Lesson End Project

Monitoring Python Web Application Using Prometheus and Grafana

Project agenda: To set up Prometheus and Grafana for monitoring a Python Flask web application, including instrumenting the application to expose metrics and configuring the tools for efficient tracking of both application and system performance

Description: You are part of a development team working on a popular e-commerce website built with Python Flask. As traffic increases, monitoring performance and ensuring reliability become crucial. You will deploy the Flask application with Prometheus and Grafana using Docker Compose, configure the Flask app to expose Prometheus metrics, set up Prometheus to collect these metrics, and configure Grafana for visualization. This setup will help your team gain insights into application behavior, detect issues proactively, and optimize performance for improved capacity planning.

Tools required: Linux operating system, Python, Flask, Docker, Grafana, Prometheus, and Node Exporter

Prerequisites: You must have Docker, docker-compose, and git installed in the lab to proceed.

Expected deliverables: A fully functional monitoring setup using Prometheus and Grafana to track and visualize metrics from a Python Flask web application deployed with Docker Compose

Steps to be followed:

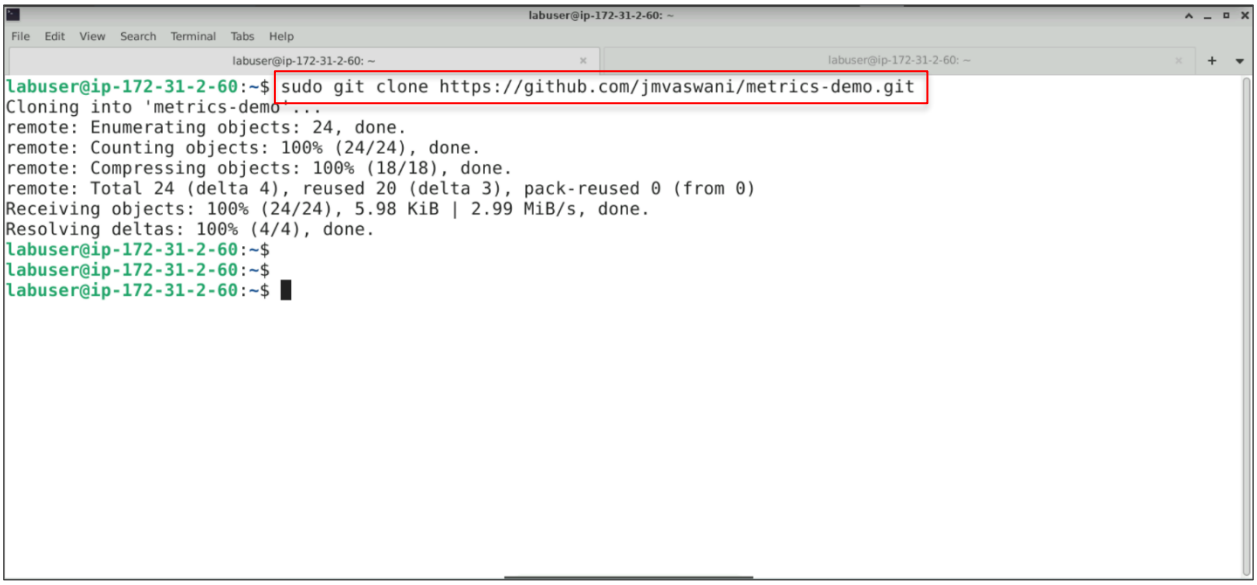
1. Clone the Python Flask repository and start the Docker services
2. Configure Prometheus as a data source in Grafana
3. Create a Grafana dashboard and visualize the Flask metrics

Note: Ensure that Prometheus server and Node exporter are already installed and running before building the container

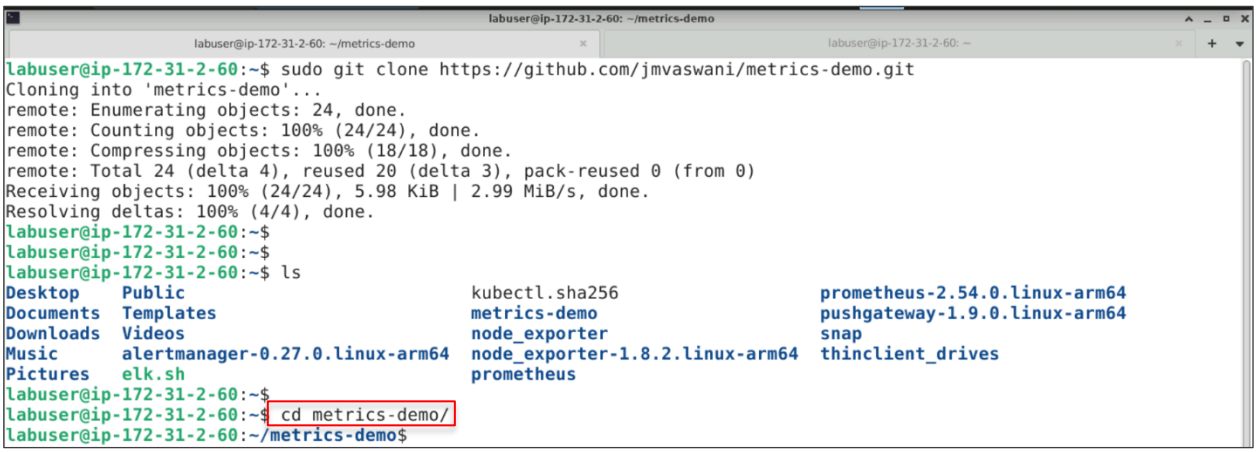
Note: Ensure that ports 3000 and 8080 are completely free from any process before building the Docker container

