Prometheus Assignment

Vikas K R

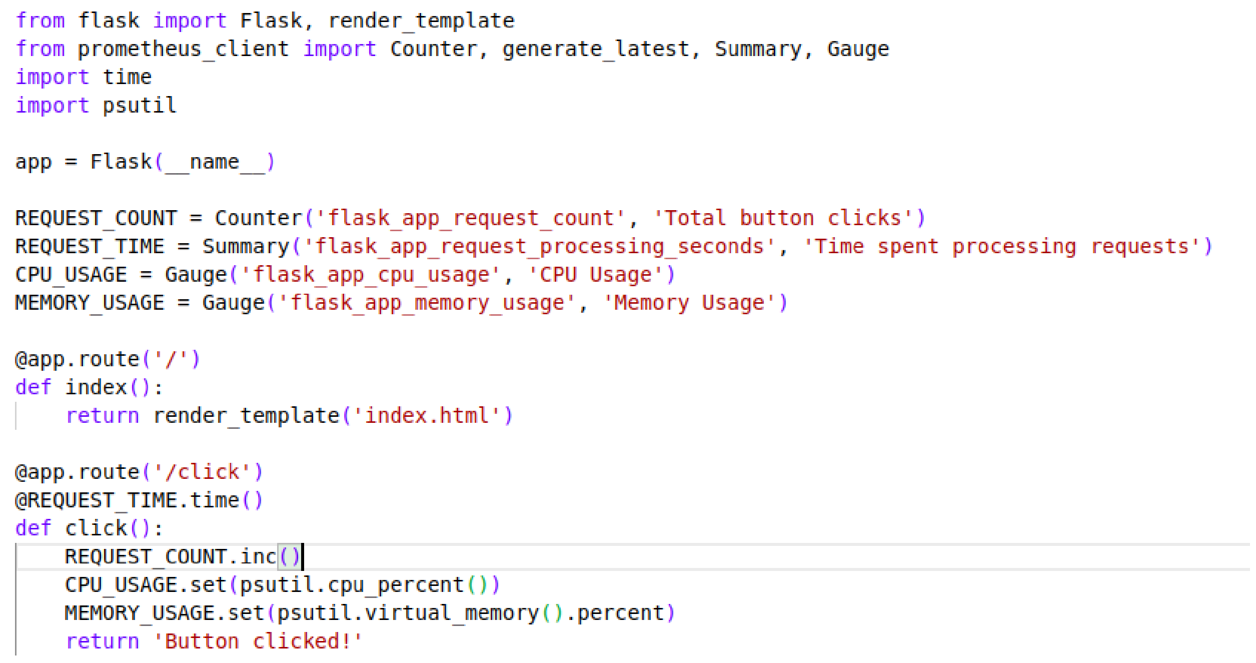
Question:

1. Export the metrics (like request per second, memory usage, cpu usage etc) in the existing mini project given to Interns
2. Install Prometheus and Grafana using Docker (with docker-compose)
3. Configure prometheus (scrape configs) such way that it can scrape the metrics from default metric path of the application job
4. Validate the entire configuration to check if the data is coming or not in Prometheus UI
5. Create the Dashboards in Grafana on top of the metrics exported by adding the Prometheus as a Datasource.

Solution:

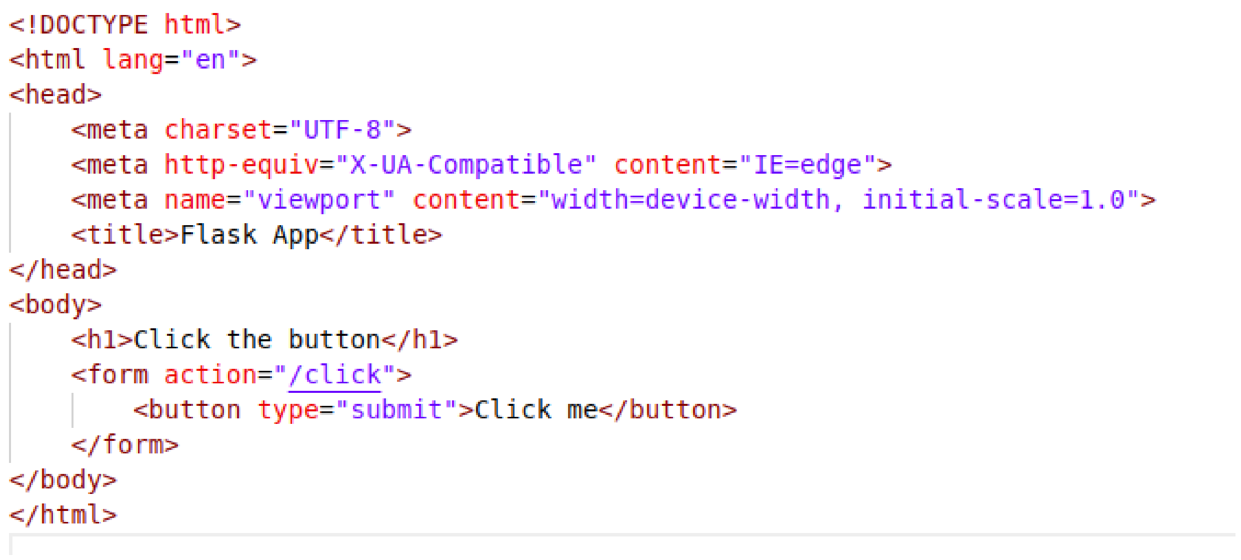
Step 1:

I created a folder where I write a simple Flask application that collects data on how many times an action is triggered using a Prometheus counter, as well as collects CPU usage and memory usage.



```python
from flask import Flask, render_template
from prometheus_client import Counter, generate_latest, Summary, Gauge
import time
import psutil

app = Flask(__name__)

REQUEST_COUNT = Counter('flask_app_request_count', 'Total button clicks')
REQUEST_TIME = Summary('flask_app_request_processing_seconds', 'Time spent processing requests')
CPU_USAGE = Gauge('flask_app_cpu_usage', 'CPU Usage')
MEMORY_USAGE = Gauge('flask_app_memory_usage', 'Memory Usage')

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/click')
@REQUEST_TIME.time()
def click():
    REQUEST_COUNT.inc()
    CPU_USAGE.set(psutil.cpu_percent())
    MEMORY_USAGE.set(psutil.virtual_memory().percent)
    return 'Button clicked!'
```

Index.html



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Flask App</title>
</head>
<body>
    <h1>Click the button</h1>
    <form action="/click">
        <button type="submit">Click me</button>
    </form>
</body>
</html>
```

And then i created docker file which contain

```dockerfile
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
COPY . .
CMD ["python", "app.py"]
```

Below is the Docker Compose file, which contains Grafana and Prometheus as you requested in the question (Install Prometheus and Grafana using Docker with Docker Compose). Prometheus is exposed on port 9096, Grafana on port 3000, and the admin password for Grafana is set to 'admin'.

```yaml
version: '3'
services:
 flask-app:
   build: .
   ports:
     - "5000:5000"
 prometheus:
   image: prom/prometheus
   volumes:
     - ./prometheus.yml:/etc/prometheus/prometheus.yml
   ports:
     - "9096:9090"

 grafana:
   image: grafana/grafana
   ports:
     - "3000:3000"
   environment:
     - GF_SECURITY_ADMIN_PASSWORD=admin
   volumes:
     - grafana-storage:/var/lib/grafana
volumes:
 grafana-storage:
```

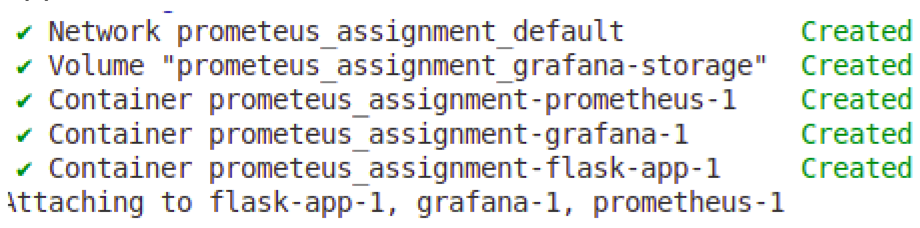
`Prometheus.yml`

This configuration sets the global scrape interval for Prometheus to 5 seconds, meaning it will collect metrics from monitored targets every 5 seconds. The scrape_configs section defines a job named "flask-app," which tells Prometheus to scrape metrics from the Flask application running on port 5000. The target for scraping is specified as flask-app:5000, allowing Prometheus to access the Flask app within the Docker network.

