Manual de integración con GPU

Manual de integración gráfica

Instalar WSL2

wsl --install

Este comando:

- Activa WSL y sus componentes necesarios
- Descarga e instala Ubuntu
- Configura todo automáticamente

```
PS C:\WINDOWS\system32> wsl --install
Descargando: Subsistema de Windows para Linux 2.4.13
Instalando: Subsistema de Windows para Linux 2.4.13
Se ha instalado Subsistema de Windows para Linux 2.4.13.
Instalando componente opcional de Windows: VirtualMachinePlatform

Herramienta Administración y mantenimiento de imágenes de implementación
Versión: 10.0.26100.1150

Versión de imagen: 10.0.26100.3775

Habilitando características
[==========================100.0%==========================]
La operación se completó correctamente.
La operación solicitada se realizó correctamente. Los cambios se aplicarán una vez que se reinicie el sistema.
La operación solicitada se realizó correctamente. Los cambios se aplicarán una vez que se reinicie el sistema.
PS C:\WINDOWS\system32>
```

Reiniciamos el equipo y verificamos que ubuntu está en WSL2

En este caso no se ha descargado, así que lo instalamos manualmente:

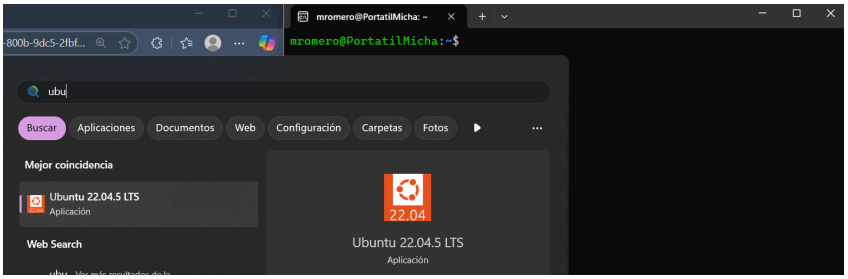
wsl.exe --install Ubuntu-22.04

```
PS C:\WINDOWS\system32> wsl.exe --list --online
A continuación, se muestra una lista de las distribuciones válidas que se pueden instalar.
Instalar con "wsl.exe --install <Distro>".

NAME                            FRIENDLY NAME
AlmaLinux-8                     AlmaLinux OS 8
AlmaLinux-9                     AlmaLinux OS 9
AlmaLinux-Kitten-10             AlmaLinux OS Kitten 10
Debian                          Debian GNU/Linux
FedoraLinux-42                  Fedora Linux 42
SUSE-Linux-Enterprise-15-SP5    SUSE Linux Enterprise 15 SP5
SUSE-Linux-Enterprise-15-SP6    SUSE Linux Enterprise 15 SP6
Ubuntu                          Ubuntu
Ubuntu-24.04                    Ubuntu 24.04 LTS
archlinux                       Arch Linux
kali-linux                      Kali Linux Rolling
openSUSE-Tumbleweed             openSUSE Tumbleweed
openSUSE-Leap-15.6              openSUSE Leap 15.6
Ubuntu-18.04                    Ubuntu 18.04 LTS
Ubuntu-20.04                    Ubuntu 20.04 LTS
Ubuntu-22.04                    Ubuntu 22.04 LTS
OracleLinux_7_9                 Oracle Linux 7.9
OracleLinux_8_7                 Oracle Linux 8.7
OracleLinux_9_1                 Oracle Linux 9.1
PS C:\WINDOWS\system32> wsl.exe --install Ubuntu-22.04
wsl: Usando registro de distribución heredado. Considere la posibilidad de usar una distribución bas
ada en tar en su lugar.
Instalando: Ubuntu 22.04 LTS
[==========================69,0%=========                    ]
```

Comprobamos que se ha instalado correctamente
wsl --list --verbose

```
PS C:\WINDOWS\system32> wsl --list --verbose
  NAME            STATE           VERSION
* Ubuntu-22.04    Stopped         2
PS C:\WINDOWS\system32>
```

Ahora si buscamos ubuntu podremos usarlo



Actualizar y configurar el sistema operativo

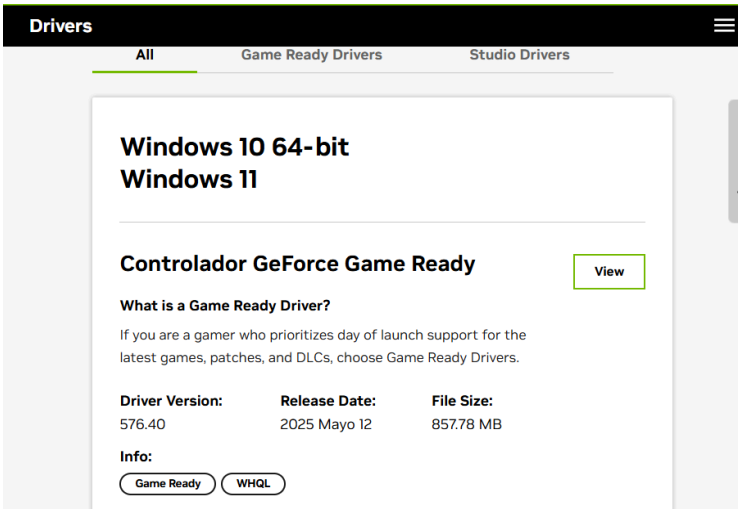
sudo apt update && sudo apt upgrade -y

```
mromero@PortatilMicha:~$ sudo apt update && sudo apt upgrade -y
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 M
B]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [
2311 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 k
```

sudo apt install curl wget build-essential -y

```
mromero@PortatilMicha:~$ sudo apt install curl wget build-essential -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (7.81.0-1ubuntu1.20).
curl set to manually installed.
```

