 1 > [ ] static_configs > { } 0 >
prometheus.json - Prometheus configuration file (prometheus.json)
1 global:
2   scrape_interval: 5s
3 scrape_configs:
4   - job_name: "prometheus_service"
5     static_configs:
6       - targets: ["prometheus:9090"]
7
8   - job_name: "node_exporter_vm"
9     static_configs:
10      - targets: ["192.168.57.2:9100"]
11
```

Ya solo queda lanzar los contenedores y comprobar que todo funciona correctamente, para lo que usaremos el apartado *Target* de *Prometheus*.

Lanzamos el contenedor:

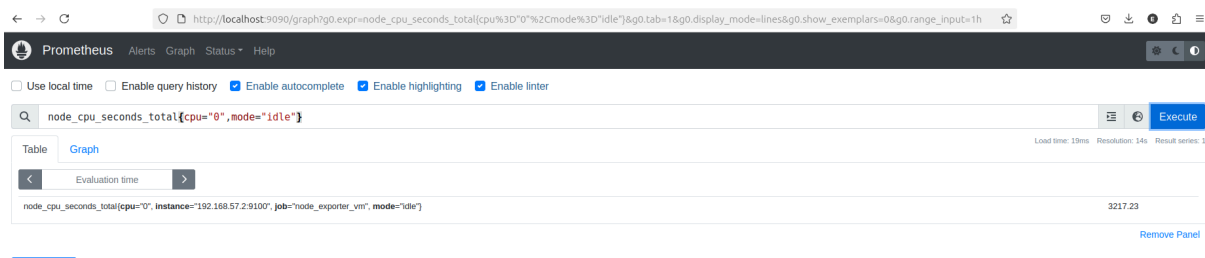
```
[lucas@lhhhost:23:13:34 progra]$ docker compose up
WARN[0000] /home/el_dramas/Desktop/Infomates/Tercer_Curso/Segundo_Cuatri/ISE/Practicas/Bloque_2/Monitoring/progra/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
WARN[0000] Found orphan containers ([progra-systemd-exporter-1 progra-node_exporter-1]) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up.
[+] Running 2/2
 ✓ Container progra-grafana-1      Created           0.0s
 ✓ Container progra-prometheus-1   Recreated      0.1s
Attaching to grafana-1, prometheus-1
prometheus-1 | ts=2025-05-19T21:13:36.658Z caller=main.go:564 level=info msg="No time or size retention was set so using the default time retention" duration=15d
prometheus-1 | ts=2025-05-19T21:13:36.658Z caller=main.go:608 level=info msg="Starting Prometheus Server" mode=server version="(version=2.50.0, branch=HEAD, revision=814b920e8a6345d35712b5857ebd4cb5e90fc107)"
prometheus-1 | ts=2025-05-19T21:13:36.658Z caller=main.go:613 level=info build_context="(go=go1.21.7, platform=linux/amd64, user=root@384077e1cf50, date=20240222-09:38:19, tags=netgo,builtinassets,stringlabels)"
prometheus-1 | ts=2025-05-19T21:13:36.658Z caller=main.go:614 level=info host_details="(Linux 6.8.0-60-generic #63-Ubuntu SMP PREEMPT_DYNAMIC Tue Apr 15 19:04:15 UTC 2025 x86_64 14c2ac87981a (none))"
prometheus-1 | ts=2025-05-19T21:13:36.658Z caller=main.go:615 level=info fd_limits="(soft=
```

Comprobamos los targets:

The screenshot shows the Prometheus web interface at <http://localhost:9090/targets?search=>. The 'Targets' page lists two target groups:

- node_exporter_vm (1/1 up)**: Contains one target with endpoint `http://192.168.57.2:9100/metrics`, state `UP`, and labels `instance="192.168.57.2:9100"` and `job="node_exporter_vm"`. The last scrape was 954.000ms ago, with a duration of 133.456ms.
- prometheus_service (1/1 up)**: Contains one target with endpoint `http://prometheus:9090/metrics`, state `UP`, and labels `instance="prometheus:9090"` and `job="prometheus_service"`. The last scrape was 1.195s ago, with a duration of 10.683ms.

Una vez hecho esto, podemos ya probar una métrica exportada de *Node Exporter* como *node_cpu_seconds_total{cpu="0",mode="idle"}* para comprobar el correcto funcionamiento del entorno de trabajo.



Hemos terminado ya la configuración del entorno de trabajo; para acabar esta sección, cabe destacar que los ejemplos se realizarán sobre un *Dashboard* ya creado que ha sido modificado por el alumno y ha sido nombrado *Monitoring my pc*.

3. Monitorización de los servicios de Apache Httpd y Sshd

Tanto para la monitorización de Apache como para la de Sshd usaremos métricas de *systemd*, de ahí la integración del *collector* en la configuración del servicio de *Node Exporter*. La métricas que usaremos son:

- **Sshd:** *node_systemd_unit_state{name="sshd.service", state="active"}*
- **Apache Httpd:** *node_systemd_unit_state{name="sshd.service", state="active"}*

Inicialmente, ambos servicios se encuentran activos; por lo que, los paneles deberán indicar que así es como se encuentran los servicios.

Sshd y Apache Httpd están activos:

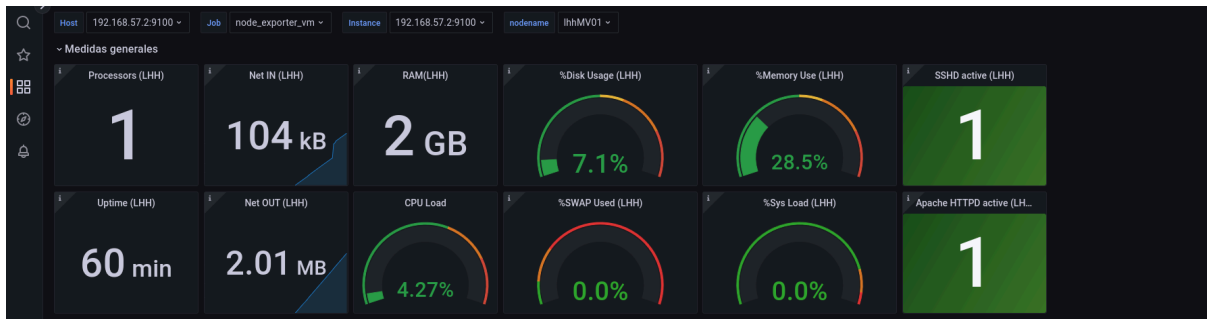
```
May 19 22:36:24 lhhMU01 systemd[1]: Starting OpenSSH server daemon: sshd.service...
[root@lhhMU01 ~]# systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2025-05-19 22:31:02 CEST; 1h 5min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Main PID: 692 (sshd)
    Tasks: 1 (limit: 11109)
   Memory: 5.7M
      CPU: 505ms
   CGroup: /system.slice/sshd.service
           └─692 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