```
global:
 scrape_interval: 5s
scrape_configs:
 - job_name: 'flask-app'
   static_configs:
     - targets: ['flask-app:5000']
```

Step 2:

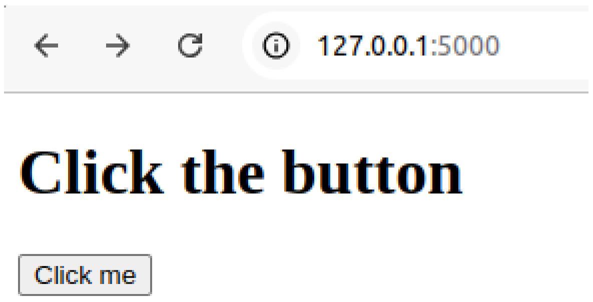
Now, run the **`docker-compose up --build`** command to execute the Docker Compose file, which will create and start the Prometheus, Grafana, and Flask app containers.



```
 ✔ Network prometeus_assignment_default             Created
 ✔ Volume "prometeus_assignment_grafana-storage"   Created
 ✔ Container prometeus_assignment-prometheus-1     Created
 ✔ Container prometeus_assignment-grafana-1        Created
 ✔ Container prometeus_assignment-flask-app-1      Created
\ttaching to flask-app-1, grafana-1, prometheus-1
```

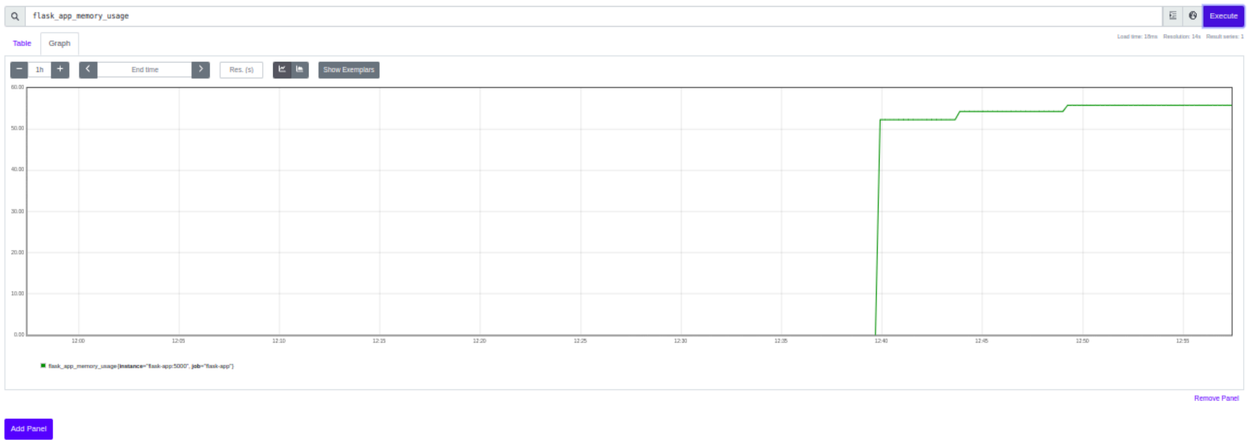
Step 3:

1. Use **`http://flask-app:5000`** to view the application interface.

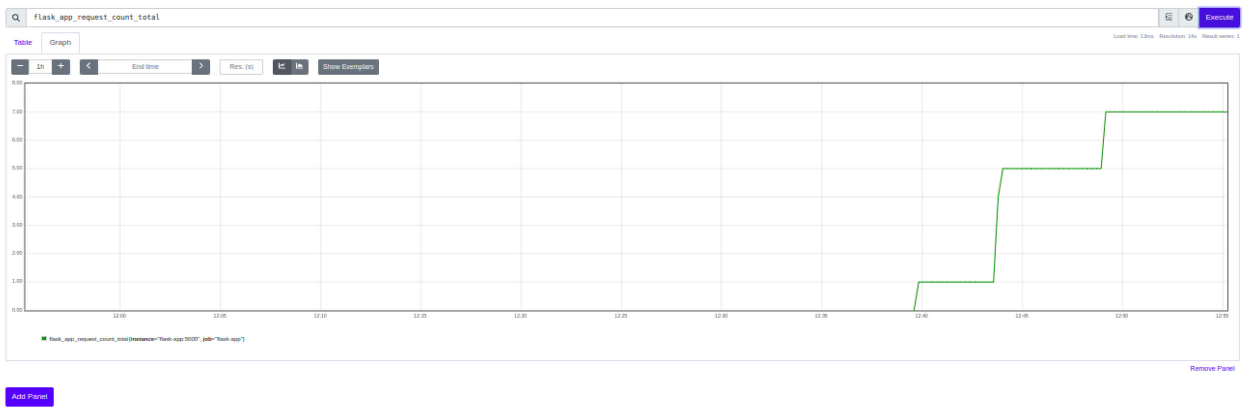


2. To view prometheus ui use http://localhost:9096 **and then you can view the different metrics in prometheus using different parameters like**

`flask_app_memory_usage`



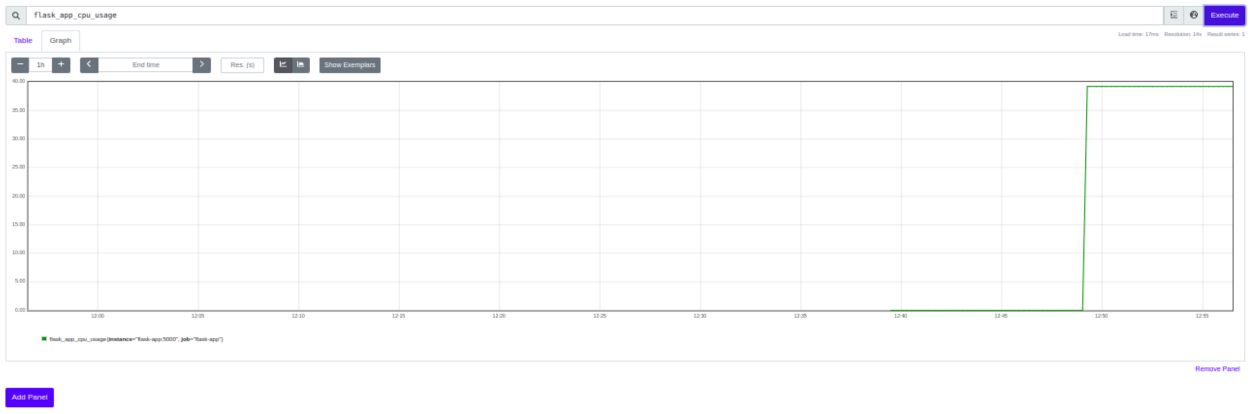
`Flask_app_cpu_usage`



