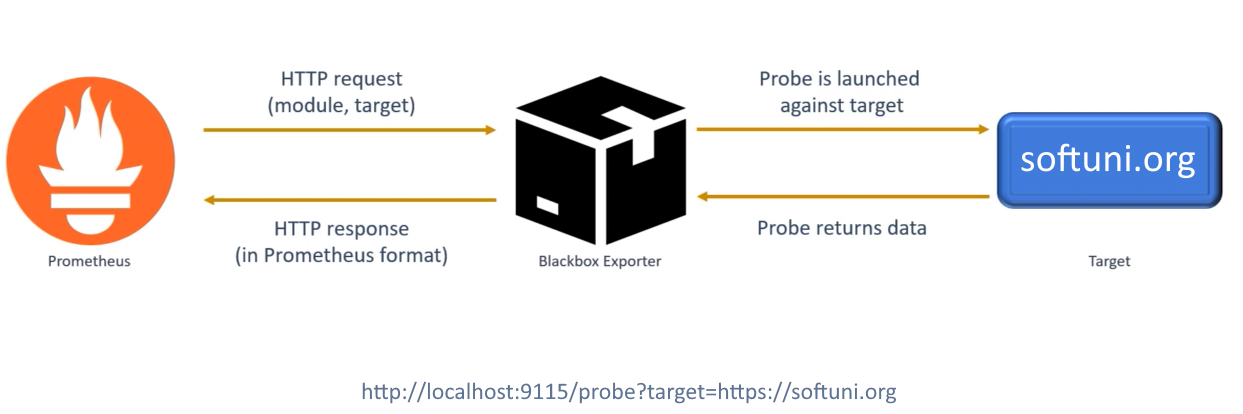
# Lab: App Monitoring

Lab for the ["Containers and Clouds"](https://softuni.bg/trainings/4359/containers-and-cloud-january-2024) course @ SoftUni

## Prometheus and Blackbox Exporter: Run Prometheus Server that Monitors SoftUni.org

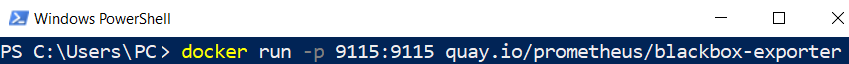
Our task is to **configure Prometheus** to **monitor** [https://softuni.org](https://softuni.org/). In order for this to happen, we shall use the **Prometheus Blackbox Exporter.**

**Prometheus Blackbox E**xporter is designed **to probe various endpoints** and **expose the results** as **metrics** that Prometheus can scrape.

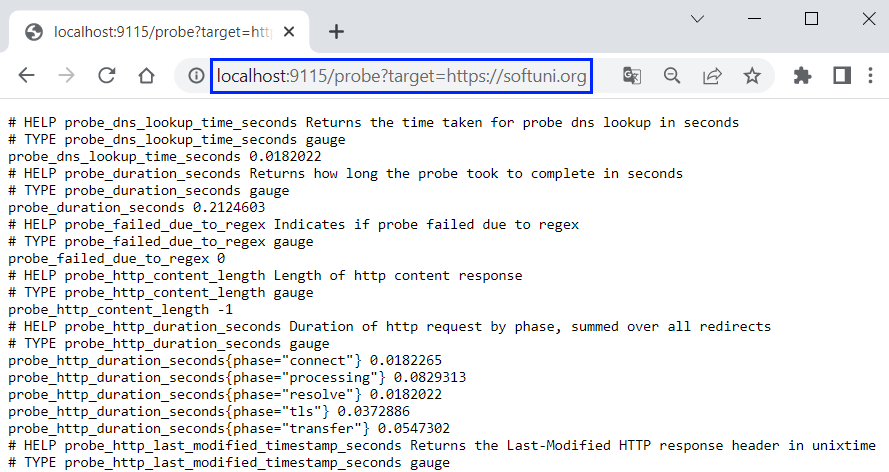


### Step 1: Install and Run Blackbox Exporter

To use the Blackbox Exporter, let's run it in a **Docker container** and **expose its port**:

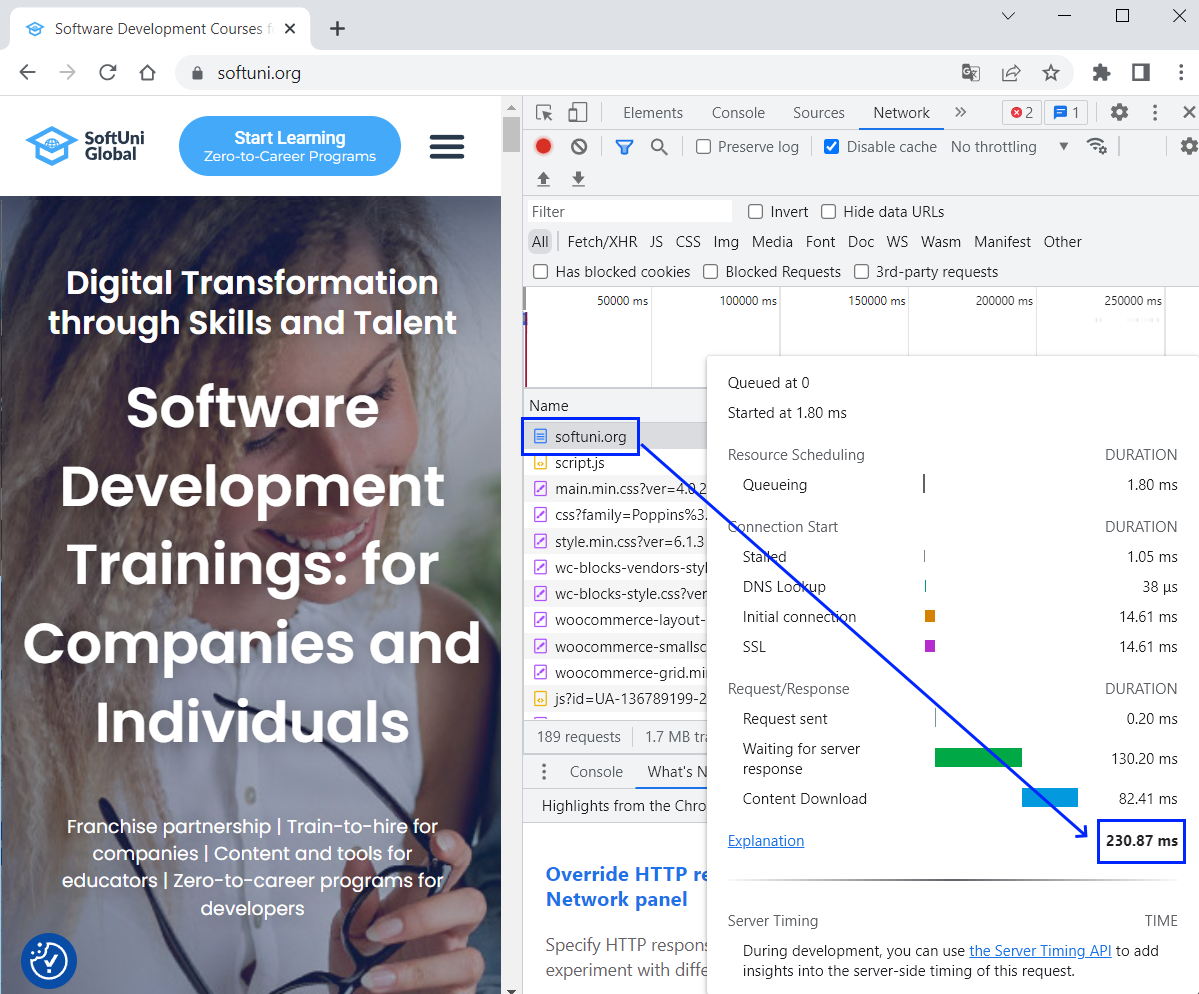


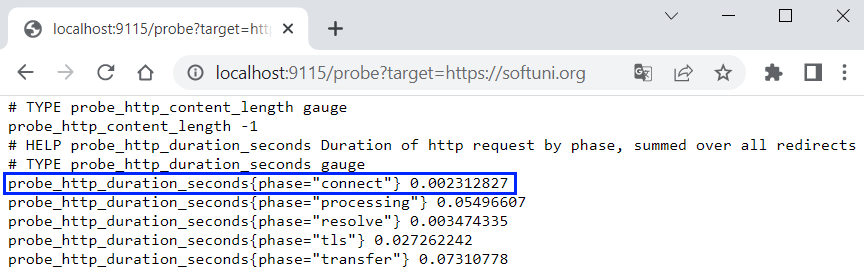
Navigate to the exporter URL in the browser with the correct target:



### Step 2: Blackbox Exporter Metrics

If you access **https://softuni.org** (**without caching**), it responds for about 23 seconds:





**NOTE:** Values **may differ** but they should be **close** enough to one another.

### Step 3: Configure and Run Prometheus

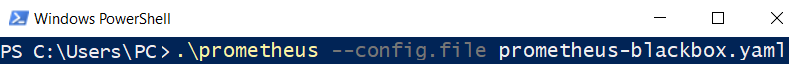
Now we should **configure** **Prometheus** to **use** the **Blackbox Exporter** metrics.

First, let's create a Prometheus configuration YAML file in the installation directory:



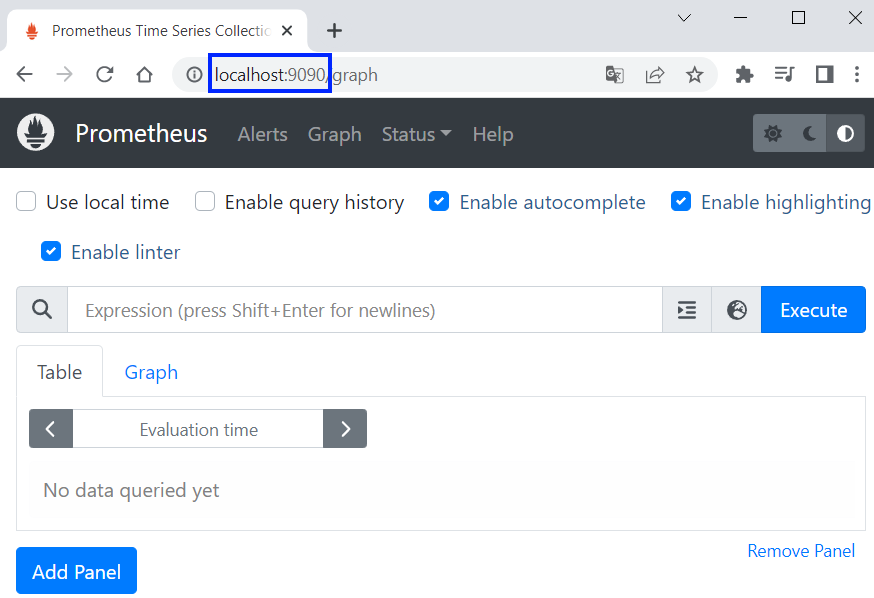
* **scrape\_interval: 15s** 
  + Target is being scraped each 15 seconds
* **metrics\_path: /probe**
  + Metrics can be accessed on **/probe**
* **- targets:  
   -** [**https://softuni.org**](https://softuni.org)
  + Define the targeted site URL
* **Replacement: 127.0.0.1:9115**
  + Blackbox exporter's **hostname:port**

Then, let's **start Prometheus server** with the **configuration file:**

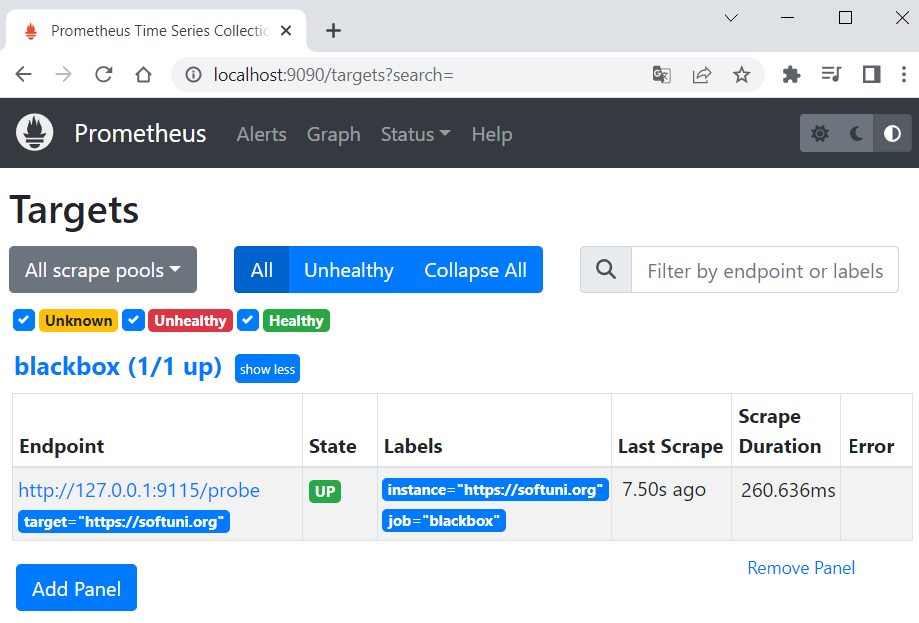


### Step 4: Access Prometheus

Now, let's access Prometheus on [**http://localhost:9090**](http://localhost:9090)



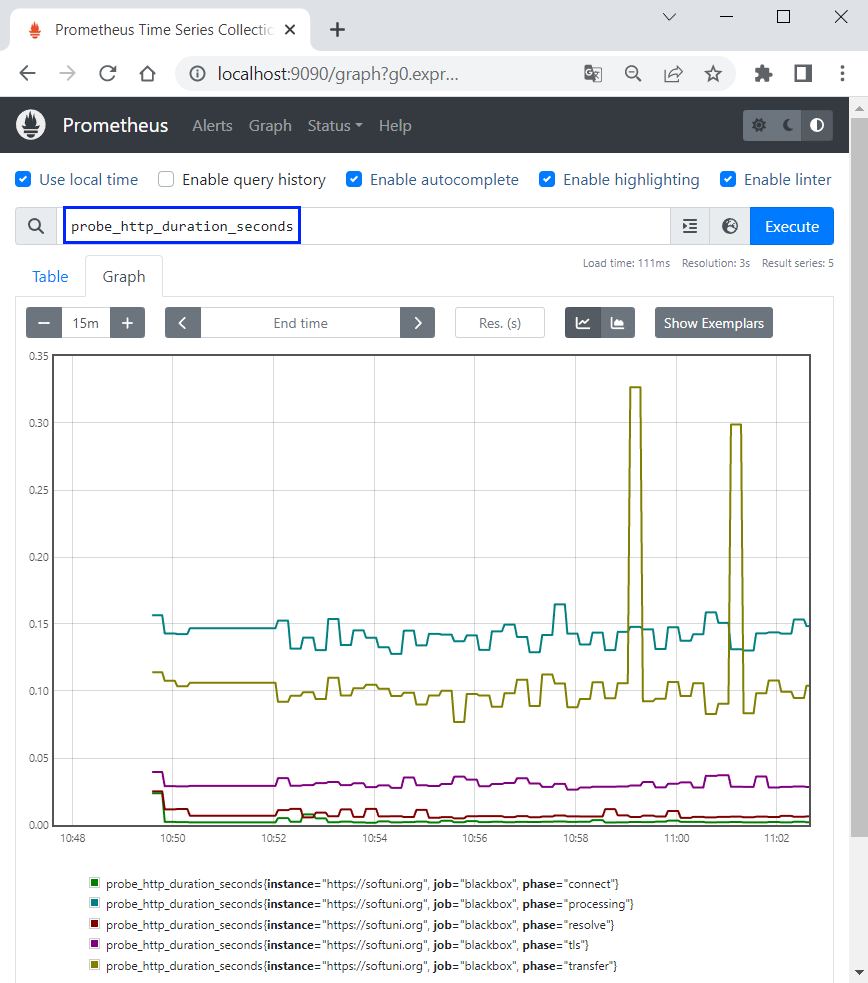
We can look at the target site status from [**Status**] → [**Targets**]:



### Step 5: Examine Metrics

Prometheus graphs are used to **visualize the metrics** collected and help you understand how **systems are performing** over time.