Step 1: Clone the Python Flask repository and start the Docker services

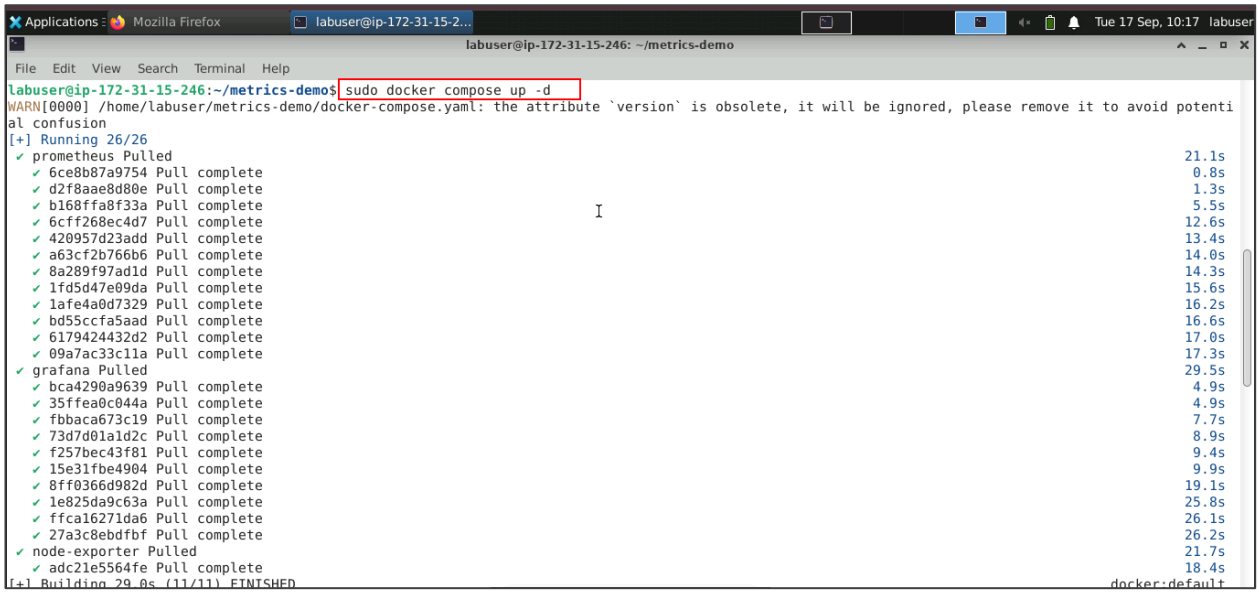
1.1 Navigate to the terminal and run the following command to clone the metrics-demo repository from GitHub to your local machine:
sudo git clone https://github.com/jmvaswani/metrics-demo.git



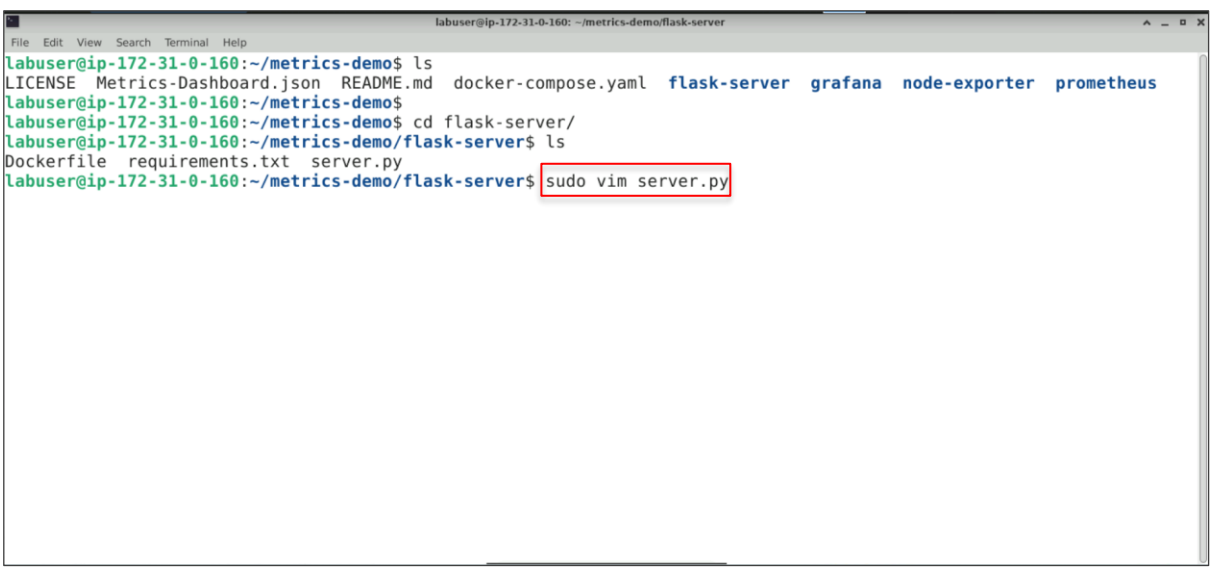
1.2 Run the following command to navigate from the current directory to the **metrics-demo** directory:
cd metrics-demo/



1.3 Run the following command to build and start the Docker containers:
sudo docker compose up -d



1.4 Run the following command to open the file **server.py** using the **vim** editor:
sudo vim server.py



Note: You can also use the cat command: **cat <file_name>** to view the configuration and Python files without making changes.

The flask code is as follows:

```
labuser@ip-172-31-0-160: ~/metrics-demo/flask-server
File  Edit  View  Search  Terminal  Help
from flask import Flask,request
app = Flask(__name__)


from prometheus_flask_exporter import PrometheusMetrics
metrics = PrometheusMetrics(app)
metrics.info('app_info', 'Application info', version='1.0.3')



@app.post("/route1")
def postroute1():
    return "Route 1 Post call"

@app.get("/route1")
def getroute1():
    return "Route 1 Get call"

@app.post("/route2")
def postroute2():
    return "Route 2 Post call"

@app.get("/route2")
def getroute2():
    return "Route 2 Get call"


# register additional default metrics
                                                                                    1,1           Top
```

```
labuser@ip-172-31-0-160: ~/metrics-demo/flask-server
File  Edit  View  Search  Terminal  Help
def postroute2():
    return "Route 2 Post call"

@app.get("/route2")
def getroute2():
    return "Route 2 Get call"


# register additional default metrics
metrics.register_default(
    metrics.counter(
        'by_path_counter', 'Request count by request paths',
        labels={'path': lambda: request.path}
    )
)



@app.route("/superSecretRoute")
@metrics.do_not_track()
def iamasecretfunction():
    return "Something secret used to be here"



if __name__ == "__main__":
    app.run(host="0.0.0.0",port=8080)
                                                                                    47,1          Bot
```

Note: This Flask application has multiple routes for handling HTTP GET and POST requests. It creates a web server that listens on port 8080 and responds with different messages based on the requested route and HTTP method.

Note: The Python Flask application integrates with the Prometheus Flask Exporter library, exposing application metrics that can be scraped by Prometheus.

