**Introduction:**

This is an application in Python which is collecting external URL metrics and producing Prometheus metrics.

The following URLS are queried

1. <https://httpstat.us/200>
2. <https://httpstat.us/503>

The metrics in response are:-

1. **URL response time in milliseconds**
2. **URL status up or down using 1(200) or 0(503).**

This Python application will be built into docker image and then deployed to Kubernetes.

**Pre-requirements:**

1. Python3 environment
2. Kubernetes env(minikube or cloud-based)
3. Docker to be installed

**Steps :**

1. Install necessary packages from requirements.txt

*cd pythonapp/*

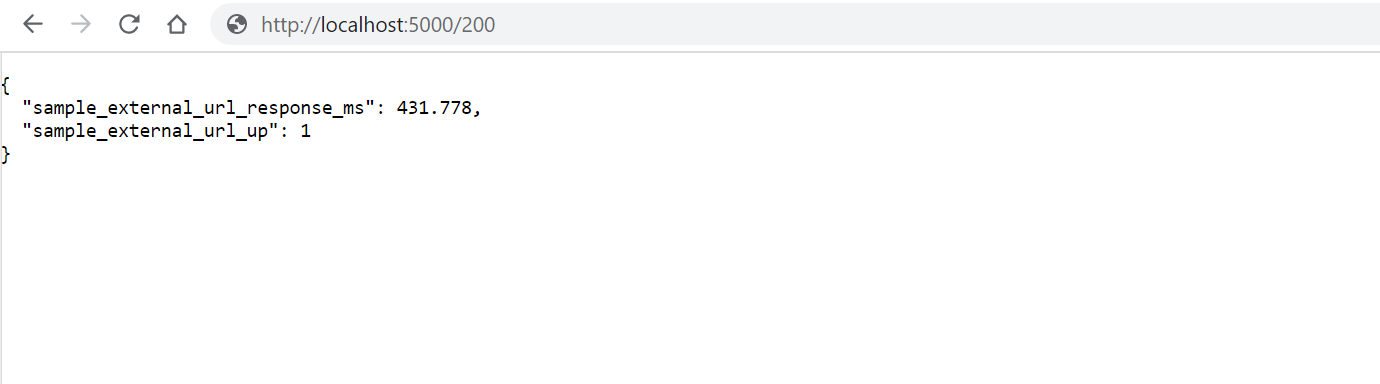
*pip install -r requirements.txt*

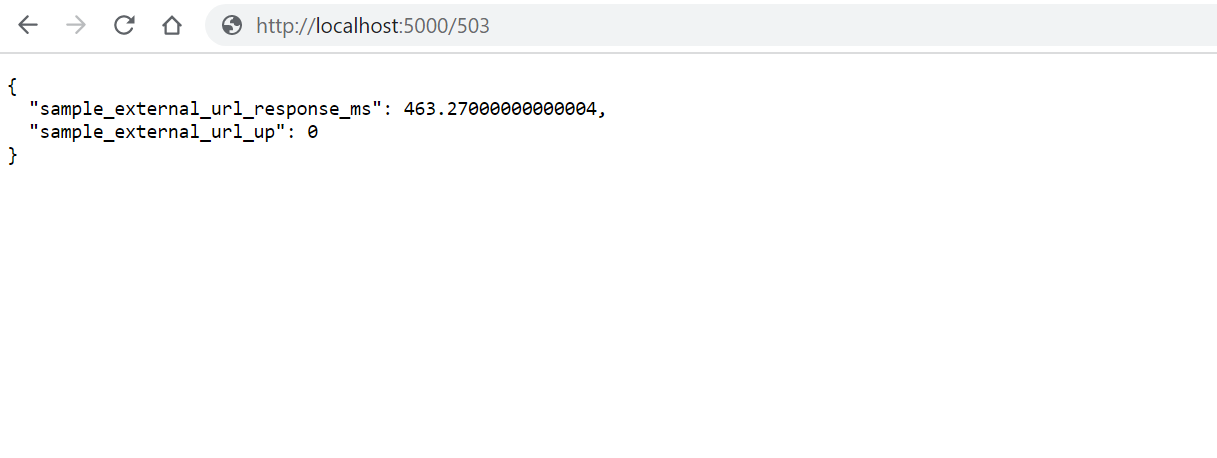
1. Run the application

*python pythonapp.py*

1. The application is designed to run on localhost. Endpoint /200 will hit <https://httpstat.us/200> and give suitable response. Similarly , the endpoint /503 will hit <https://httpstat.us/503> and give its response. Any other endpoints will throw error .