In order to look at Prometheus graph, navigate to [**Graph**], choose a metric to visualize and click [**Execute**]. From there you can switch from [**Table**] to [**Graph**], where you can examine the visualization:



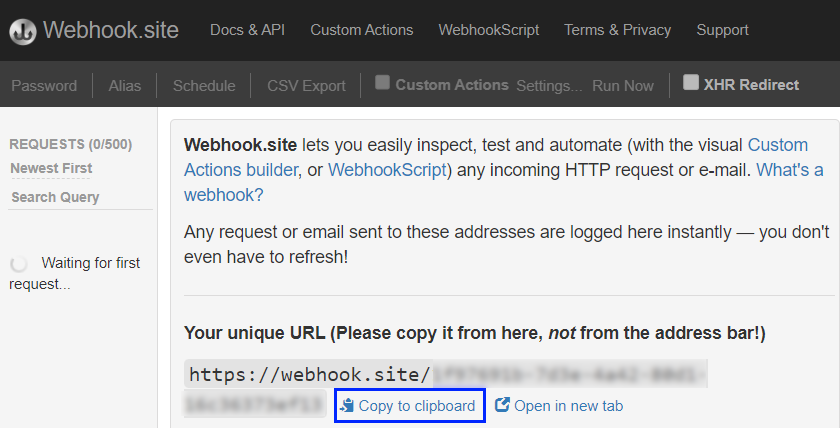
## Prometheus and Alertmanager

In this task, we will create alerts for Prometheus Metrics. If the connection to SoftUni.org takes more than 25 milliseconds, we want to fire an alert to Alertmanager. Then, Alertmanager should forward alerts to a webhook on [Webhook.site](https://webhook.site/). Everything, that is sent to it, should be shown instantly.

### Step 1: Prometheus, Blackbox Exporter and Webhook

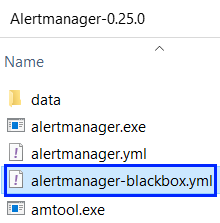
First, run **Prometheus** and **Blackbox Exporter** again.

Then, open [Webhook.site](https://webhook.site/) and copy your unique URL for the Alertmanager configuration:



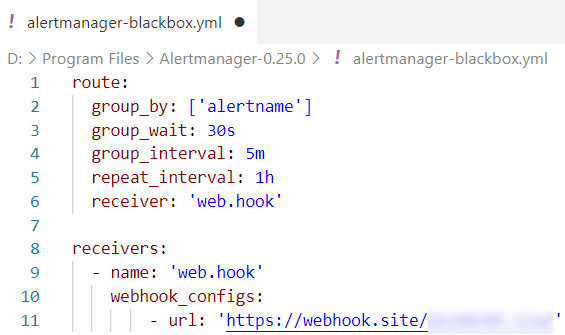
### Step 2: Configure Alertmanager

Now, let's create a YAML file in the Alertmanager directory:



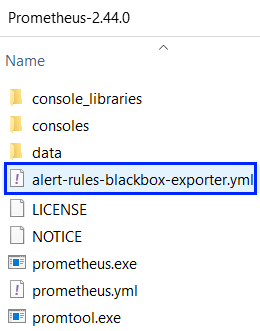
The configuration should:

* Sets the timeout for alert resolution to 5 minutes
* Specifies that alerts are sent to the "**webhook\_receiver**" receiver
* Configures the "**webhook\_receiver**" receiver
  + It should send requests to the URL that **Webhook.site** provided

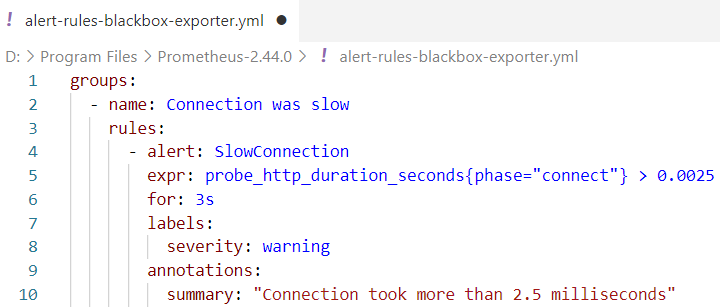


### Step 3: Configure Alerting Rules

Now, we need to configure the alerting rules, which means that we have to add rules to the Prometheus configuration. In order to do that, create a YAML file in the **Prometheus** directory:



Create the file like this:



* **- name: connection was slow**
  + The name of the rule group
* **- alert: SlowConnection**
  + The name of the alert
* **expr: probe\_http\_duration\_seconds{phase="connect"}** 
  + the Prometheus expression that **defines the condition for firing an alert**. In this case, the duration of the "connect" phase during an HTTP probe
* **for: 3s**
  + the minimum time for the expression to be true, in order to fire an alert
* **labels:  
   severity: warning**
  + the severity of the alert
* **annotations:  
   summary: "Connection took more than 2.5 milliseconds"**
  + the summary for the alert

### Step 4: Configure Prometheus

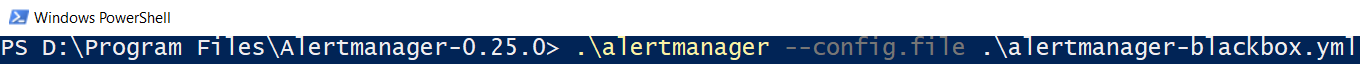
Now, let's configure Prometheus. In order to do that, we need a Prometheus YAML configuration file. We can use the **configuration from the previous demo** and add:

* Evaluation interval to define rules evaluation intervals
* The name of the rules file
* Alerting section that defines the Alertmanager configuration. Keep in mind that Alertmanager is accessed on <http://localhost:9093> (default URL).

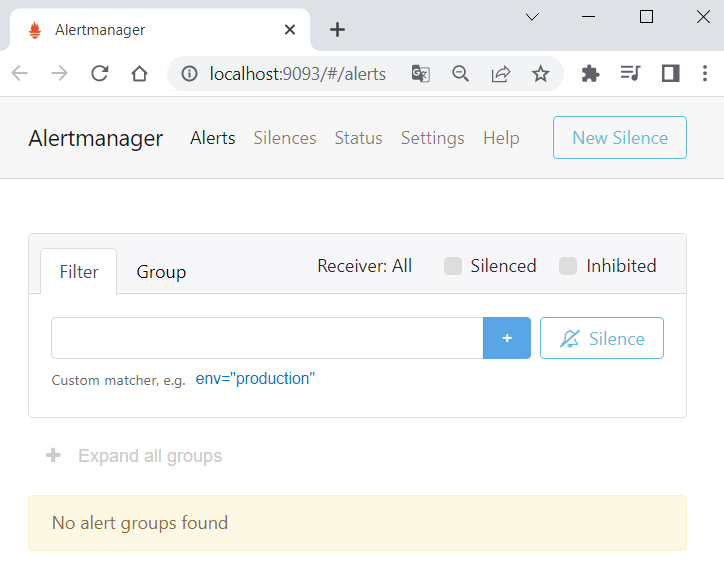


### Step 5: Run Alertmanager and Prometheus

Now, start Alertmanager with the configuration file:



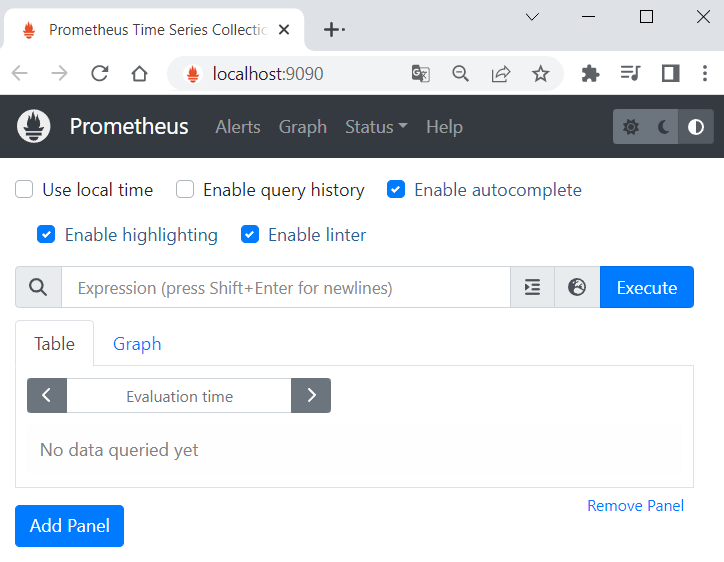
You can access it on <http://localhost:9093>:



Next, start Prometheus, too:

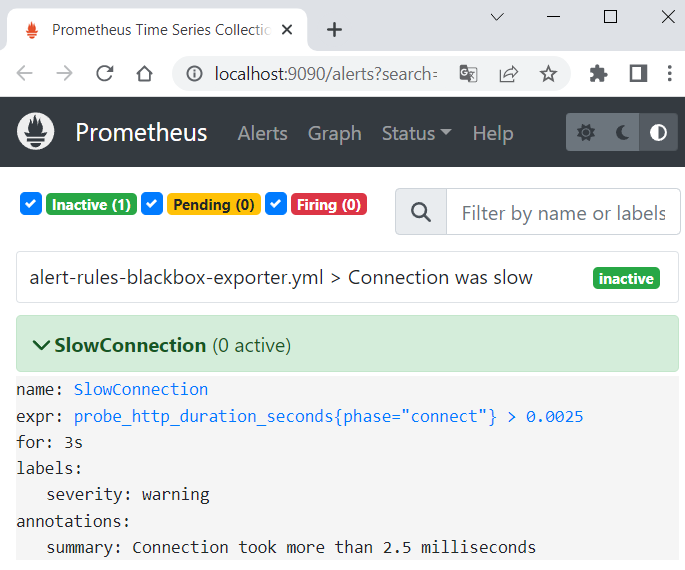


You can access it on [**http://localhost:9090**](http://localhost:9090)**:**

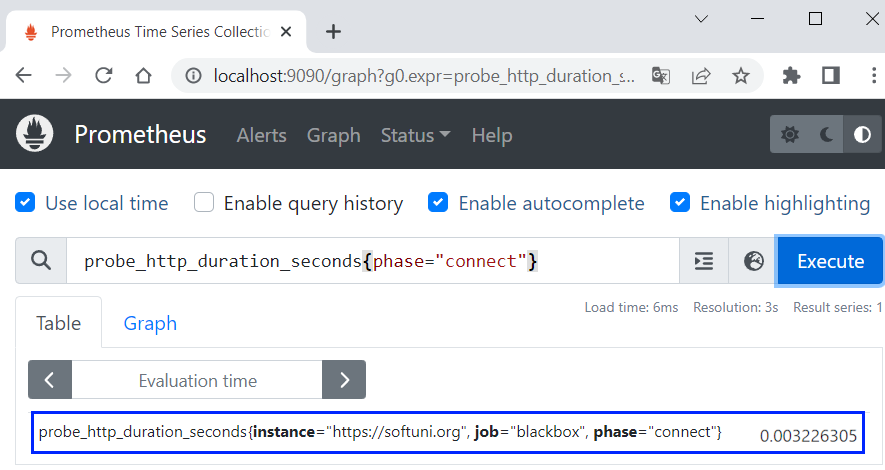


### Step 6: Fire and Examine Alert in Prometheus

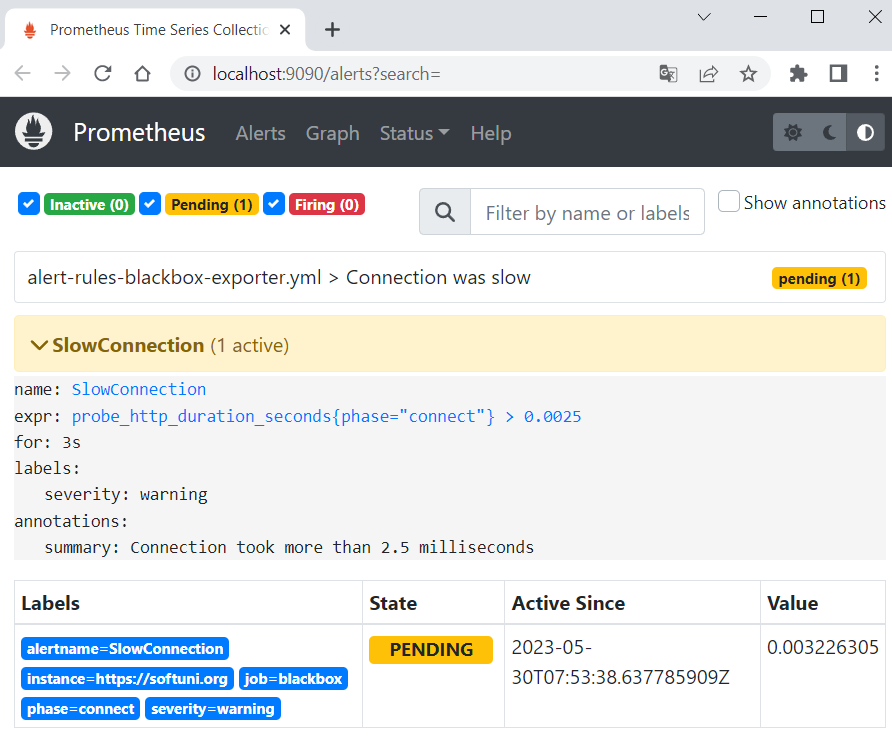
Navigate to **[Alerts]** in **Prometheus** and look at the inactive alert:



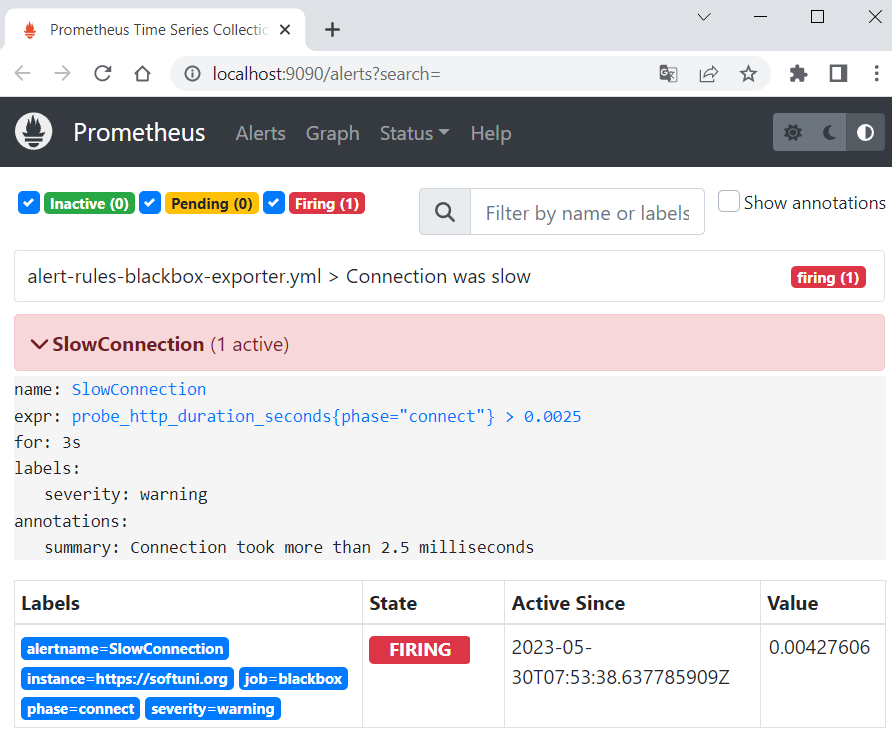
You can look at metric values to see when it **exceeds 0.0025**. This will be when the alert is fired:



Wait for **3 seconds** for the **alert to be fired** and refresh the page. It should be **pending**:

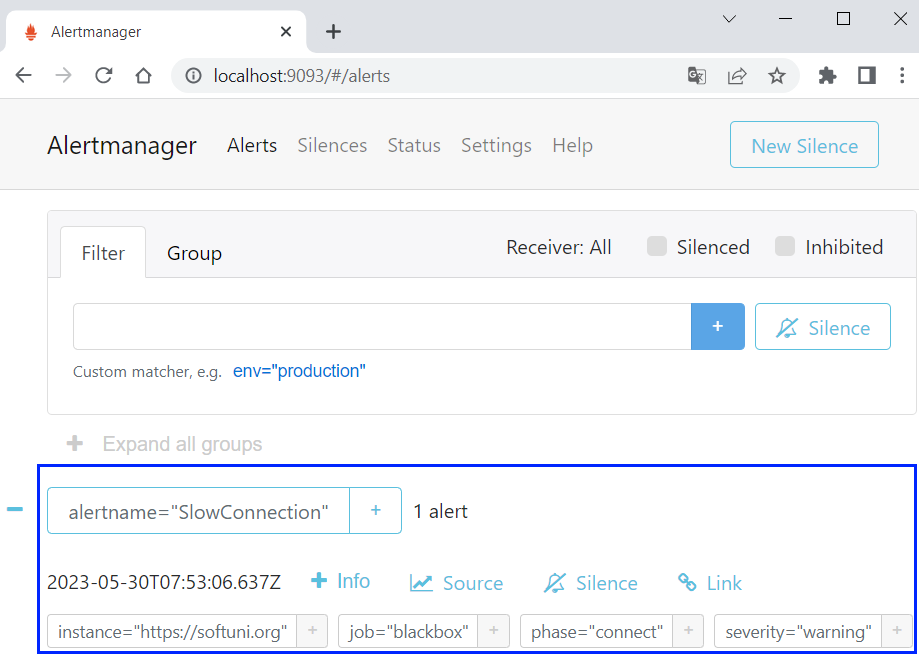


On refresh, the alert should be firing. It will become **inactive** **again** when the metric **value is <=** **0.0025:**

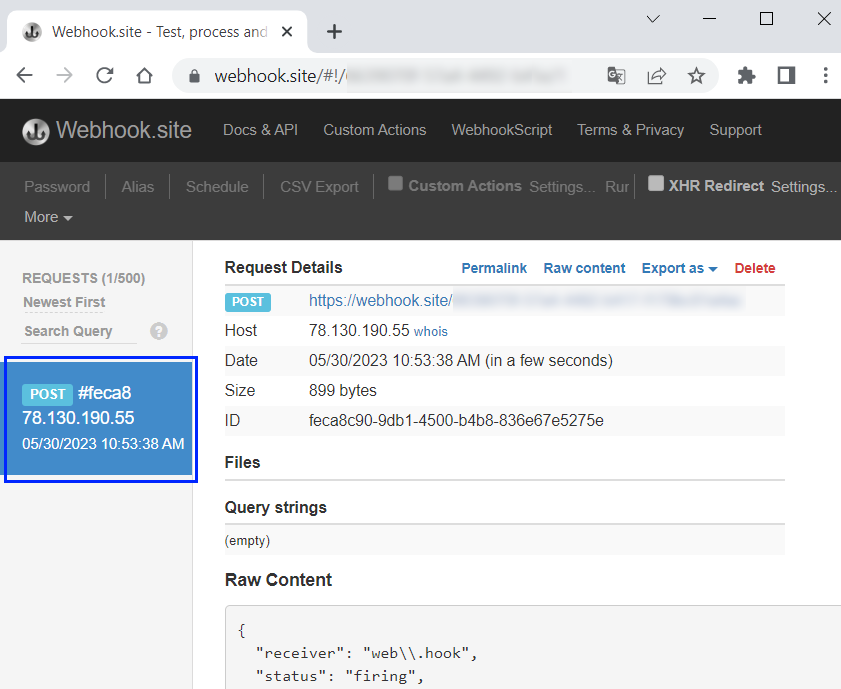


### Step 7: Examine Alert in Alertmanager and Webhook

Now, let's go to Alertmanager and we should see the fired alert:



It should also be **forwarded** to the **Webhook.site:**

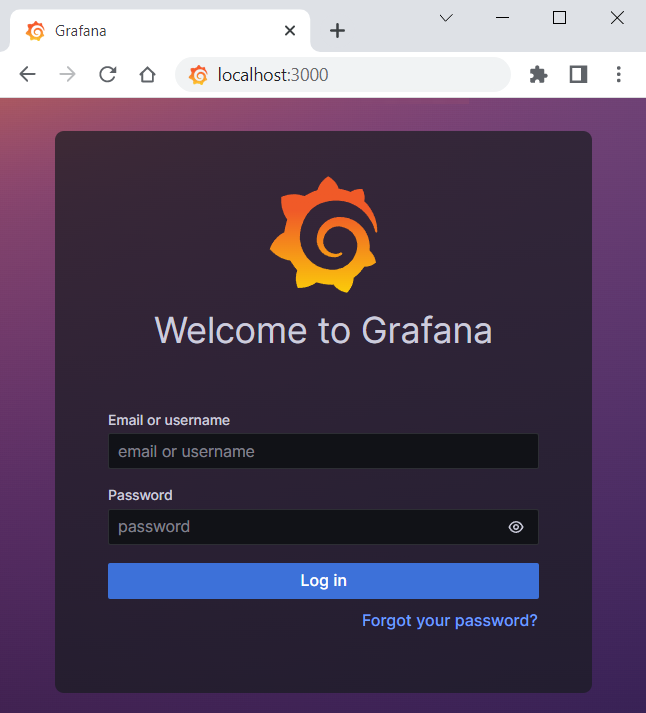


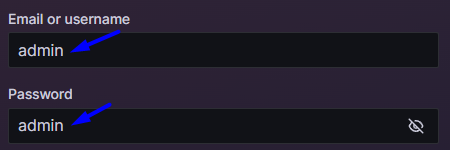
## Grafana and Prometheus

### Step 1: Log in to Grafana

By default, Grafana will be listening on [**http://localhost:3000**](http://localhost:3000). You can log in with the default credentials:

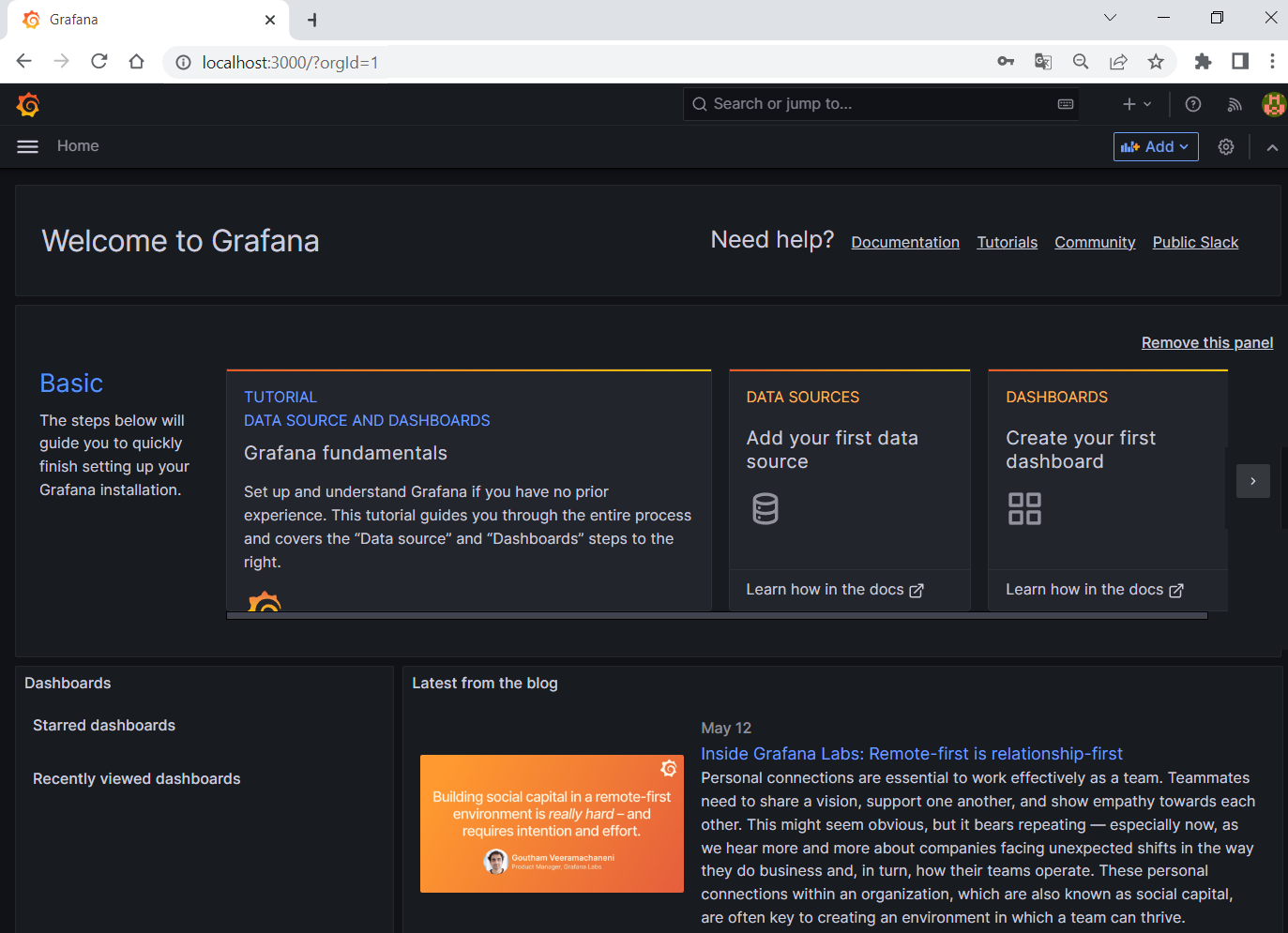
* Email or username: **admin**
* Password: **admin**





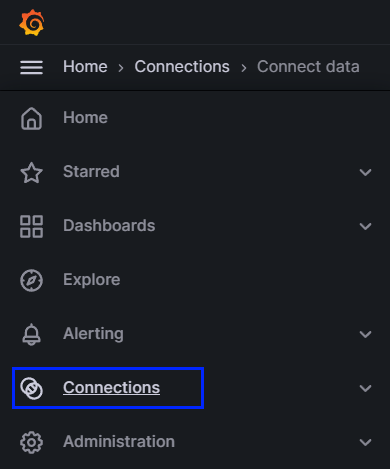
After entering the credentials, you will be asked to **set a new password**. You can do that, or you can just **skip** **this** **step**.

Then, you will be redirected to the **Welcome** page:

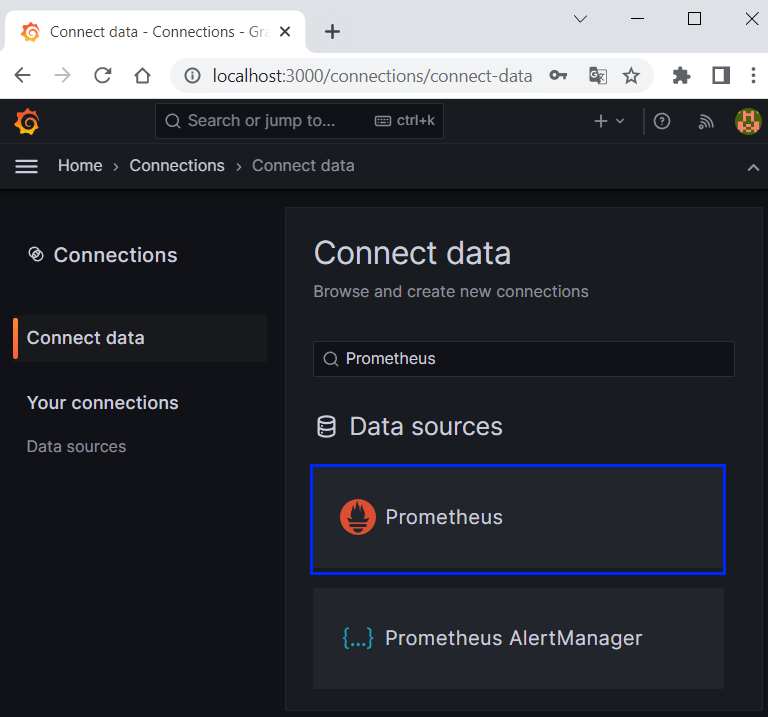


### Step 2: Create Prometheus Data Source in Grafana

Open the sidebar on the left and go to [**Connections**]:

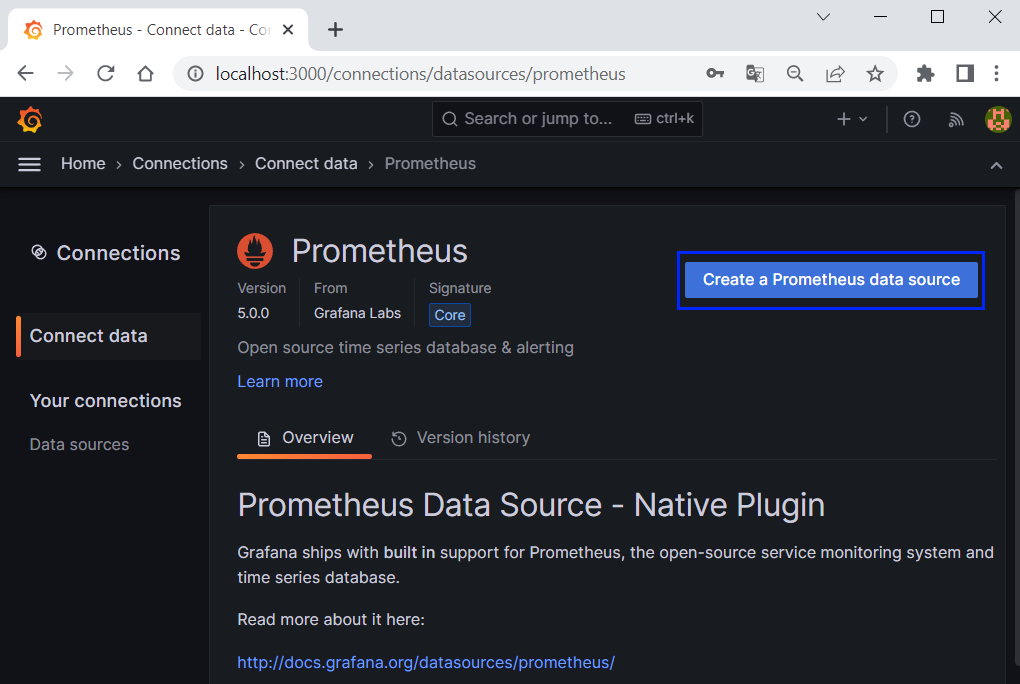


Then, search for "**Prometheus**" data source and click on it:

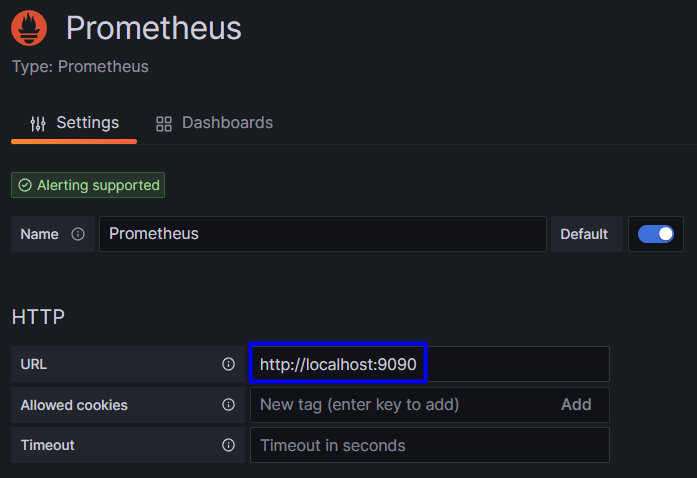


### Step 3: Configure Prometheus Data Source

Now, let's create a Prometheus data source:



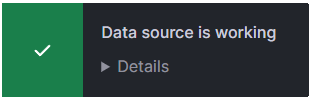
We have to set the Prometheus server URL:



Save the settings by scrolling to the bottom of the page and click on [**Save & test**]:

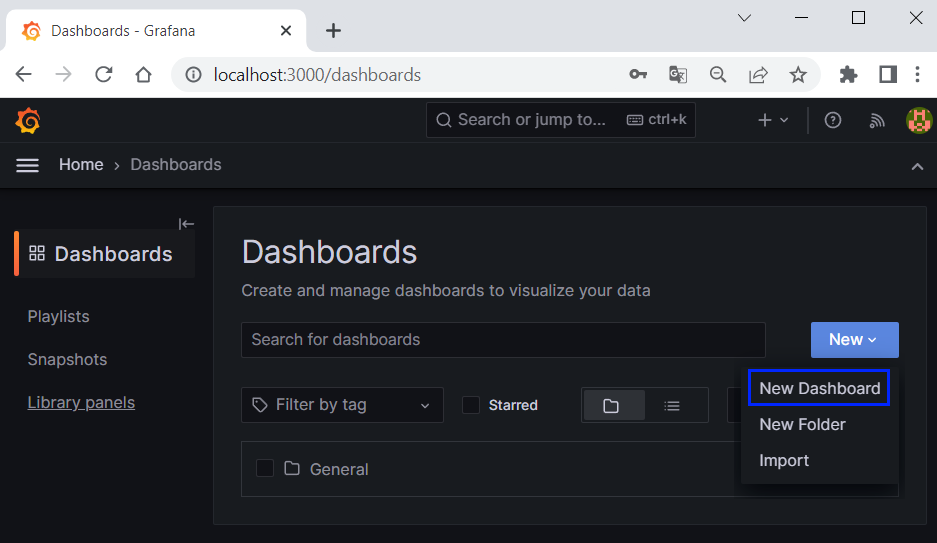


You should see the success message in a few moments:

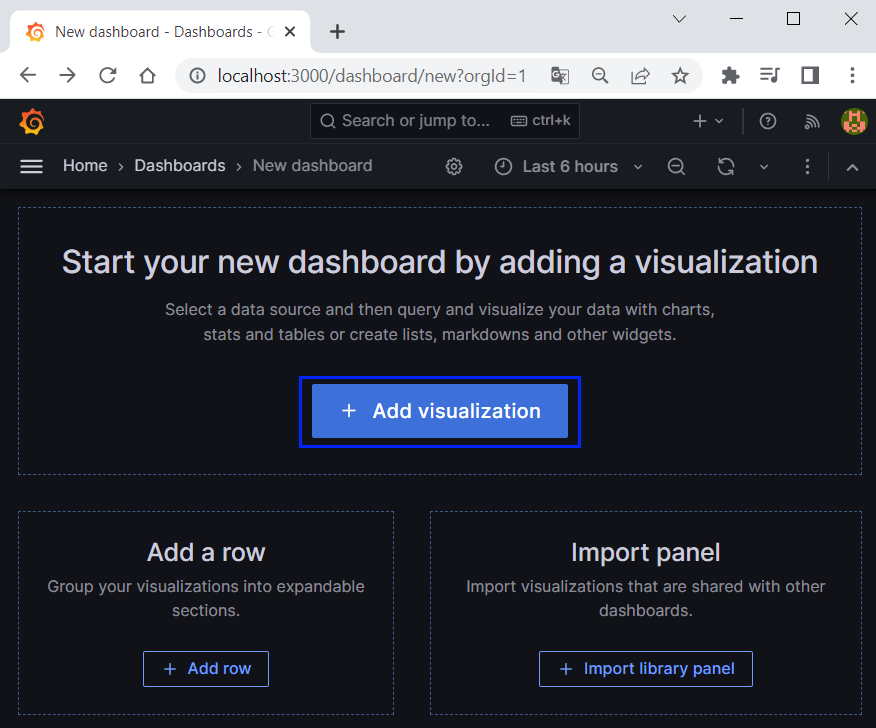


### Step 4: Create a Grafana Dashboard

Now, let's create a **new** dashboard. Go to the left sidebar again and click on [**Dashboards**]. Click on the [New] button and select [**New Dashboard**]:

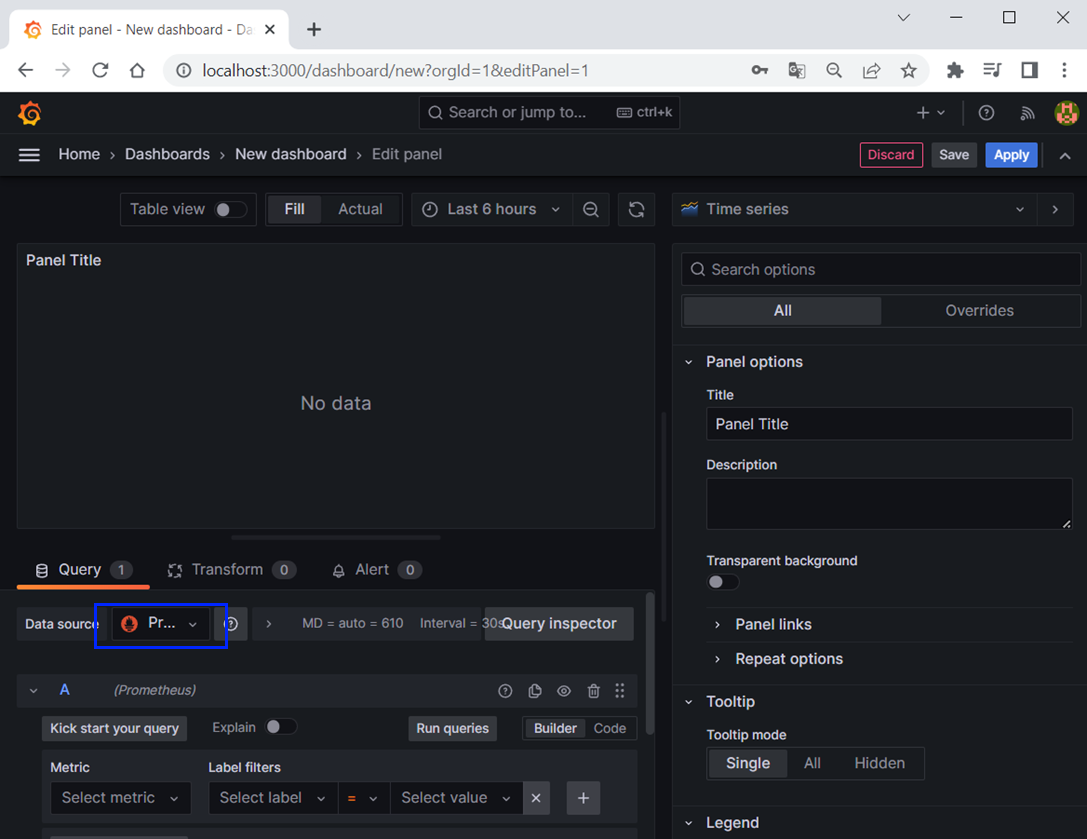


Now, click on the [**+ Add visualization**] button to create a visualization (panel):

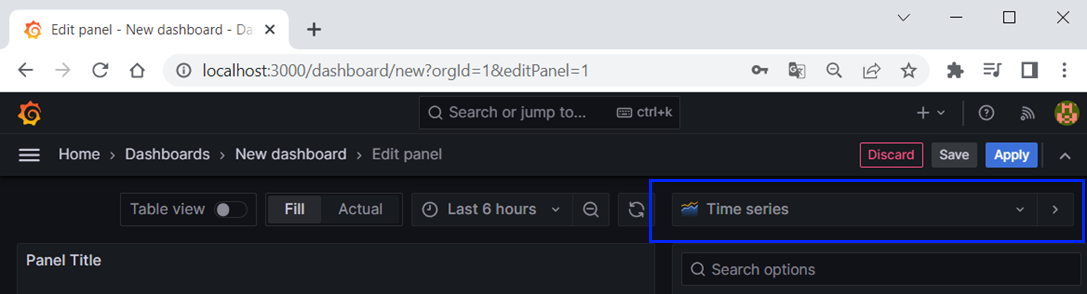


For example, you can create a histogram for the HTTP probe duration metric.

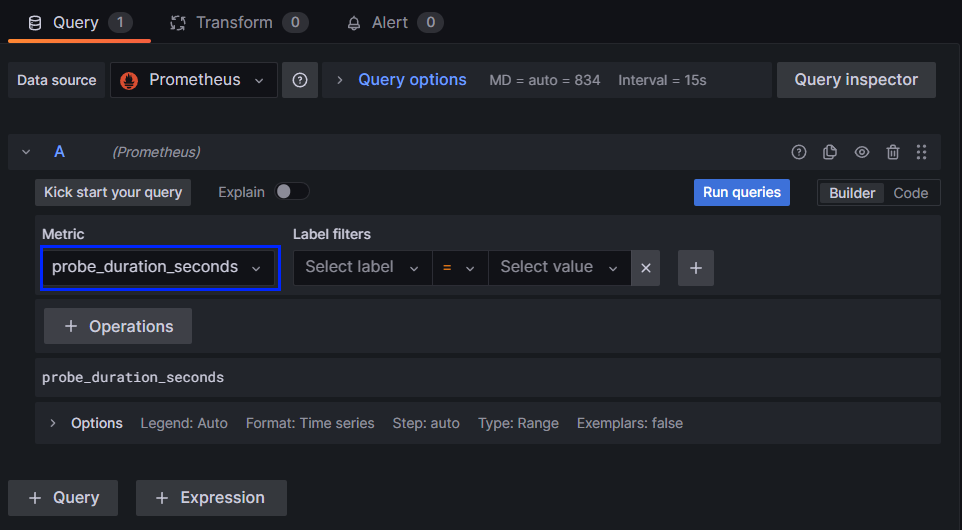
First, select the **Prometheus** data source from the Data source dropdown menu:



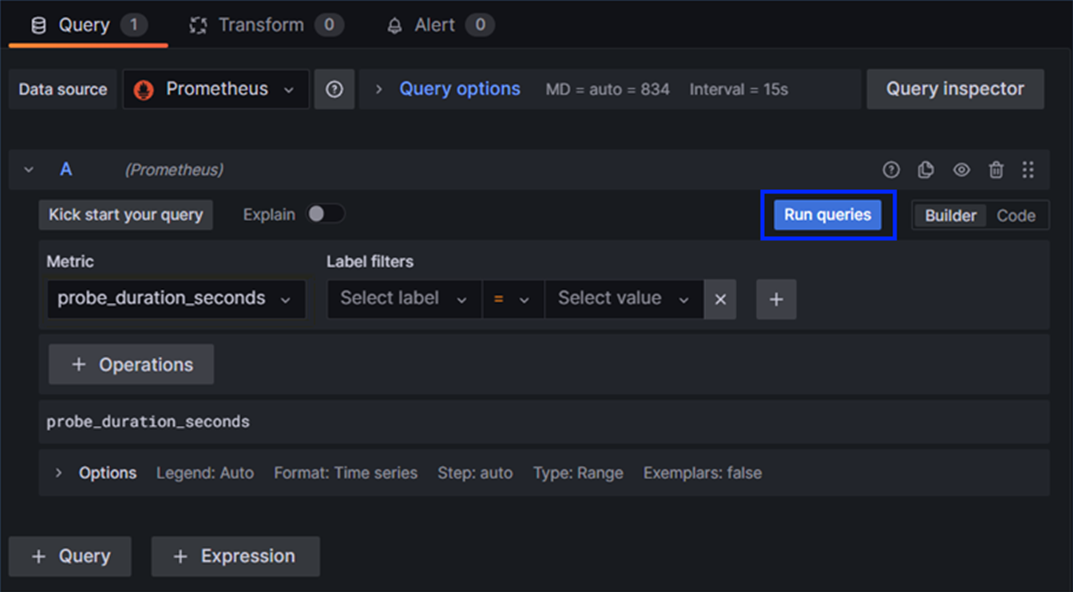
Next, click on the **Visualization** menu and select **Histogram**:

 → 

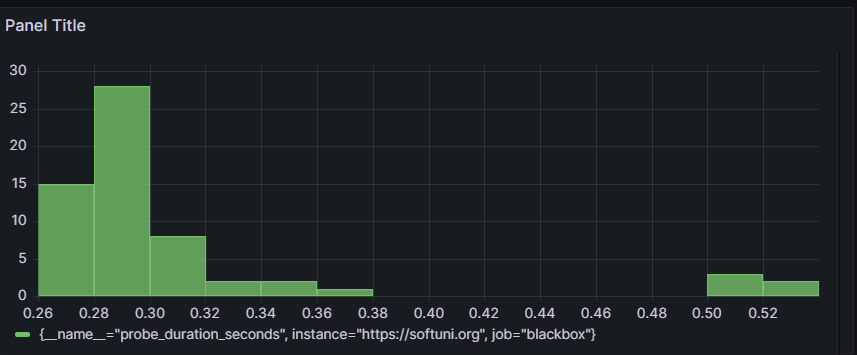
Next, select the metric for **lookup**:



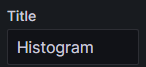
After that, click on the [**Run queries**] button:



And you should be able to see the panel:



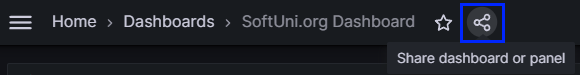
In order to change the panel title, click on the "**Title**" input field in the **Panel options** section:



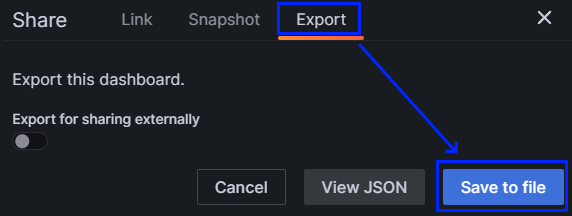
### Step 5: Import and Export Dashboards

You can export and import Grafana dashboards as **JSON files.**

In order to **export a dashboard, open Dashboards** andthen **click on [Share]:**



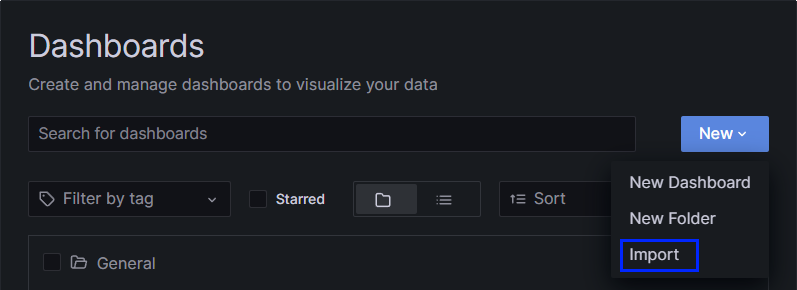
Then, click on **[Export]** → **[Save to file]:**



You should be able to **save** the dashboard as a **JSON** file

 → 

If you want to import a dashboard from a JSON file, go to **Dashboards** and click on **[New]** → **[Import]:**



You should be able to see the upload menu. You can choose to drag and drop the JSON file or just upload it:

