Creando un sistema de monitorización en red con Raspberry Pi

Xinyuan Li,Sahir Vargas, Pablo Rui Zhang guo

*Server *rp1G2**

Instalamos prometheus:

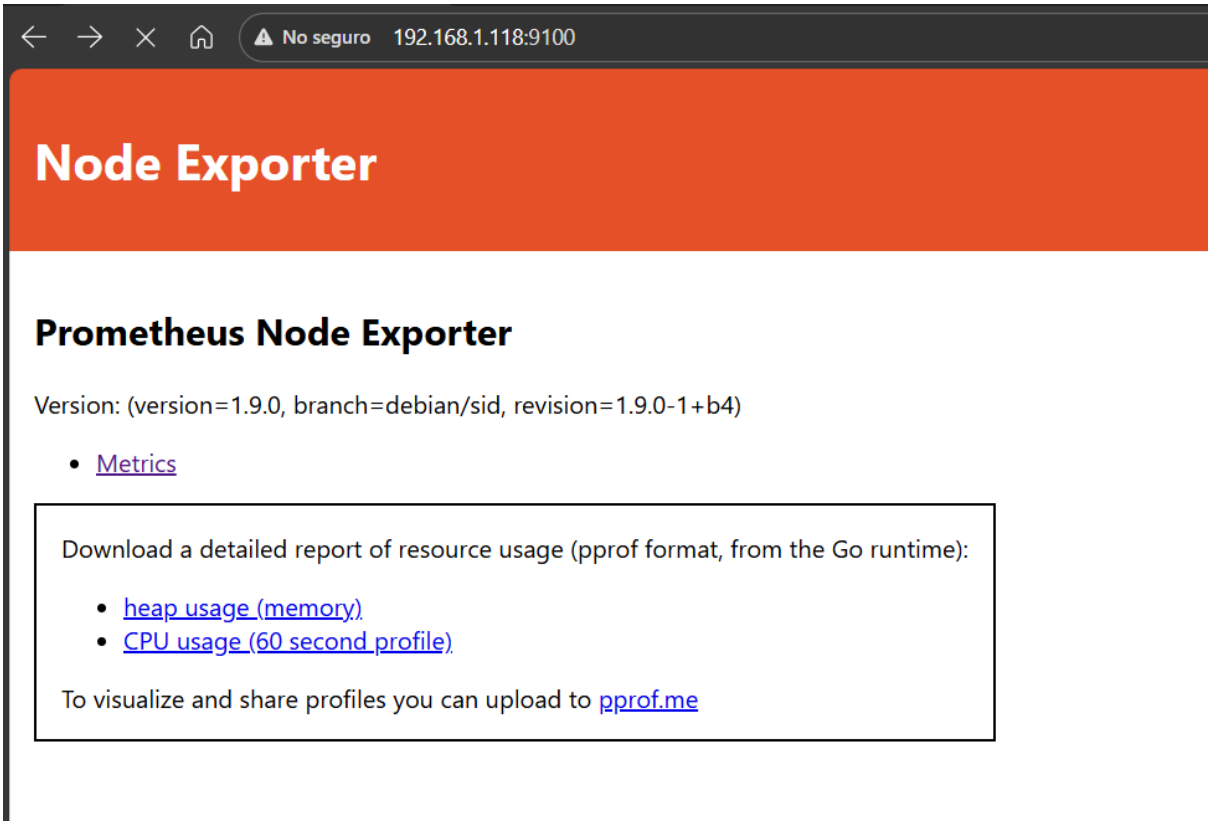
```
pi@rp1G2-srv:~ $ sudo apt install prometheus
Installing:
  prometheus

Installing dependencies:
  fonts-glyphicons-halflings          libjs-jquery-hotkeys    node-jquery
  freeipmi-common                     libjs-moment            nvme-cli
  ipmitool                            libjs-moment-timezone   openipmi
  jq                                  libjs-mustache          prometheus-node-exporter
  libfreeipmi17                       libjs-popper.js         prometheus-node-exporter-collect
  libio-pty-perl                      libjs-rickshaw          promtool
  libipc-run-perl                     libjs-sizzle            python3-decorator
  libjq1                              libonig5                python3-prometheus-client
  libjs-bootstrap                     libopenipmi0t64         smartmontools
```

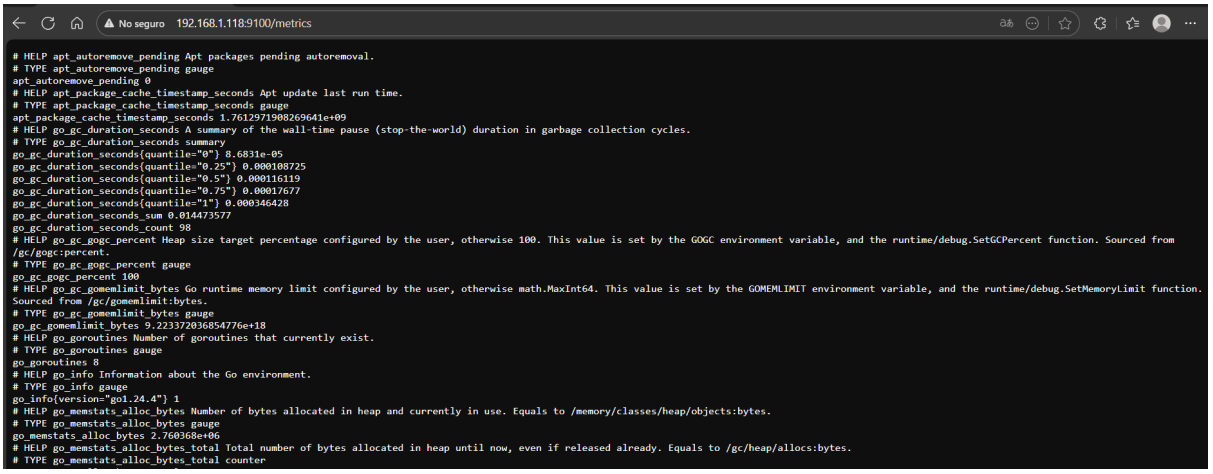
Editamos el archivo de configuración y añadimos las ip de los clientes.

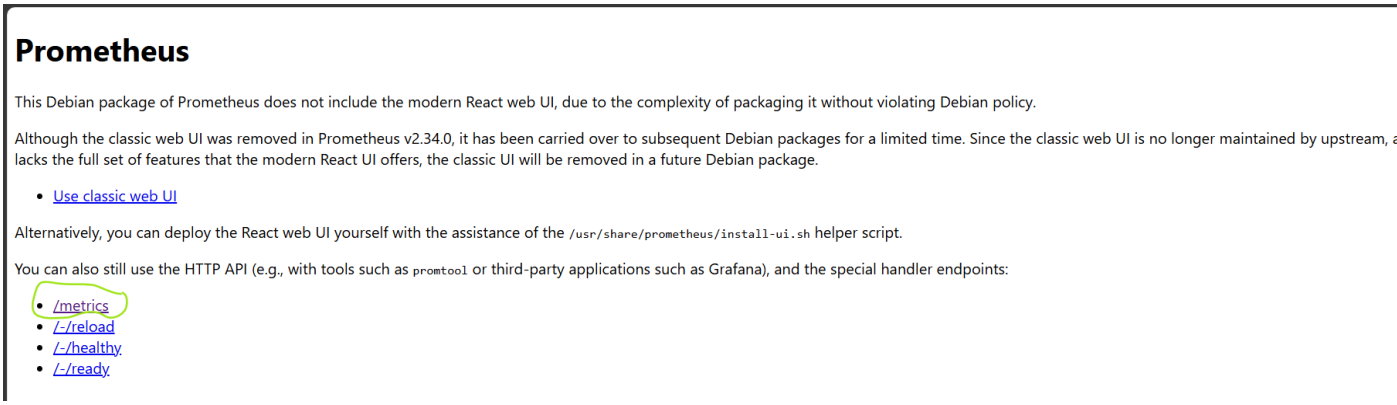
```
pi@rp1G2-srv:~ $ cd /etc/prometheus/
pi@rp1G2-srv:/etc/prometheus $ ls
console_libraries  consoles  prometheus.yml
pi@rp1G2-srv:/etc/prometheus $ sudo nano prometheus.yml
```

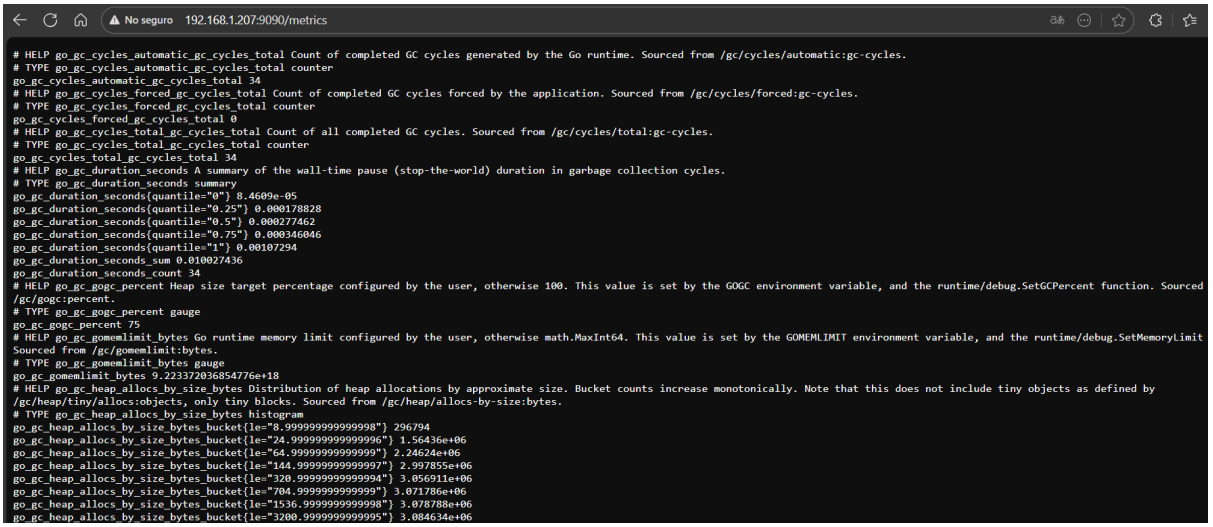
```
# The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
- job_name: 'prometheus'

  # Override the global default and scrape targets from this job every 5 seconds.
  scrape_interval: 5s
  scrape_timeout: 5s

  # metrics_path defaults to '/metrics'
  # scheme defaults to 'http'.

  static_configs:
    - targets: ['localhost:9090']

- job_name: node
  # If prometheus-node-exporter is installed, grab stats about the local
  # machine by default.
  static_configs:
    - targets: ['localhost:9100']

- job_name: 'rpi-nodes'
  scrape_interval: 15s
  static_configs:
    - targets:
        - '192.168.1.118:9100'
```

Y comprobamos:


No seguro 192.168.1.118:9100
Node Exporter
Prometheus Node Exporter
Version: (version=1.9.0, branch=debian/sid, revision=1.9.0-1+b4)
Metrics
Download a detailed report of resource usage (pprof format, from the Go runtime):
heap usage (memory)
CPU usage (60 second profile)
To visualize and share profiles you can upload to pprof.me


No seguro 192.168.1.118:9100/metrics
HELP apt_autoremove_pending Apt packages pending autoremoval.
TYPE apt_autoremove_pending gauge
apt_autoremove_pending 0
HELP apt_package_cache_timestamp_seconds Apt update last run time.
TYPE apt_package_cache_timestamp_seconds gauge
apt_package_cache_timestamp_seconds 1.7612971908269641e+09
HELP go_gc_duration_seconds A summary of the wall-time pause (stop-the-world) duration in garbage collection cycles.
TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 8.6831e-05
go_gc_duration_seconds{quantile="0.25"} 0.000108725
go_gc_duration_seconds{quantile="0.5"} 0.000116119
go_gc_duration_seconds{quantile="0.75"} 0.00017677
go_gc_duration_seconds{quantile="1"} 0.000346428
go_gc_duration_seconds_sum 0.014473577
go_gc_duration_seconds_count 98
HELP go_gc_gogc_percent Heap size target percentage configured by the user, otherwise 100. This value is set by the GOGC environment variable, and the runtime/debug.SetGCPercent function. Sourced from /gc/gogc:percent.
TYPE go_gc_gogc_percent gauge
go_gc_gogc_percent 100
HELP go_gc_gomemlimit_bytes Go runtime memory limit configured by the user, otherwise math.MaxInt64. This value is set by the GOMEMLIMIT environment variable, and the runtime/debug.SetMemoryLimit function. Sourced from /gc/gomemlimit:bytes.
TYPE go_gc_gomemlimit_bytes gauge
go_gc_gomemlimit_bytes 9.223372036854776e+18
HELP go_goroutines Number of goroutines that currently exist.
TYPE go_goroutines gauge
go_goroutines 8
HELP go_info Information about the Go environment.
TYPE go_info gauge
go_info{version="go1.24.4"} 1
HELP go_memstats_alloc_bytes Number of bytes allocated in heap and currently in use. Equals to /memory/classes/heap/objects:bytes.
TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.760368e+06
HELP go_memstats_alloc_bytes_total Total number of bytes allocated in heap until now, even if released already. Equals to /gc/heap/allocs:bytes.
TYPE go_memstats_alloc_bytes_total counter