Example screenshots:





1. Run the pytest to verify the outputs if you get them as expected.

**Building Docker Containers**

1. Build the Docker image

*docker build -t pyapp pythonapp/ .*

1. Run docker-compose file to check if app is working fine and if you are able to see Prometheus metrics on 9090 and Grafana on 3000.

*nohup docker-compose -f docker-compose.yaml up > $(date +%Y%m%d%H%M%S).txt 2>&1 &*

(App endpoint : <http://localhost:5000> , Prometheus endpoint : <http://localhost:9090> ,

Grafana endpoint : <http://localhost:3000> )

**Deploying the Containers On K8s Cluster**

The file kubernetesmain.yaml is used to create deployments, services on Kubernetes. Secret.yaml is used to store Grafana login credentials.

The file contains:-

1. **Deployment** – It contains deployment of python application(local docker image), Prometheus and Grafana (public docker images)
2. **Service** – Contains the service endpoints for python application, Prometheus and Grafana.
3. Create a namespace

*kubectl create -n newdeploy*

1. Deploy secret first

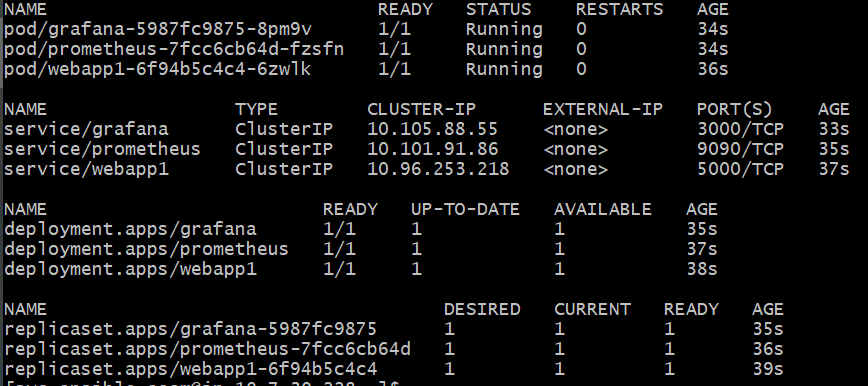
*kubectl apply -f secret.yaml -n newdeploy*

1. *Deploy the deployments and services.*

*kubectl apply -f kubernetesmain.yaml -n newdeploy*

1. Display all the components deployed

*kubectl get all -n newdeploy*



1. Forward all service endpoints to listen to node ports

*kubectl port-forward service/grafana -n newdeploy 3000:3000*

*kubectl port-forward service/prometheus -n newdeploy 5000:5000*

*kubectl port-forward service/webapp1 -n newdeploy 9090:9090*

1. Login to Prometheus pod and copy contents of prometheus/prometheus.yaml

To listen to the webapp . instead of local host, change the ip to the clusterip.