The following method sets up the **/metrics** route to export all collected metrics:
metrics = PrometheusMetrics(app)

```
labuser@ip-172-31-0-160: ~/metrics-demo/flask-server
File  Edit  View  Search  Terminal  Help
from flask import Flask,request
app = Flask(__name__)


from prometheus_flask_exporter import PrometheusMetrics
metrics = PrometheusMetrics(app)
metrics.info('app_info', 'Application info', version='1.0.3')



@app.post("/route1")
def postroute1():
    return "Route 1 Post call"

@app.get("/route1")
def getroute1():
    return "Route 1 Get call"

@app.post("/route2")
def postroute2():
    return "Route 2 Post call"

@app.get("/route2")
def getroute2():
    return "Route 2 Get call"


# register additional default metrics
                                                                                    1,1           Top
```

1.5 Run the following commands to change the directory to **node_exporter** and open the file:
cd node-exporter/
docker-compose.yml
sudo vim docker-compose.yaml

```
labuser@ip-172-31-0-160: ~/metrics-demo/node-exporter
File  Edit  View  Search  Terminal  Help
labuser@ip-172-31-0-160:~/metrics-demo$ ls
LICENSE  Metrics-Dashboard.json  README.md  docker-compose.yaml  flask-server  grafana  node-exporter  prometheus
labuser@ip-172-31-0-160:~/metrics-demo$
labuser@ip-172-31-0-160:~/metrics-demo$ cd flask-server/
labuser@ip-172-31-0-160:~/metrics-demo/flask-server$ ls
Dockerfile  requirements.txt  server.py
labuser@ip-172-31-0-160:~/metrics-demo/flask-server$ sudo vim server.py
labuser@ip-172-31-0-160:~/metrics-demo/flask-server$
labuser@ip-172-31-0-160:~/metrics-demo/flask-server$ cd ..
labuser@ip-172-31-0-160:~/metrics-demo$ ls
LICENSE  Metrics-Dashboard.json  README.md  docker-compose.yaml  flask-server  grafana  node-exporter  prometheus
labuser@ip-172-31-0-160:~/metrics-demo$
labuser@ip-172-31-0-160:~/metrics-demo$ cd node-exporter/
labuser@ip-172-31-0-160:~/metrics-demo/node-exporter$ ls
docker-compose.yaml
labuser@ip-172-31-0-160:~/metrics-demo/node-exporter$
labuser@ip-172-31-0-160:~/metrics-demo/node-exporter$ sudo vim docker-compose.yaml _
```

The following configuration configures the node exporter to use the Docker image available to run the exporter.

```
labuser@ip-172-31-0-160: ~/metrics-demo/node-exporter
File  Edit  View  Search  Terminal  Help
version: "3"
services:
  node-exporter:
    image: bitnami/node-exporter:latest
    container_name: node-exporter
    ports:
      - "9100:9100"
~
"docker-compose.yaml" 7L, 145B                                    1,1           All
```

1.6 Run the following command to change the directory to **Prometheus:**
cd prometheus

```
Applications  Mozilla Firefox  labuser@ip-172-31-15-2...  Tue 17 Sep, 10:34  labuser
labuser@ip-172-31-15-246: ~/metrics-demo
File  Edit  View  Search  Terminal  Help
labuser@ip-172-31-15-246:~/metrics-demo$ sudo vim server.py
labuser@ip-172-31-15-246:~/metrics-demo$ sudo vim docker-compose.yaml
labuser@ip-172-31-15-246:~/metrics-demo$ ls
LICENSE  Metrics-Dashboard.json  README.md  docker-compose.yaml  flask-server  grafana  node-exporter  prometheus  server.py
labuser@ip-172-31-15-246:~/metrics-demo$ cd prometheus
```

1.7 Run the following command to open the file **config.yml** to review the configuration:
sudo vim config.yml

```
labuser@ip-172-31-0-160: ~/metrics-demo/prometheus
File  Edit  View  Search  Terminal  Help
labuser@ip-172-31-0-160:~/metrics-demo$ ls
LICENSE  Metrics-Dashboard.json  README.md  docker-compose.yaml  flask-server  grafana  node-exporter  prometheus
labuser@ip-172-31-0-160:~/metrics-demo$
labuser@ip-172-31-0-160:~/metrics-demo$ cd flask-server/
labuser@ip-172-31-0-160:~/metrics-demo/flask-server$ ls
Dockerfile  requirements.txt  server.py
labuser@ip-172-31-0-160:~/metrics-demo/flask-server$ sudo vim server.py
labuser@ip-172-31-0-160:~/metrics-demo/flask-server$
labuser@ip-172-31-0-160:~/metrics-demo/flask-server$ cd ..
labuser@ip-172-31-0-160:~/metrics-demo$ ls
LICENSE  Metrics-Dashboard.json  README.md  docker-compose.yaml  flask-server  grafana  node-exporter  prometheus
labuser@ip-172-31-0-160:~/metrics-demo$
labuser@ip-172-31-0-160:~/metrics-demo$ cd node-exporter/
labuser@ip-172-31-0-160:~/metrics-demo/node-exporter$ ls
docker-compose.yaml
labuser@ip-172-31-0-160:~/metrics-demo/node-exporter$
labuser@ip-172-31-0-160:~/metrics-demo/node-exporter$ sudo vim docker-compose.yaml
labuser@ip-172-31-0-160:~/metrics-demo/node-exporter$
labuser@ip-172-31-0-160:~/metrics-demo/node-exporter$ cd ..
labuser@ip-172-31-0-160:~/metrics-demo$ cd prometheus/
labuser@ip-172-31-0-160:~/metrics-demo/prometheus$
labuser@ip-172-31-0-160:~/metrics-demo/prometheus$ ls
config-debug.yml  config.yml  docker-compose.yaml
labuser@ip-172-31-0-160:~/metrics-demo/prometheus$
labuser@ip-172-31-0-160:~/metrics-demo/prometheus$ sudo vim config.yml
```