Instalamos el driver de NVIDIA con soporte CUDA para WSL2



Comprobamos que nuestro ubuntu detecta la GPU

nvidia-smi

```
mromero@PortatilMicha:~$ nvidia-smi
Wed May 14 12:50:30 2025
+-----------------------------------------------------------------------------
--------------+
| NVIDIA-SMI 575.55.01              Driver Version: 576.40         CUDA Vers
ion: 12.9     |
|-----------------------------------------+------------------------+--------
--------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatil
e Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Uti
l  Compute M. |
|                                         |                        |
       MIG M. |
|=========================================+========================+========
==============|
|   0  NVIDIA GeForce RTX 3050 ...    On  |   00000000:01:00.0 Off |
          N/A |
| N/A   55C    P8              3W /   75W |       0MiB /   4096MiB |      0%
      Default |
|                                         |                        |
          N/A |
+-----------------------------------------+------------------------+--------
--------------+


+-----------------------------------------------------------------------------
--------------+
| Processes:
                |
|  GPU   GI   CI              PID   Type   Process name
   GPU Memory |
|        ID   ID
   Usage      |
|=============================================================================
==============|
|  No running processes found
                |
+-----------------------------------------------------------------------------
--------------+
```

Instalar Ollama

Dentro de ubuntu ejecutamos:

curl -fsSL https://ollama.com/install.sh | sh

```
mromero@PortatilMicha:~$ curl -fsSL https://ollama.com/install.sh | sh
>>> Installing ollama to /usr/local
[sudo] password for mromero:
>>> Downloading Linux amd64 bundle
##################                                                    26.1%
```

Descargamos el modelo llama3 desde su repositorio oficial.
ollama run llama3

```
mromero@PortatilMicha:~$ ollama run llama3
pulling manifest
pulling 6a0746a1ec1a: 100%                                                    4.7 GB
pulling 4fa551d4f938: 100%                                                     12 KB
pulling 8ab4849b038c: 100%                                                    254 B
pulling 577073ffcc6c: 100%                                                    110 B
pulling 3f8eb4da87fa: 100%                                                    485 B
verifying sha256 digest
writing manifest
success
>>> Send a message (/? for help)
```

Mientras se ejecuta el comando abrimos otra ventana y ejecutamos el siguiente comando (debería aparecer un proceso “Ollama” usando la GPU):
nvidia-smi

```
mromero@PortatilMicha:~$ nvidia-smi
Mon May 19 08:53:30 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 575.55.01              Driver Version: 576.40         CUDA Version: 12.9     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA GeForce RTX 3050 ...    On  |   00000000:01:00.0 Off |                  N/A |
| N/A   48C    P8              3W /   75W |    2375MiB /   4096MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI              PID   Type   Process name                        GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A             592      C   /ollama                                 N/A   |
+-----------------------------------------------------------------------------------------+
```

Ya tendríamos ollama dentro de ubuntu en WSL2, ahora solo tendríamos que comprobar que funciona.

```
>>> que eres capaz de hacer?
Soy un modelo de lenguaje entrenado para realizar una variedad de tareas. Aquí hay algunas cosas que puedo hacer:

1. **Conversar**: Puedo mantener conversaciones naturales sobre temas como la vida, el mundo, las noticias, los
hobbies, etc.
2. **Responder preguntas**: Puedo responder a preguntas sobre una amplia variedad de temas, desde historia hasta
tecnología.
3. **Generar texto**: Puedo generar texto basado en un tema o idea que me proporciones.
4. **Traducir**: Puedo traducir textos de una lengua a otra, aunque mi precisión puede variar dependiendo del
contexto y la complejidad del texto.
5. **Generar ideas**: Puedo generar ideas y sugerencias sobre temas específicos, como proyectos o historias.
6. **Crear contenido**: Puedo crear contenido original en forma de artículos, historias cortas, poemas, etc.
7. **Jugar**: Puedo jugar juegos de texto con personas, como 20 Questions o Hangman.

Estas son solo algunas de las cosas que puedo hacer. Si tienes una tarea específica en mente, no dudes en
preguntar y ver si puedo ayudarte!

>>> 2+2-4*8
¡Vamos a resolver la ecuación!

La ecuación es: 2 + 2 - 4 * 8

 Primero, vamos a evaluar el multiplicación:

4 * 8 = 32

Ahora, la ecuación se convierte en:

2 + 2 - 32

Segundo, vamos a sumar los dos primeros términos:

2 + 2 = 4

Y finalmente, vamos a restar 32 de 4:

4 - 32 = -28

¡La respuesta es -28!
```

Manual de monitorización con GPU:

Descargar Prometheus

Ejecutamos este comando:

curl -LO https://github.com/prometheus/prometheus/releases/download/v2.52.0/prometheus-2.52.0.linux-amd64.tar.gz

```
mromero@PortatilMicha:~$ curl -LO https://github.com/prometheus/prometheus/r
eleases/download/v2.52.0/prometheus-2.52.0.linux-amd64.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Curr
ent
                                 Dload  Upload   Total   Spent    Left  Spee
d
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--
 0
  0 99.8M    0 32750    0     0  28404      0  1:01:24  0:00:01  1:01:23 284
 15 99.8M   15 15.4M    0     0  8207k      0  0:00:12  0:00:01  0:00:11 19.
 40 99.8M   40 40.7M    0     0  13.9M      0  0:00:07  0:00:02  0:00:05 22.
 66 99.8M   66 66.8M    0     0  16.9M      0  0:00:05  0:00:03  0:00:02 24.
 93 99.8M   93 92.8M    0     0  18.8M      0  0:00:05  0:00:04  0:00:01 24.
100 99.8M  100 99.8M    0     0  18.6M      0  0:00:05  0:00:05 --:--:-- 23.
7M
```

Lo descomprimimos y accedemos a él con:

tar -xvzf prometheus-2.52.0.linux-amd64.tar.gz

cd prometheus-2.52.0.linux-amd64

```
mromero@PortatilMicha:~$ tar -xvzf prometheus-2.52.0.linux-amd64.tar.gz
prometheus-2.52.0.linux-amd64/
prometheus-2.52.0.linux-amd64/LICENSE
prometheus-2.52.0.linux-amd64/promtool
prometheus-2.52.0.linux-amd64/console_libraries/
prometheus-2.52.0.linux-amd64/console_libraries/prom.lib
prometheus-2.52.0.linux-amd64/console_libraries/menu.lib
prometheus-2.52.0.linux-amd64/prometheus
prometheus-2.52.0.linux-amd64/consoles/
prometheus-2.52.0.linux-amd64/consoles/node-overview.html
prometheus-2.52.0.linux-amd64/consoles/prometheus.html
prometheus-2.52.0.linux-amd64/consoles/node.html
prometheus-2.52.0.linux-amd64/consoles/index.html.example
prometheus-2.52.0.linux-amd64/consoles/node-disk.html
prometheus-2.52.0.linux-amd64/consoles/node-cpu.html
prometheus-2.52.0.linux-amd64/consoles/prometheus-overview.html
prometheus-2.52.0.linux-amd64/prometheus.yml
prometheus-2.52.0.linux-amd64/NOTICE
mromero@PortatilMicha:~$ cd prometheus-2.52.0.linux-amd64
mromero@PortatilMicha:~/prometheus-2.52.0.linux-amd64$
```