May 19 22:37:21 lhhMU01 sshd[1388]: Accepted publickey for lucas from 192.168.57.1 port 54182 ssh2:
May 19 22:37:21 lhhMU01 sshd[1388]: pam_unix(sshd:session): session opened for user lucas(uid=1000)

May 19 22:39:37 lhhMU01 systemd[1]: Starting The Apache HTTP Server: httpd.service...
[root@lhhMU01 ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2025-05-19 22:31:03 CEST; 1h 8min ago
     Docs: man:httpd.service(8)
  Main PID: 690 (httpd)
   Status: "Total requests: 0; Idle/Busy workers 100/0; Requests/sec: 0; Bytes served/sec: 0 B/s"
    Tasks: 177 (limit: 11109)
   Memory: 26.9M
      CPU: 2.667s
   CGroup: /system.slice/httpd.service
           └─690 /usr/sbin/httpd -DFOREGROUND
             └─725 /usr/sbin/httpd -DFOREGROUND
               └─726 /usr/sbin/httpd -DFOREGROUND
                 └─727 /usr/sbin/httpd -DFOREGROUND
                   └─728 /usr/sbin/httpd -DFOREGROUND

May 19 22:31:02 lhhMU01 systemd[1]: Starting The Apache HTTP Server...
```


Por tanto, el estado de los paneles con los nombres **SSHD Active** y **Apache HTTPD active** debe ser verdadero o 1:



Procedemos a detener los servicios para ver que los paneles pasan a mostrar 0 o falso:

Apagamos servicios:

```
root@lhhMV01:~# systemctl status httpd
■ httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: inactive (dead) since Mon 2025-05-19 23:44:04 CEST; 3min 58s ago
     Docs: man:httpd.service(8)
  Process: 690 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=0/SUCCESS)
 Main PID: 690 (code=exited, status=0/SUCCESS)
    Status: "Total requests: 0; Idle/Busy workers 100/0; Requests/sec: 0; Bytes served/sec: 0 B/s"
      CPU: 2.847s

May 19 22:31:02 lhhMV01 systemd[1]: Starting The Apache HTTP Server...
May 19 22:31:03 lhhMV01 httpd[690]: AH00558: httpd: Could not reliably determine the server's fully
```

```
[root@lhhMV01-23:49:07 tmp]# systemctl stop sshd
[root@lhhMV01-23:49:10 tmp]# systemctl status httpd
■ httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: inactive (dead) since Mon 2025-05-19 23:44:04 CEST; 5min ago
     Docs: man:httpd.service(8)
  Process: 690 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=0/SUCCESS)
 Main PID: 690 (code=exited, status=0/SUCCESS)
    Status: "Total requests: 0; Idle/Busy workers 100/0; Requests/sec: 0; Bytes served/sec: 0 B/s"
      CPU: 2.847s