*kubectl exec -it prometheus-7fcc6cb64d-cf658 -n newdeploy -- /bin/sh*

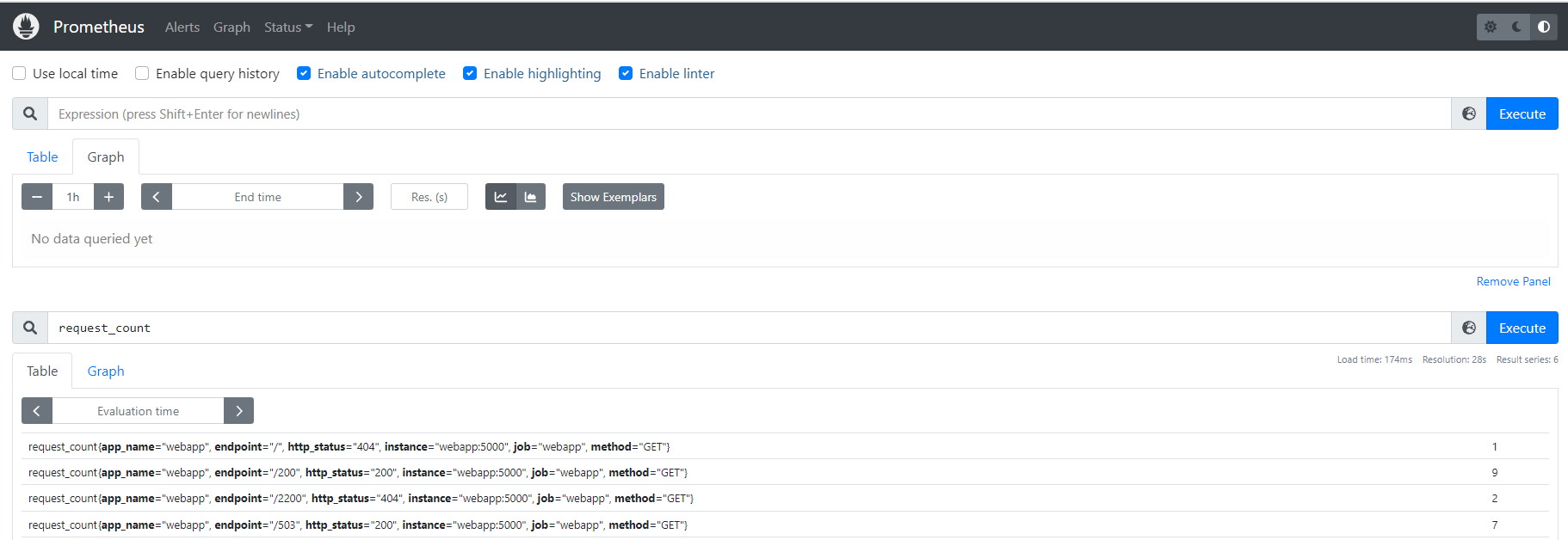
*vi /etc/prometheus/prometheus.yaml*

1. Open the endpoints again (App endpoint : <http://localhost:5000> , Prometheus endpoint : <http://localhost:9090> ,Grafana endpoint : <http://localhost:3000> ) to check if they are loading as expected.