Prometheus
This Debian package of Prometheus does not include the modern React web UI, due to the complexity of packaging it without violating Debian policy.
Although the classic web UI was removed in Prometheus v2.34.0, it has been carried over to subsequent Debian packages for a limited time. Since the classic web UI is no longer maintained by upstream, a
lacks the full set of features that the modern React UI offers, the classic UI will be removed in a future Debian package.
Use classic web UI
Alternatively, you can deploy the React web UI yourself with the assistance of the /usr/share/prometheus/install-ui.sh helper script.
You can also still use the HTTP API (e.g., with tools such as promtool or third-party applications such as Grafana), and the special handler endpoints:
/metrics
/-/reload
/-/healthy
/-/ready



También instalamos Grafana:

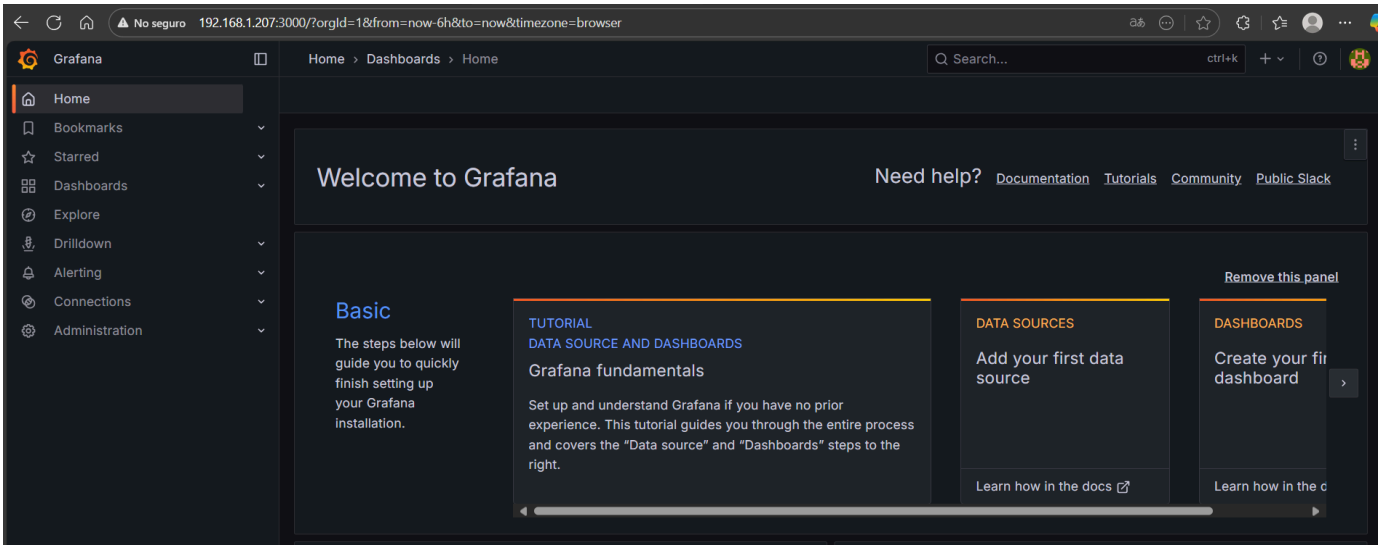
```
pi@rp1G2-srv:~ $ apt-cache search software-properties-common
pi@rp1G2-srv:~ $ echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://packages.grafana.com/oss/deb stable main"
| sudo tee /etc/apt/sources.list.d/grafana.list
deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://packages.grafana.com/oss/deb stable main
pi@rp1G2-srv:~ $ sudo mkdir -p /etc/apt/keyrings/
pi@rp1G2-srv:~ $ wget -q -O - https://packages.grafana.com/gpg.key | sudo gpg --dearmor -o /etc/apt/keyrings/grafana.gp
g
pi@rp1G2-srv:~ $ echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://packages.grafana.com/oss/deb stable main"
| sudo tee /etc/apt/sources.list.d/grafana.list
deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://packages.grafana.com/oss/deb stable main
pi@rp1G2-srv:~ $ sudo apt update
Hit:1 http://deb.debian.org/debian trixie InRelease
Hit:2 http://archive.raspberrypi.com/debian trixie InRelease
Hit:3 http://deb.debian.org/debian trixie-updates InRelease
Hit:4 http://deb.debian.org/debian-security trixie-security InRelease
Get:5 https://packages.grafana.com/oss/deb stable InRelease [7,661 B]
Get:6 https://packages.grafana.com/oss/deb stable/main arm64 Packages [398 kB]
Get:7 https://packages.grafana.com/oss/deb stable/main armhf Packages [476 kB]
Fetched 882 kB in 2s (371 kB/s)
All packages are up to date.
pi@rp1G2-srv:~ $ sudo apt install grafana
Installing:
  grafana

Installing dependencies:
  musl

Summary:
  Upgrading: 0, Installing: 2, Removing: 0, Not Upgrading: 0
```

Activamos el servicio:

```
Processing triggers for man-db (2.13.1-1) ...
pi@rp1G2-srv:~ $ sudo systemctl enable grafana-server
Synchronizing state of grafana-server.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable grafana-server
Created symlink '/etc/systemd/system/multi-user.target.wants/grafana-server.service' → '/usr/lib/systemd/system/grafana
-server.service'.
pi@rp1G2-srv:~ $
```

Y comprobamos

```
pi@rp1G2-srv:~ $ sudo systemctl start grafana-server
pi@rp1G2-srv:~ $ sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
     Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-10-24 10:51:56 CEST; 1min 0s ago
 Invocation: 0d1998caecc844148a6092c4454ee320
       Docs: http://docs.grafana.org
   Main PID: 16937 (grafana)
      Tasks: 19 (limit: 3906)
        CPU: 16.277s
     CGroup: /system.slice/grafana-server.service
             └─16937 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/gr
```



*Clientes *rp2G2**

Instalar Node Exporter:

- wget https://github.com/prometheus/node_exporter/releases/download/v1.8.2/node_exporter-1.8.2.linux-armv7.tar.gz

```
pi@rp2G2:~ $ cd /tmp
pi@rp2G2:/tmp $ wget https://github.com/prometheus/node_exporter/releases/download/v1.8.2/node_exporter-1.8.2.linux-armv7.tar.gz
```

- tar xvf node_exporter-1.8.2.linux-armv7.tar.gz

```
pi@rp2G2:/tmp $ tar xvf node_exporter-1.8.2.linux-armv7.tar.gz
node_exporter-1.8.2.linux-armv7/
node_exporter-1.8.2.linux-armv7/NOTICE
node_exporter-1.8.2.linux-armv7/node_exporter
node_exporter-1.8.2.linux-armv7/LICENSE
```

- sudo mv node_exporter-1.8.2.linux-armv7/node_exporter /usr/local/bin/

```
pi@rp2G2:/tmp $ sudo mv node_exporter-1.8.2.linux-armv7/node_exporter /usr/local/bin/
```

Crea un usuario de sistema:

- sudo useradd -rs /bin/false node_exporter

```
pi@rp2G2:/tmp $ sudo useradd -rs /bin/false node_exporter
pi@rp2G2:/tmp $
```

Crea el servicio systemd:

- sudo nano /etc/systemd/system/node_exporter.service

```
pi@rp2G2:/tmp $ sudo nano /etc/systemd/system/node_exporter.service
```

Pega el siguiente contenido:

```
[Unit]
Description=Node Exporter
After=network.target

[Service]
User=node_exporter
ExecStart=/usr/local/bin/node_exporter
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

```
  GNU nano 8.4                                  /etc/systemd/system/node_exporter.service *
[Unit]
Description=Node Exporter  # Descripción del servicio
After=network.target       # Iniciar después de que la red esté lista

[Service]
User=node_exporter         # Ejecutar como usuario
ExecStart=/usr/local/bin/node_exporter  # Comando de inicio
Restart=on-failure         # Reinicio automático en caso de fallo.

