

1. [2.0 Puntos (Funcionamiento + Sustentación)] Instalación y Configuración de Prometheus y Exploración de Métricas de Linux con el Node Exporter

Paso 1: Instalación y Configuración de Prometheus

a. Instalación de Prometheus

Descargar la última versión de Prometheus

wget <https://github.com/prometheus/prometheus/releases/download/v2.30.3/prometheus-2.30.3.linux-amd64.tar.gz>

```
final — vagrant@vagrant: ~ — ssh - vagrant ssh serviciofinal — 96x38

This system is built by the Bento project by Chef Software
More information can be found at https://github.com/chef/bento
vagrant@vagrant:~$ wget https://github.com/prometheus/prometheus/releases/download/v2.30.3/prometheus-2.30.3.linux-amd64.tar.gz
--2023-11-09 23:37:16-- https://github.com/prometheus/prometheus/releases/download/v2.30.3/prometheus-2.30.3.linux-amd64.tar.gz
Resolving github.com (github.com)... 140.82.113.3
Connecting to github.com (github.com)|140.82.113.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/6838921/bc9e0970-09b3-4893-a66b-5fb918691a1e?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20231109%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20231109T233719Z&X-Amz-Expires=300&X-Amz-Signature=336b295e923261a2bda4fbc8da3f264cfe9740b0c8f5357568d4c513a24275f3&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=6838921&response-content-disposition=attachment%3B%20filename%3Dprometheus-2.30.3.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream [following]
--2023-11-09 23:37:17-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/6838921/bc9e0970-09b3-4893-a66b-5fb918691a1e?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20231109%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20231109T233719Z&X-Amz-Expires=300&X-Amz-Signature=336b295e923261a2bda4fbc8da3f264cfe9740b0c8f5357568d4c513a24275f3&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=6838921&response-content-disposition=attachment%3B%20filename%3Dprometheus-2.30.3.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.109.133, 185.199.111.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 72638078 (69M) [application/octet-stream]
Saving to: 'prometheus-2.30.3.linux-amd64.tar.gz'

prometheus-2.30.3.l 100%[=====] 69.27M 30.2MB/s in 2.3s

2023-11-09 23:37:19 (30.2 MB/s) - 'prometheus-2.30.3.linux-amd64.tar.gz' saved [72638078/72638078]

vagrant@vagrant:~$
```

Descomprimir el archivo

```
tar -xzf prometheus-2.30.3.linux-amd64.tar.gz
```

Mover la carpeta a /opt

```
sudo mv prometheus-2.30.3.linux-amd64 /opt/prometheus
```

Eliminar el archivo comprimido

```
rm prometheus-2.30.3.linux-amd64.tar.gz
```

b. Configuración de Prometheus

Crea un archivo de configuración para Prometheus. Puedes utilizar un archivo de configuración simple para comenzar. Crea un archivo prometheus.yml con el siguiente contenido:

Ubicación: `cd /opt/prometheus/prometheus.yml`

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
```

Actualizado

```
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['localhost:9100']
```

Mueve este archivo a la carpeta /opt/prometheus/.

c. Configuración del servicio de Prometheus

Creemos un servicio systemd para Prometheus. Creamos un archivo llamado /etc/systemd/system/prometheus.service con el siguiente contenido:

`sudo vim /etc/systemd/system/prometheus.service`

```
[Unit]
Description=Prometheus
After=network.target

[Service]
ExecStart=/opt/prometheus/prometheus --config.file=/opt/prometheus/prometheus.yml
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=always

[Install]
WantedBy=default.target
```

Recarga systemd y habilita el servicio:

```
sudo systemctl daemon-reload
sudo systemctl start prometheus
sudo systemctl enable prometheus
sudo systemctl status prometheus
```

```
[vagrant@vagrant:~]$ sudo systemctl daemon-reload
[vagrant@vagrant:~]$ sudo systemctl start prometheus
[vagrant@vagrant:~]$ sudo systemctl enable prometheus
Created symlink /etc/systemd/system/default.target.wants/prometheus.service → /etc/systemd/system/prometheus.service.
[vagrant@vagrant:~]$ sudo systemctl status prometheus
● prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2023-11-09 23:58:00 UTC; 38s ago
     Main PID: 1883 (prometheus)
        Tasks: 8 (limit: 2275)
      Memory: 20.0M
         CPU: 154ms
       CGroup: /system.slice/prometheus.service
              └─1883 /opt/prometheus/prometheus --config.file=/opt/prometheus/prometheus.yml

