

**Configuration of .Net Core Api with   
Grafana and Prometheus**

**Version History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SI | Date | Ver | Description | Changed by | Approved/Reviewed by |
| 01 | 07-March-21 | 1.0 | Configuration of netCore Api with Grafana and Prometheus | Maidul Islam  Software Engineer | Md. Mahedee Hasan  Head of Software Development |

Contents

[Introduction 2](#_Toc97567236)

[Features Offered by Grafana 2](#_Toc97567237)

[Pros and Cons of Grafana 3](#_Toc97567238)

[Configuration Requirements 3](#_Toc97567239)

[Configure Prometheus 4](#_Toc97567240)

[Configure Grafana 6](#_Toc97567241)

[Custom Dashboard Configuration 11](#_Toc97567242)

[References 15](#_Toc97567243)

# Introduction

Grafana is a general purpose dashboard and graph composer. It's focused on providing rich ways to visualize time series metrics, mainly though graphs but supports other ways to visualize data through a pluggable panel architecture. It currently has rich support for for Graphite, InfluxDB and OpenTSDB. But supports other data sources via plugins.

With growing reliance on information technology and integrated systems, the need for data analysis across enterprise IT environments is critical for troubleshooting, auditing, security, reporting, and operational insights. Visualization tools collect massive amounts of data from various logs and systems to provide analytics to staff who can make informed decisions based on a tool’s output. Without these tools, administrators and other IT staff face frustrations and difficulties proactively maintaining network appliances, that can lead to unforeseen downtime. Visualization tools and analytics help administrators remediate issues prior to them causing critical crashes.

# Features Offered by Grafana

This open-source framework takes care of all the analytics of our app. We can easily query, visualize, set up alerts, understand the data with the help of metrics.

The dashboard is pretty equipped, & continually evolving, to make sense of complex data. From displaying graphs to heatmaps, histograms, Geo maps. The tool has a plethora of visualization options to understand data as per our business requirements.

Alerts are set up & triggered like trip wires whenever an anticipated scenario occurs. These happenings can be notified on Slack or whatever communication platform the monitoring team uses.

Grafana has native support for approx. a dozen databases. And with many more, facilitated by respective plugins.

Either host it on-prem or any cloud platform of your choice.

It has built-in support for Graphite & expressions like add, filter, avg, min, max functions etc. to custom fetch data.

# Pros and Cons of Grafana

Pros:

* Broad compatibility with multiple data sources
* Support for Elasticsearch and Prometheus
* Report options are available without many customizations or configurations
* An active developer community, so new features are added regularly

Cons:

* Support for a wide range of data sources adds complexity to the setup
* Limitations on selection of reporting types

# Configuration Requirements

I have already build an application using .Net Core Web API and Grafana with Prometheus. Please download the application from NextGen Repository POC folder.

I have used following libraries to configure metrics.

Installation Pacakages:

* App.Metrics.AspNetCore
* App.Metrics.AspNetCore.Endpoints
* App.Metrics.AspNetCore.Tracking
* App.Metrics.Formatters.Prometheus

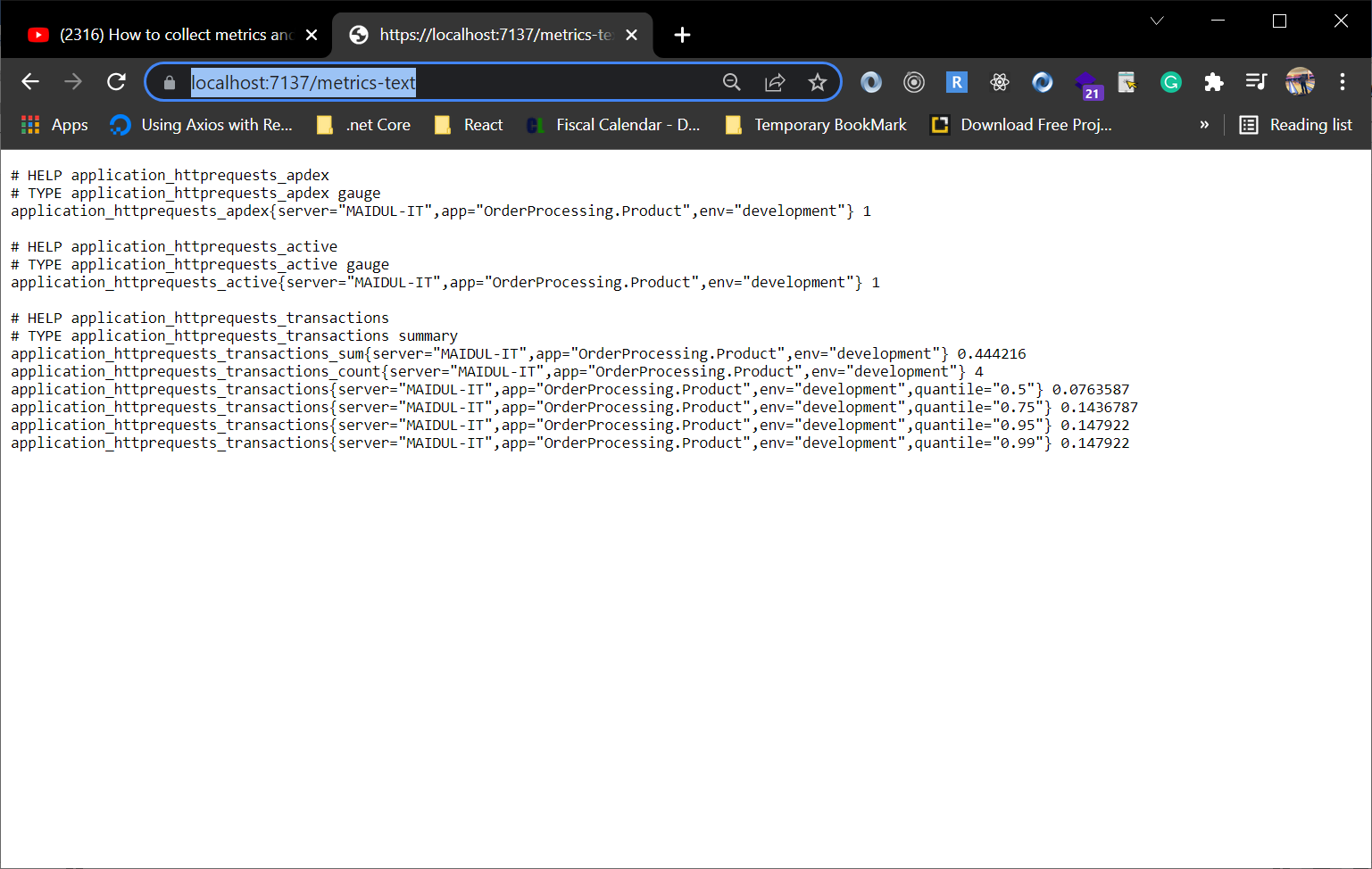
Prometheus support for different types of metrics.

1. Gauge
2. Counter
3. Summary

Open the application and run, now I am going through the summary configuration of .net core with Grafana and Prometheus.

As the application is already configured, run the application and go to below link to check the metrics report in text. You can check either on browser or postman.

https://{url} /metrics-text



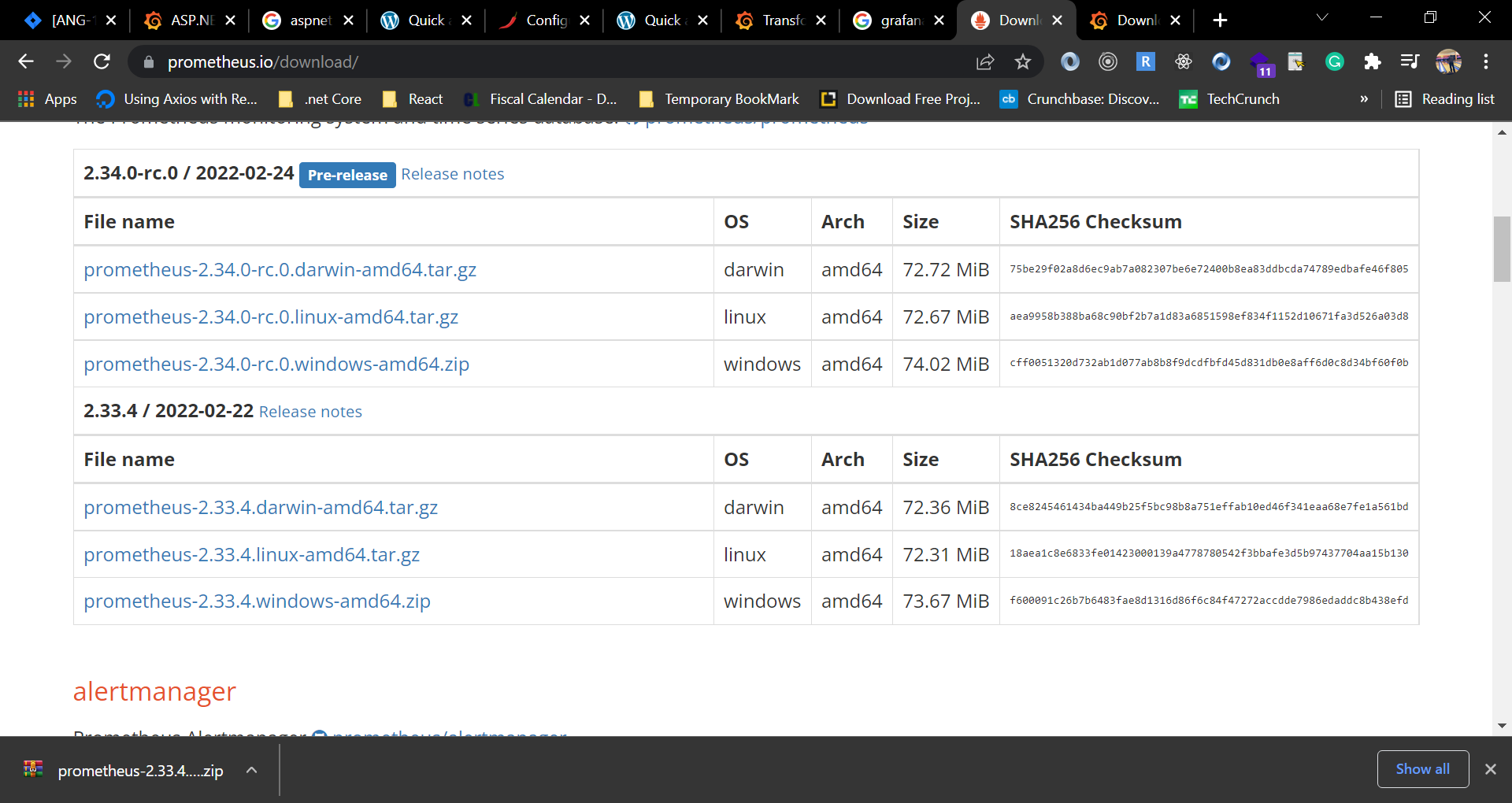
Now we are going to configure to display all metrics data on Prometheus Dashboard.