[Install]
WantedBy=multi-user.target # Iniciar en modo multiusuario
```

Después de guardar, vuelva a cargar el servicio, habilítelo e inícielo:

- sudo systemctl daemon-reload
- sudo systemctl enable node_exporter
- sudo systemctl start node_exporter

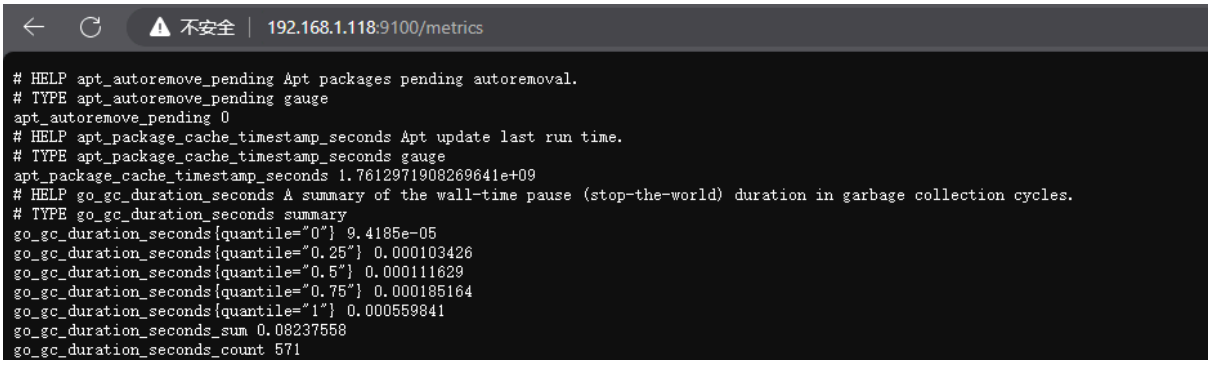
```
pi@rp2G2:/tmp $ sudo systemctl daemon-reload
pi@rp2G2:/tmp $ sudo systemctl enable node_exporter
Created symlink '/etc/systemd/system/multi-user.target.wants/node_exporter.service' → '/etc/systemd/system/node_exporter.service'
.
pi@rp2G2:/tmp $ sudo systemctl start node_exporter
```

Introduzca el siguiente comando para ver el estado operativo:

- sudo systemctl status prometheus-node-exporter

```
pi@rp2G2:/tmp $ sudo systemctl status prometheus-node-exporter
● prometheus-node-exporter.service - Prometheus exporter for machine metrics
     Loaded: loaded (/usr/lib/systemd/system/prometheus-node-exporter.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-10-24 11:16:43 CEST; 31min ago
 Invocation: d239cf64f396424b99ab0de6cc3bcecd
       Docs: https://github.com/prometheus/node_exporter
   Main PID: 15206 (prometheus-node)
      Tasks: 6 (limit: 3906)
        CPU: 22.929s
     CGroup: /system.slice/prometheus-node-exporter.service
             └─15206 /usr/bin/prometheus-node-exporter
```

Abre **http://<IP del cliente>:9100/metrics** en tu navegador.
Verás una amplia gama de métricas del sistema (como CPU, memoria, red, etc.).



Añadir supervisión de la temperatura de la CPU
Crear directorio de recopilación de datos:

- sudo mkdir -p /var/lib/node_exporter/textfile_collector

```
pi@rp2G2:/tmp $ sudo mkdir -p /var/lib/node_exporter/textfile_collector
```

Escribir script de recopilación de temperatura:

- sudo nano /usr/local/bin/rpi_temp.sh

```
pi@rp2G2:/tmp $ sudo nano /usr/local/bin/rpi_temp.sh
```

Introduzca el siguiente código:

```
#!/bin/bash
temp=$(cat /sys/class/thermal/thermal_zone0/temp)
temp_c=$(awk "BEGIN {print $temp/1000}")
echo "rpi_cpu_temp_celsius $temp_c" >
/var/lib/node_exporter/textfile_collector/rpi_temp.prom
```

```
 GNU nano 8.4                                    /usr/local/bin/rpi_temp.sh *
#!/bin/bash
temp=$(cat /sys/class/thermal/thermal_zone0/temp)  # Leer temperatura bruta (unidad: 0,1 °C)
temp_c=$(awk "BEGIN {print $temp/1000}")          # Convertir a grados Celsius
echo "rpi_cpu_temp_celsius $temp_c" > /var/lib/node_exporter/textfile_collector/rpi_temp.prom
```

Conceder permisos de ejecución y configurar tareas programadas:

- sudo chmod +x /usr/local/bin/rpi_temp.sh

```
pi@rp2G2:/tmp $ sudo chmod +x /usr/local/bin/rpi_temp.sh
```

- echo "* * * * * root /usr/local/bin/rpi_temp.sh" | sudo tee /etc/cron.d/rpi_temp

```
pi@rp2G2:/tmp $ echo "* * * * * root /usr/local/bin/rpi_temp.sh" | sudo tee /etc/cron.d/rpi_temp
* * * * * root /usr/local/bin/rpi_temp.sh
```

Clientes *rp3G2*

Nos movemos a la carpeta /tmp

Luego emplearemos un comando para instalar *Node Exporter*

```
pi@rp3G2:/tmp $ wget https://github.com/prometheus/node_exporter/releases/download/v1.8.2/node_exporter-1.8.2.linux-armv7.tar.gz
--2025-10-24 11:30:43--  https://github.com/prometheus/node_exporter/releases/download/v1.8.2/node_exporter-1.8.2.linux-armv7.tar
Resolving github.com (github.com)... 140.82.121.3
Connecting to github.com (github.com)|140.82.121.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://release-assets.githubusercontent.com/github-production-release-asset/9524057/f4932a7d-0556-4154-af14-c89dd16171
r=https&se=2025-10-24T10%3A09%3A25Z&rscd=attachment%3B+filename%3Dnode_exporter-1.8.2.linux-armv7.tar.gz&rsct=application%2Foctet
3a1-aedd-ab1947aa7ab0&sktid=398a6654-997b-47e9-b12b-9515b896b4de&skt=2025-10-24T09%3A08%3A35Z&ske=2025-10-24T10%3A09%3A25Z&sks=b&
di79k23C9VFmOMYGCXSLJ8EnKu0nihjGE%3D&jwt=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJnaXRodWIuY29tIiwiYXVkIjoicmVsZWFzZS1hc3N
Y29tIiwia2V5Ijoia2V5MSIsImV4cCI6MTc2MTI5ODU0NCwibmJmIjoxNzYxMjk4MjQ0LCJwYXRoIjoicmVsZWFzZWFzZ2V0cHJvZHVjdGlvbi5ibG9iLmNvcmUud2luZ
r834202hdgGKZawTwbbNaOy_SwyA&response-content-disposition=attachment%3B%20filename%3Dnode_exporter-1.8.2.linux-armv7.tar.gz&respo
2Foctet-stream [following]
--2025-10-24 11:30:45--  https://release-assets.githubusercontent.com/github-production-release-asset/9524057/f4932a7d-0556-4154-
8-11-09&sr=b&spr=https&se=2025-10-24T10%3A09%3A25Z&rscd=attachment%3B+filename%3Dnode_exporter-1.8.2.linux-armv7.tar.gz&rsct=appl
96c2d410-5711-43a1-aedd-ab1947aa7ab0&sktid=398a6654-997b-47e9-b12b-9515b896b4de&skt=2025-10-24T09%3A08%3A35Z&ske=2025-10-24T10%3A
&sig=lkF8EJQNmOdi79k23C9VFmOMYGCXSLJ8EnKu0nihjGE%3D&jwt=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJnaXRodWIuY29tIiwiYXVkIjoi
XNlcmNvbnRlbnQuY29tIiwia2V5Ijoia2V5MSIsImV4cCI6MTc2MTI5ODU0NCwibmJmIjoxNzYxMjk4MjQ0LCJwYXRoIjoicmVsZWFzZWFzZ2V0cHJvZHVjdGlvbi5ibG9
mwUCqBVkOyA7F0Qr834202hdgGKZawTwbbNaOy_SwyA&response-content-disposition=attachment%3B%20filename%3Dnode_exporter-1.8.2.linux-armv
pe=application%2Foctet-stream
Resolving release-assets.githubusercontent.com (release-assets.githubusercontent.com)... 185.199.109.133, 185.199.108.133, 185.19
Connecting to release-assets.githubusercontent.com (release-assets.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9850356 (9.4M) [application/octet-stream]
Saving to: ‘node_exporter-1.8.2.linux-armv7.tar.gz’

node_exporter-1.8.2.linux-armv7.tar.gz  100%[=========================================================================>]   9.

2025-10-24 11:31:02 (576 KB/s) - ‘node_exporter-1.8.2.linux-armv7.tar.gz’ saved [9850356/9850356]

pi@rp3G2:/tmp $ tar xvf node_exporter-1.8.2.linux-armv7.tar.gz
node_exporter-1.8.2.linux-armv7/
node_exporter-1.8.2.linux-armv7/NOTICE
node_exporter-1.8.2.linux-armv7/node_exporter
node_exporter-1.8.2.linux-armv7/LICENSE
pi@rp3G2:/tmp $ sudo mv node_exporter-1.8.2.linux-armv7/node_exporter /usr/local/bin/
```

Agregamos un usuario y luego creamos un nano

```
pi@rp3G2:/tmp $ sudo useradd -rs /bin/false node_exporter
pi@rp3G2:/tmp $ sudo nano /etc/systemd/system/node_exporter.service
```

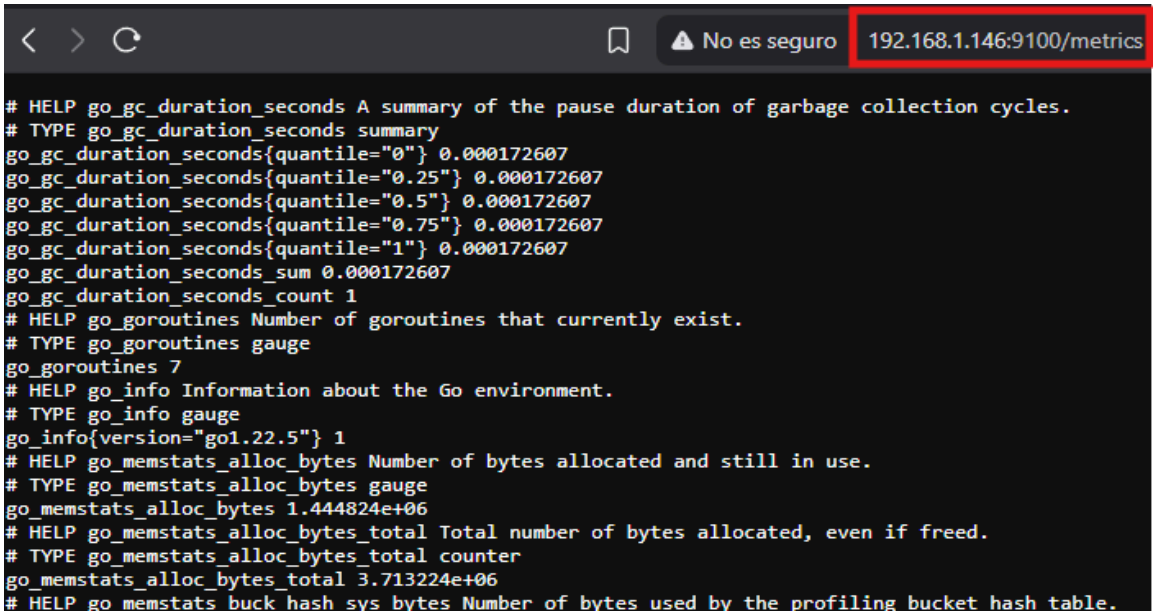
Realizamos lo siguiente

```
#!/bin/bash/

[Unit]
Description=Node Exporter
After=network.target

[Service]
User=node_exporter
ExecStart=/usr/local/bin/node_exporter
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Ahora guardaremos, habilitamos y arrancamos

```
pi@rp3G2:/tmp $ sudo systemctl daemon-reload
pi@rp3G2:/tmp $ sudo systemctl enable node_exporter
Created symlink '/etc/systemd/system/multi-user.target.wants/node_exporter.service' → '
pi@rp3G2:/tmp $ sudo systemctl start node_exporter
```

Comprobamos poniendo el siguiente comando desde un navegador
http://192.168.1.146:9100/metrics



Procederemos a crear una carpeta y un nano donde tendremos que escribir un script

```
pi@rp3G2:/tmp $ sudo mkdir -p /var/lib/node_exporter/textfile_collector
pi@rp3G2:/tmp $ sudo nano /usr/local/bin/rpi_temp.sh
```

Este es el script que escribiremos

```
  GNU nano 8.4                                          /usr/local/bin/rpi_temp.sh
#!/bin/bash
temp=$(cat /sys/class/thermal/thermal_zone0/temp)
temp_c=$(awk "BEGIN {print $temp/1000}")
echo "rpi_cpu_temp_celsius $temp_c" > /var/lib/node_exporter/textfile_collector/rpi_temp.prom
```

Lo hacemos ahora ejecutable con el siguiente comando

```
pi@rp3G2:/tmp $ sudo chmod +x /usr/local/bin/rpi_temp.sh
```

Ahora lo probamos manualmente

```
pi@rp3G2:/tmp $ sudo /usr/local/bin/rpi_temp.sh
pi@rp3G2:/tmp $ cat /var/lib/node_exporter/textfile_collector/rpi_temp.prom
rpi_cpu_temp_celsius 46.251
```