Nov 09 23:58:00 vagrant prometheus[1883]: level=info ts=2023-11-09T23:58:00.090Z caller=head.go
Nov 09 23:58:00 vagrant prometheus[1883]: level=info ts=2023-11-09T23:58:00.090Z caller=head.go
Nov 09 23:58:00 vagrant prometheus[1883]: level=info ts=2023-11-09T23:58:00.091Z caller=tls_con
Nov 09 23:58:00 vagrant prometheus[1883]: level=info ts=2023-11-09T23:58:00.092Z caller=head.go
Nov 09 23:58:00 vagrant prometheus[1883]: level=info ts=2023-11-09T23:58:00.092Z caller=head.go
Nov 09 23:58:00 vagrant prometheus[1883]: level=info ts=2023-11-09T23:58:00.094Z caller=main.go
Nov 09 23:58:00 vagrant prometheus[1883]: level=info ts=2023-11-09T23:58:00.094Z caller=main.go
Nov 09 23:58:00 vagrant prometheus[1883]: level=info ts=2023-11-09T23:58:00.094Z caller=main.go
Nov 09 23:58:00 vagrant prometheus[1883]: level=info ts=2023-11-09T23:58:00.094Z caller=main.go
Nov 09 23:58:00 vagrant prometheus[1883]: level=info ts=2023-11-09T23:58:00.095Z caller=main.go
Nov 09 23:58:00 vagrant prometheus[1883]: level=info ts=2023-11-09T23:58:00.095Z caller=main.go
[vagrant@vagrant:~]$
```

Paso 2: Exploración de Métricas con Node Exporter

a. Instalación de Node Exporter

Descarga e instala el Node Exporter. Podemos hacerlo siguiendo estos pasos:

Descargar la última versión de Node Exporter

wget

https://github.com/prometheus/node_exporter/releases/download/v1.2.2/node_exporter-1.2.2.linux-amd64.tar.gz

```
[vagrant@vagrant:~]$ tar -xzf node_exporter-1.2.2.linux-amd64.tar.gz
[vagrant@vagrant:~]$ sudo mv node_exporter-1.2.2.linux-amd64/node_exporter /usr/local/bin/
[vagrant@vagrant:~]$ rm -rf node_exporter-1.2.2.linux-amd64.tar.gz node_exporter-1.2.2.linux-amd64
```

```
final — vagrant@vagrant: ~ — s
[Unit]
Description=Node Exporter
After=network.target

[Service]
ExecStart=/usr/local/bin/node_exporter

[Install]
WantedBy=default.target
```

```
sudo systemctl daemon-reload
```

```
sudo systemctl start node_exporter
sudo systemctl enable node_exporter
sudo systemctl status node_exporter
```

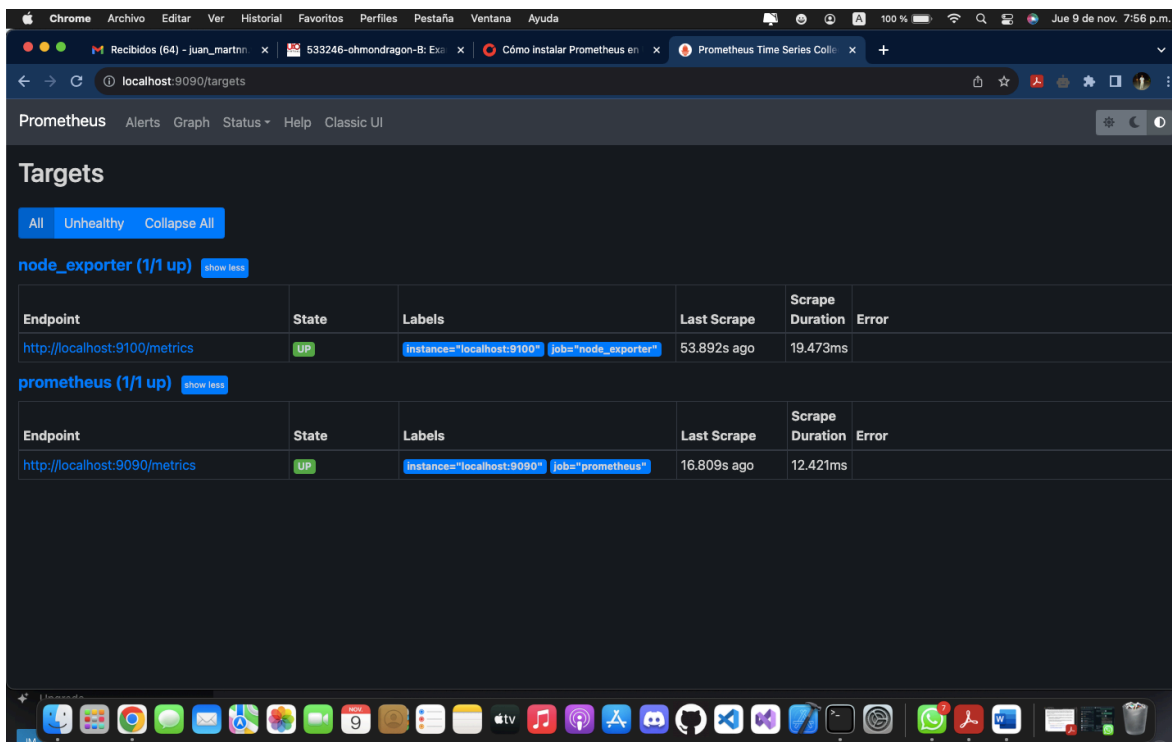
```
vagrant@vagrant:~$ sudo systemctl status node_exporter
● node_exporter.service - Node Exporter
   Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2023-11-10 00:28:17 UTC; 18s ago
     Main PID: 2050 (node_exporter)
        Tasks: 4 (limit: 2275)
       Memory: 2.2M
          CPU: 23ms
      CGroup: /system.slice/node_exporter.service
              └─2050 /usr/local/bin/node_exporter

Nov 10 00:28:17 vagrant node_exporter[2050]: level=info ts=2023-11-10T00:28:17.737Z caller=node>
Nov 10 00:28:17 vagrant node_exporter[2050]: level=info ts=2023-11-10T00:28:17.737Z caller=node>
Nov 10 00:28:17 vagrant node_exporter[2050]: level=info ts=2023-11-10T00:28:17.737Z caller=node>
Nov 10 00:28:17 vagrant node_exporter[2050]: level=info ts=2023-11-10T00:28:17.737Z caller=node>
Nov 10 00:28:17 vagrant node_exporter[2050]: level=info ts=2023-11-10T00:28:17.738Z caller=node>
Nov 10 00:28:17 vagrant node_exporter[2050]: level=info ts=2023-11-10T00:28:17.738Z caller=node>
Nov 10 00:28:17 vagrant node_exporter[2050]: level=info ts=2023-11-10T00:28:17.738Z caller=node>
Nov 10 00:28:17 vagrant node_exporter[2050]: level=info ts=2023-11-10T00:28:17.738Z caller=node>
Nov 10 00:28:17 vagrant node_exporter[2050]: level=info ts=2023-11-10T00:28:17.739Z caller=node>
Nov 10 00:28:17 vagrant node_exporter[2050]: level=info ts=2023-11-10T00:28:17.739Z caller=tl>
lines 1-20/20 (END)
vagrant@vagrant:~$
```

Paso 3: Verificación

Verificamos que tanto Prometheus como Node Exporter estén en funcionamiento:

Vamos a la interfaz web de Prometheus (<http://localhost:9090>) y utilizamos la barra de búsqueda para buscar node_exporter. Deberíamos ver los objetivos (targets) relacionados con node_exporter si la configuración se realizó correctamente.



2. [1.0 Puntos (Funcionamiento + Sustentación)] API REST.

En la maquina host cree una carpeta en mi escritorio(repofinal) y en ella vamos a clonar el repo:

git clone <https://github.com/omondragon/APIRestFlaskMySQLUbuntu.git>

```
juanmartinvasquezcaicedo@JUANS-MacBook-Air-5 ~ % cd desktop
juanmartinvasquezcaicedo@JUANS-MacBook-Air-5 desktop % cd repofinal
juanmartinvasquezcaicedo@JUANS-MacBook-Air-5 repofinal % git clone https://github.com/omondragon/APIRestFlaskMySQLUbuntu.git
Clonando en 'APIRestFlaskMySQLUbuntu'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0
Recibiendo objetos: 100% (6/6), listo.
juanmartinvasquezcaicedo@JUANS-MacBook-Air-5 repofinal % cd APIRestFlaskMySQLUbuntu/
juanmartinvasquezcaicedo@JUANS-MacBook-Air-5 APIRestFlaskMySQLUbuntu %
```

cd APIRestFlaskMySQLUbuntu/

```
juanmartinvasquezcaicedo@JUANS-MacBook-Air-5 APIRestFlaskMySQLUbuntu % cd APIRestFlaskMySQLUbuntu/
cd: no such file or directory: APIRestFlaskMySQLUbuntu/
juanmartinvasquezcaicedo@JUANS-MacBook-Air-5 APIRestFlaskMySQLUbuntu %
```

Levantamos y aprovisionamos la máquina virtual:

vagrant up

JMartinVasquez

Ejecutamos la aplicación:

```
export FLASK_APP=apiREST_mysql.py
```

```
python3 -m flask run --host=0.0.0.0
```

```
APIRestFlaskMySQLUbuntu — vagrant@servidorRest: ~ — ssh < vagrant ssh...

[vagrant@servidorRest:~]$ export FLASK_APP=apiREST_mysql.py
[vagrant@servidorRest:~]$ python3 -m flask run --host=0.0.0.0
 * Serving Flask app 'apiREST_mysql.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use
e a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://10.0.2.15:5000
Press CTRL+C to quit
```

PRUEBA CURL

```
curl http://192.168.60.3:5000/books
```

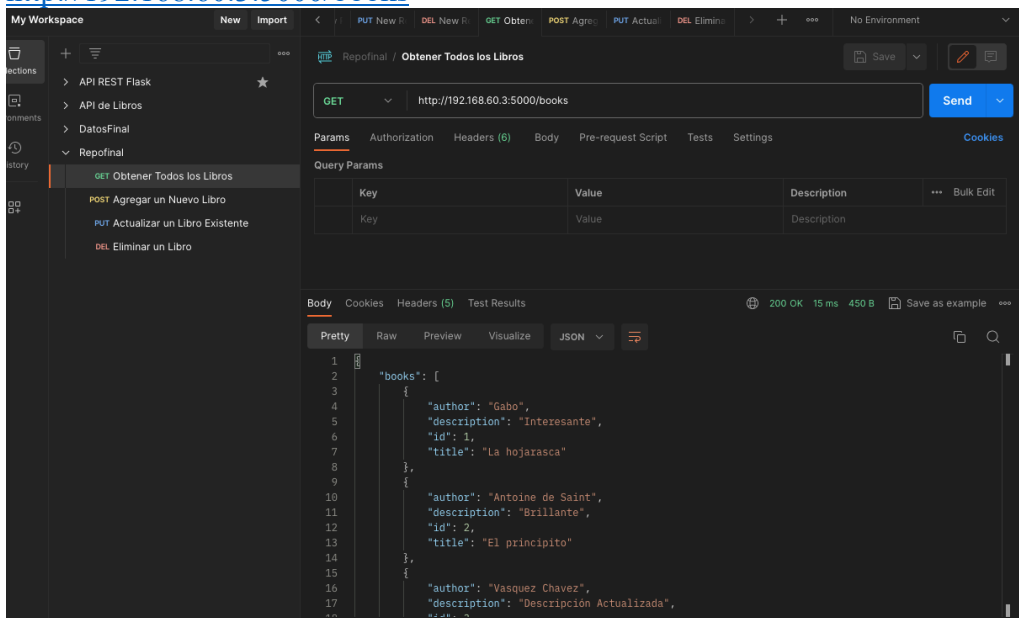
```
juanmartinvasquezcaicedo — zsh — 91x24

Last login: Tue Nov 14 10:25:52 on ttys001
[juanmartinvasquezcaicedo@JUANS-MacBook-Air-5 ~ % curl http://192.168.60.3:5000/books
{"books":[{"author":"Gabo","description":"Interesante","id":1,"title":"La hojarasca"}, {"author":"Vasquez Chavez","description":"Descripci\u00f3n Actualizada","id":3,"title":"Libro Actualizado"}]}
juanmartinvasquezcaicedo@JUANS-MacBook-Air-5 ~ %
```

PRUEBAS DESDE POSTMAN

Petición GET

<http://192.168.60.3:5000/books>



Petición POST

The screenshot shows the Postman interface with a workspace named 'My Workspace'. The left sidebar shows a collection 'API REST Flask' with a sub-collection 'API de Libros'. The 'Reporfinal' folder is expanded, showing three requests: 'GET Obtener Todos los Libros', 'POST Agregar un Nuevo Libro' (selected), and 'PUT Actualizar un Libro Existente'. The main panel shows the 'POST' request to 'http://192.168.60.3:5000/books'. The 'Body' tab is selected, showing a JSON payload:

```
{  "title": "Nuevo Libro",  "description": "Descripción del Nuevo Libro",  "author": "Autor del Nuevo Libro"}
```

. The response is shown in the 'Body' tab, indicating a '201 CREATED' status with a response time of 30 ms and 287 B of data. The response body is a JSON object:

```
{  "book": {    "author": "Autor del Nuevo Libro",    "description": "Descripción del Nuevo Libro",    "title": "Nuevo Libro"  }}
```

Petición PUT

The screenshot shows the Postman interface with a workspace named 'My Workspace'. The left sidebar shows the same collection 'API REST Flask' with the 'PUT Actualizar un Libro Existente' request selected. The main panel shows the 'PUT' request to 'http://192.168.60.3:5000/books/3'. The 'Body' tab is selected, showing a JSON payload:

```
{  "title": "Libro Actualizado",  "description": "Descripción Actualizada",  "author": "Vasquez Chavez"}
```

. The response is shown in the 'Body' tab, indicating a '200 OK' status with a response time of 32 ms and 289 B of data. The response body is a JSON object:

```
{  "book": {    "author": "Autor del Nuevo Libro",    "description": "Descripción del Nuevo Libro",    "id": 3,    "title": "Nuevo Libro"  }}
```

Petición DELETE

The screenshot shows the Postman interface with a workspace named 'My Workspace'. The left sidebar shows the same collection 'API REST Flask' with the 'DEL Eliminar un Libro' request selected. The main panel shows the 'DELETE' request to 'http://192.168.60.3:5000/books/2'. The 'Body' tab is selected, showing a JSON payload:

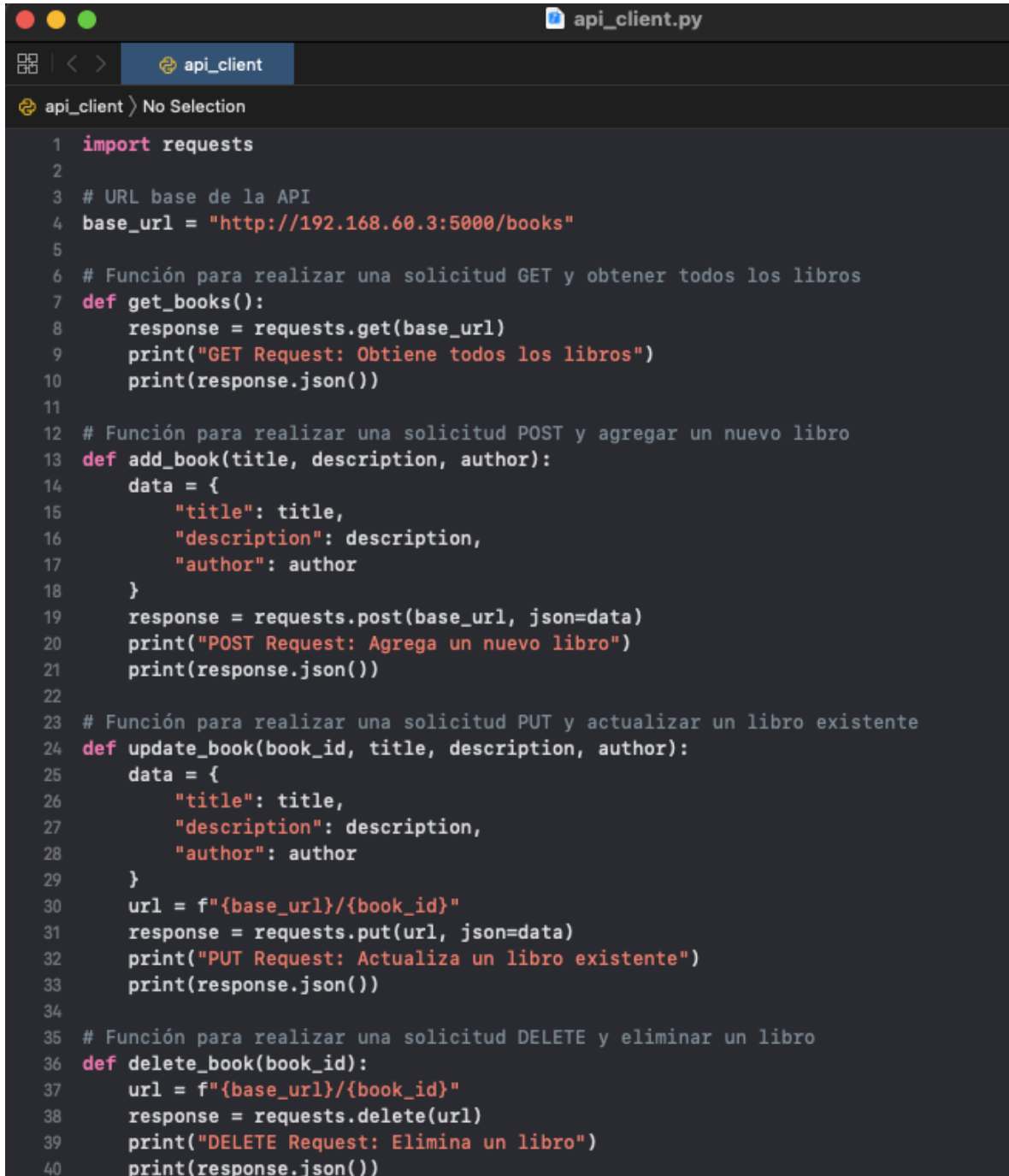
```
{  "result": true}
```

. The response is shown in the 'Body' tab, indicating a '200 OK' status with a response time of 34 ms and 182 B of data.


```
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.0.2.15:5000
Press CTRL+C to quit
192.168.60.1 - - [14/Nov/2023 15:41:25] "GET /books HTTP/1.1" 200 -
192.168.60.1 - - [14/Nov/2023 15:41:25] "GET /favicon.ico HTTP/1.1" 404 -
192.168.60.1 - - [14/Nov/2023 15:45:33] "GET /books HTTP/1.1" 200 -
192.168.60.1 - - [14/Nov/2023 15:47:04] "POST /books HTTP/1.1" 201 -
192.168.60.1 - - [14/Nov/2023 15:47:51] "GET /books HTTP/1.1" 200 -
{'id': 3, 'title': 'Nuevo Libro', 'description': 'Descripción del Nuevo Libro', 'author': 'Autor del Nuevo Libro'}
192.168.60.1 - - [14/Nov/2023 15:48:42] "PUT /books/3 HTTP/1.1" 200 -
192.168.60.1 - - [14/Nov/2023 15:48:55] "GET /books HTTP/1.1" 200 -
192.168.60.1 - - [14/Nov/2023 15:49:56] "DELETE /books/2 HTTP/1.1" 200 -
192.168.60.1 - - [14/Nov/2023 15:56:18] "GET /books HTTP/1.1" 200 -
192.168.60.1 - - [14/Nov/2023 16:10:04] "GET /books HTTP/1.1" 200 -
192.168.60.1 - - [14/Nov/2023 16:10:04] "POST /books HTTP/1.1" 201 -
192.168.60.1 - - [14/Nov/2023 16:10:04] "GET /books HTTP/1.1" 200 -
```

b. [0.5 Puntos]

Creamos un cliente simple en Python que pueda realizar las operaciones básicas (GET, POST, PUT, DELETE) en la API REST



```
1 import requests
2
3 # URL base de la API
4 base_url = "http://192.168.60.3:5000/books"
5
6 # Función para realizar una solicitud GET y obtener todos los libros
7 def get_books():
8     response = requests.get(base_url)
9     print("GET Request: Obtiene todos los libros")
10    print(response.json())
11
12 # Función para realizar una solicitud POST y agregar un nuevo libro
13 def add_book(title, description, author):
14     data = {
15         "title": title,
16         "description": description,
17         "author": author
18     }
19     response = requests.post(base_url, json=data)
20     print("POST Request: Agrega un nuevo libro")
21     print(response.json())
22
23 # Función para realizar una solicitud PUT y actualizar un libro existente
24 def update_book(book_id, title, description, author):
25     data = {
26         "title": title,
27         "description": description,
28         "author": author
29     }
30     url = f"{base_url}/{book_id}"
31     response = requests.put(url, json=data)
32     print("PUT Request: Actualiza un libro existente")
33     print(response.json())
34
35 # Función para realizar una solicitud DELETE y eliminar un libro
36 def delete_book(book_id):
37     url = f"{base_url}/{book_id}"
38     response = requests.delete(url)
39     print("DELETE Request: Elimina un libro")
40     print(response.json())
```

```
41
42 # Ejemplos de uso
43 get_books()
44
45 add_book("Nuevo Libro", "Descripción del Nuevo Libro", "Autor del Nuevo Libro")
46
47 get_books()
48
49 update_book(1, "Libro Actualizado", "Descripción Actualizada", "Autor Actualizado")
50
51 get_books()
52
53 delete_book(1)
54
55 get_books()
```

PRUEBA

Probamos desde el host

cd desktop

cd api_cliente

python3 api_client.py

```
[juanmartinvasquezcaicedo@JUANS-MacBook-Air-5 ~ % cd desktop
[juanmartinvasquezcaicedo@JUANS-MacBook-Air-5 desktop % cd api_cliente
cd: no such file or directory: api_cliente
[juanmartinvasquezcaicedo@JUANS-MacBook-Air-5 desktop % cd api_cliente
juanmartinvasquezcaicedo@JUANS-MacBook-Air-5 api_cliente % python3 api_client.py
/Users/juanmartinvasquezcaicedo/Library/Python/3.9/lib/python/site-packages/urllib3/__init__.py
:34: NotOpenSSLWarning: urllib3 v2.0 only supports OpenSSL 1.1.1+, currently the 'ssl' module i
s compiled with 'LibreSSL 2.8.3'. See: https://github.com/urllib3/urllib3/issues/3020
  warnings.warn(
GET Request: Obtiene todos los libros
{'books': [{'author': 'Gabo', 'description': 'Interesante', 'id': 1, 'title': 'La hojarasca'},
{'author': 'Vasquez Chavez', 'description': 'Descripción Actualizada', 'id': 3, 'title': 'Libro
Actualizado'}]}
POST Request: Agrega un nuevo libro
{'book': {'author': 'Autor del Nuevo Libro', 'description': 'Descripción del Nuevo Libro', 'tit
le': 'Nuevo Libro'}}
GET Request: Obtiene todos los libros
{'books': [{'author': 'Gabo', 'description': 'Interesante', 'id': 1, 'title': 'La hojarasca'},
{'author': 'Vasquez Chavez', 'description': 'Descripción Actualizada', 'id': 3, 'title': 'Libro
Actualizado'}, {'author': 'Autor del Nuevo Libro', 'description': 'Descripción del Nuevo Libro
', 'id': 4, 'title': 'Nuevo Libro'}]}
PUT Request: Actualiza un libro existente
{'book': {'author': 'Gabo', 'description': 'Interesante', 'id': 1, 'title': 'La hojarasca'}}
GET Request: Obtiene todos los libros
{'books': [{'author': 'Autor Actualizado', 'description': 'Descripción Actualizada', 'id': 1, '
title': 'Libro Actualizado'}, {'author': 'Vasquez Chavez', 'description': 'Descripción Actualiz
ada', 'id': 3, 'title': 'Libro Actualizado'}, {'author': 'Autor del Nuevo Libro', 'description'
: 'Descripción del Nuevo Libro', 'id': 4, 'title': 'Nuevo Libro'}]}
DELETE Request: Elimina un libro
{'result': True}
GET Request: Obtiene todos los libros
{'books': [{'author': 'Vasquez Chavez', 'description': 'Descripción Actualizada', 'id': 3, 'tit
le': 'Libro Actualizado'}, {'author': 'Autor del Nuevo Libro', 'description': 'Descripción del
Nuevo Libro', 'id': 4, 'title': 'Nuevo Libro'}]}
juanmartinvasquezcaicedo@JUANS-MacBook-Air-5 api_cliente %
```

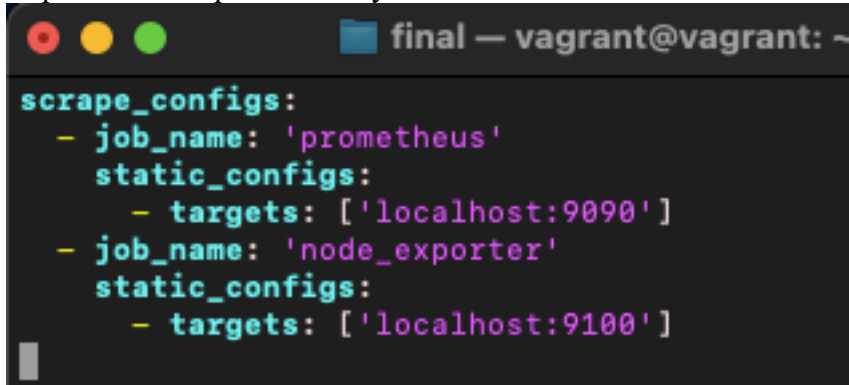
```
{'id': 1, 'title': 'La hojarasca', 'description': 'Interesante', 'author': 'Gabo'}  
192.168.60.1 - - [14/Nov/2023 16:10:04] "PUT /books/1 HTTP/1.1" 200 -  
192.168.60.1 - - [14/Nov/2023 16:10:04] "GET /books HTTP/1.1" 200 -  
192.168.60.1 - - [14/Nov/2023 16:10:04] "DELETE /books/1 HTTP/1.1" 200 -  
192.168.60.1 - - [14/Nov/2023 16:10:04] "GET /books HTTP/1.1" 200 -
```

Algunos comandos:

Buscar archivo:

```
find / -name prometheus.yml
```

Explicacion del prometheus.yml



```
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['localhost:9100']
```

Este bloque de configuración se encuentra en el archivo prometheus.yml y se utiliza para definir qué y cómo se "raspará" (scrape) la información de los objetivos o servicios que se están monitoreando.

- **scrape_configs:** Aquí defines las configuraciones de scraping, que son conjuntos de reglas para recolectar métricas de servicios específicos.
- **job_name:** Es un nombre que le das a la tarea de scraping. Cada **job** representa una instancia del servicio que estás monitoreando.
- **static_configs:** Aquí defines la configuración estática para este trabajo, que implica especificar los objetivos específicos que se van a raspar.
- **targets:** Son las direcciones y puertos donde el servicio objetivo expone sus métricas. En este caso:
- Para el trabajo '**prometheus**', se rasparán las métricas de Prometheus en **localhost:9090**.
- Para el trabajo '**node_exporter**', se rasparán las métricas del node_exporter en **localhost:9100**.

En resumen, este archivo le dice a Prometheus que raspe las métricas de dos trabajos: el propio Prometheus y el node_exporter, y especifica las direcciones y puertos donde encontrar esas métricas.

Explicacion del prometheus.service

```
final — vagrant@vagrant: ~ — ssh • vagrant ssh serviciofinal — 96x38
[Unit]
Description=Prometheus
After=network.target

[Service]
ExecStart=/opt/prometheus/prometheus --config.file=/opt/prometheus/prometheus.yml
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=always

[Install]
WantedBy=default.target
```

- **[Unit]:** Aquí defines las propiedades de la unidad systemd para el servicio.
 - **Description:** Proporciona una descripción del servicio. En este caso, simplemente indica que es para Prometheus.
 - **After=network.target:** Indica que este servicio debe iniciarse después de que la red esté disponible.
- **[Service]:** Define cómo systemd debe manejar el servicio.
 - **ExecStart:** Especifica la ruta del ejecutable y los argumentos necesarios para iniciar el servicio. En este caso, se ejecuta Prometheus con el archivo de configuración prometheus.yml en la ubicación **/opt/prometheus/**.
 - **ExecReload:** Indica cómo recargar el servicio cuando hay cambios en la configuración. En este caso, se envía la señal de recarga (**SIGHUP**) al proceso principal.
 - **KillMode=process:** Define cómo systemd debería enviar señales de finalización al servicio. En este caso, se hace enviando la señal directamente al proceso principal.
 - **Restart=always:** Indica que el servicio debe reiniciarse siempre que termine, sin importar la razón.
- **[Install]:** Proporciona información sobre cómo instalar o habilitar el servicio.
 - **WantedBy=default.target:** Indica que este servicio se activará al iniciar el sistema y se detendrá al apagar el sistema.

En resumen, este fragmento de configuración systemd define cómo iniciar, recargar y gestionar el ciclo de vida del servicio Prometheus en el sistema. Además, asegura que el servicio se inicie después de que la red esté disponible.

Explicacion node_exporter.service

```
final — vagrant@vagrant: ~ — s
[Unit]
Description=Node Exporter
After=network.target

[Service]
ExecStart=/usr/local/bin/node_exporter

[Install]
WantedBy=default.target
```

- **[Unit]:** Aquí defines las propiedades de la unidad systemd para el servicio del Node Exporter.
 - **Description:** Proporciona una descripción del servicio. En este caso, indica que es el Node Exporter, que es una herramienta para exportar métricas del sistema.
 - **After=network.target:** Indica que este servicio debe iniciarse después de que la red esté disponible.
- **[Service]:** Define cómo systemd debe manejar el servicio del Node Exporter.
 - **ExecStart:** Especifica la ruta del ejecutable para iniciar el servicio. En este caso, se ejecuta el binario **node_exporter** desde la ubicación **/usr/local/bin/**.
- **[Install]:** Proporciona información sobre cómo instalar o habilitar el servicio.
 - **WantedBy=default.target:** Indica que este servicio se activará al iniciar el sistema y se detendrá al apagar el sistema.

En resumen, este fragmento de configuración systemd establece las propiedades para el servicio del Node Exporter, asegurando que se inicie después de que la red esté disponible y especificando la ubicación del ejecutable para iniciar el servicio. Además, se configura para activarse al iniciar el sistema y detenerse al apagar el sistema.