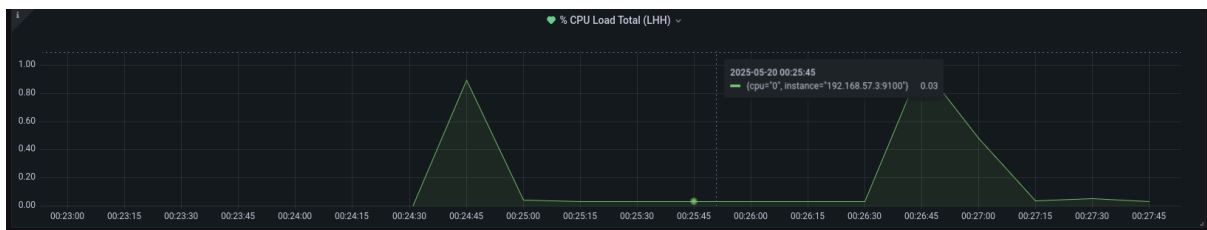
May 19 22:31:02 lhhMV01 systemd[1]: Starting The Apache HTTP Server...
May 19 22:31:03 lhhMV01 httpd[690]: AH00558: httpd: Could not reliably determine the server's fully
May 19 22:31:03 lhhMV01 httpd[690]: Server configured, listening on: port 80
```

Y como cabía de esperar, el resultado que muestran los paneles es el que debería ser:

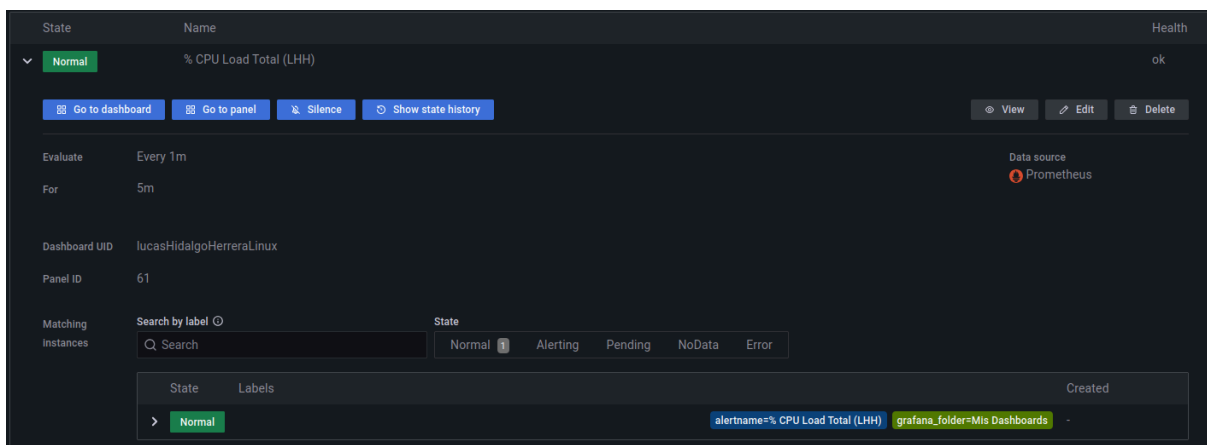


4. Monitorización de uso de la CPU y alerta de sobrecarga

Para monitorizar el uso de la CPU he usado una serie temporal que representa el porcentaje de uso de la cpu en cada instante de tiempo. Inicialmente, el panel tiene este aspecto:



Donde se ve que se ha creado una alerta y que, en estos instantes, todo es correcto; la configuración de la alerta es la siguiente:



Donde pasado el **75%** de forma sostenida durante **5 minutos** provoca que se dispare la alerta.

Para probar que la alerta funciona correctamente vamos a sobrecargar la CPU de la máquina virtual haciendo uso de la herramienta *stress*, mediante la cual vamos a generar 4 procesos realizando iteraciones calculando raíces cuadradas.

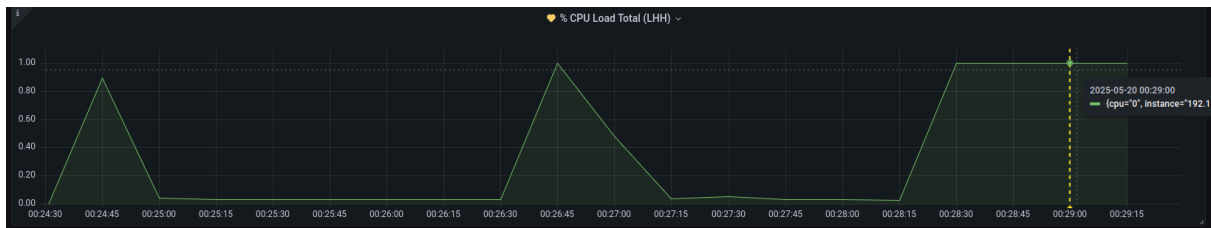
Lucas Hidalgo Herrera

```

[Lucas@lhhMU01-00:27:07 ~]$ stress -c 4
stress: info: [1348] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
^C
[Lucas@lhhMU01-00:35:24 ~]$

```

Tras unos segundos, el porcentaje de uso de la CPU ha aumentado hasta el **80%** entrando en fase de **Warning** de manera que, si pasan **5 minutos** desde este momento en los que el porcentaje de uso no baja del **75%** saltará la alerta.



Pasados esos 5 minutos, ha saltado la alerta comprobando así el correcto funcionamiento de la práctica.