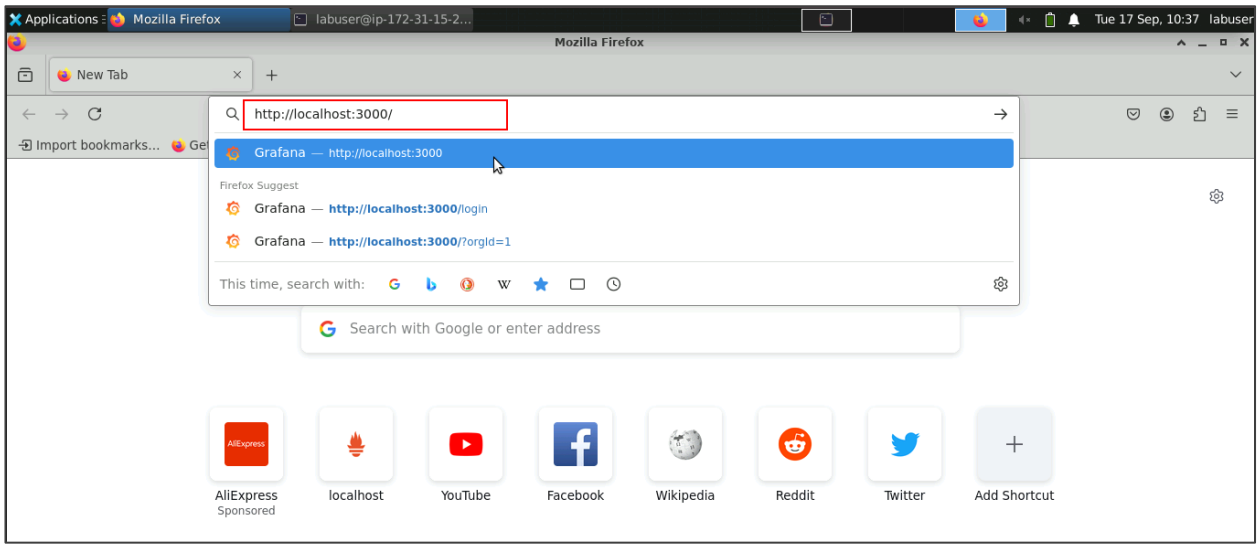
The configuration file appears as shown below:

```
labuser@ip-172-31-0-160: ~/metrics-demo/prometheus
File  Edit  View  Search  Terminal  Help
global:
  scrape_interval: 3s

  external_labels:
    monitor: "example-app"

rule_files:

scrape_configs:
  - job_name: "prometheus"
    static_configs:
      - targets: ["localhost:9090"]

  - job_name: "node-exporter"
    static_configs:
      - targets: ["node-exporter:9100"]

  - job_name: "python-app"
    static_configs:
      - targets: ["flask-app:8080"]
~
~
~
~
~
~
~
~
"config.yml" 20L, 365B                                                    1,1           All
```

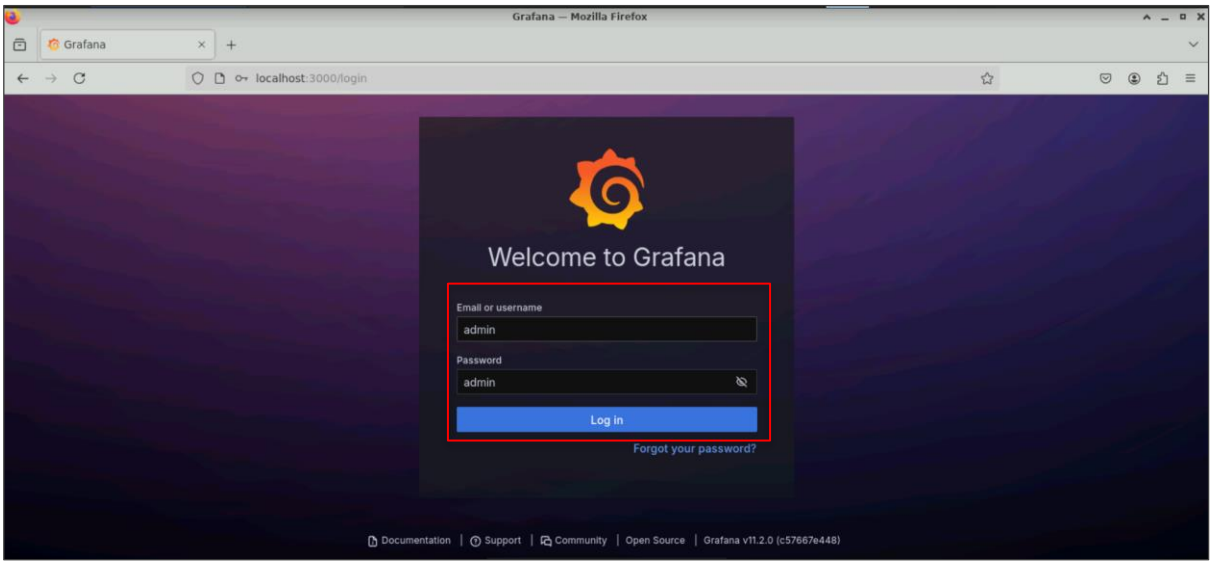
Note: The Prometheus config file is used to fetch metrics. This file defines three metrics servers, each with a name (used by Grafana to segregate data) and a target where the **/metrics** request will be routed.

Step 2: Configure Prometheus as a data source in Grafana

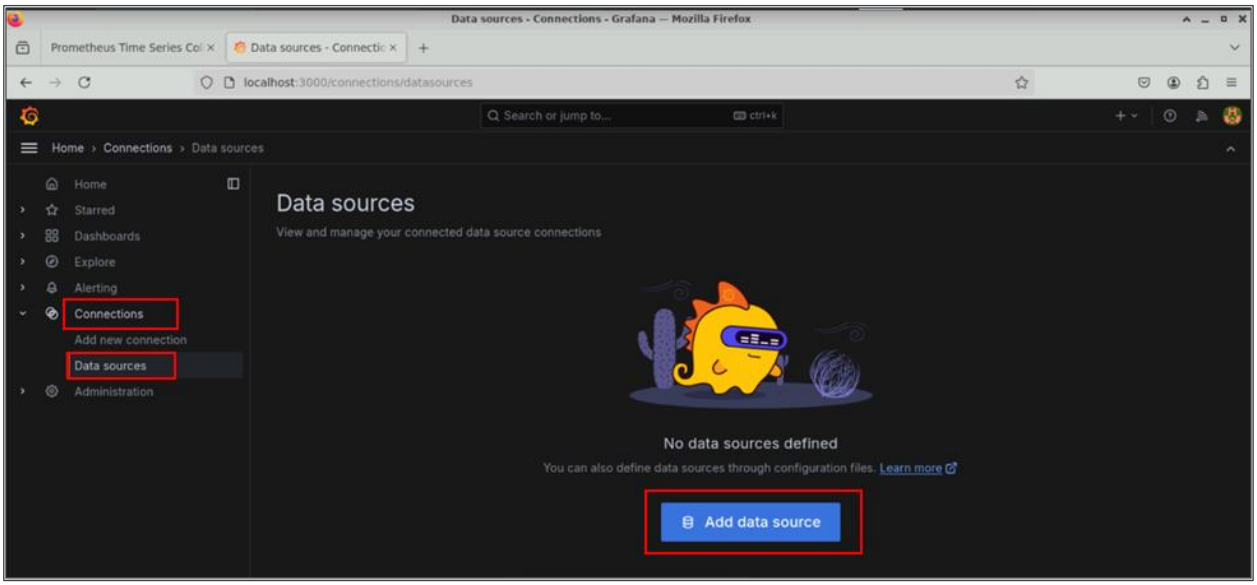
2.1 Open a web browser and access the Grafana dashboard using the following URL: **http://localhost:3000/**