# Configure Prometheus

1. Go to below link to download Prometheus.

<https://prometheus.io/download/>

Download the stable version.



1. Extract the zip file.
2. Open Prometheus.yml file in vs code or any other editor.
3. Now we are going to add some additional configuration at the bottom of the file. Please copy and paste below text.

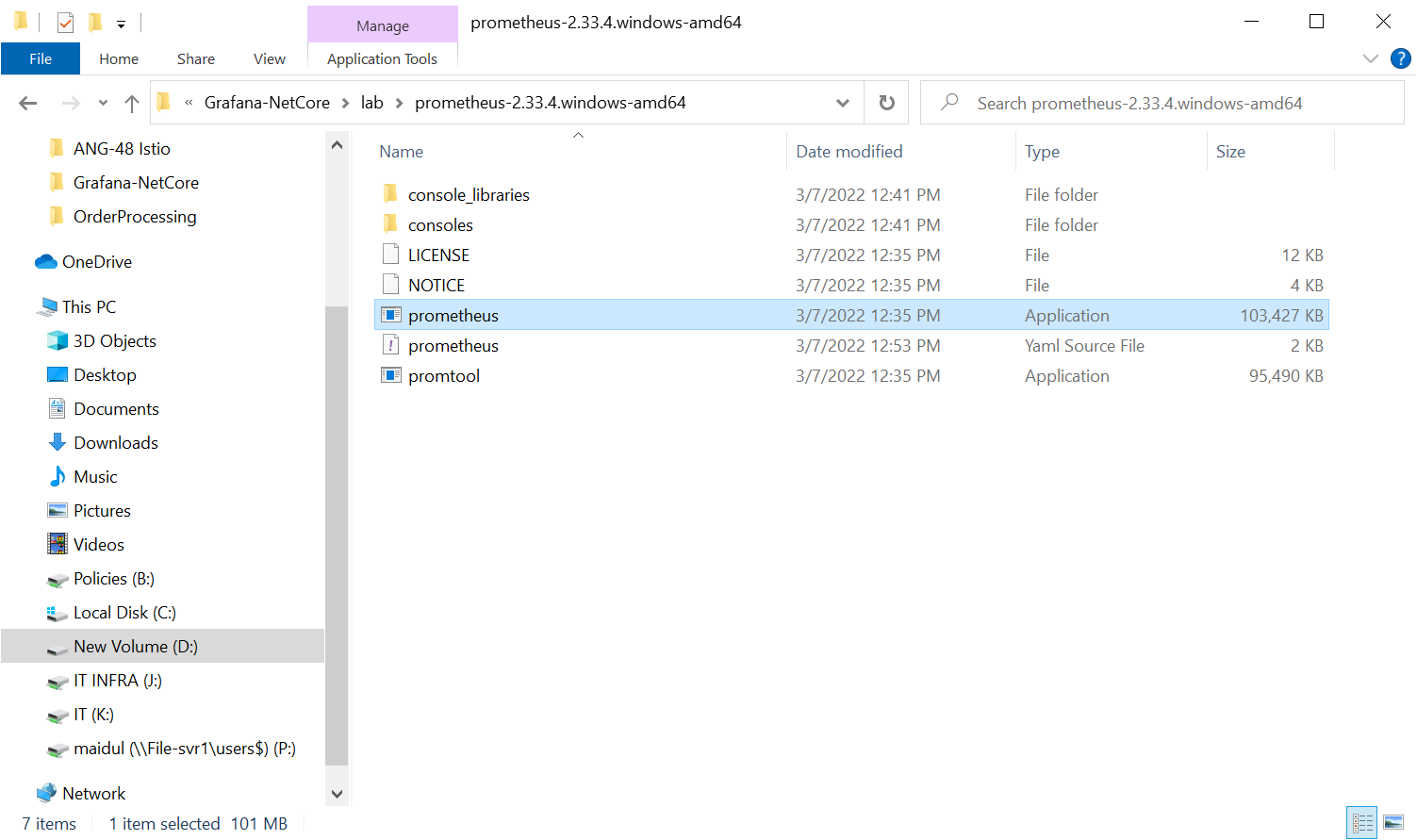
  - job\_name: "product-api"

    static\_configs:

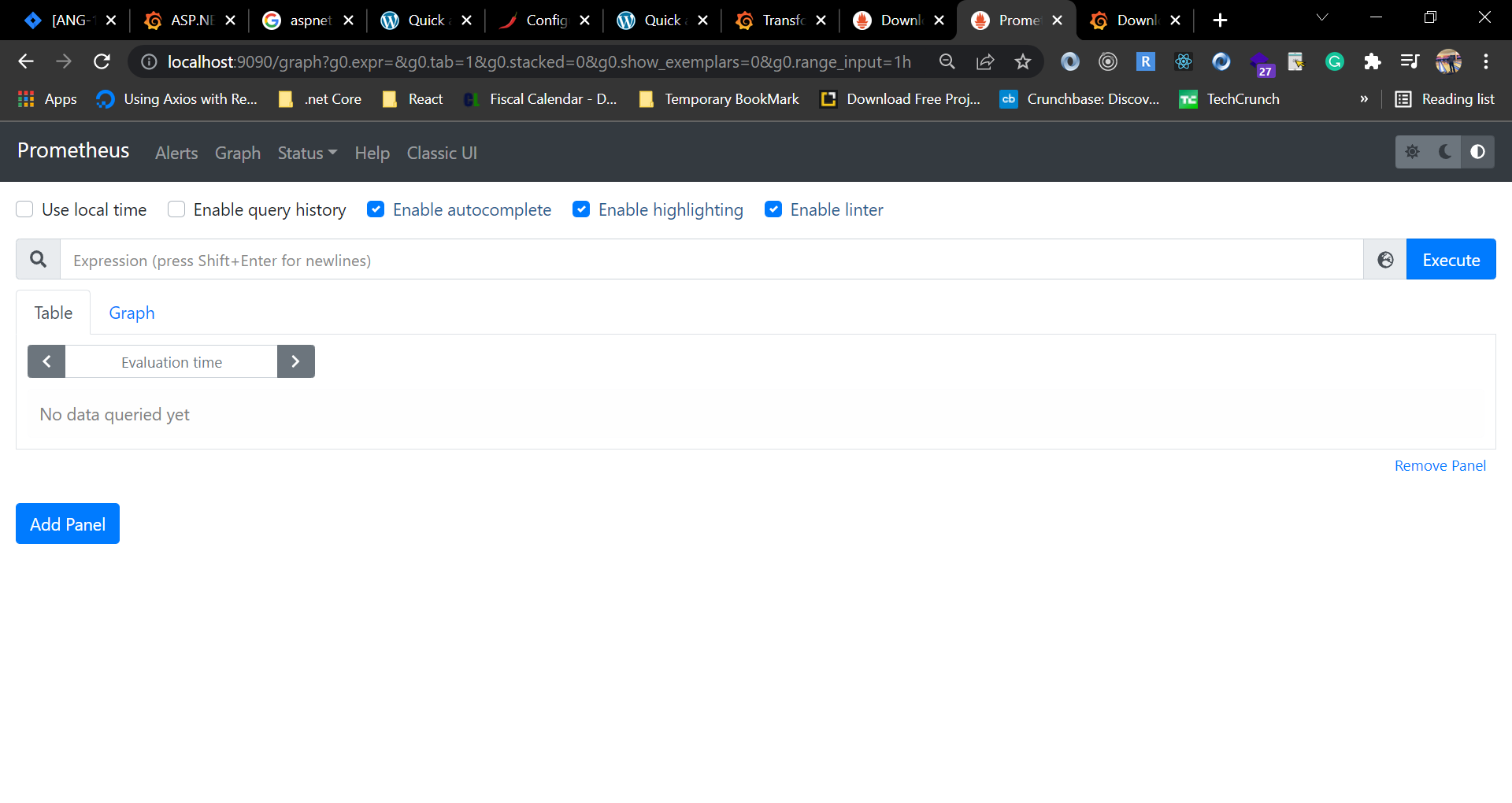
      - targets: ["localhost:30644"]

    metrics\_path: /metrics-text

1. Save and close the file.
2. Now open **Prometheus.exe** file and keep the console run.



1. Now go to browser and visit <http://localhost:9090/>



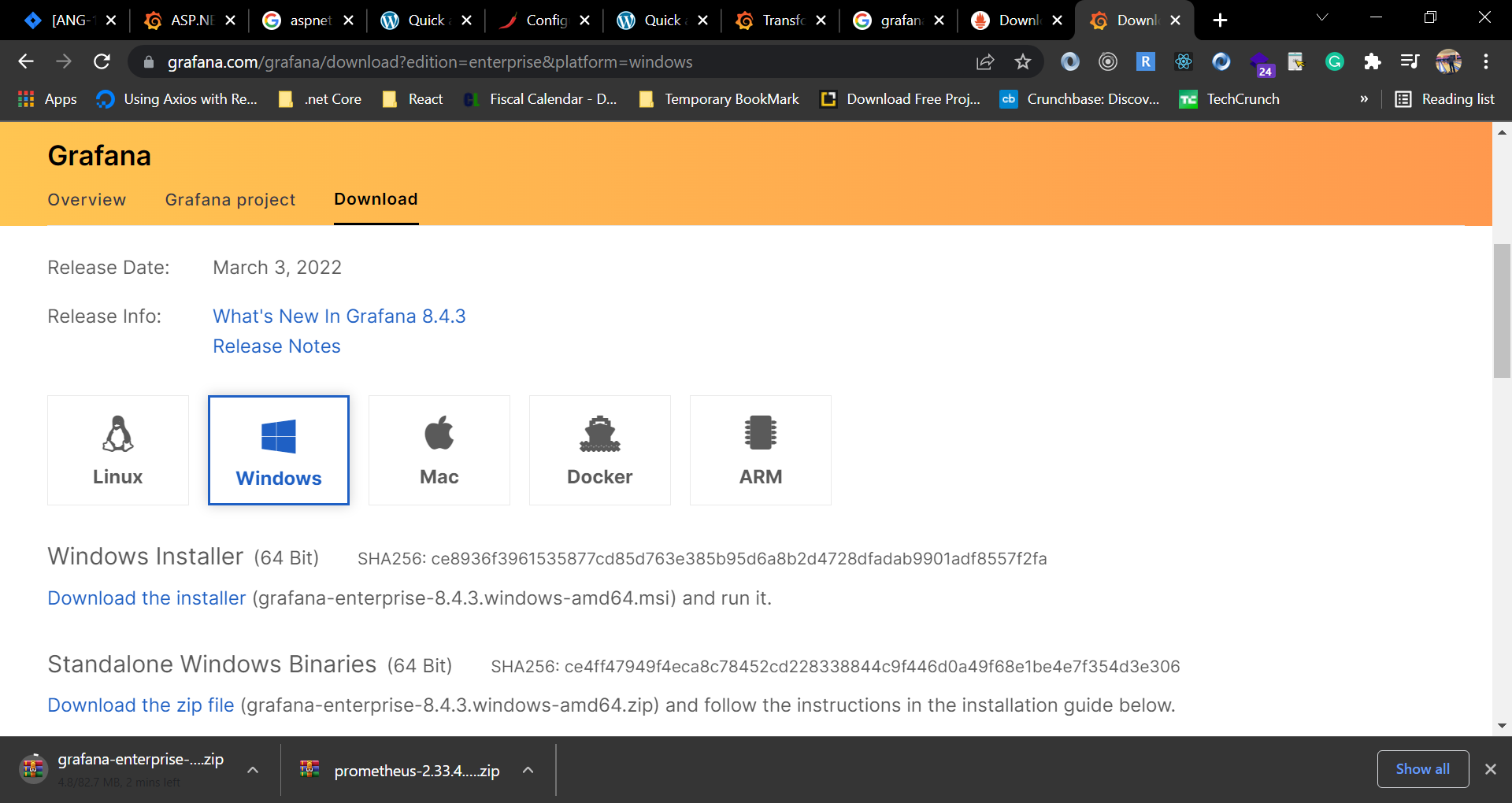
As you can see that Prometheus is configured, you can try with different query to verify or move to next step.