Movemos los binarios:

sudo mv prometheus promtool /usr/local/bin/

```
mromero@PortatilMicha:~/prometheus-2.52.0.linux-amd64$ ls
LICENSE  NOTICE  console_libraries  consoles  prometheus.yml
```

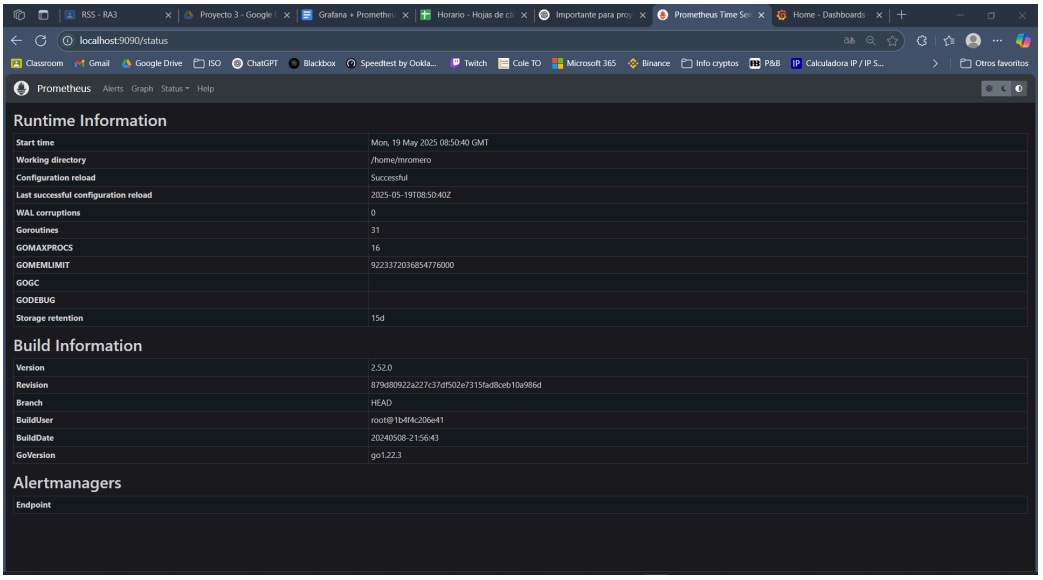
Creamos los directorios

sudo mkdir /etc/prometheus /var/lib/prometheus
sudo cp -r consoles console_libraries /etc/prometheus/
sudo cp prometheus.yml /etc/prometheus/

```
mromero@PortatilMicha:~/prometheus-2.52.0.linux-amd64$ sudo mkdir /etc/prome
theus /var/lib/prometheus
sudo cp -r consoles console_libraries /etc/prometheus/
sudo cp prometheus.yml /etc/prometheus/
mromero@PortatilMicha:~/prometheus-2.52.0.linux-amd64$ ls
LICENSE  NOTICE  console_libraries  consoles  prometheus.yml
```

Ejecutar Prometheus

prometheus --config.file=/etc/prometheus/prometheus.yml
Y lo abrimos en el navegador con http://localhost:9090



Para Grafana añadiremos el repositorio oficial:
sudo apt install -y software-properties-common
sudo add-apt-repository "deb [arch=amd64] https://packages.grafana.com/oss/deb stable main"
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -

```
mromero@PortatilMicha:~$ sudo apt install -y software-properties-common
sudo add-apt-repository "deb [arch=amd64] https://packages.grafana.com/oss/d
eb stable main"
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
software-properties-common is already the newest version (0.99.22.9).
software-properties-common set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Repository: 'deb [arch=amd64] https://packages.grafana.com/oss/deb stable ma
in'
Description:
Archive for codename: stable components: main
More info: https://packages.grafana.com/oss/deb
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_packages_grafa
na_com_oss_deb-jammy.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_p
ackages_grafana_com_oss_deb-jammy.list
Hit:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
```

Instalamos Grafana

sudo apt update

sudo apt install grafana -y

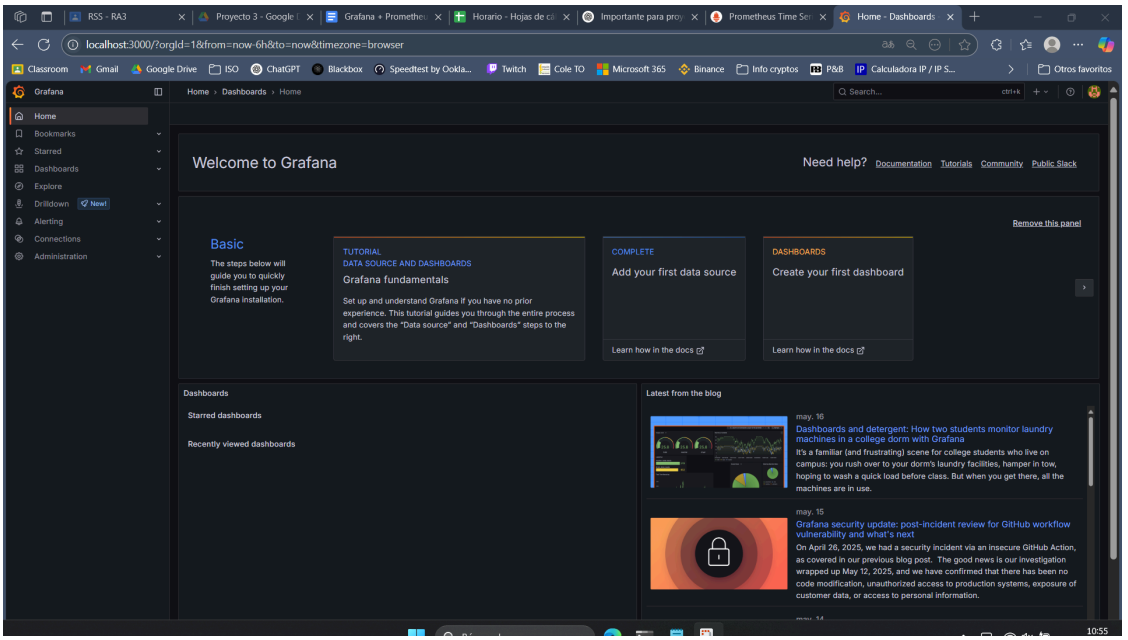
```
mromero@PortatilMicha:~$ sudo apt update
sudo apt install grafana -y
Get:1 https://packages.grafana.com/oss/deb stable InRelease [7661 B]
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:4 https://packages.grafana.com/oss/deb stable/main amd64 Packages [394 kB]
Hit:5 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:6 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Fetched 394 kB in 1s (564 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
W: https://packages.grafana.com/oss/deb/dists/stable/InRelease: Key is stored in legacy tru
sted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for deta
ils.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  musl
The following NEW packages will be installed:
  grafana musl
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 175 MB of archives.
After this operation, 649 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 musl amd64 1.2.2-4 [407 kB]
```

Iniciamos Grafana

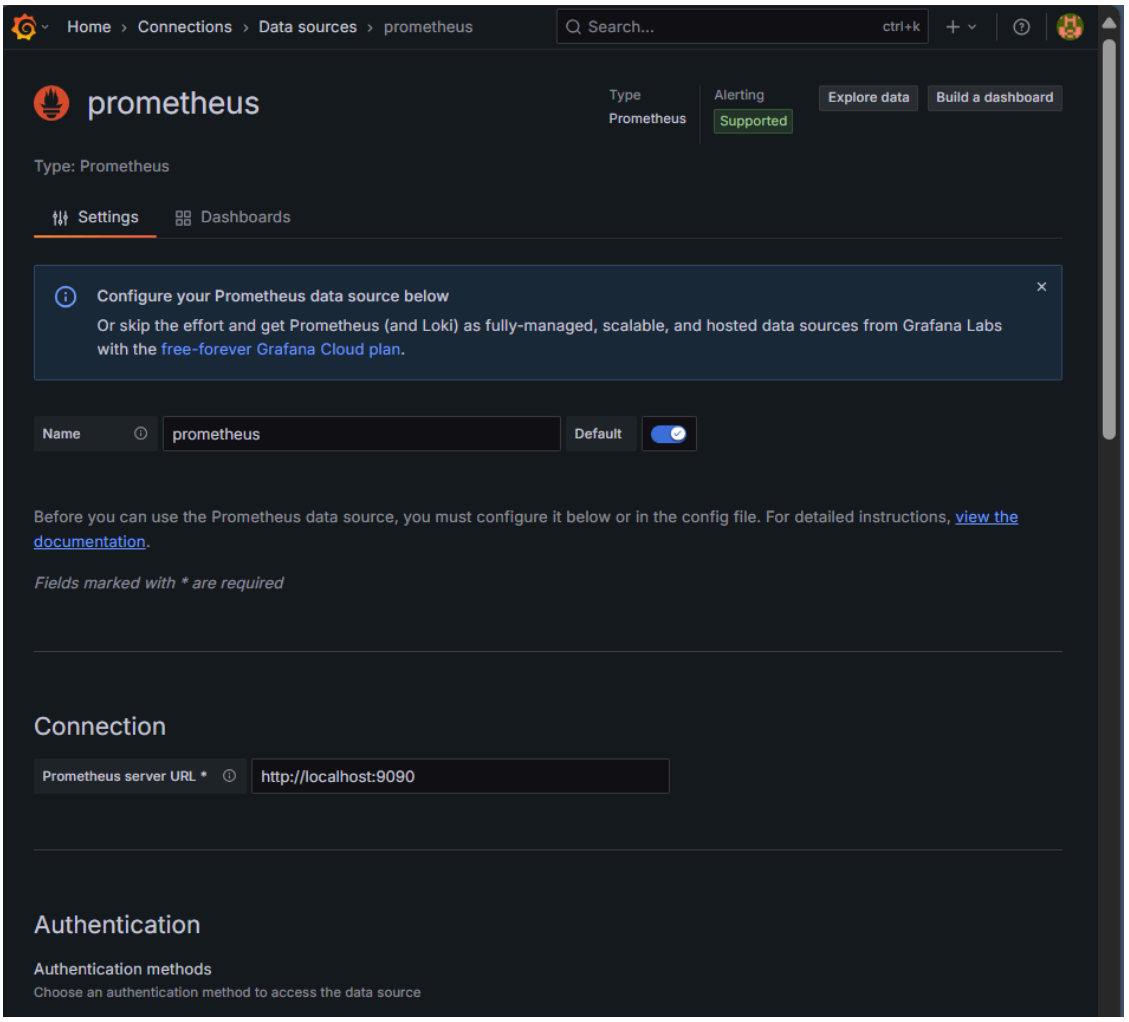
sudo systemctl start grafana-server

sudo systemctl enable grafana-server

Y lo abrimos en el buscador http://localhost:3000



Añadimos Prometheus a Grafana como fuente de datos



Instalar y lanzar node_exporter

wget https://github.com/prometheus/node_exporter/releases/download/v1.9.1/node_exporter-1.9.1.linux-amd64.tar.gz
tar -xvzf node_exporter-1.9.1.linux-amd64.tar.gz
cd node_exporter-1.9.1.linux-amd64
./node_exporter &

```
mromero@PortatilMicha:~$ wget https://github.com/prometheus/node_exporter/releases/download
/v1.9.1/node_exporter-1.9.1.linux-amd64.tar.gz
--2025-05-19 11:18:01--  https://github.com/prometheus/node_exporter/releases/download/v1.9
.1/node_exporter-1.9.1.linux-amd64.tar.gz
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://objects.githubusercontent.com/github-production-release-asset-2e65be/9524
057/dc8ec09c-2975-42a2-9591-57dd1ffff7b7?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=
releaseassetproduction%2F20250519%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20250519T09180
1Z&X-Amz-Expires=300&X-Amz-Signature=f09da210b00df95dea601c0d75fb51e099cfe4471fb980de61f95d
0cc0cdd6ff&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3
Dnode_exporter-1.9.1.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream [f
ollowing]
```

```
mromero@PortatilMicha:~$ tar -xvzf node_exporter-1.9.1.linux-amd64.tar.gz
node_exporter-1.9.1.linux-amd64/
node_exporter-1.9.1.linux-amd64/LICENSE
node_exporter-1.9.1.linux-amd64/NOTICE
node_exporter-1.9.1.linux-amd64/node_exporter
mromero@PortatilMicha:~$ cd node_exporter-1.9.1.linux-amd64
mromero@PortatilMicha:~/node_exporter-1.9.1.linux-amd64$ ./node_exporter &
[1] 3195
mromero@PortatilMicha:~/node_exporter-1.9.1.linux-amd64$ time=2025-05-19T09:22:24.079Z leve
l=INFO source=node_exporter.go:216 msg="Starting node_exporter" version="(version=1.9.1, br
anch=HEAD, revision=f2ec547b49af53815038a50265aa2adcd1275959)"
time=2025-05-19T09:22:24.079Z level=INFO source=node_exporter.go:217 msg="Build context" bu
ild_context="(go=go1.23.7, platform=linux/amd64, user=root@7023beaa563a, date=20250401-15:1
9:01, tags=unknown)"
```

Añadimos lo siguiente en el apartado de scrape_configs del fichero prometheus.yml

nano prometheus.yml

- job_name: 'node_exporter'

static_configs:

- targets: ['localhost:9100']

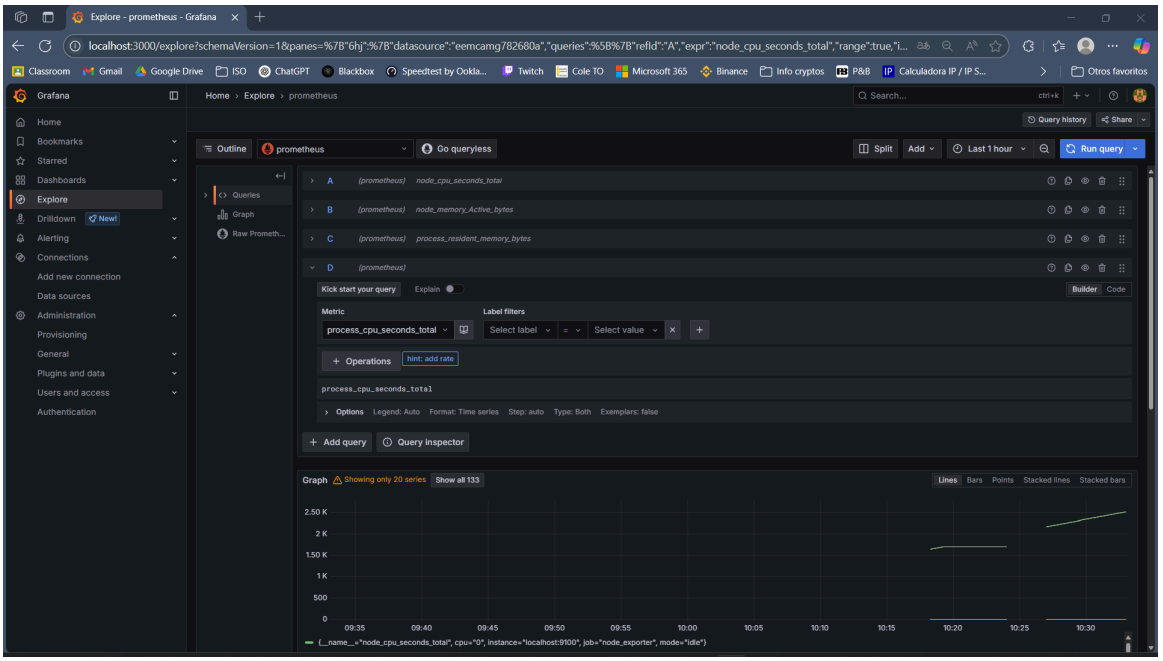
```
  GNU nano 6.2                          prometheus.yml
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 m>
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minut>
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_inter>
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['localhost:9100']

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

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

    static_configs:
      - targets: ["localhost:9090"]
```

Desde el directorio en el que tenemos el binario de Prometheus reiniciamos Prometheus

./prometheus --config.file=~/prometheus-config/prometheus.yml --storage.tsdb.path=~/prometheus-data

```
mromero@PortatilMicha:~$ cd /usr/local/bin
mromero@PortatilMicha:/usr/local/bin$ ls
ollama  prometheus  promtool
mromero@PortatilMicha:/usr/local/bin$ ./prometheus --config.file=~/prometheus-config/promet
heus.yml --storage.tsdb.path=~/prometheus-data
ts=2025-05-19T10:00:14.521Z caller=main.go:521 level=error msg="Error loading config (--con
fig.file=~/prometheus-config/prometheus.yml)" file=/usr/local/bin/~/prometheus-config/prome
theus.yml err="open ~/prometheus-config/prometheus.yml: no such file or directory"
mromero@PortatilMicha:/usr/local/bin$
```

Ya con todo instalado vamos a grafana y en Data sources en la fuente de prometheus vamos a “explore data” y añadimos las siguientes métricas. Este dashboard monitoriza la RAM y el CPU



Guardamos el dashboard

Instalar Nvidia GPU exporter para monitorizar la gpu

git clone https://github.com/mindprince/nvidia_gpu_prometheus_exporter.git

```
mromero@PortatilMicha:~$ git clone https://github.com/mindprince/nvidia_gpu_
prometheus_exporter.git
Cloning into 'nvidia_gpu_prometheus_exporter'...
remote: Enumerating objects: 188, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 188 (delta 21), reused 19 (delta 19), pack-reused 147 (from 1
Receiving objects: 100% (188/188), 298.78 KiB | 1.23 MiB/s, done.
Resolving deltas: 100% (28/28), done.
mromero@PortatilMicha:~$ ls
data                                nvidia_gpu_prometheus_exporter
node_exporter-1.9.1.linux-amd64     prometheus-2.52.0.linux-amd64
```

Para compilar los paquetes instalamos Go

sudo apt install golang-go

```
mromero@PortatilMicha:~$ sudo apt install golang-go
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  golang-1.18-go golang-1.18-src golang-src pkg-config
Suggested packages:
  bzr | brz mercurial subversion
```

En el directorio de nvidia ejecutaremos los siguientes comandos para que genere el binario “nvidia_gpu_prometheus_exporter”

go mod tidy -e

go mod vendor

go build

```
mromero@PortatilMicha:~/nvidia_gpu_prometheus_exporter$ go mod tidy -e
go: finding module for package github.com/golang/protobuf/proto/testdata
github.com/mindprince/nvidia_gpu_prometheus_exporter imports
        github.com/prometheus/client_golang/prometheus imports
        github.com/prometheus/common/expfmt imports
        github.com/matttproud/golang_protobuf_extensions/pbutil tested by
        github.com/matttproud/golang_protobuf_extensions/pbutil.test imports
        github.com/golang/protobuf/proto/testdata: module github.com/golang/
protobuf@latest found (v1.5.4), but does not contain package github.com/gola
ng/protobuf/proto/testdata
mromero@PortatilMicha:~/nvidia_gpu_prometheus_exporter$ go build
go: inconsistent vendoring in /home/mromero/nvidia_gpu_prometheus_exporter:
        github.com/mindprince/gonvml@v0.0.0-20180514031326-b364b296c732: is
explicitly required in go.mod, but not marked as explicit in vendor/modules.
txt
        github.com/prometheus/client_golang@v0.8.0: is explicitly required i
n go.mod, but not marked as explicit in vendor/modules.txt
        github.com/beorn7/perks@v0.0.0-20180321164747-3a771d992973: is expli
citly required in go.mod, but not marked as explicit in vendor/modules.txt
        github.com/golang/protobuf@v1.1.0: is explicitly required in go.mod,
 but not marked as explicit in vendor/modules.txt
        github.com/matttproud/golang_protobuf_extensions@v1.0.0: is explicit
ly required in go.mod, but not marked as explicit in vendor/modules.txt
        github.com/prometheus/client_model@v0.0.0-20171117100541-99fa1f4be8e
5: is explicitly required in go.mod, but not marked as explicit in vendor/mo
dules.txt
        github.com/prometheus/common@v0.0.0-20180518154759-7600349dcfe1: is
explicitly required in go.mod, but not marked as explicit in vendor/modules.
txt
        github.com/prometheus/procfs@v0.0.0-20180408092902-8b1c2da0d56d: is
explicitly required in go.mod, but not marked as explicit in vendor/modules.
txt
        golang.org/x/sync@v0.14.0: is explicitly required in go.mod, but not
 marked as explicit in vendor/modules.txt

        To ignore the vendor directory, use -mod=readonly or -mod=mod.
        To sync the vendor directory, run:
                go mod vendor
mromero@PortatilMicha:~/nvidia_gpu_prometheus_exporter$ go mod vendor
mromero@PortatilMicha:~/nvidia_gpu_prometheus_exporter$ go build
mromero@PortatilMicha:~/nvidia_gpu_prometheus_exporter$ ls
Dockerfile  LICENSE    go.mod   nvidia_gpu_prometheus_exporter
```

Ejecutamos el exporter

./nvidia_gpu_prometheus_exporter

Con este comando podemos ejecutarlo en segundo plano

nohup ./nvidia_gpu_prometheus_exporter > exporter.log 2>&1 &

```
mromero@PortatilMicha:~/nvidia_gpu_prometheus_exporter$ ./nvidia_gpu_prometh
eus_exporter
2025/05/20 12:27:28 SystemDriverVersion(): 576.40
```

Añadimos lo siguiente al archivo prometheus.yml

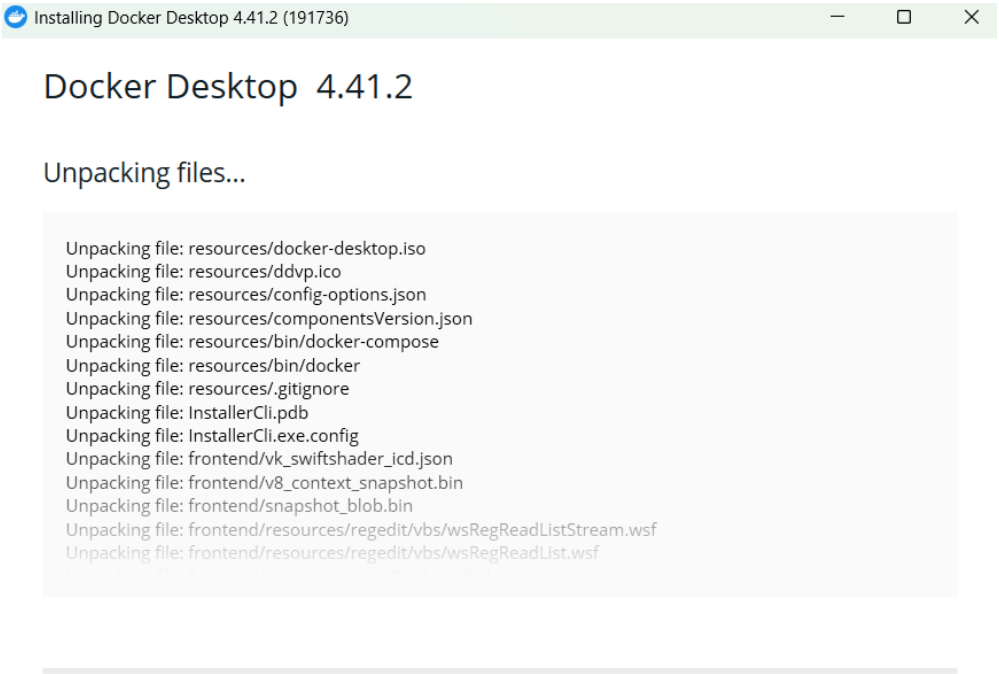
- job_name: 'nvidia_gpu'
static_configs:
- targets: ['localhost:9445']

```
scrape_configs:
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['localhost:9100']

  - job_name: 'nvidia_gpu'
    static_configs:
      - targets: ['localhost:9445']
```

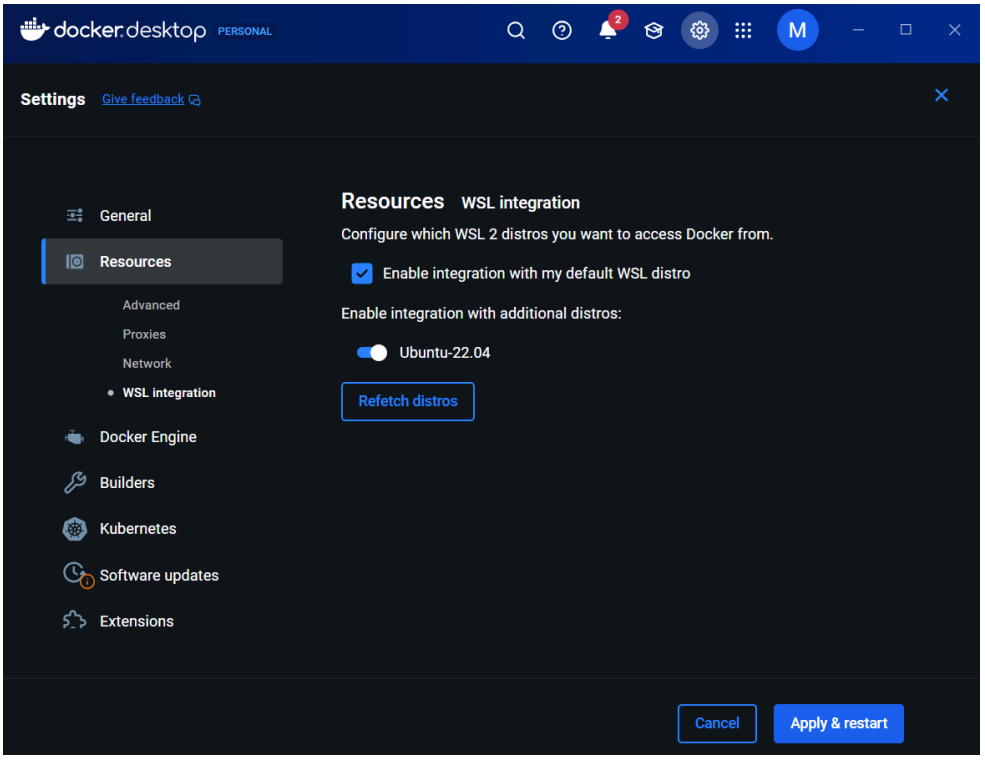
Como en grafana solo nos aparece nvidia_gpu_num_devices instalamos docker para poder ejecutar DCGM exporter.

Instalamos la versión correspondiente a nuestro sistema y procesador de docker. En nuestro caso instalamos el de Windows AMD64.

En la instalación habilitaremos la opción de integración para WSL2



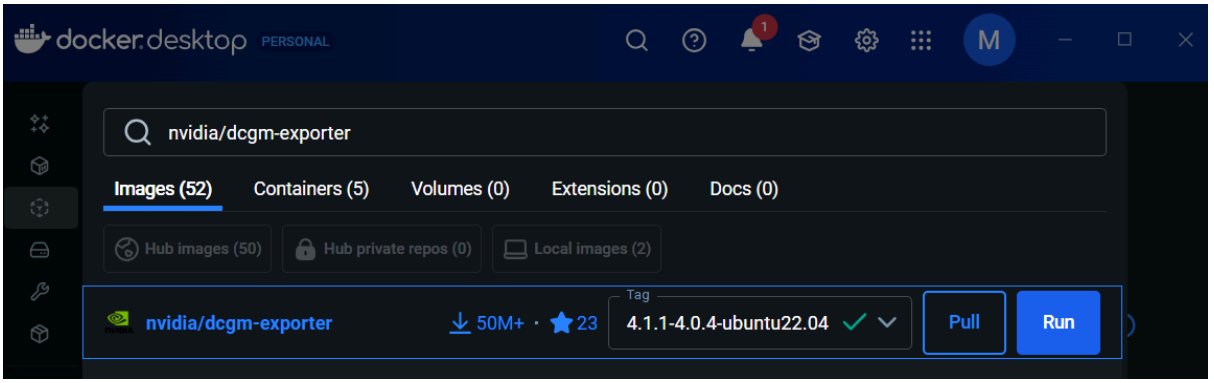
Cuando acabe la instalación nos pedirá que reiniciemos el equipo
Con docker instalado, iniciamos sesión y habilitamos la integración de WSL2



Comprobamos en WSL que Docker funcione

```
mromero@PortatilMicha:~$ docker --version
Docker version 28.1.1, build 4eba377
mromero@PortatilMicha:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:dd01f97f252193ae3210da231b1dca0cffab4aadb3566692d6730bf93f123
a48
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent
it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Una vez estamos seguros de que WSL funciona con Docker “instalamos” la siguiente imagen



Usamos este comando para ejecutar el contenedor y comprobar que está activo:
docker run -d --gpus all -p 9400:9400 nvidia/dcgm-exporter:4.1.1-4.0.4-ubuntu22.04
y docker ps

```
mromero@PortatilMicha:~$ docker ps
CONTAINER ID   IMAGE                                                COMMAND
           CREATED          STATUS          PORTS                    NAMES
c39f18bf23fe   nvidia/dcgm-exporter:4.1.1-4.0.4-ubuntu22.04   "/usr/local/dc
gm/dcg…"   24 minutes ago   Up 24 minutes   0.0.0.0:9400->9400/tcp   xenodoc
hial_visvesvaraya
mromero@PortatilMicha:~$ docker run -d --gpus all -p 9400:9400 nvidia/dcgm-e
xporter:4.1.1-4.0.4-ubuntu22.04
```

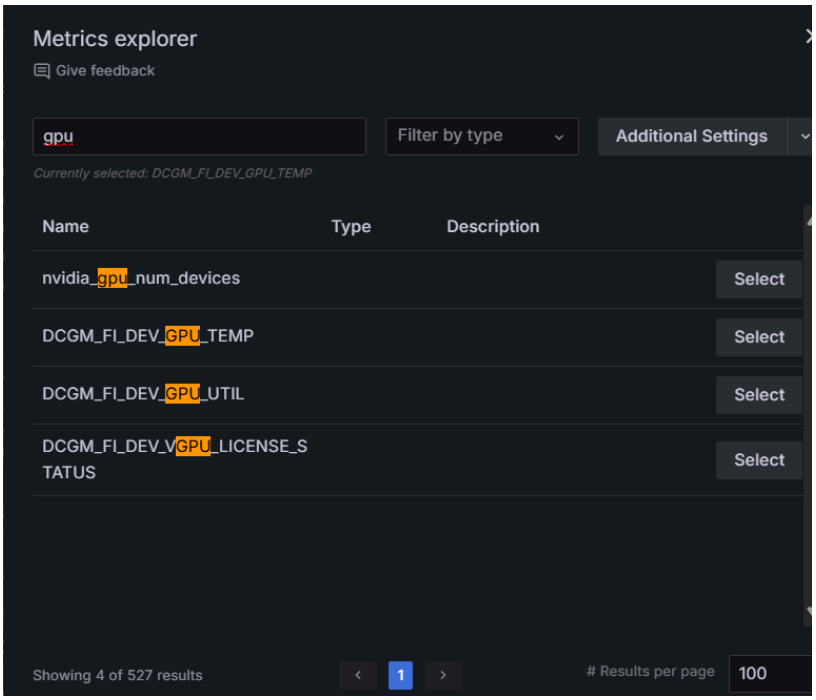
Ahora iremos al archivo de configuración de prometheus.yml y añadiremos lo siguiente

- job_name: 'dcgm-exporter'
static_configs:
- targets: ['localhost:9400']

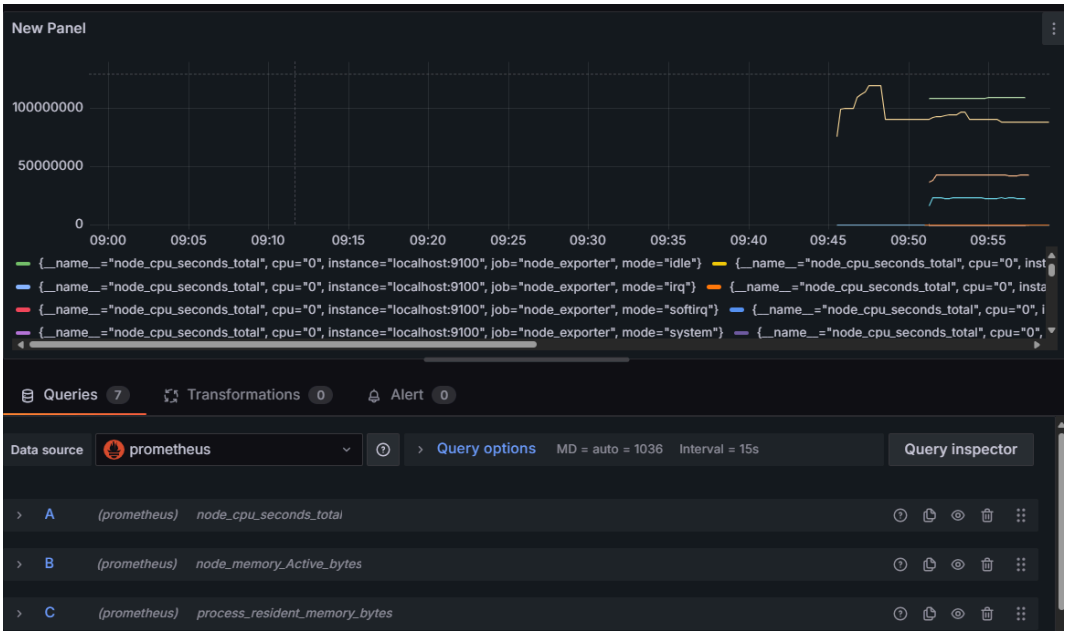
```
scrape_configs:
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['localhost:9100']

  - job_name: 'nvidia_gpu'
    static_configs:
      - targets: ['localhost:9445']

  - job_name: 'dcgm-exporter'
    static_configs:
      - targets: ['localhost:9400']
```

Ahora tenemos que reiniciar prometheus e ir a grafana

Con todo configurado nos aparecerán estas métricas, de las cuales vamos a usar:
DCGM_FI_DEV_GPU_TEMP
DCGM_FI_DEV_GPU_UTIL



Así es como debe quedar el dashboard hasta este momento



Asi se veria mi dashboard con llama3 corriendo

```
mromero@PortatilMicha:~$ ollama run llama3
>>> que tal
Hola! Que tengas un buen día. ¿En qué puedo ayudarte hoy?
```

En nuestro caso estas serían las métricas usadas para el dashboard

›	A	(prometheus)	node_cpu_seconds_total
›	B	(prometheus)	node_memory_Active_bytes
›	C	(prometheus)	process_resident_memory_bytes
›	D	(prometheus)	process_cpu_seconds_total
›	E	(prometheus)	nvidia_gpu_num_devices
›	F	(prometheus)	DCGM_FI_DEV_GPU_TEMP
›	G	(prometheus)	DCGM_FI_DEV_GPU_UTIL

A - node_cpu_seconds_total: Tiempo total que la CPU ha pasado en diferentes modos (usuario, sistema, inactivo, etc.).

B - node_memory_Active_bytes: Cantidad de memoria RAM activa actualmente en uso.

C - process_resident_memory_bytes: Memoria RAM usada por un proceso en particular (Se puede filtrar por procesos).

D - process_cpu_seconds_total: Tiempo total de CPU consumido por un proceso específico.

E - nvidia_gpu_num_devices: Número total de dispositivos GPU detectados por NVIDIA. (En nuestro caso solo monitoriza uno, pero en un entorno real más preparado hay varias GPUs)

F - DCGM_FI_DEV_GPU_TEMP: Temperatura actual de la GPU.

G - DCGM_FI_DEV_GPU_UTIL: Porcentaje de utilización de la GPU.