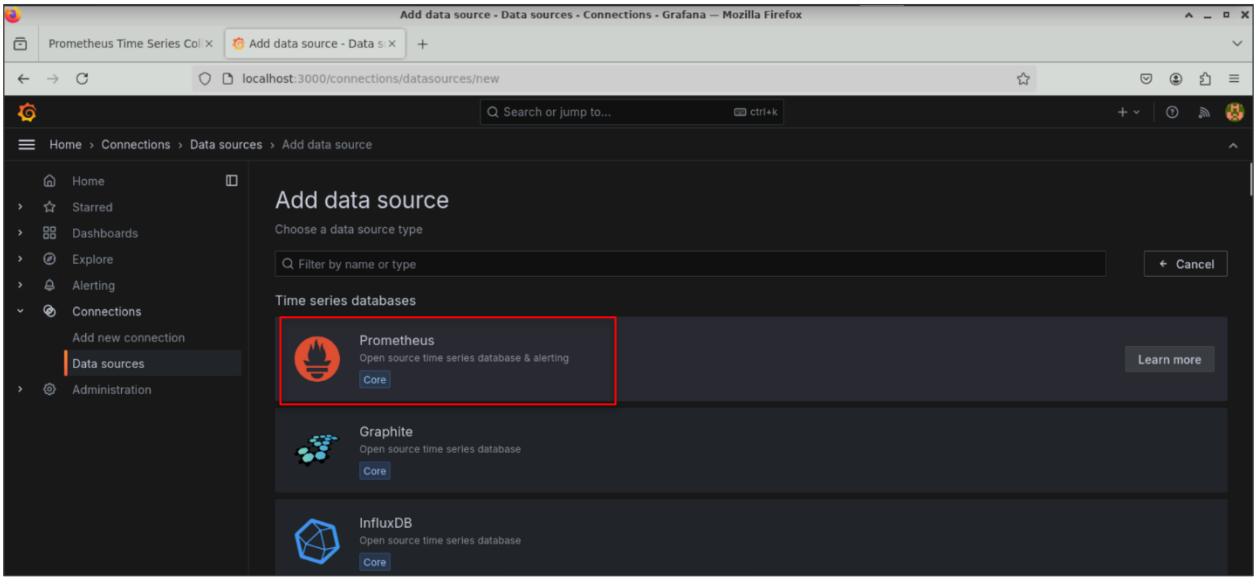
2.2 Enter **admin** as both the default **Username** and **Password**, then click **Log in**



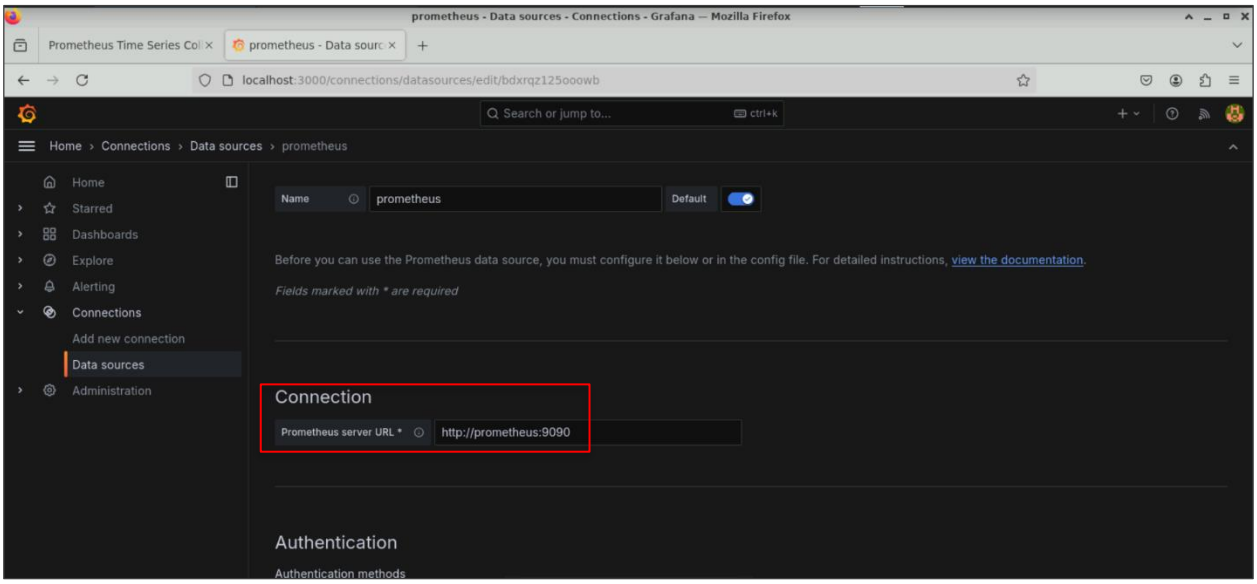
2.3 In the Grafana dashboard, select **Connection,** then choose **Data sources** from the menu on the left side, and then click **Add data source**



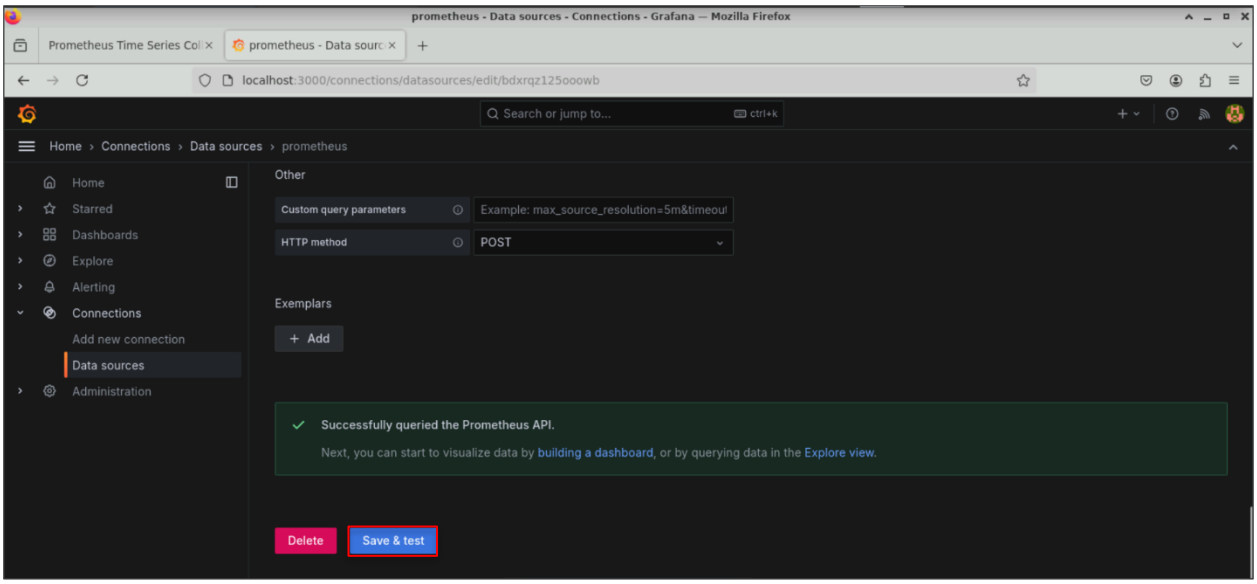
2.4 Select **Prometheus** as a data source from the list



2.5 In the **Connection** section, set the **Prometheus server URL** to **http://prometheus:9090**

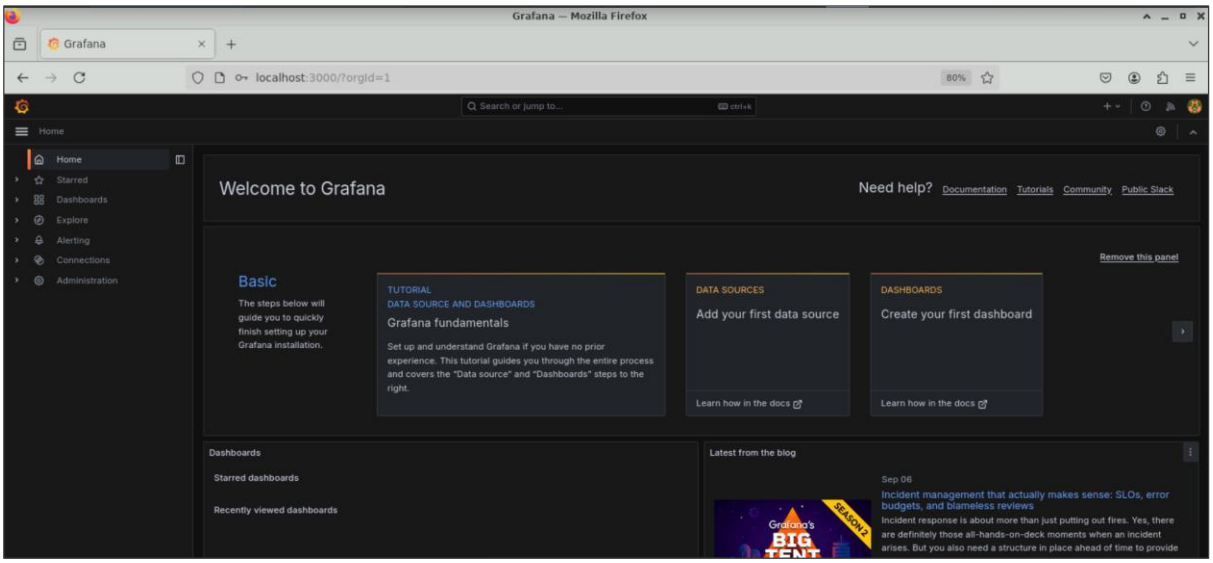


2.6 Scroll to the bottom, then click **Save & test** to save the Prometheus settings

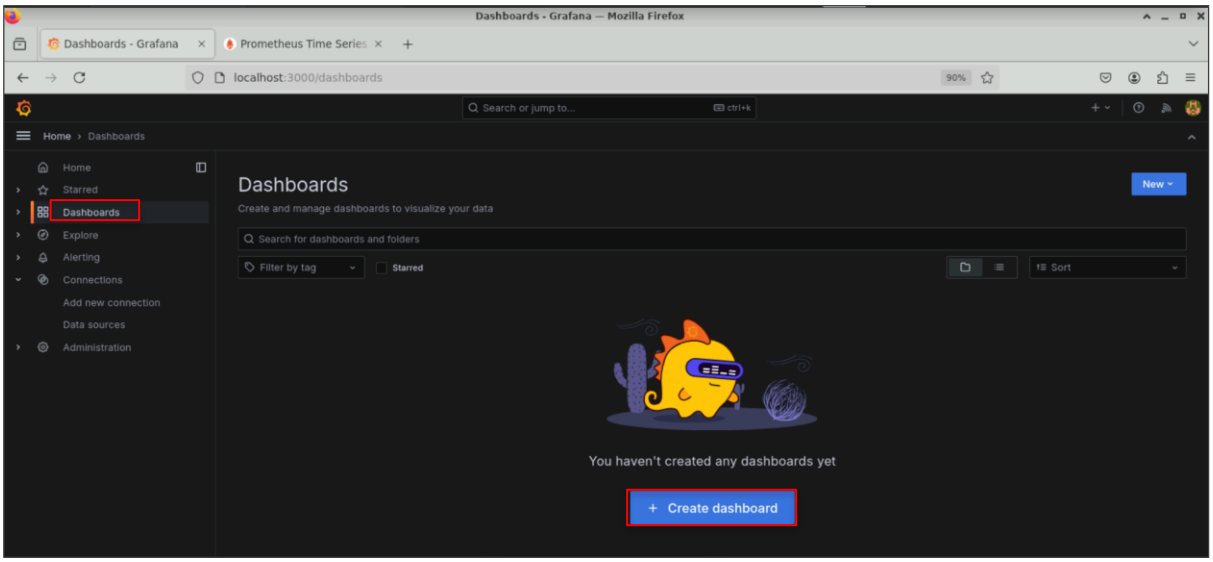


Step 3: Create a Grafana dashboard and visualize the Flask metrics

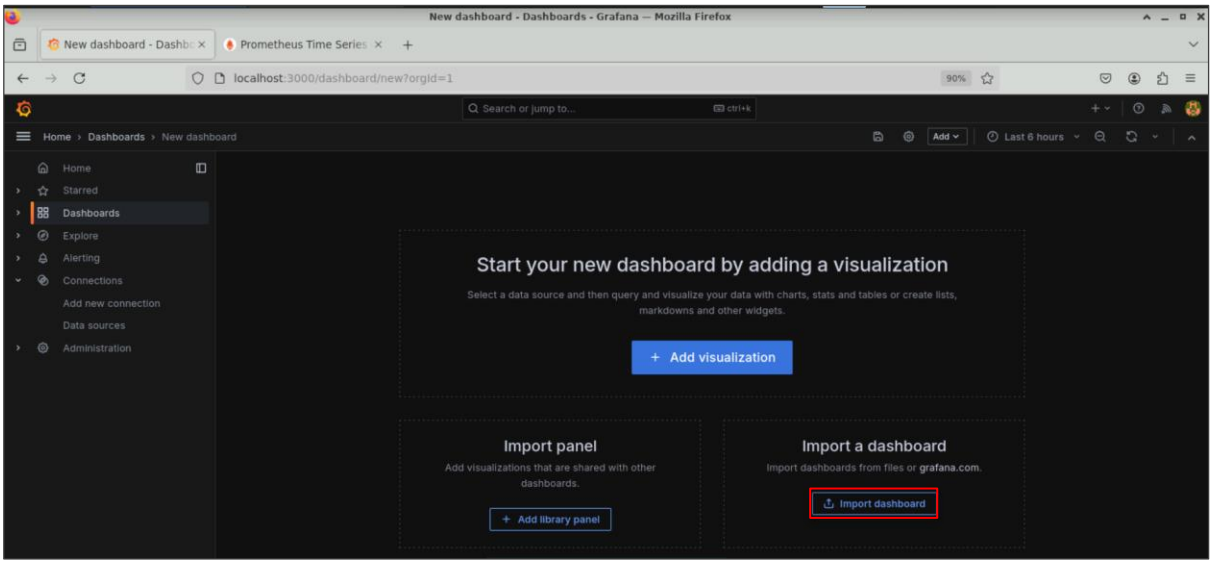
3.1 Navigate back to the Grafana dashboard using the URL **http://localhost:3000/**



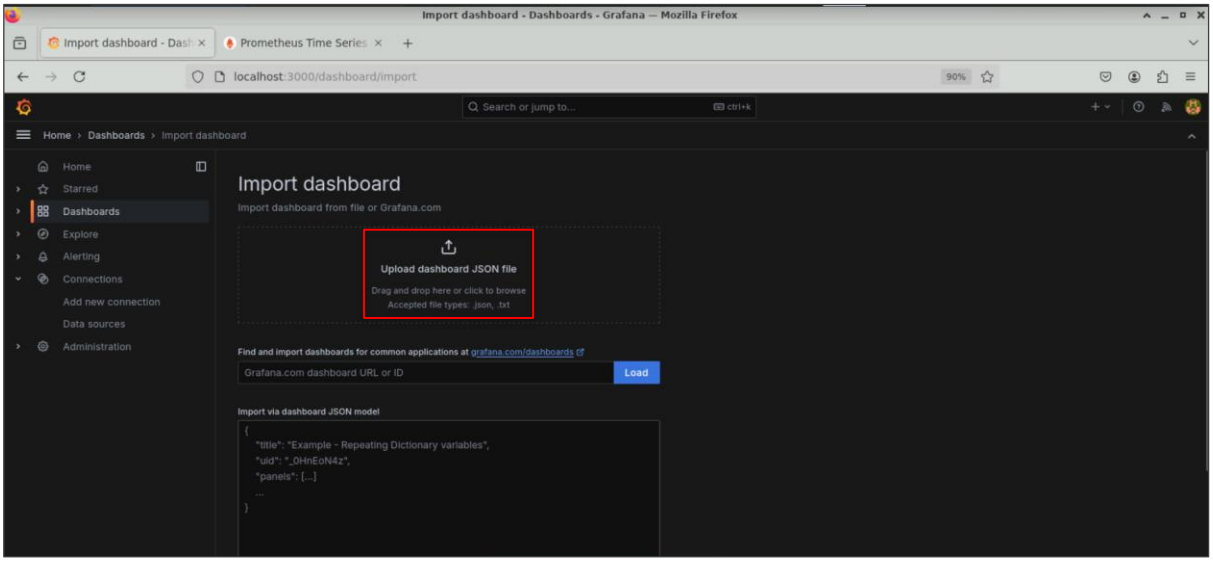
3.2 Navigate to the **Dashboards** section from the left-side menu, and click **+ Create dashboard**



3.3 Select **Import dashboard**

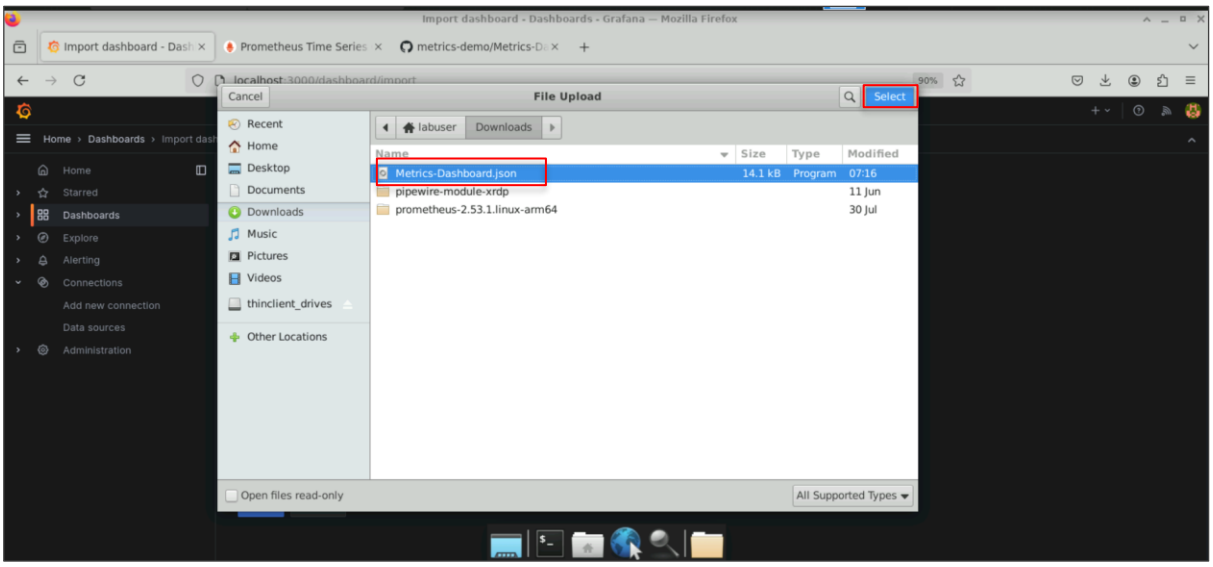


3.4 Click on **Upload dashboard JSON file**

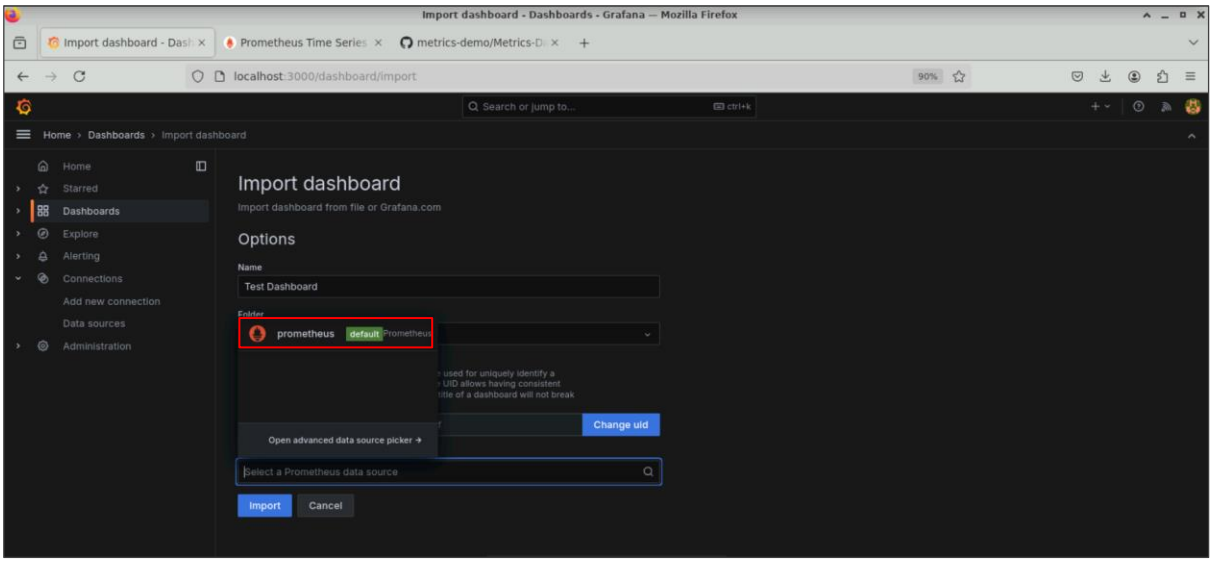


Note: Download the JSON file to your local drive from the link: **https://github.com/jmvaswani/metrics-demo/blob/bb305b09df825f3174e9e75dddb95fc0796baf3b/Metrics-Dashboard.json**

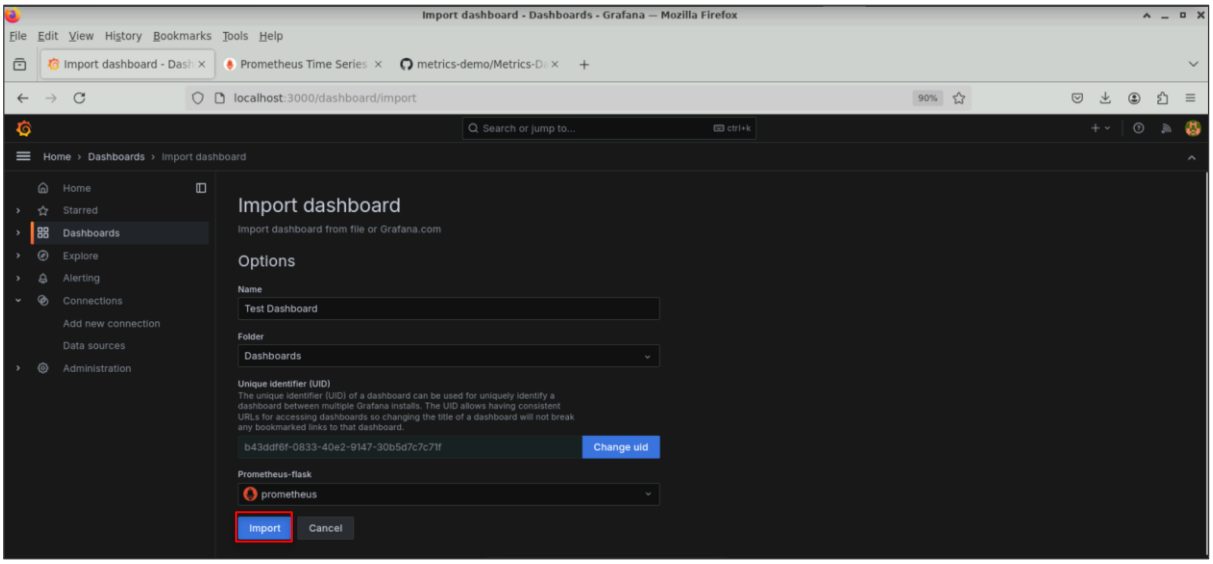
3.5 Navigate to the folder where the JSON file was downloaded, choose the JSON file, and click on **Select**



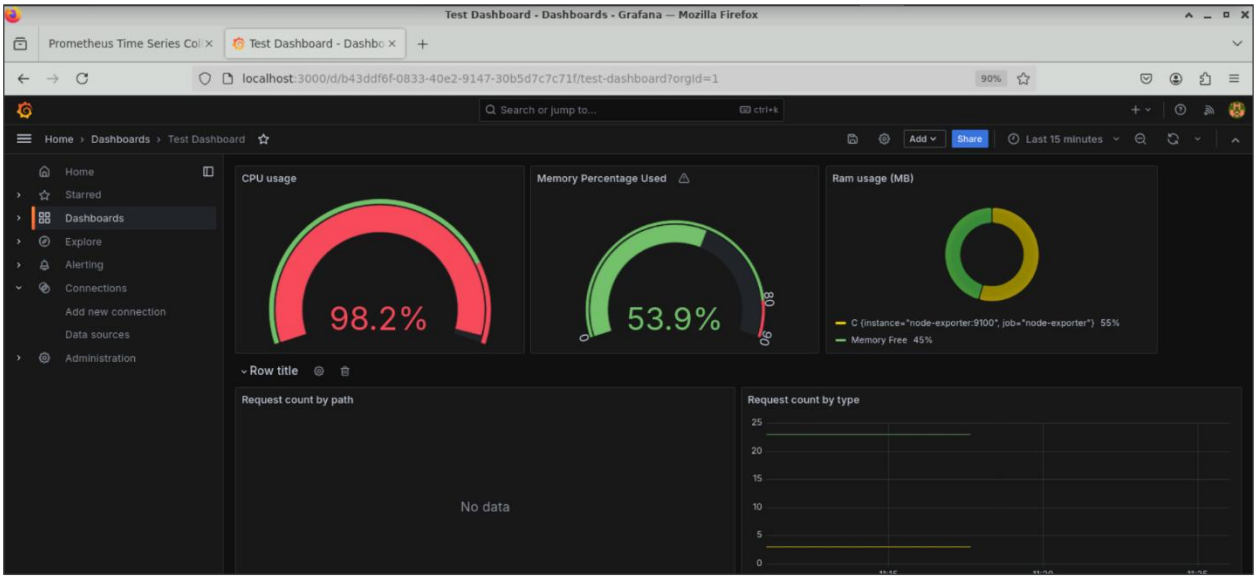
3.6 Select **prometheus** as the data source from the drop-down menu



3.7 Click on the **Import** button to visualize the metrics



You will see the following interface:



By following these steps, you have successfully set up Prometheus and Grafana to monitor a Python Flask application. This includes configuring Prometheus to scrape metrics and visualizing them in Grafana through customized dashboards.