# Configure Grafana

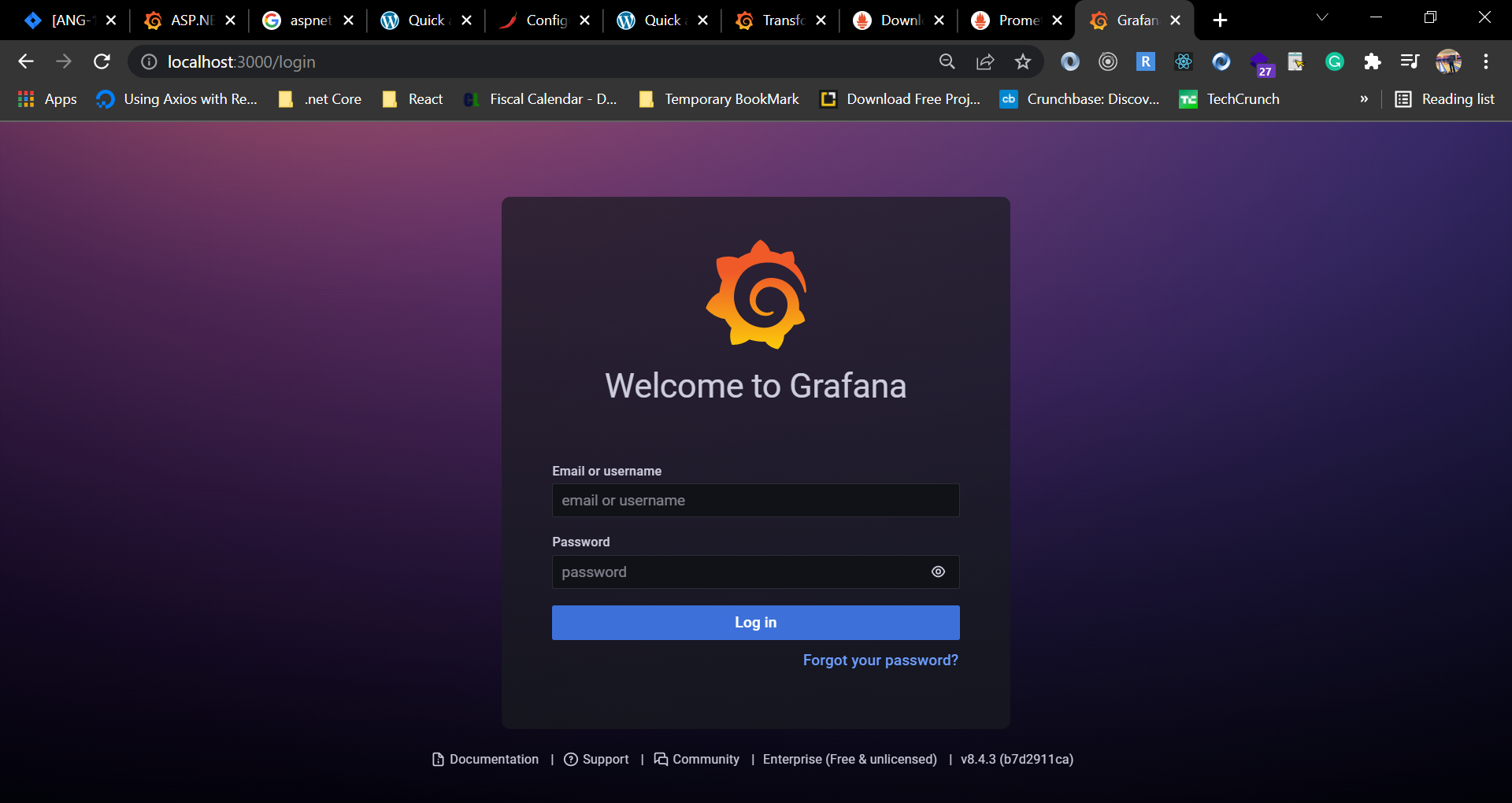
1. Go to below link to download Prometheus.

<https://grafana.com/grafana/download>

Download the stable and correct version to match with your OS.



1. Extract the zip file.
2. Go to Grafana > bin directory.
3. Now execute the **grafana-server.exe.**
4. Keep the console open and visit <http://localhost:3000/> (it might take some time to load initially)



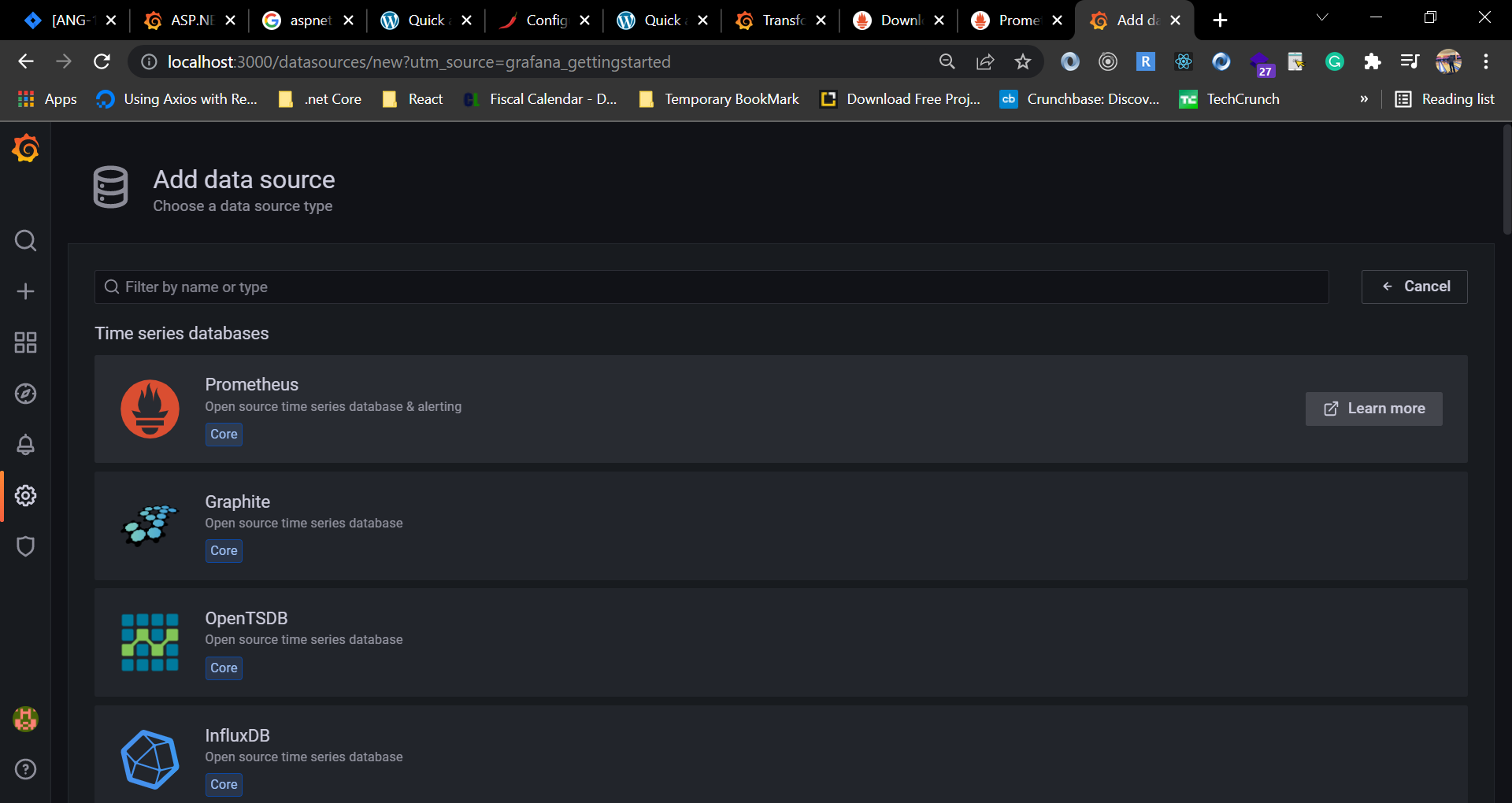
1. Provide login credential, **user: admin, password: admin**

You can either change the password or skip to keep the default user credential.

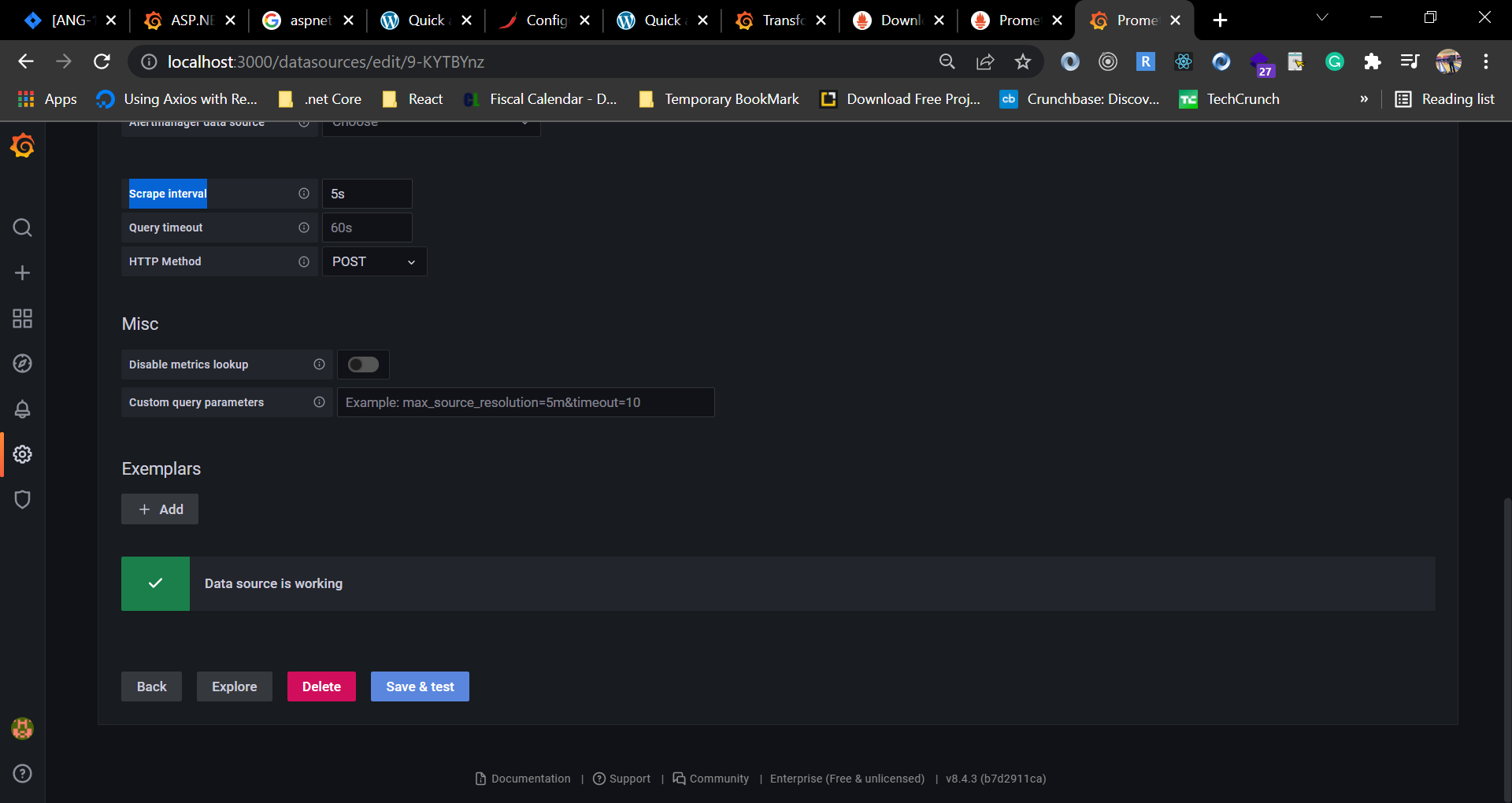
1. You are going to see a dashboard like below.



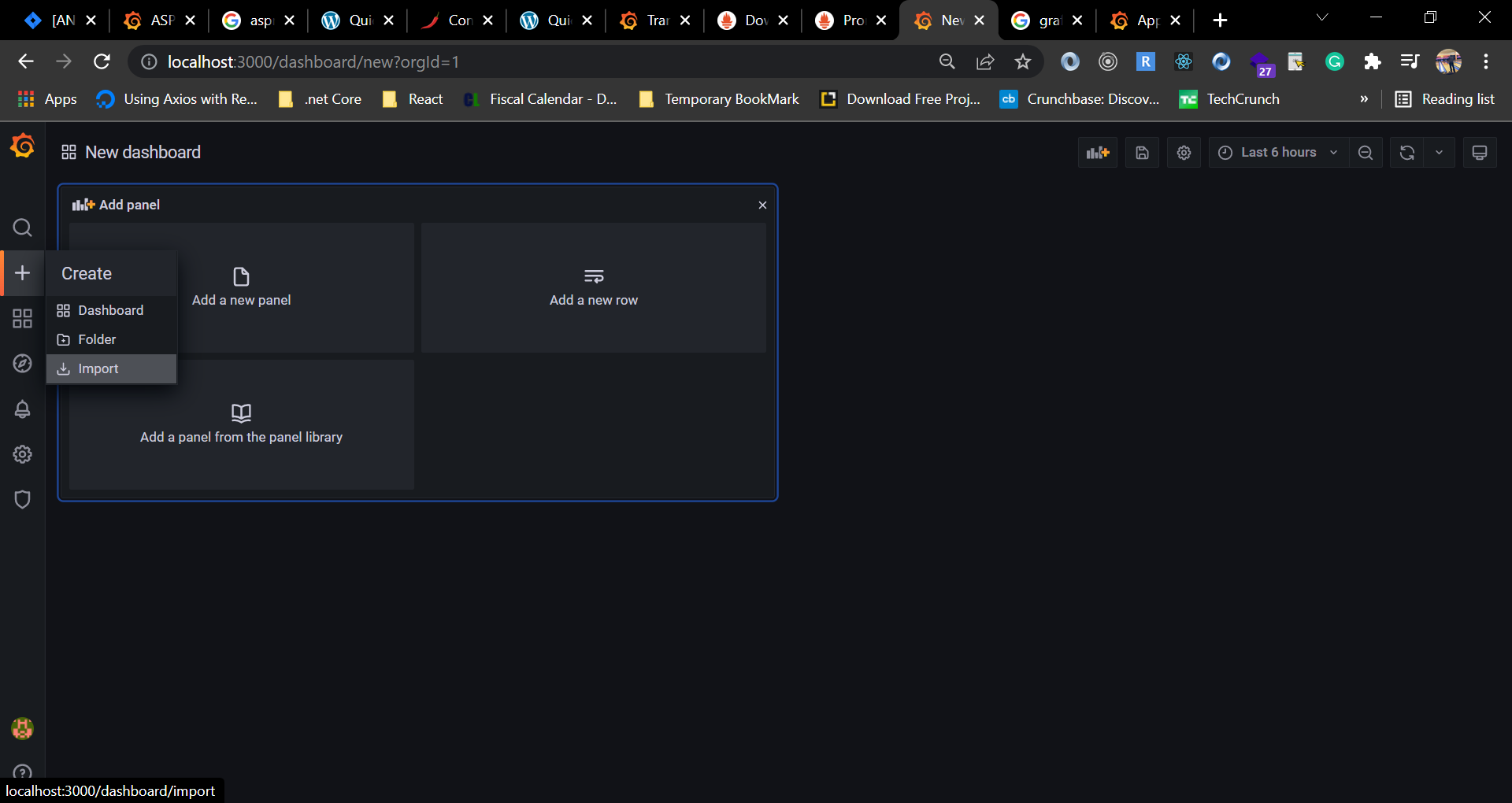
1. Now we need to add data source. Click on Data Source to add first data source.
2. Select Prometheus.

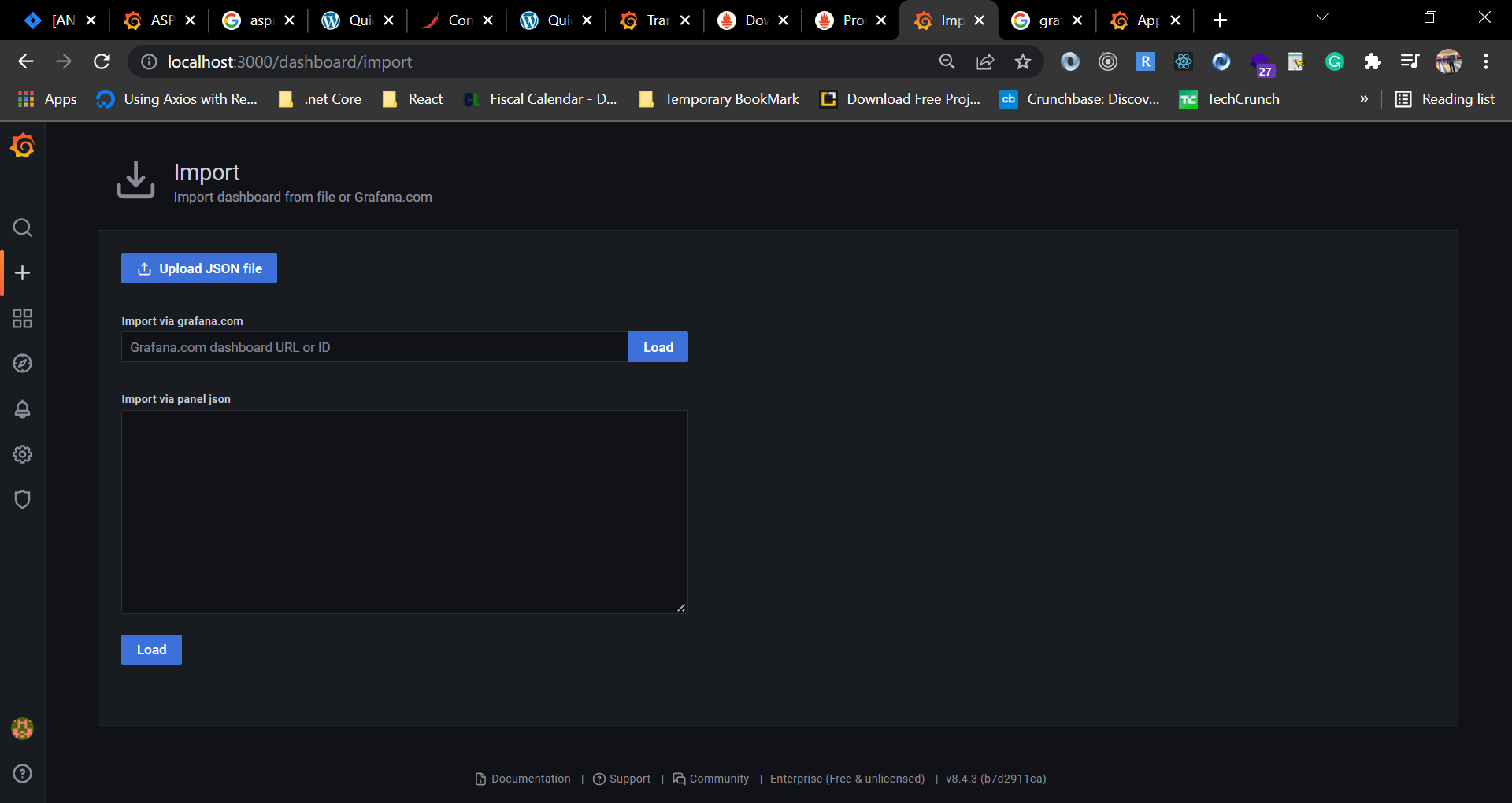


1. At the URL filed write [**http://localhost:9090**](http://localhost:9090)and Scrape interval **5s** keep every things else default.
2. Now click Save & test button at the bottom.



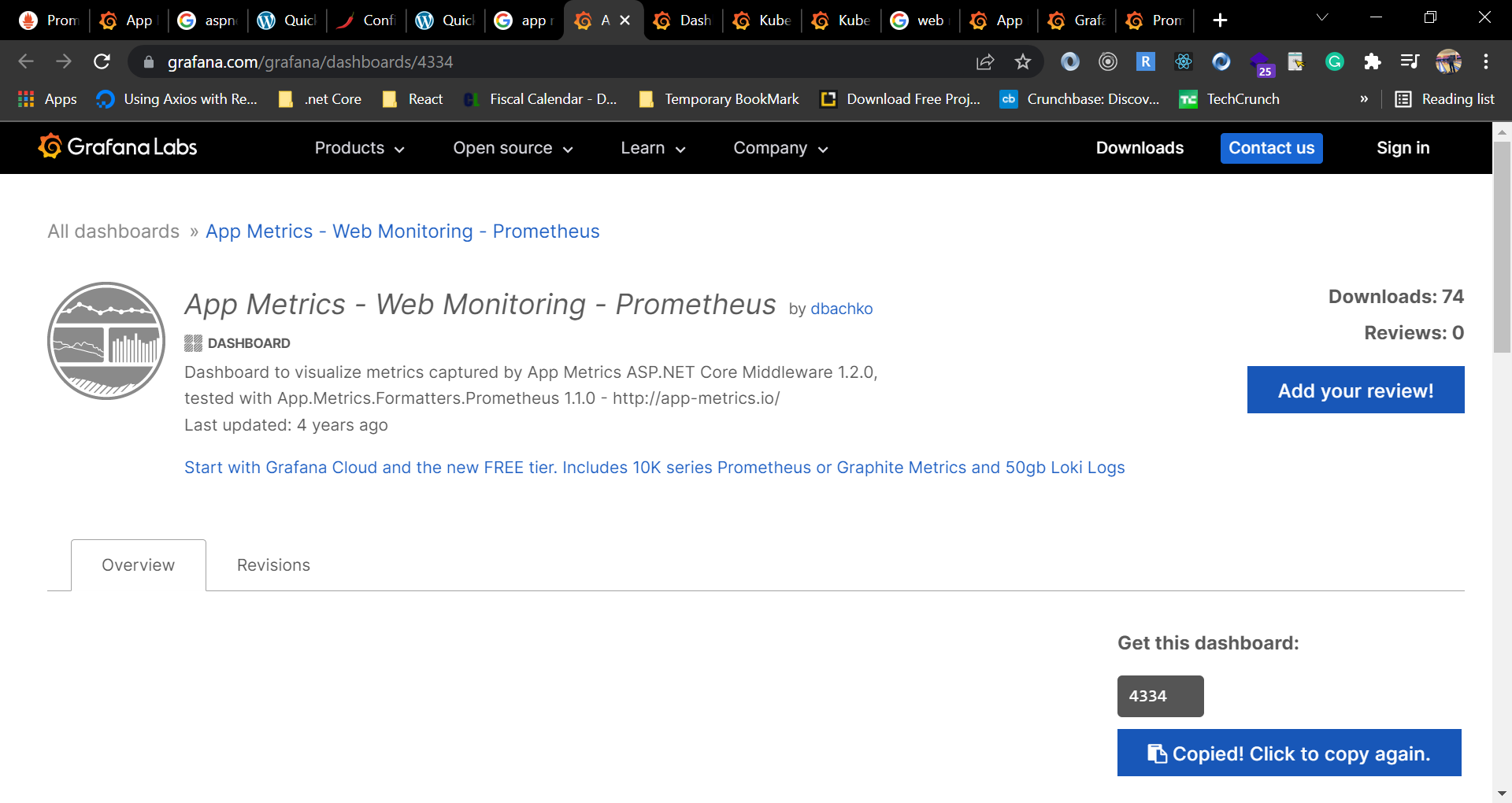
1. A message will show if data source works.
2. Now go back to home page and we are going to configure dashboard.
3. At the left nav click on plue(+) button and click on Import.



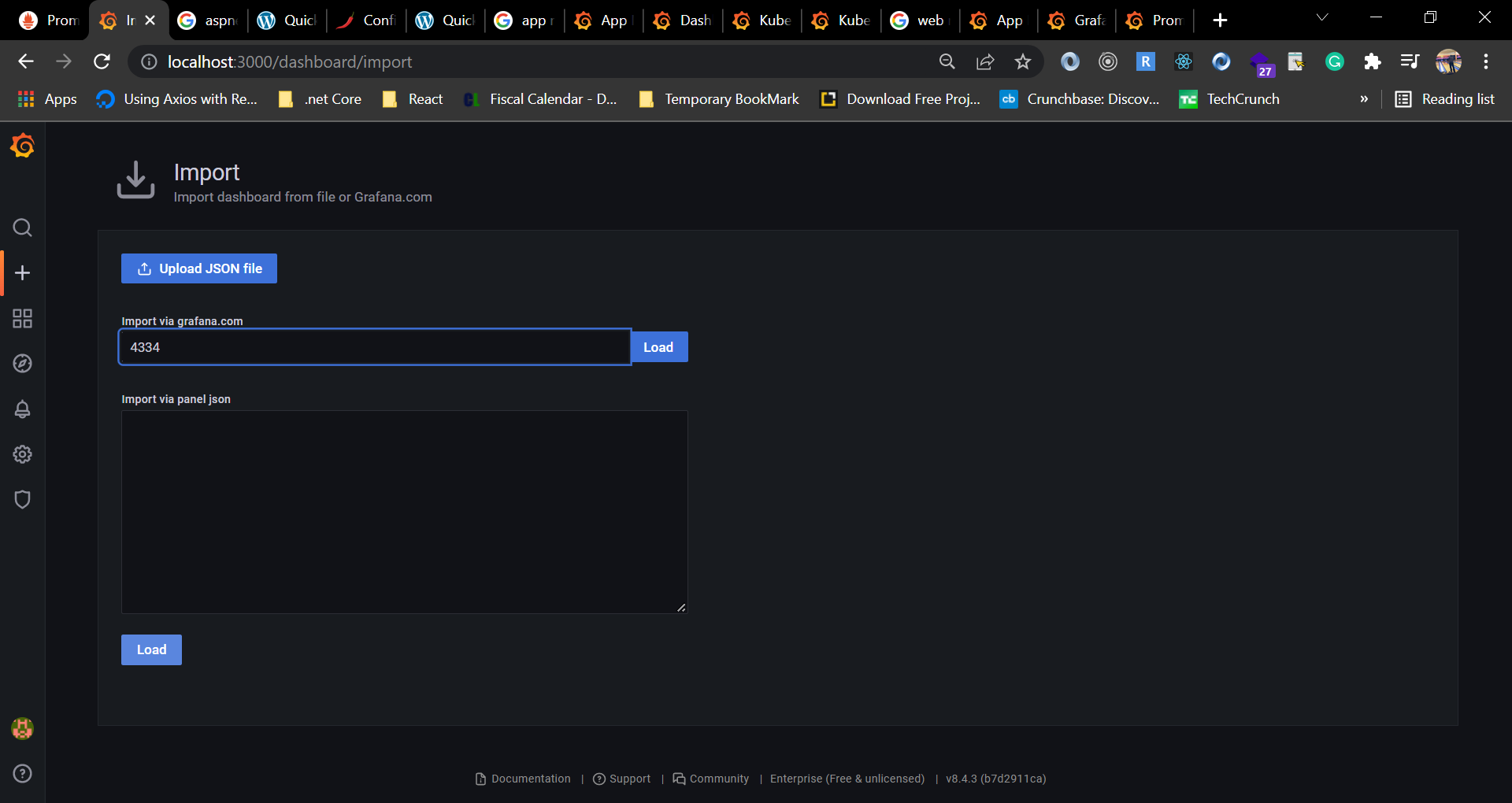


We need to specify here id or URL from Grafana.com.

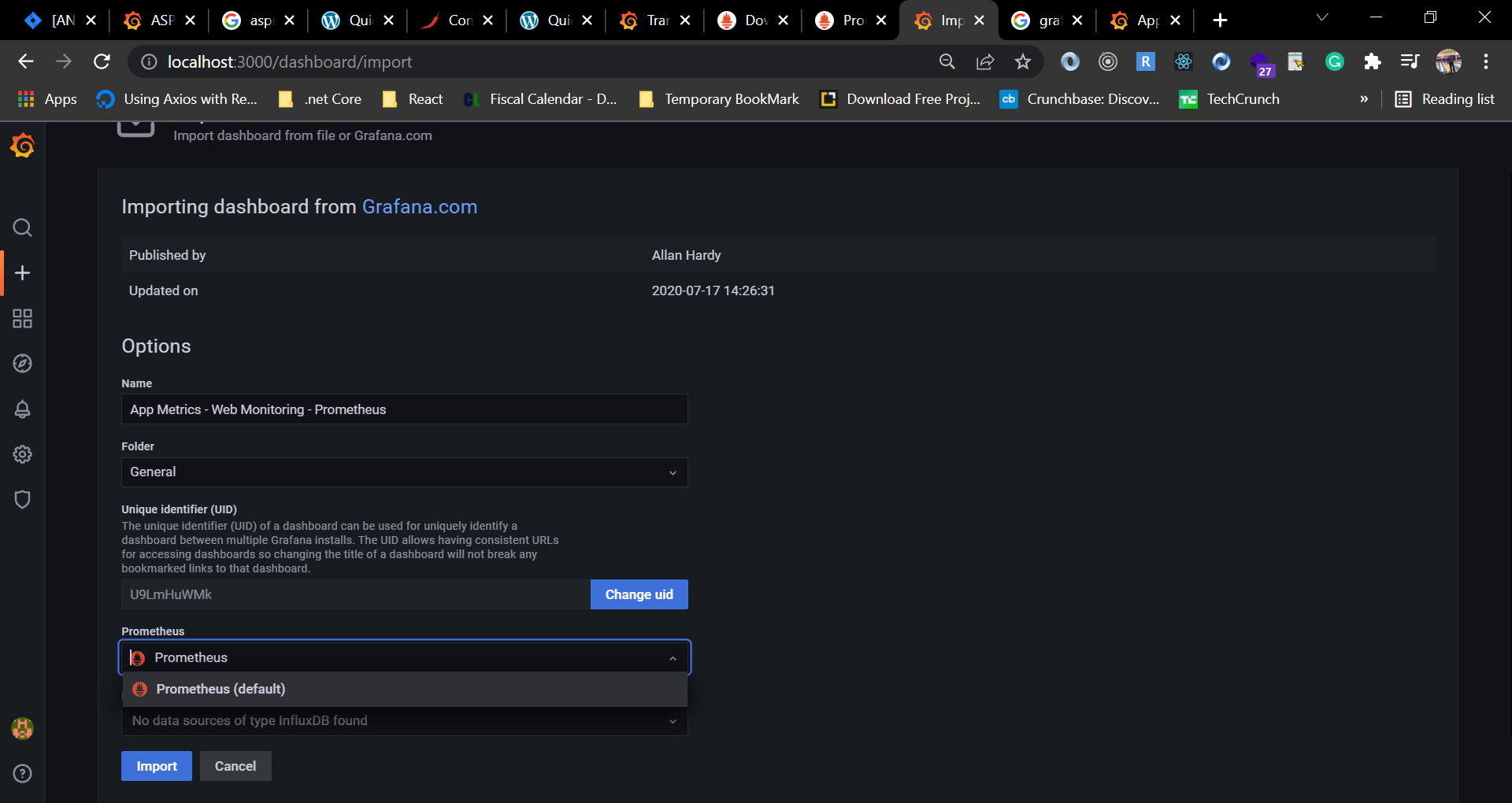
1. On your browser visit <https://grafana.com/grafana/dashboards/4334>



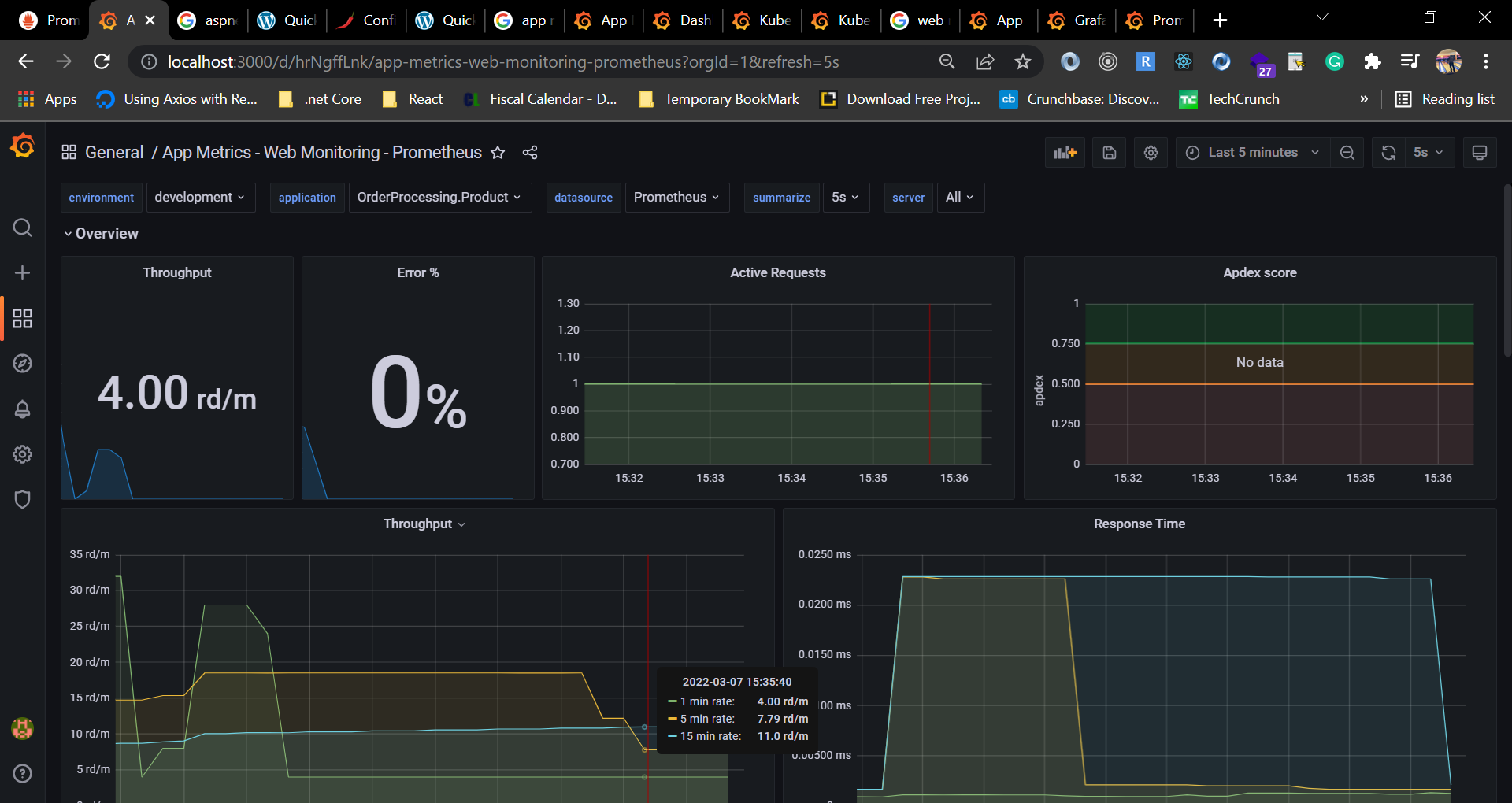
1. Now copy the id under the section of “Get this dashboard”.
2. Go back to grafana dashboard and paste the id and click on load button.



1. A screen like below will appear and select Prometheus at the section.



1. Select Prometheus as data source and click on Import button. A dashboard like below will be shown.



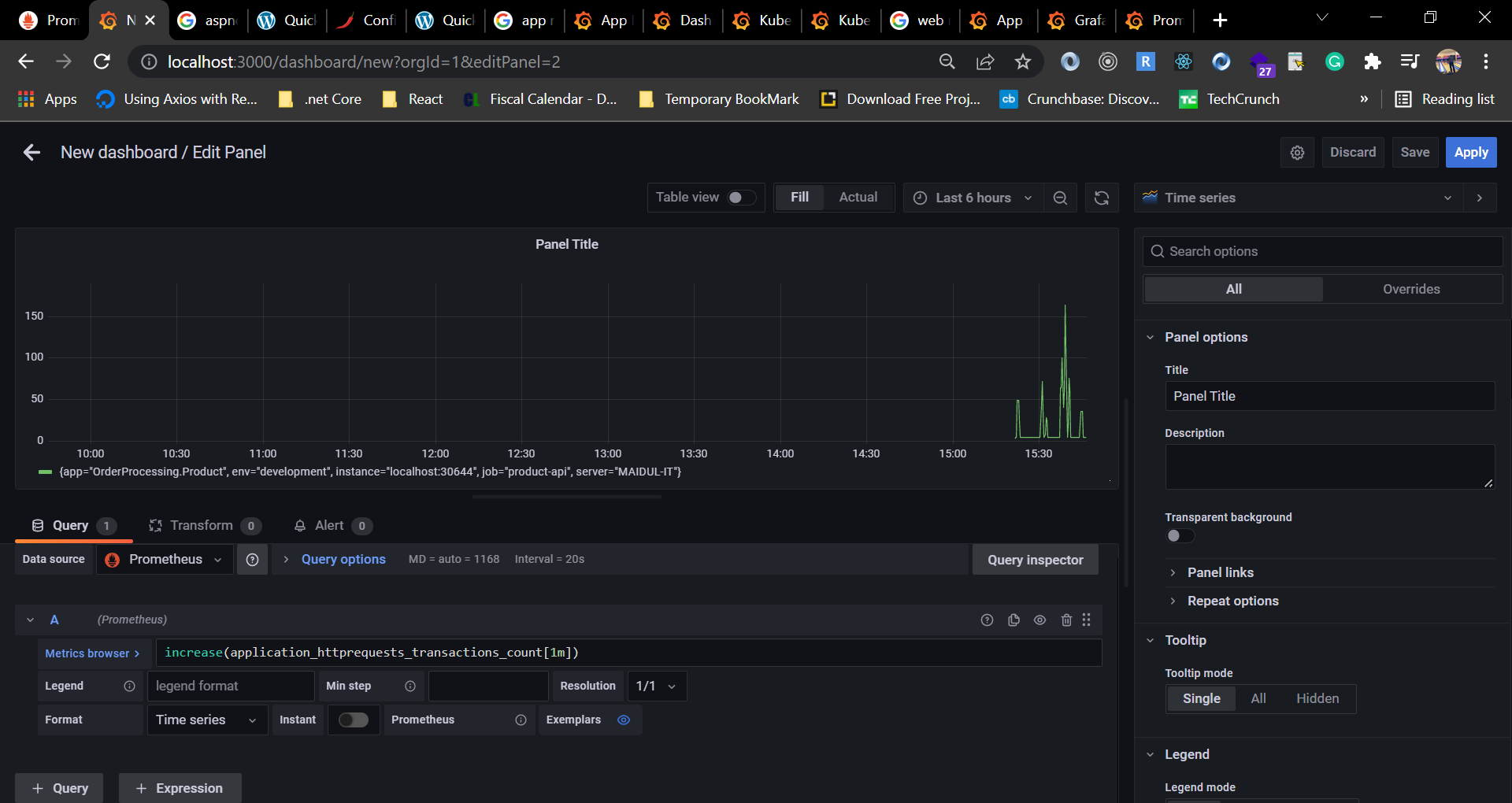
1. We can browse our application and see the differences on the metrics dashboard.

# Custom Dashboard Configuration

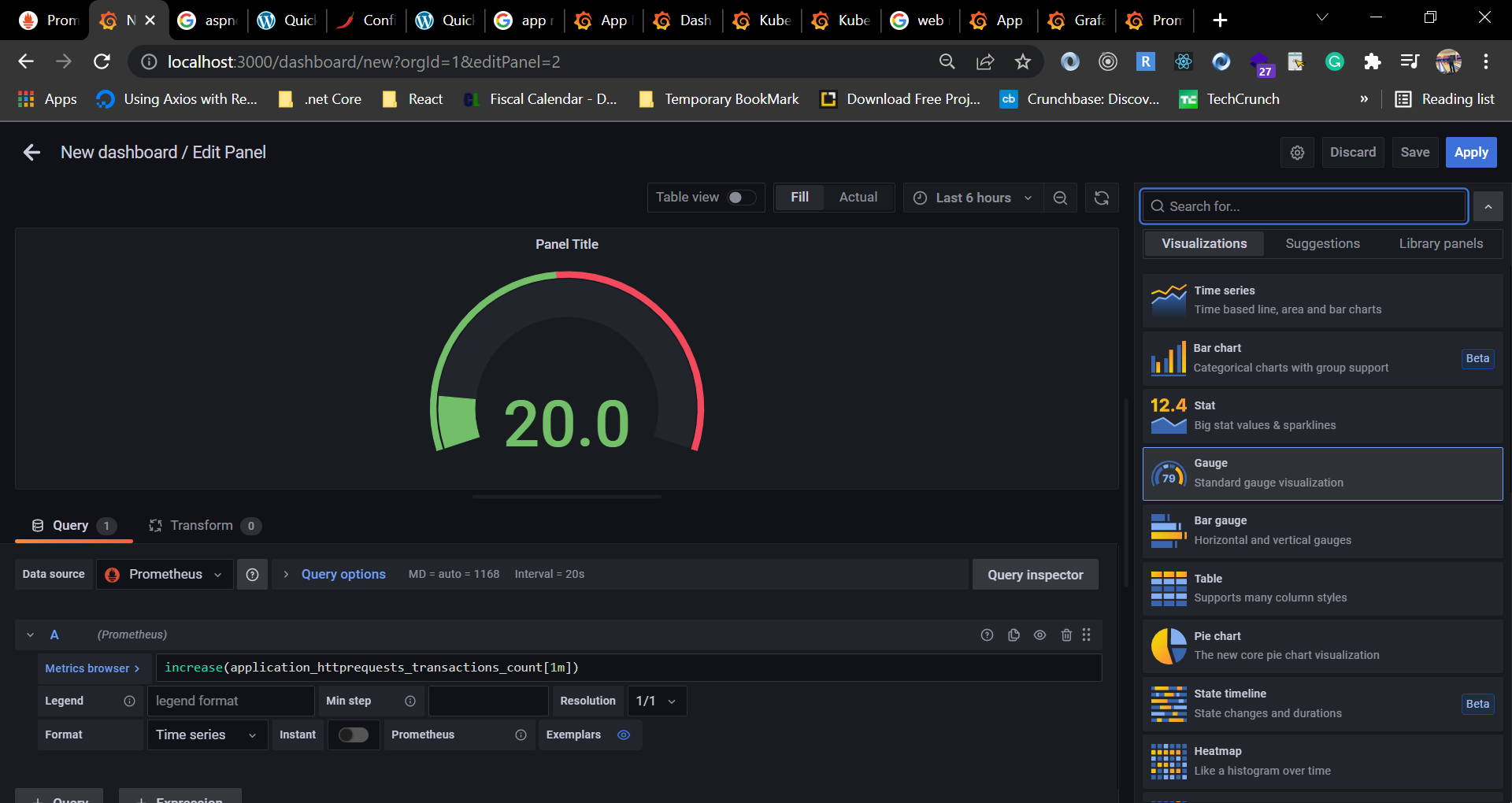
Now we are going to create our own dashboard.

1. Click on plus(+) from side nav and click on create.
2. Select “Add a new panel”.
3. In the field of Metrics Browser write below command and check the graph.

increase(application\_httprequests\_transactions\_count[1m])



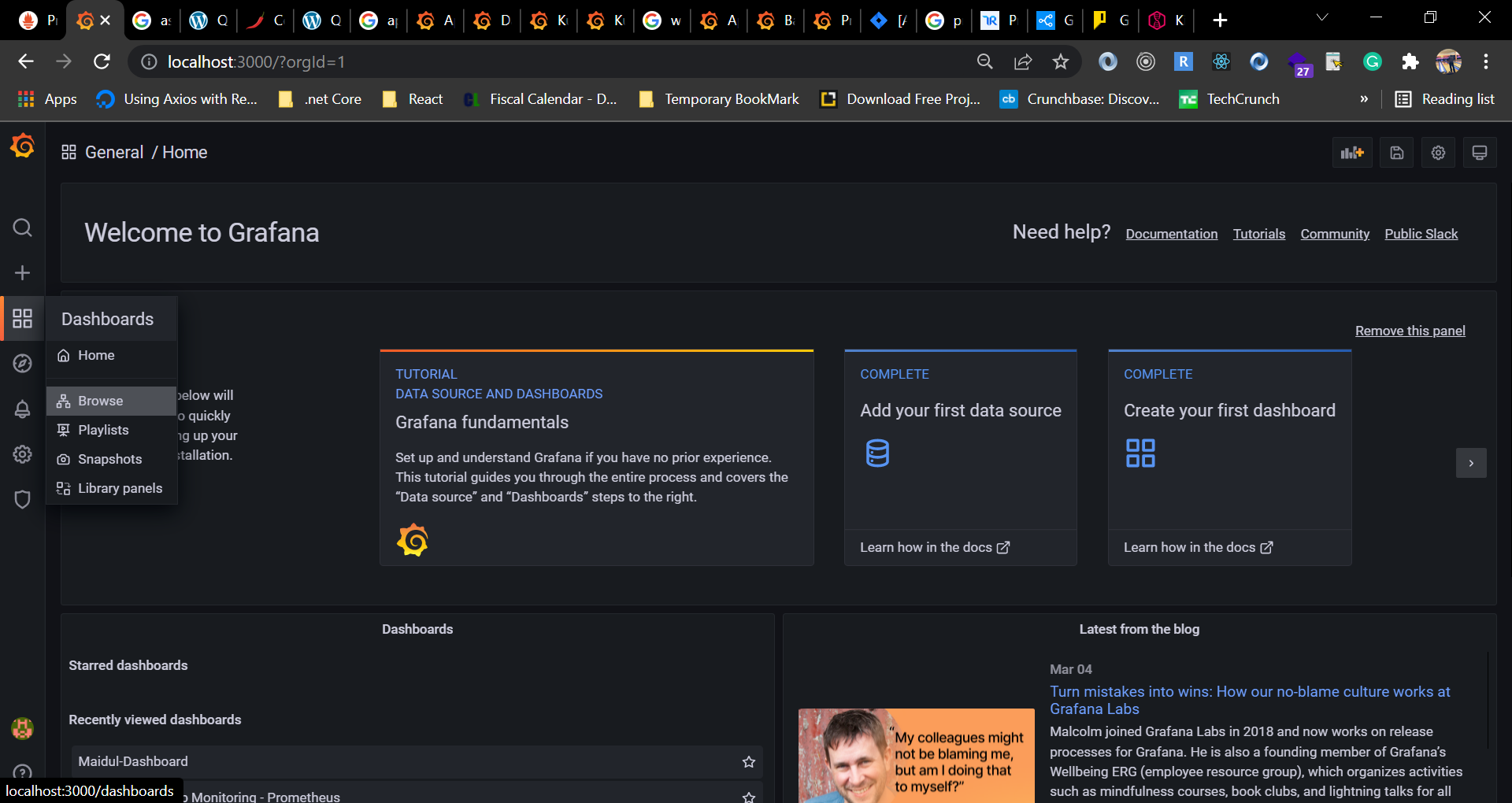
1. We can also choose visualization from side nav.



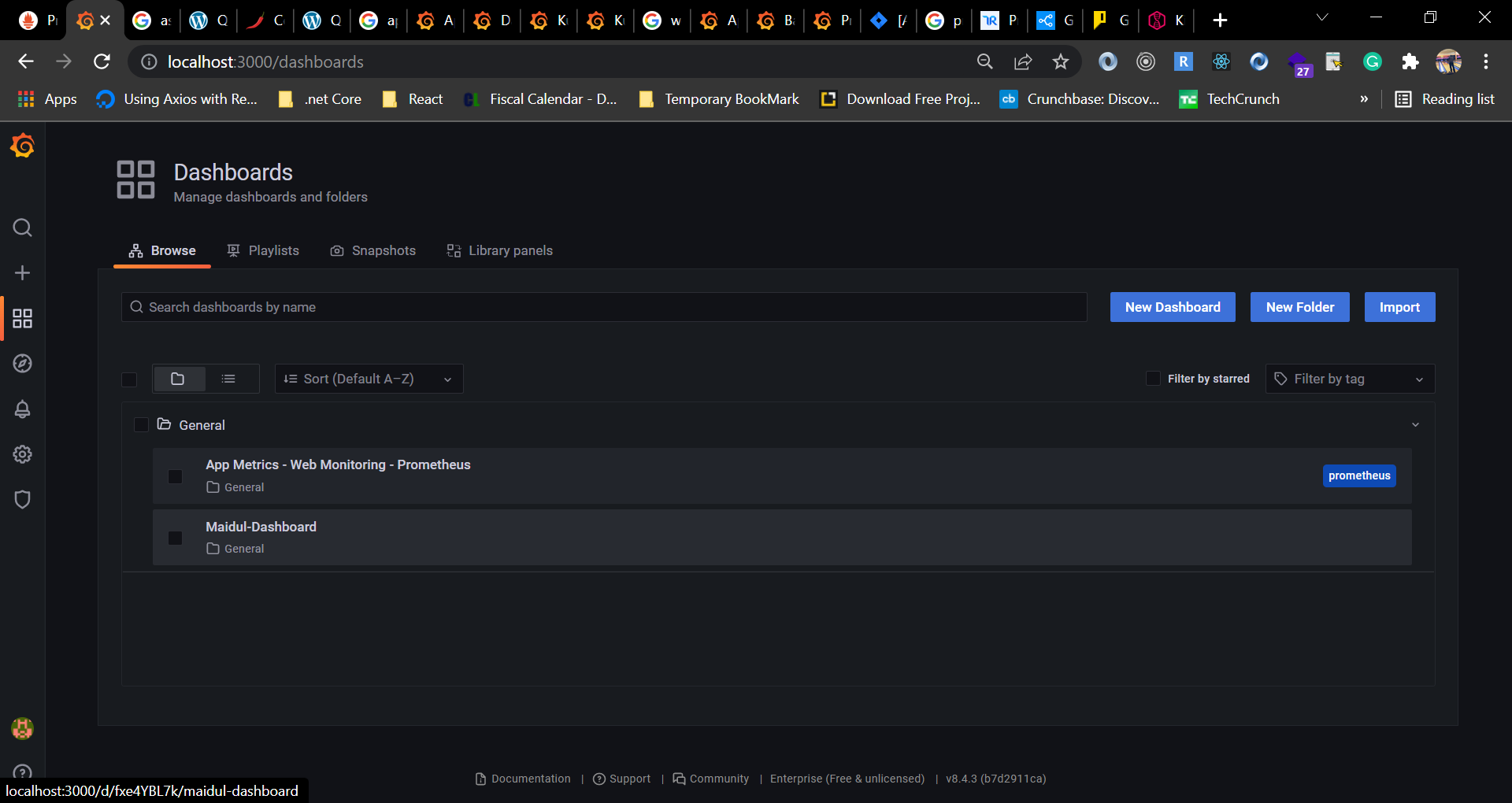
1. Change the name and save. It will be redirected to custom created dashboard or you can go to your dashboard from browser.
2. Let’s create another panel to check total error of last 10 miniutes.
3. At the field of metrics browser write below command.

increase(application\_httprequests\_error\_rate\_total[10m])

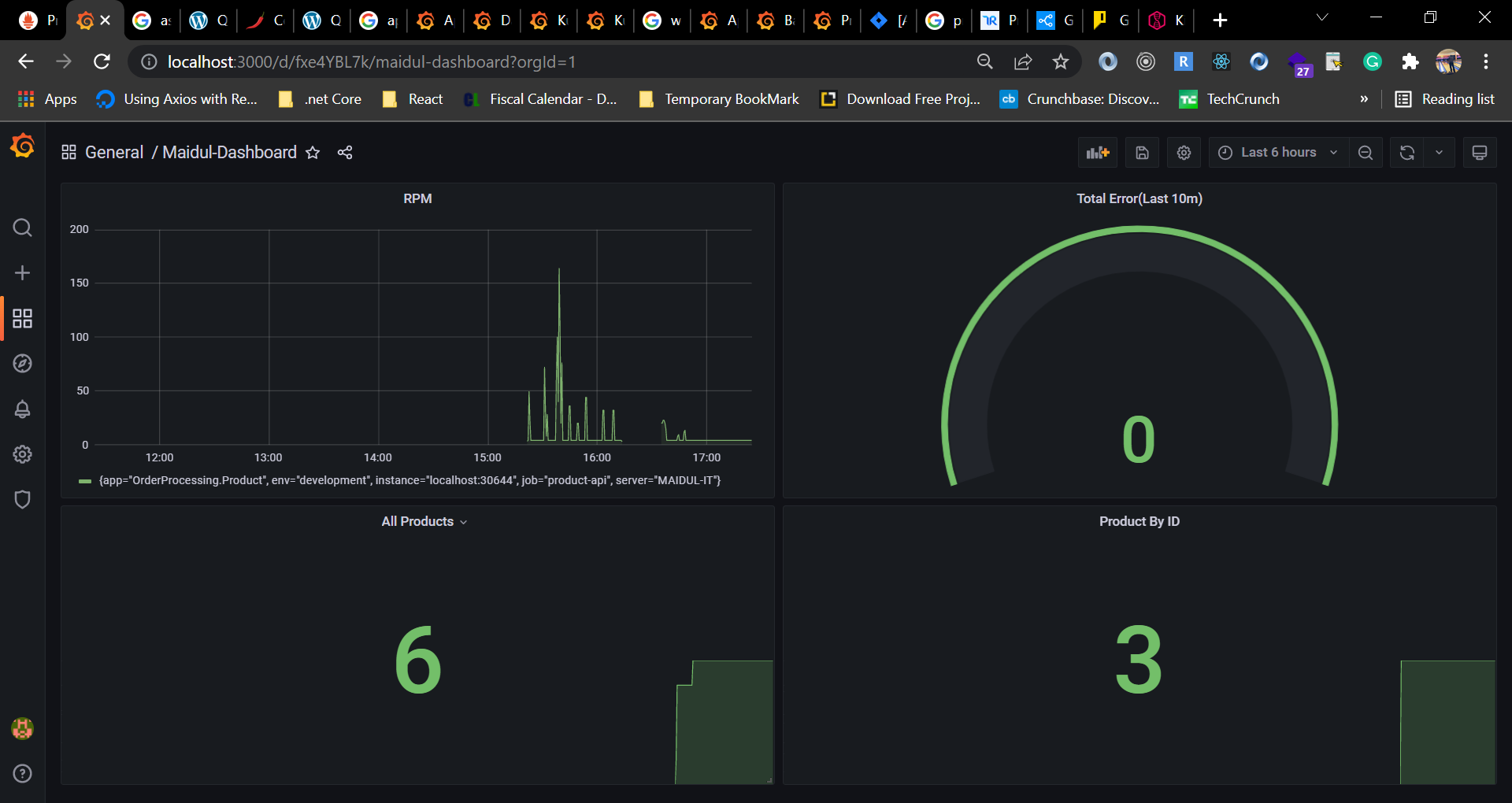
1. Select a graph and save. Now go back to your created dashboard and check.
2. Go to Dashboard > Browse.



1. Select your created dashboard, as I have save my dashboard name **Maidul-Dashboard.**



1. You will see an output like below.



In our .net application I have configured to verify total number of API request to get all products and get all products by Id. Following instruction to visualize the configuration.

1. At the top of your custom dashboard, there is an icon to add new panel, click there and select “Add a new panel”.
2. In the Metrics Browser field write below command.

productapi\_get\_all\_product

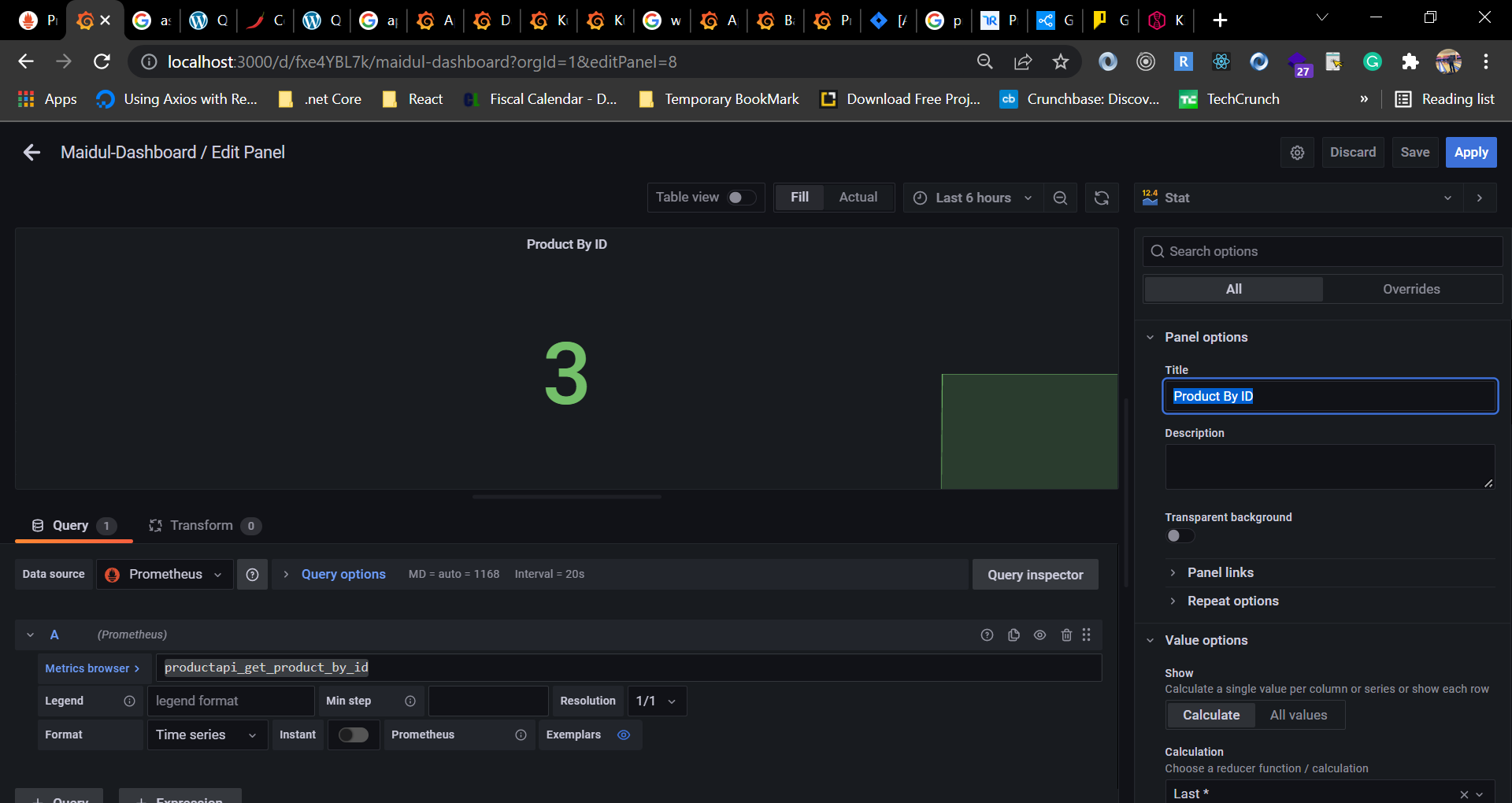
1. Select the Stat graph.



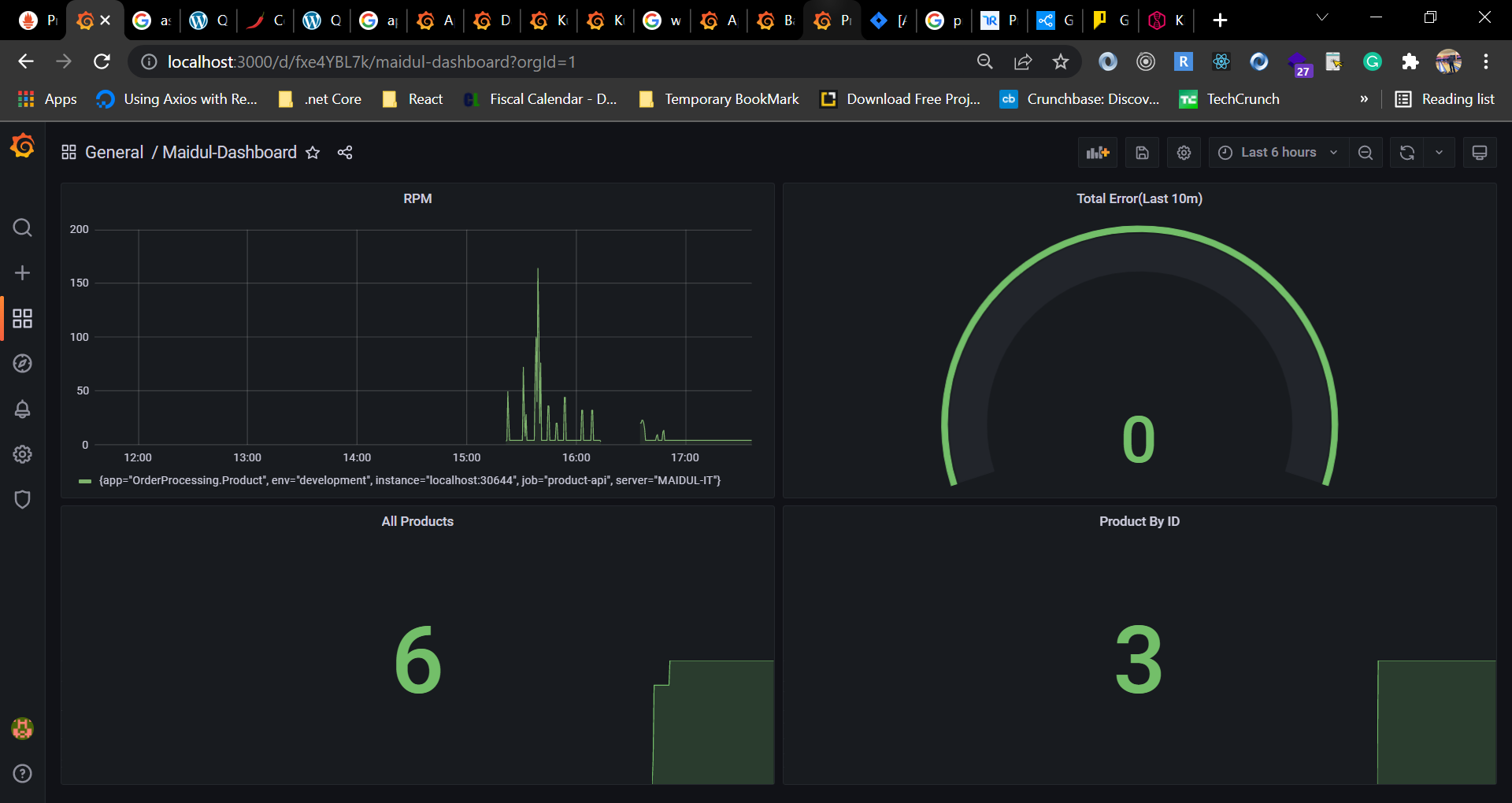
1. Select last 6 hours.
2. Change the title to **All Products**.
3. Apply > Save > Save
4. Now click again the add new panel icon from your custom dashboard and select “Add a new panel”.
5. In the Metrics Browser field write below command.

productapi\_get\_product\_by\_id

1. Select the Stat graph.
2. Select last 6 hours.
3. Change the title to **Product By ID**.
4. Apply > Save > Save.



1. Go back to your custom created dashboard and final output should look like below.



We can add more panel as many as needed.

# References

<https://www.logdna.com/learn-observability/kibana-vs-grafana-vs-prometheus-vs-logdna>

<https://play.grafana.org/d/000000012/grafana-play-home?orgId=1>

<https://grafana.com/grafana/dashboards/4334>

<https://devlinduldulao.pro/quick-and-dirty-asp-net-core-3-1-prometheus-and-grafana-demo/>

<https://devlinduldulao.pro/quick-and-dirty-asp-net-core-3-1-prometheus-and-grafana-demo/>

<https://devlinduldulao.pro/quick-and-dirty-asp-net-core-3-1-prometheus-and-grafana-demo/>

<https://prometheus.io/download/>

<https://grafana.com/grafana/download>