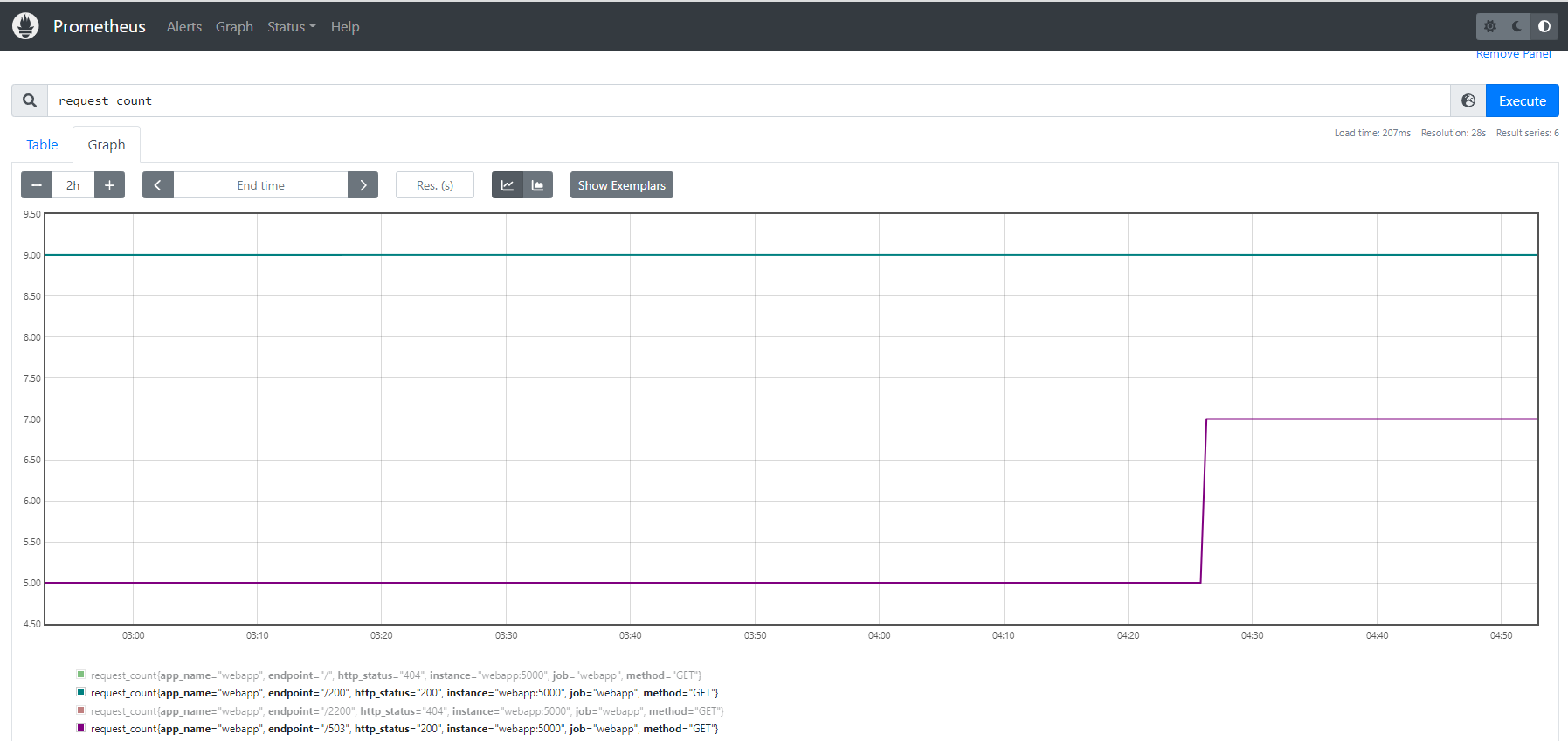
**Data Observability in Prometheus**

In prometheus(<http://localhost:9090> ), go to “graph” section , and in table , choose

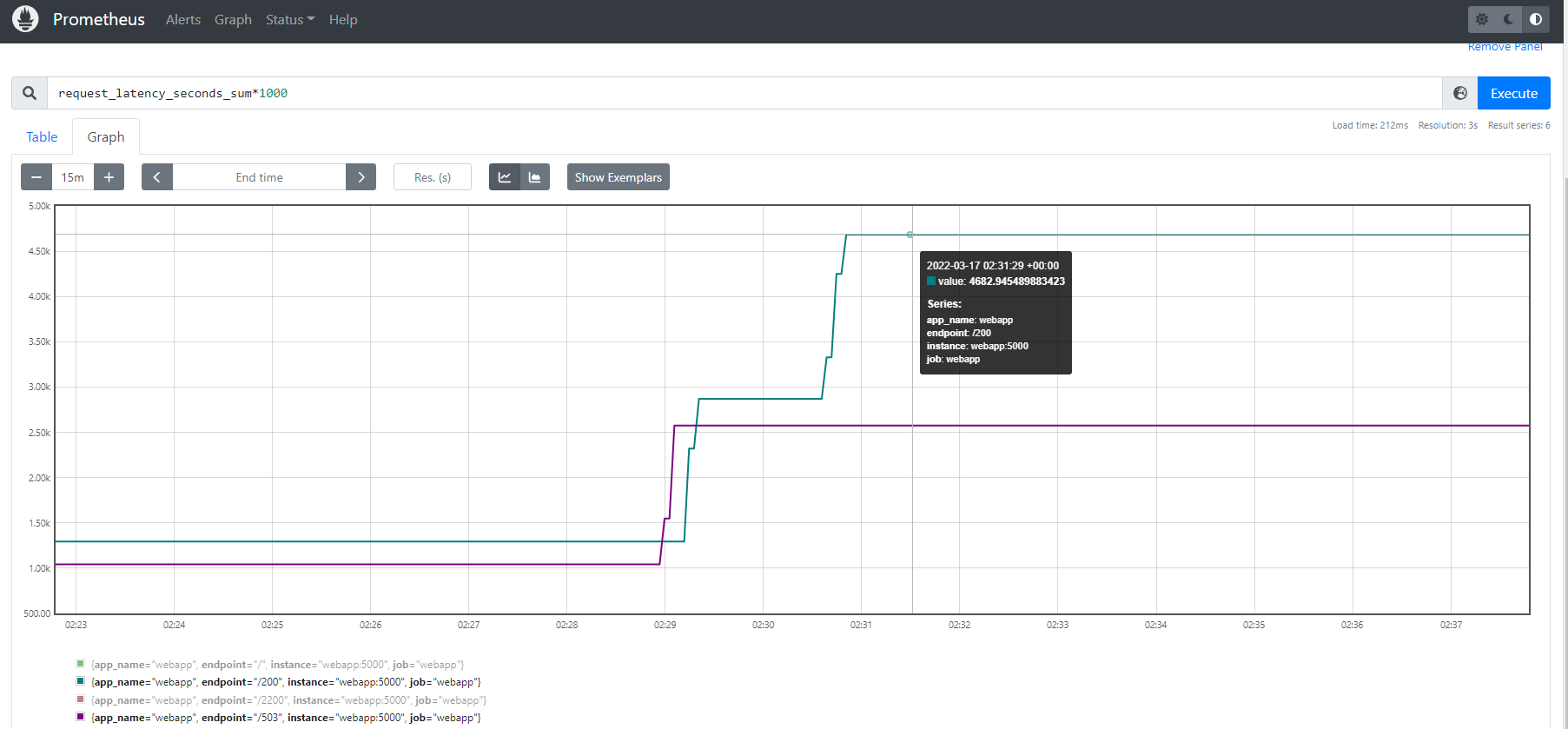
**request\_count** to see the number of requests for each url.



It can be represented as graph also



We can also check the total latency for each request over the time period in milliseconds.





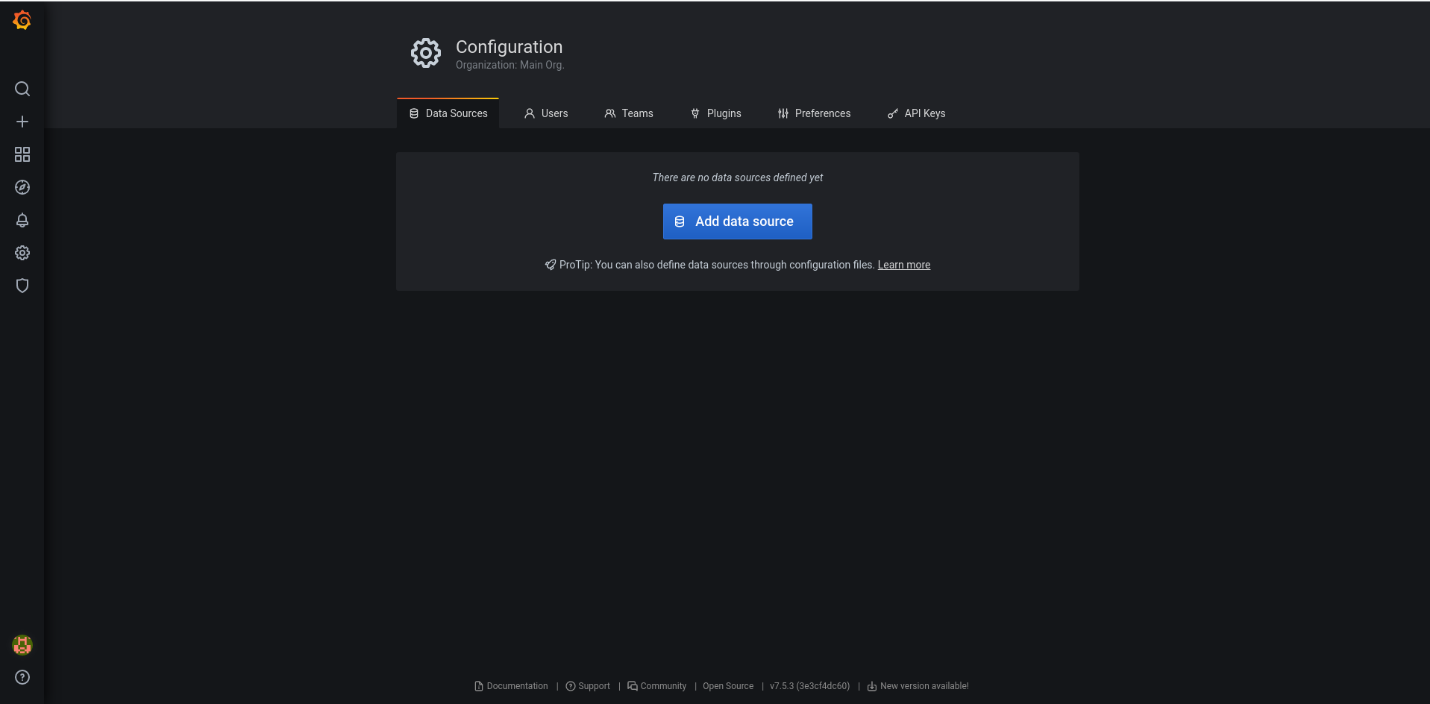
**Adding Prometheus Data Source To Grafana**

1. Open Grafana

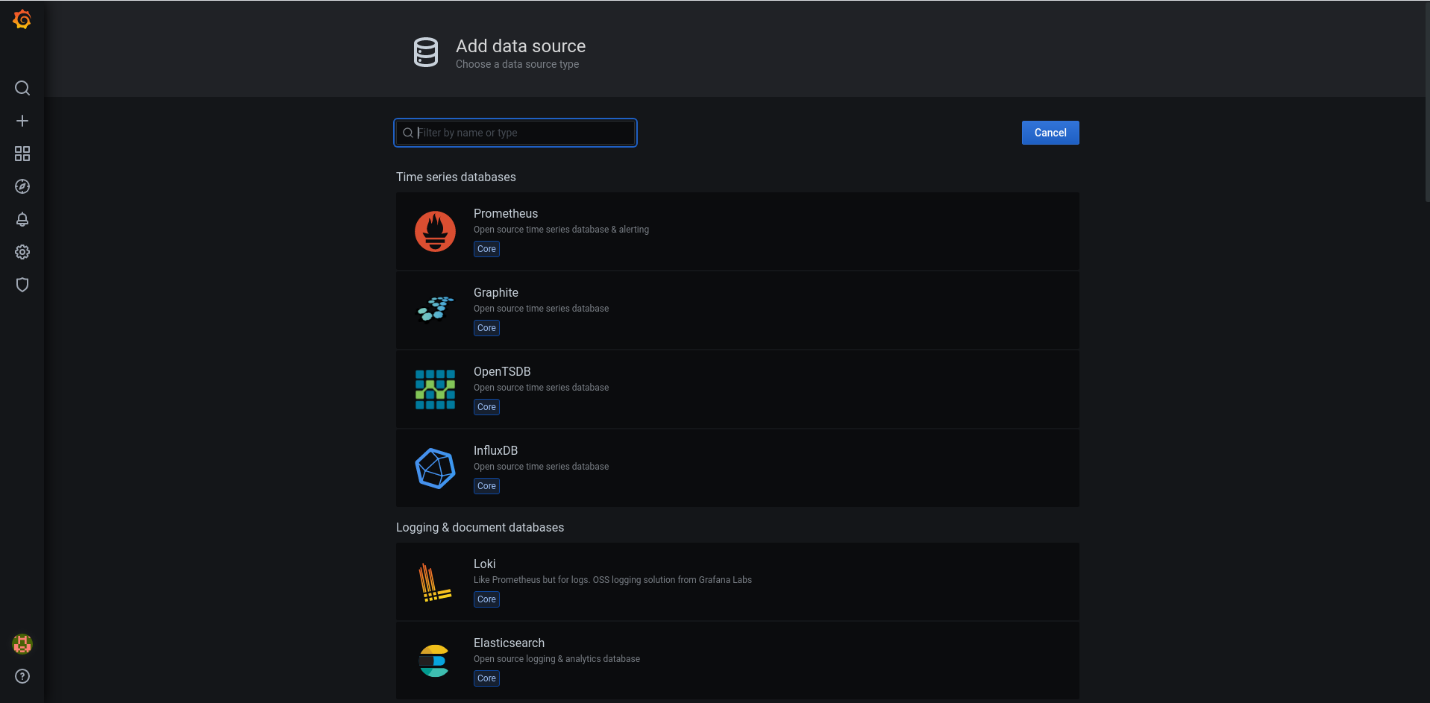
Open your browser and point to [http://localhost:3000](http://localhost:3000/) you will see **Grafana Login**.