```
flask_app_request_count_total
```

flask_app_request_count_total{instance="flask-app:5000", job="flask-app"}

`Flask_app_request_count_total`



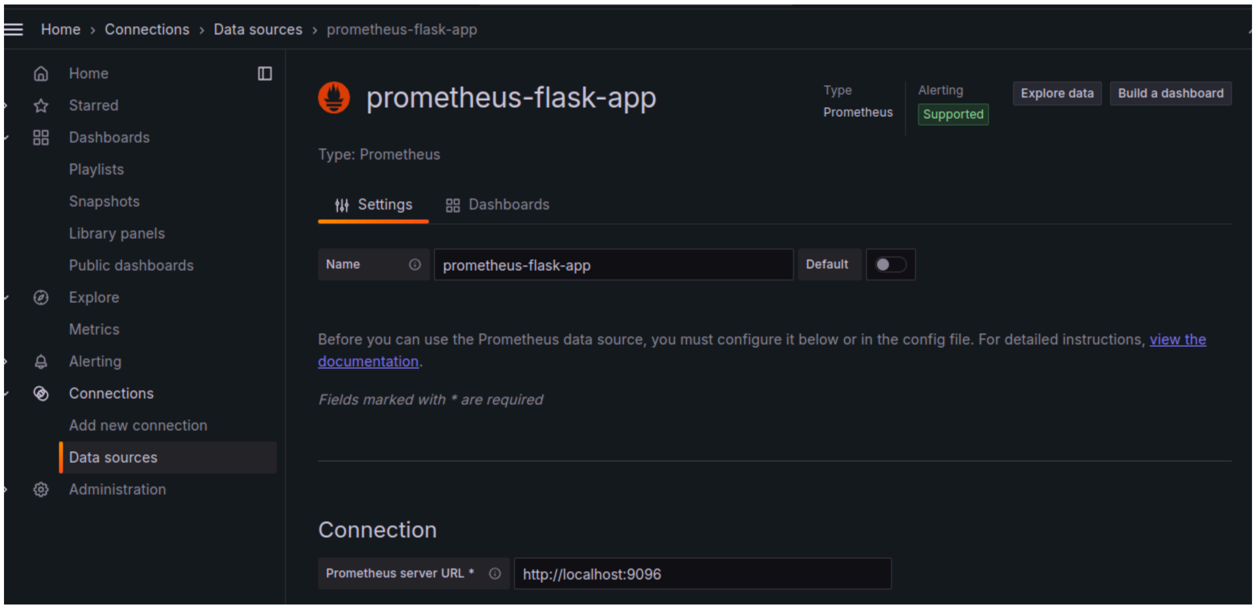
```
flask_app_cpu_usage
```

flask_app_cpu_usage{instance="flask-app:5000", job="flask-app"}

3. Grafana Configuration

Add Prometheus as Data Source:

- Go to Grafana UI at `http://localhost:3000`.
- Navigate to **Configuration > Data Sources** and ensure Prometheus is added.
- Check that the URL is set to `http://prometheus:9090`



prometheus-flask-app

Type: Prometheus

Alerting: Supported

Type: Prometheus

Name: prometheus-flask-app

Default

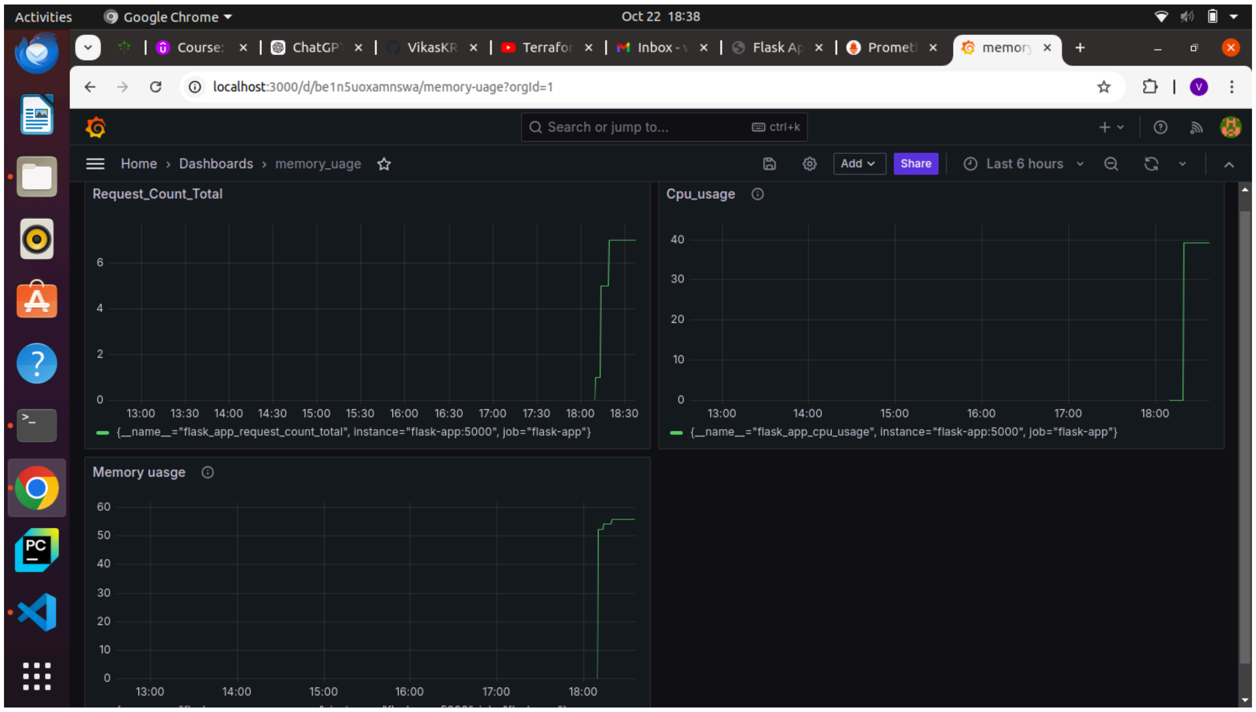
Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, view the documentation.

*Fields marked with * are required*

Connection

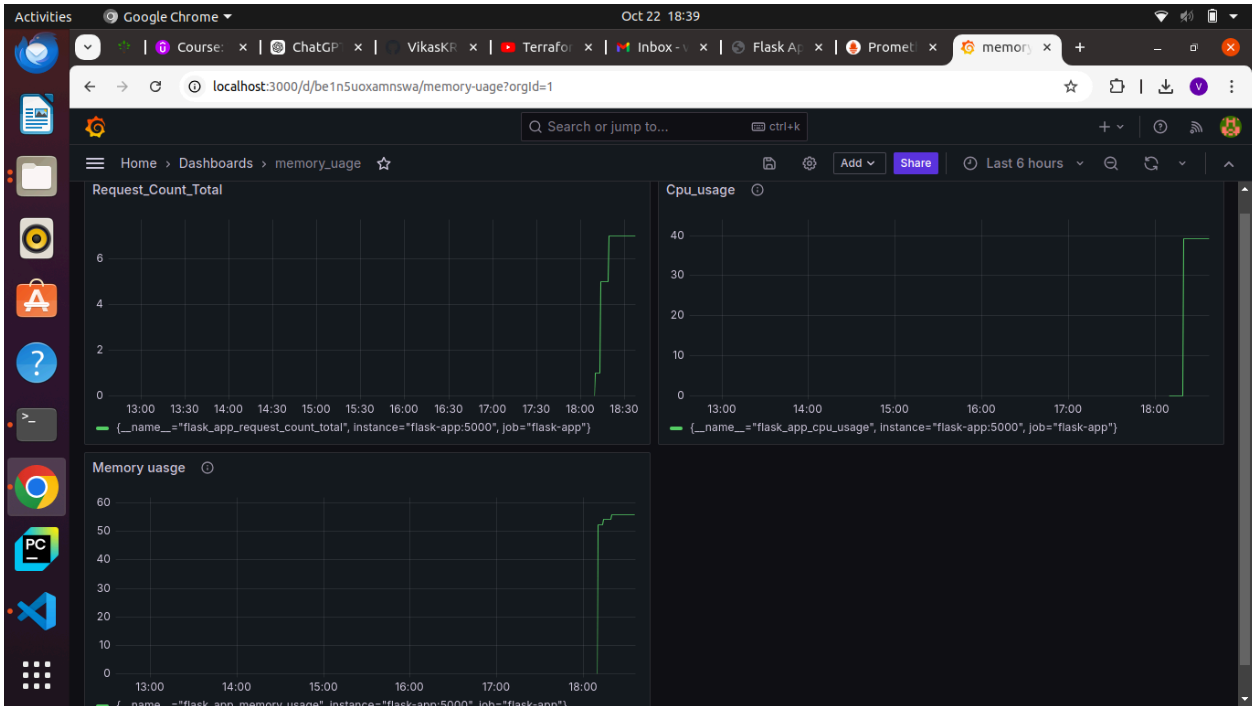
Prometheus server URL *: http://localhost:9096

4. Grafana Chart



{__name__="flask_app_request_count_total", instance="flask-app:5000", job="flask-app"}

{__name__="flask_app_cpu_usage", instance="flask-app:5000", job="flask-app"}



```
{__name__="flask_app_request_count_total", instance="flask-app:5000", job="flask-app"}
{__name__="flask_app_cpu_usage", instance="flask-app:5000", job="flask-app"}
{__name__="flask_app_memory_usage", instance="flask-app:5000", job="flask-app"}
```