Enter the username and password to login.(default username is admin and password can be retrieved from secrets)

1. Click on Configuration > Data Sources
2. Click on Add data source

[](https://raw.githubusercontent.com/himadriganguly/sample_external_url/main/screenshots/grafan-configuration.png)

1. Select Prometheus as the data source

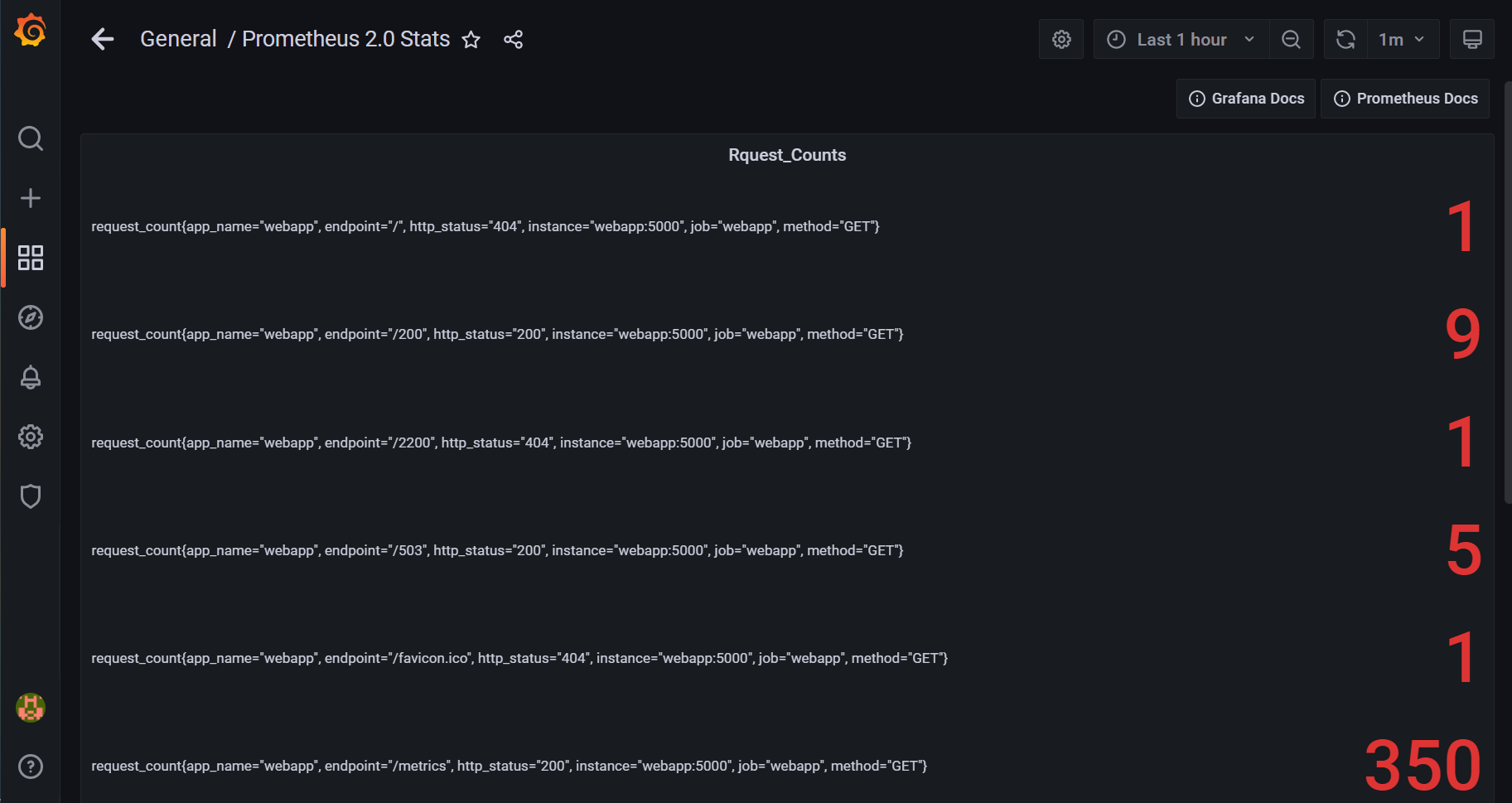
[](https://raw.githubusercontent.com/himadriganguly/sample_external_url/main/screenshots/grafan-configuration-add-data-source.png)

1. Add the Prometheus Cluster IP in URL.
2. Click Save & Test

**Grafana Dashboard**

1. Click on **Create** > **New Panel**
2. In metrics , add **request\_count .**
3. set Panel name , legend , choose color palette , remove unwanted endpoints from transformation option.
4. Repeat the same by choosing **request\_latency\_seconds\_sum** as the metric.
5. Apply and save to view the metrics in Grafana dashboard.

Request counts dashboard:



Requests latency dashboard:

