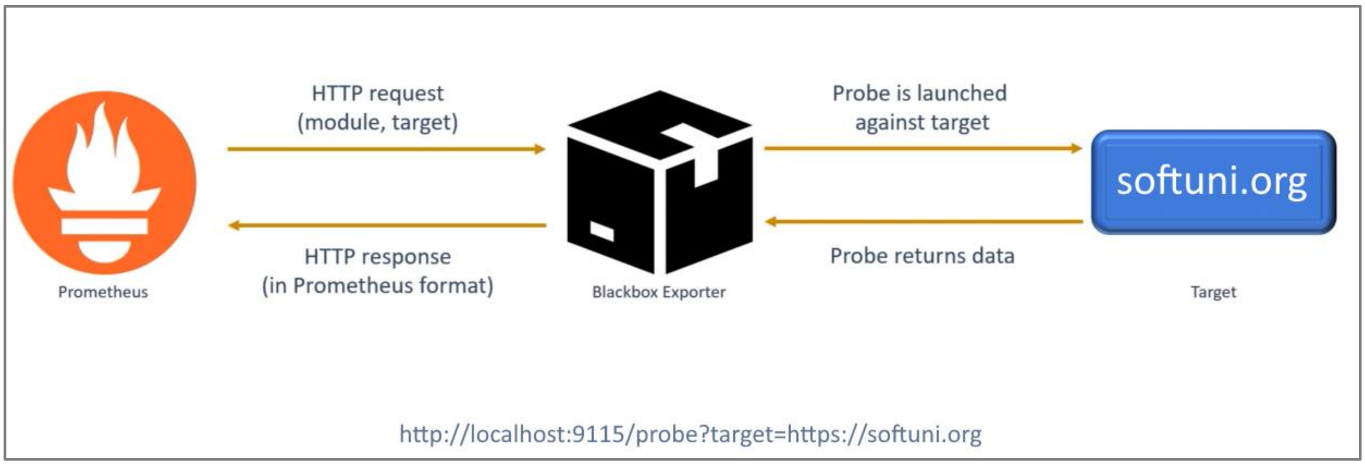
Lab: App Monitoring

Lab for the "Containers and Clouds" course @ SoftUni

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

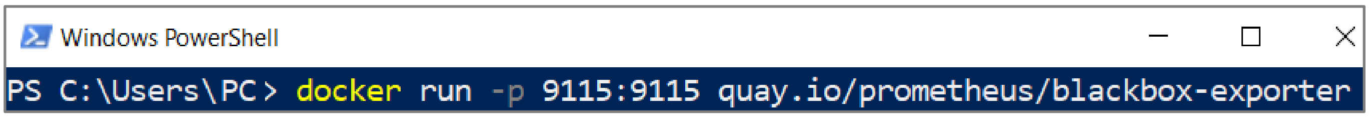
Our task is to **configure Prometheus** to **monitor** https://softuni.org. In order for this to happen, we shall use the **Prometheus Blackbox Exporter.**

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



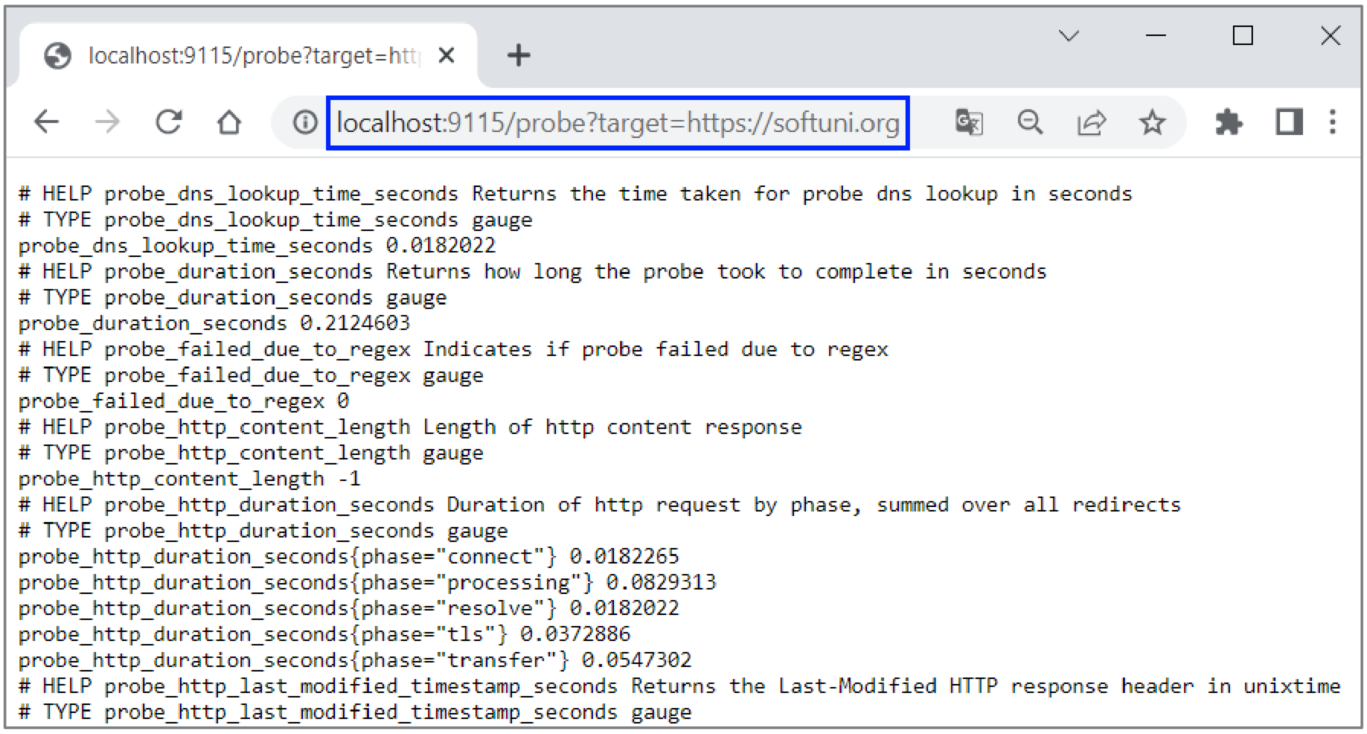
Step 1: Install and Run Blackbox Exporter

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



```
PS C:\Users\PC> docker run -p 9115:9115 quay.io/prometheus/blackbox-exporter
```

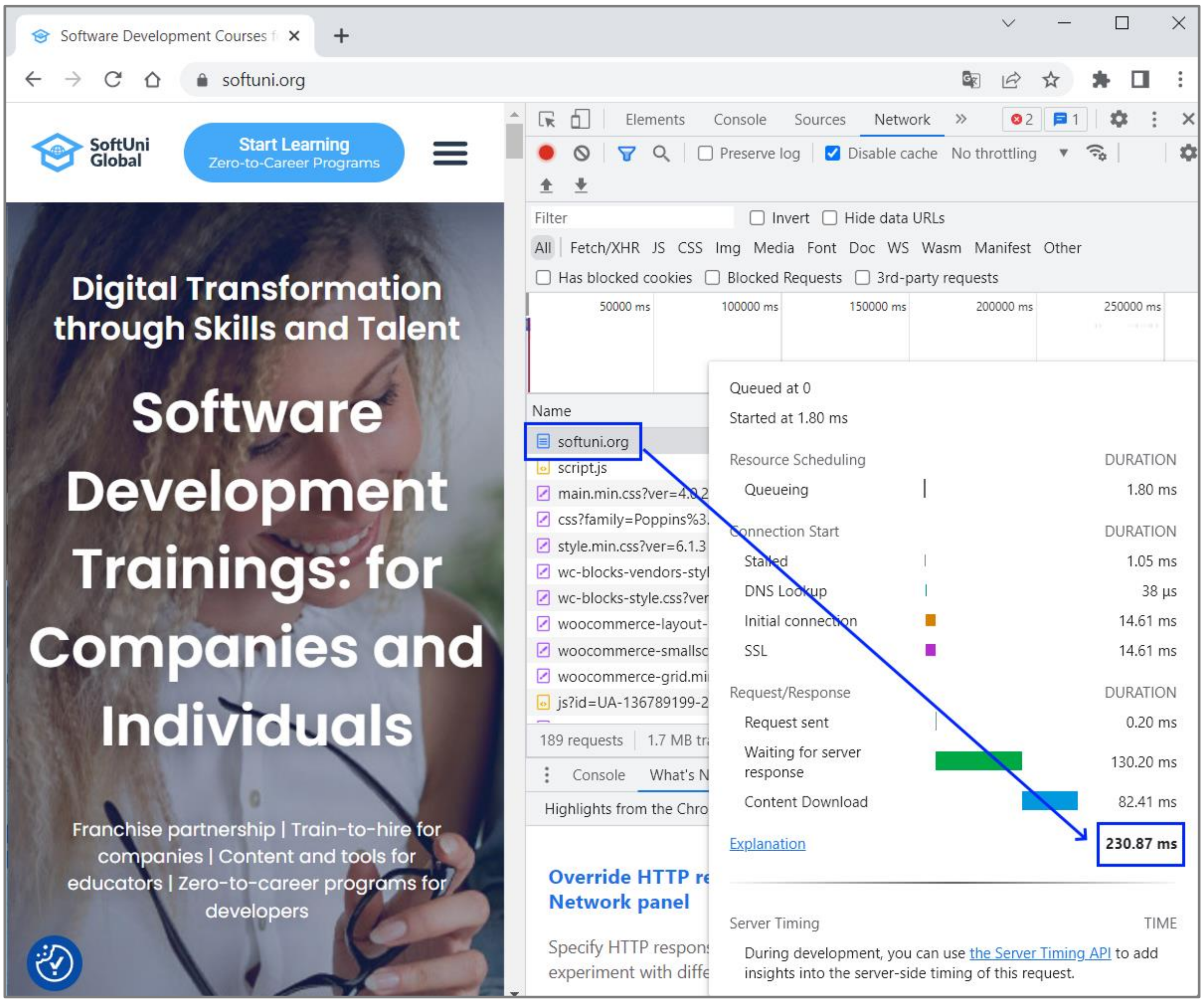
Navigate to the exporter URL in the browser with the correct target:



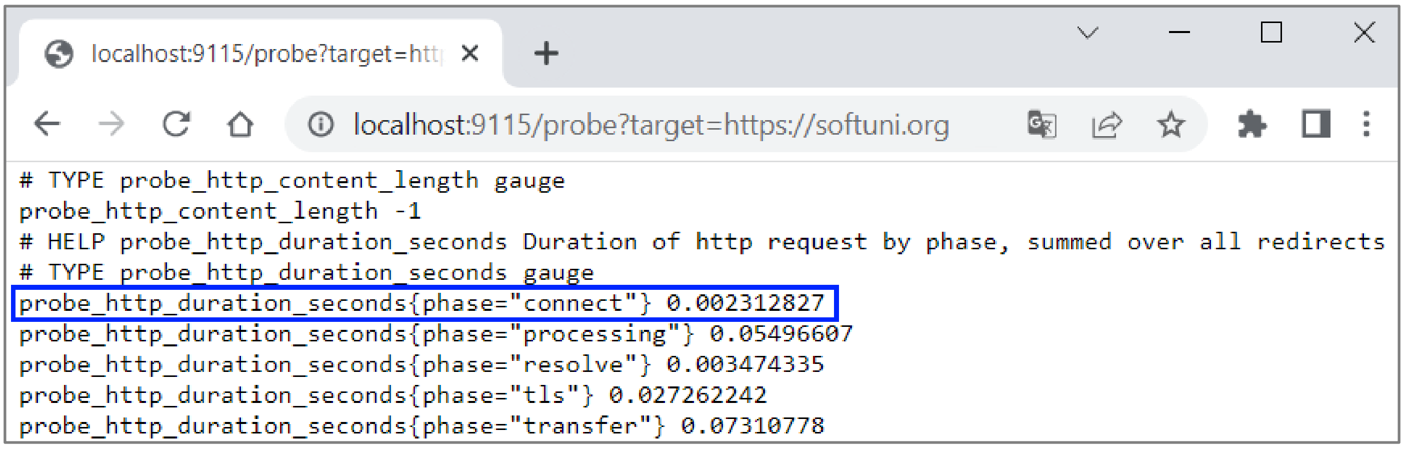
```
# HELP probe_dns_lookup_time_seconds Returns the time taken for probe dns lookup in seconds
# TYPE probe_dns_lookup_time_seconds gauge
probe_dns_lookup_time_seconds 0.0182022
# HELP probe_duration_seconds Returns how long the probe took to complete in seconds
# TYPE probe_duration_seconds gauge
probe_duration_seconds 0.2124603
# HELP probe_failed_due_to_regex Indicates if probe failed due to regex
# TYPE probe_failed_due_to_regex gauge
probe_failed_due_to_regex 0
# HELP probe_http_content_length Length of http content response
# TYPE probe_http_content_length gauge
probe_http_content_length -1
# HELP probe_http_duration_seconds Duration of http request by phase, summed over all redirects
# TYPE probe_http_duration_seconds gauge
probe_http_duration_seconds{phase="connect"} 0.0182265
probe_http_duration_seconds{phase="processing"} 0.0829313
probe_http_duration_seconds{phase="resolve"} 0.0182022
probe_http_duration_seconds{phase="tls"} 0.0372886
probe_http_duration_seconds{phase="transfer"} 0.0547302
# HELP probe_http_last_modified_timestamp_seconds Returns the Last-Modified HTTP response header in unixtime
# TYPE probe_http_last_modified_timestamp_seconds gauge
```

Step 2: Blackbox Exporter Metrics

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


Software Development Courses f
softuni.org
Elements
Console
Sources
Network
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
All
Fetch/XHR
JS
CSS
Img
Media
Font
Doc
WS
Wasm
Manifest
Other
Has blocked cookies
Blocked Requests
3rd-party requests
Start Learning
Zero-to-Career Programs
Digital Transformation through Skills and Talent
Software Development Trainings: for Companies and Individuals
Franchise partnership | Train-to-hire for companies | Content and tools for educators | Zero-to-career programs for developers

```
Name
softuni.org
script.js
main.min.css?ver=4.0.2
css?family=Poppins%3.
style.min.css?ver=6.1.3
wc-blocks-vendors-styl
wc-blocks-style.css?ver
woocommerce-layout-
woocommerce-smallsc
woocommerce-grid.mi
js?id=UA-136789199-2
189 requests | 1.7 MB tr

Queued at 0
Started at 1.80 ms

Resource Scheduling          DURATION
  Queueing                   1.80 ms
Connection Start             DURATION
  Stalled                    1.05 ms
  DNS Lookup                 38 µs
  Initial connection         14.61 ms
  SSL                        14.61 ms
Request/Response             DURATION
  Request sent               0.20 ms
  Waiting for server
  response                   130.20 ms
  Content Download           82.41 ms
Explanation                  230.87 ms

Server Timing                TIME
  During development, you can use the Server Timing API to add
  insights into the server-side timing of this request.
```


localhost:9115/probe?target=htt
localhost:9115/probe?target=https://softuni.org

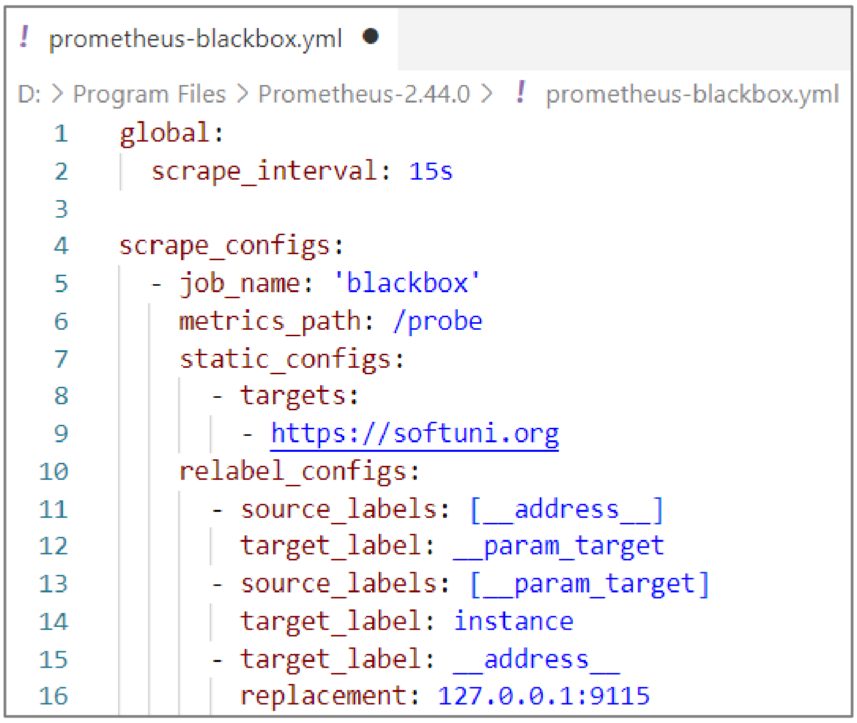
```
# TYPE probe_http_content_length gauge
probe_http_content_length -1
# HELP probe_http_duration_seconds Duration of http request by phase, summed over all redirects
# TYPE probe_http_duration_seconds gauge
probe_http_duration_seconds{phase="connect"} 0.002312827
probe_http_duration_seconds{phase="processing"} 0.05496607
probe_http_duration_seconds{phase="resolve"} 0.003474335
probe_http_duration_seconds{phase="tls"} 0.027262242
probe_http_duration_seconds{phase="transfer"} 0.07310778
```

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.

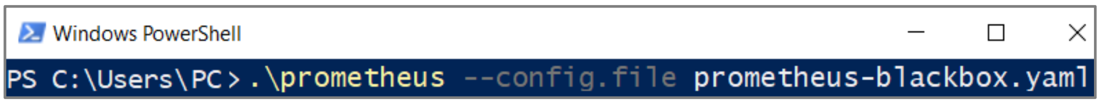
First, let's create a Prometheus configuration YAML file in the installation directory:


prometheus-blackbox.yml
D: > Program Files > Prometheus-2.44.0 > prometheus-blackbox.yml

```yaml
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'blackbox'
    metrics_path: /probe
    static_configs:
      - targets:
        - https://softuni.org
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
      - target_label: __address__
        replacement: 127.0.0.1:9115
```

- `scrape_interval: 15s`
  - Target is being scraped each 15 seconds
- `metrics_path: /probe`
  - Metrics can be accessed on **`/probe`**
- **`- targets:`**
  `- https://softuni.org`
  - Define the targeted site URL
- **`Replacement: 127.0.0.1:9115`**
  - Blackbox exporter's **`hostname:port`**

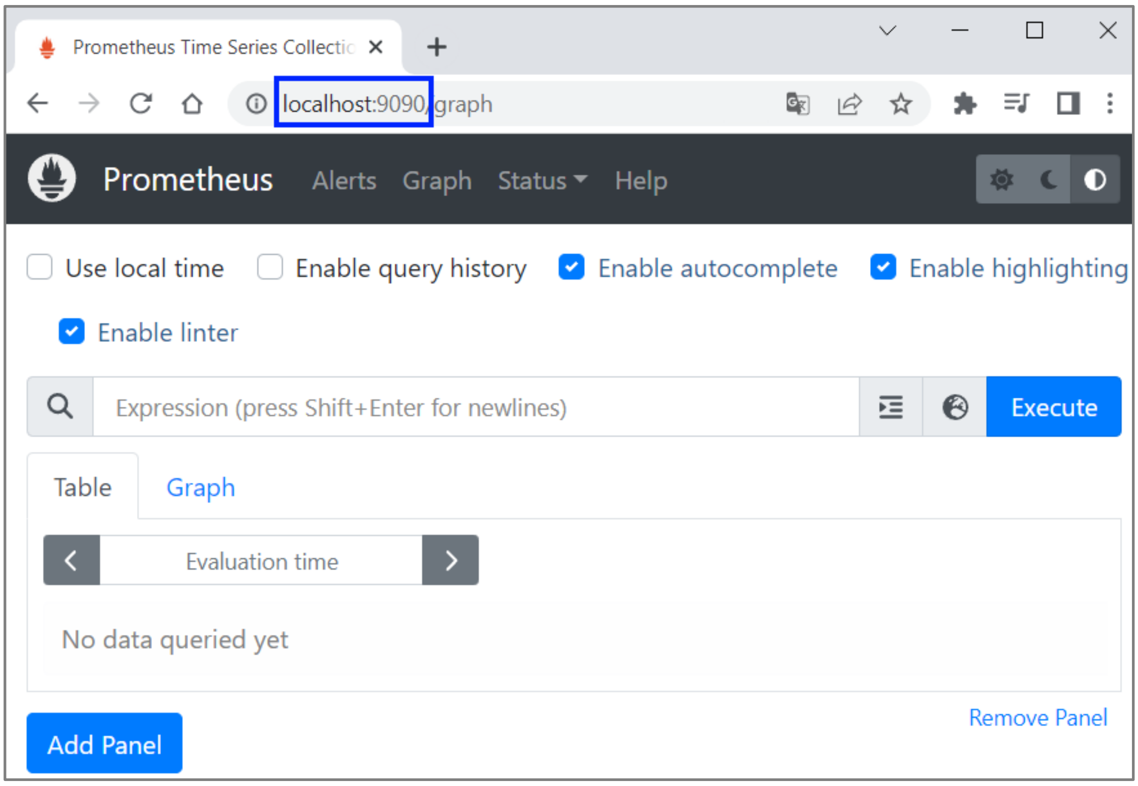
Then, let's **start Prometheus server** with the **configuration file:**



```
PS C:\Users\PC> .\prometheus --config.file prometheus-blackbox.yaml
```

Step 4: Access Prometheus

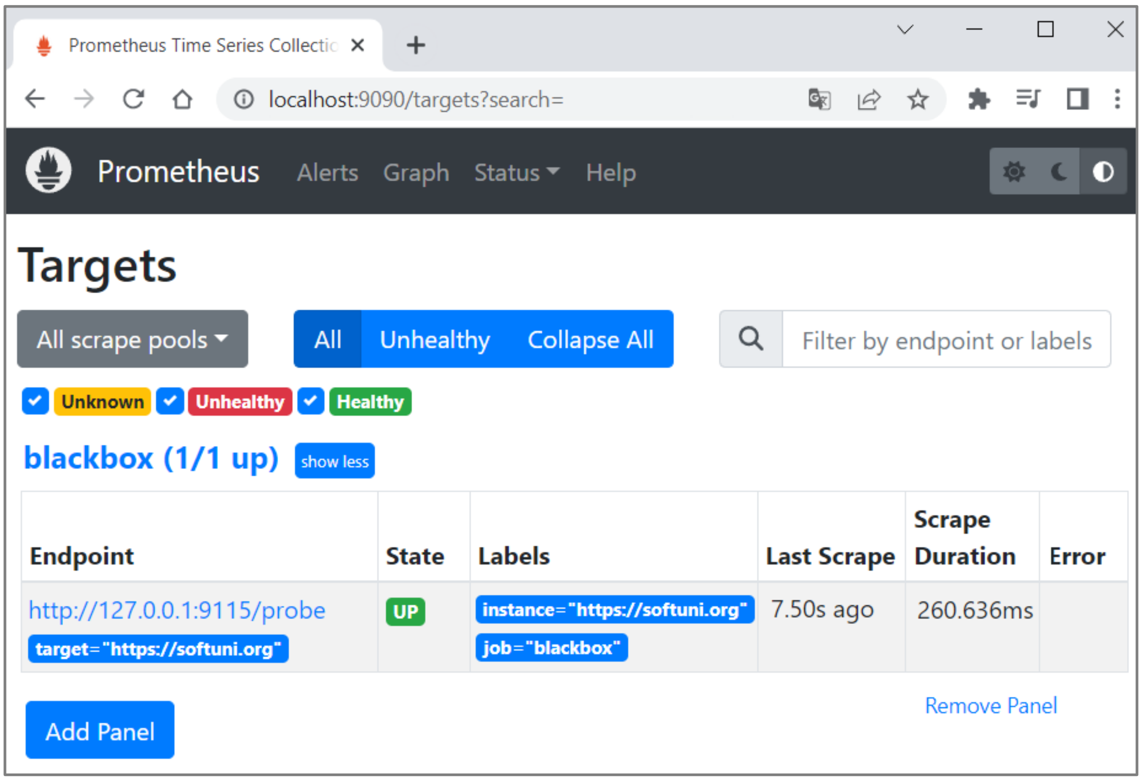
Now, let's access Prometheus on http://localhost:9090



☐ Use local time ☐ Enable query history ☑ Enable autocomplete ☑ Enable highlighting ☑ Enable linter

No data queried yet

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


Prometheus Time Series Collectic
localhost:9090/targets?search=
Prometheus
Alerts
Graph
Status
Help
All scrape pools
All
Unhealthy
Collapse All
Filter by endpoint or labels
☑ Unknown ☑ Unhealthy ☑ Healthy
show less
Add Panel
Remove Panel

Targets

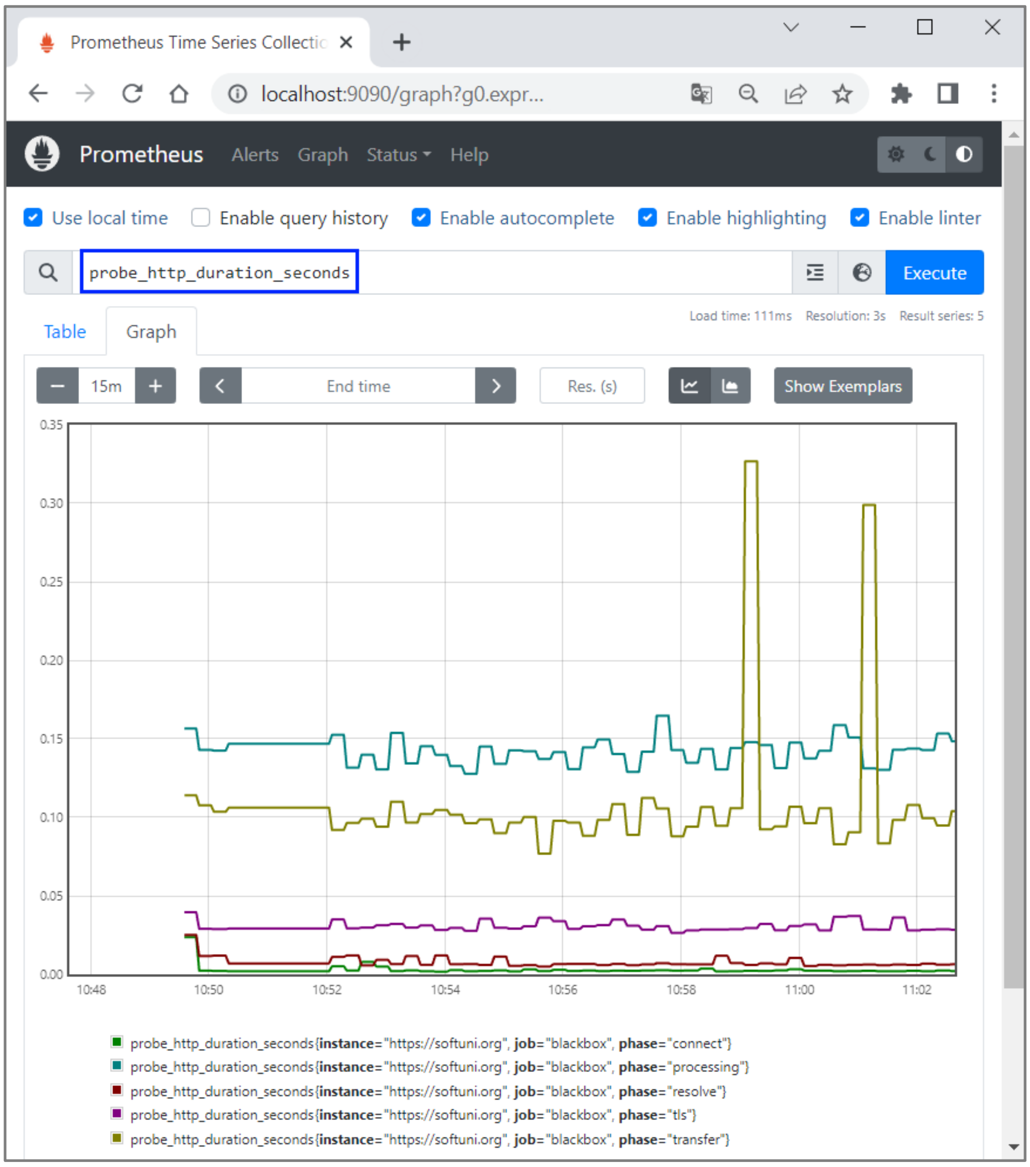
blackbox (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://127.0.0.1:9115/probe target="https://softuni.org"	UP	instance="https://softuni.org" job="blackbox"	7.50s ago	260.636ms	

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:



☑ Use local time ☐ Enable query history ☑ Enable autocomplete ☑ Enable highlighting ☑ Enable linter

```
probe_http_duration_seconds
```

Load time: 111ms Resolution: 3s Result series: 5

- 🟩 probe_http_duration_seconds{**instance**="https://softuni.org", **job**="blackbox", **phase**="connect"}
- 🟩 probe_http_duration_seconds{**instance**="https://softuni.org", **job**="blackbox", **phase**="processing"}
- 🟥 probe_http_duration_seconds{**instance**="https://softuni.org", **job**="blackbox", **phase**="resolve"}
- 🟪 probe_http_duration_seconds{**instance**="https://softuni.org", **job**="blackbox", **phase**="tls"}
- 🟨 probe_http_duration_seconds{**instance**="https://softuni.org", **job**="blackbox", **phase**="transfer"}

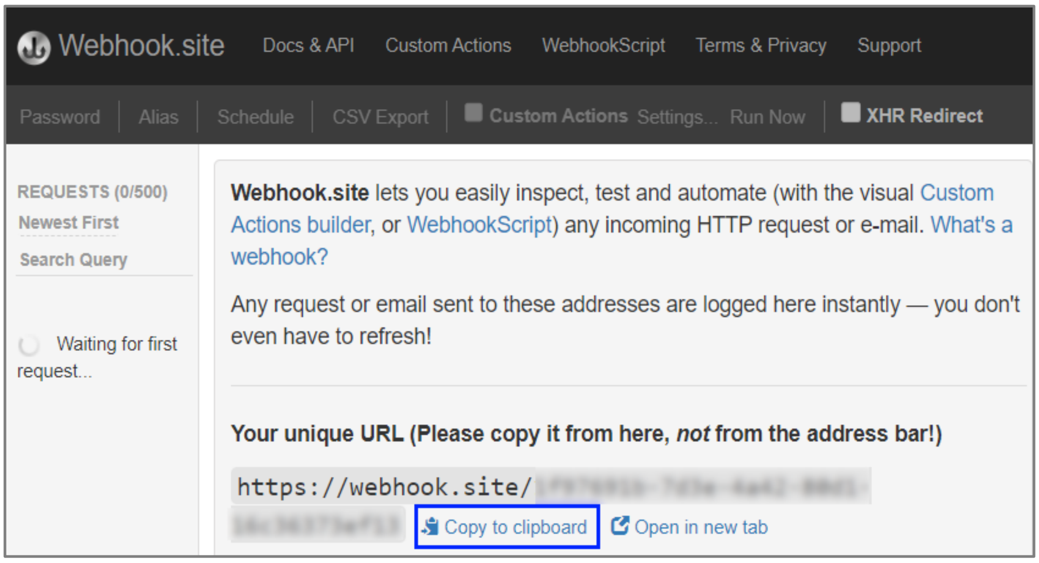
2. Prometheus and Alertmanager

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. 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 and copy your unique URL for the Alertmanager configuration:



Webhook.site lets you easily inspect, test and automate (with the visual Custom Actions builder, or WebhookScript) any incoming HTTP request or e-mail. What's a webhook?

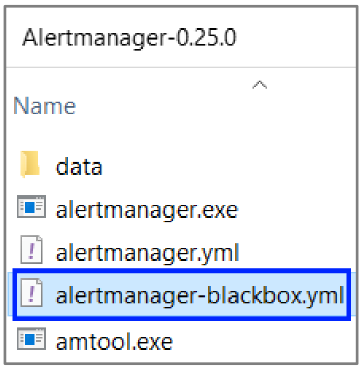
Any request or email sent to these addresses are logged here instantly — you don't even have to refresh!

Your unique URL (Please copy it from here, *not* from the address bar!)

`https://webhook.site/`[redacted]

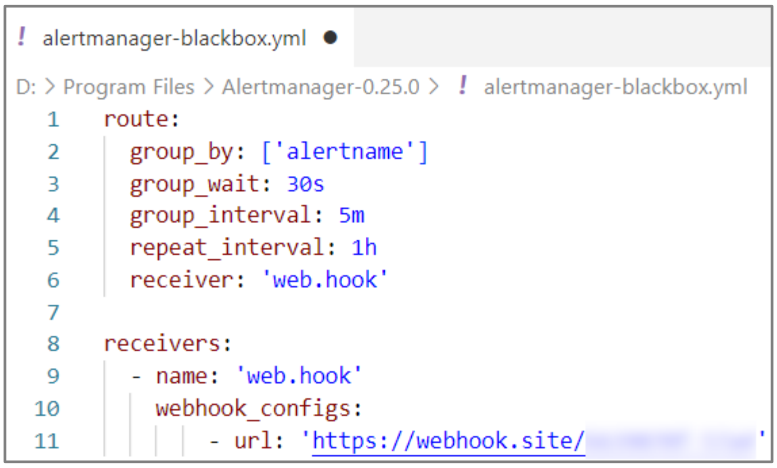
Step 2: Configure Alertmanager

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



The configuration should:

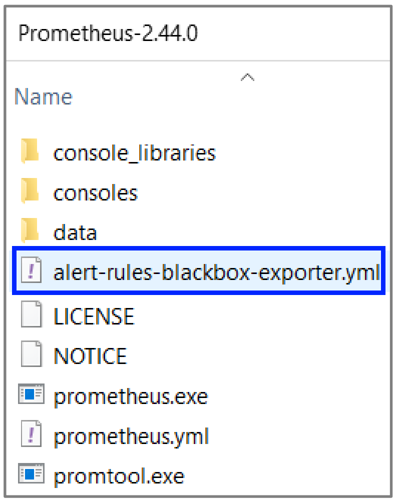
- Sets the timeout for alert resolution to 5 minutes
- Specifies that alerts are sent to the "**`webhook_receiver`**" receiver
- Configures the "**`webhook_receiver`**" receiver
  - It should send requests to the URL that **`Webhook.site`** provided


alertmanager-blackbox.yml
D: > Program Files > Alertmanager-0.25.0 > alertmanager-blackbox.yml

```yaml
route:
  group_by: ['alertname']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 1h
  receiver: 'web.hook'

receivers:
  - name: 'web.hook'
    webhook_configs:
      - url: 'https://webhook.site/[redacted]'
```

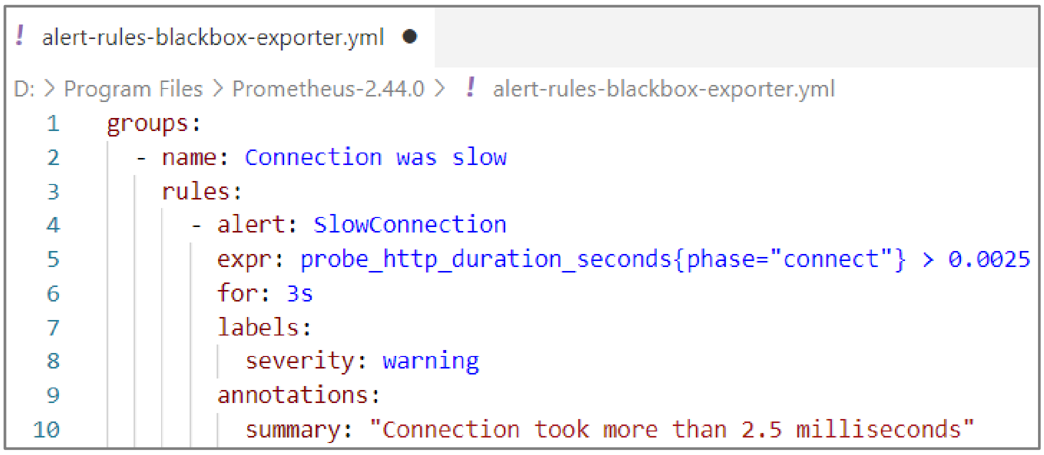
Step 3: Configure Alerting Rules

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:



```
Prometheus-2.44.0
Name
console_libraries
consoles
data
alert-rules-blackbox-exporter.yml
LICENSE
NOTICE
prometheus.exe
prometheus.yml
promtool.exe
```

Create the file like this:



```
alert-rules-blackbox-exporter.yml
D: > Program Files > Prometheus-2.44.0 > alert-rules-blackbox-exporter.yml
groups:
  - name: Connection was slow
    rules:
      - alert: SlowConnection
        expr: probe_http_duration_seconds{phase="connect"} > 0.0025
        for: 3s
        labels:
          severity: warning
        annotations:
          summary: "Connection took more than 2.5 milliseconds"
```

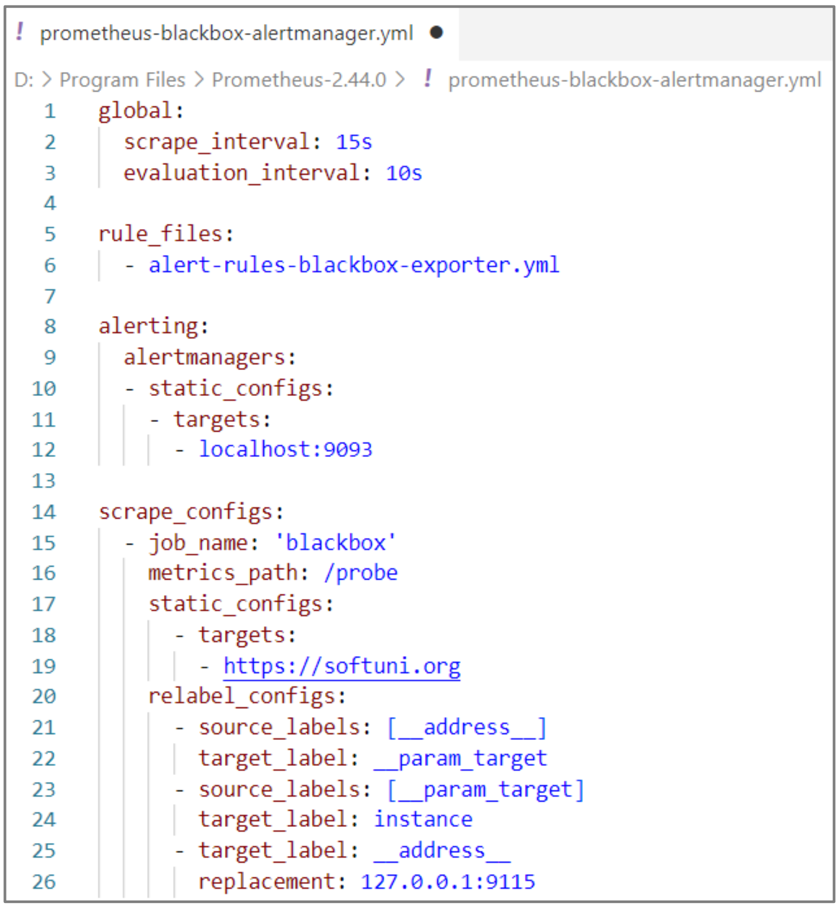
- **`- name: connection was slow`**
  - The name of the rule group
- **`- alert: SlowConnection`**
  - The name of the alert
- **`expr: probe_http_duration_seconds{phase="connect"}`**
  - the Prometheus expression that **defines the condition for firing an alert**. In this case, the duration of the "connect" 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).



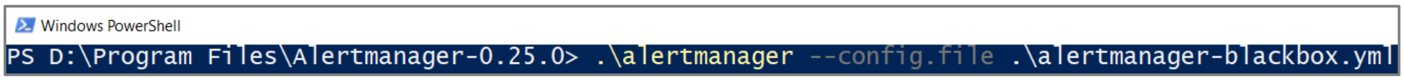
prometheus-blackbox-alertmanager.yml

D: > Program Files > Prometheus-2.44.0 > prometheus-blackbox-alertmanager.yml

```yaml
global:
  scrape_interval: 15s
  evaluation_interval: 10s

rule_files:
  - alert-rules-blackbox-exporter.yml

alerting:
  alertmanagers:
  - static_configs:
    - targets:
      - localhost:9093

scrape_configs:
  - job_name: 'blackbox'
    metrics_path: /probe
    static_configs:
      - targets:
        - https://softuni.org
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
      - target_label: __address__
        replacement: 127.0.0.1:9115
```

Step 5: Run Alertmanager and Prometheus

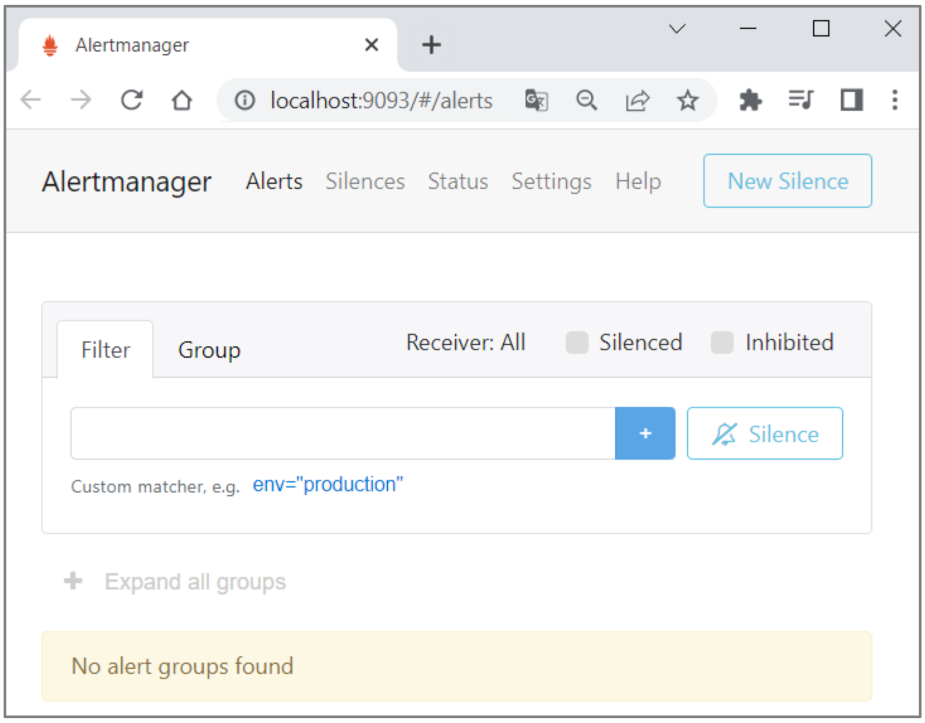
Now, start Alertmanager with the configuration file:



Windows PowerShell

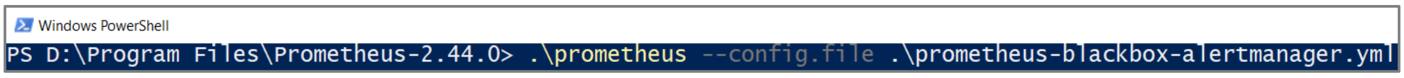
```
PS D:\Program Files\Alertmanager-0.25.0> .\alertmanager --config.file .\alertmanager-blackbox.yml
```

You can access it on `http://localhost:9093`:



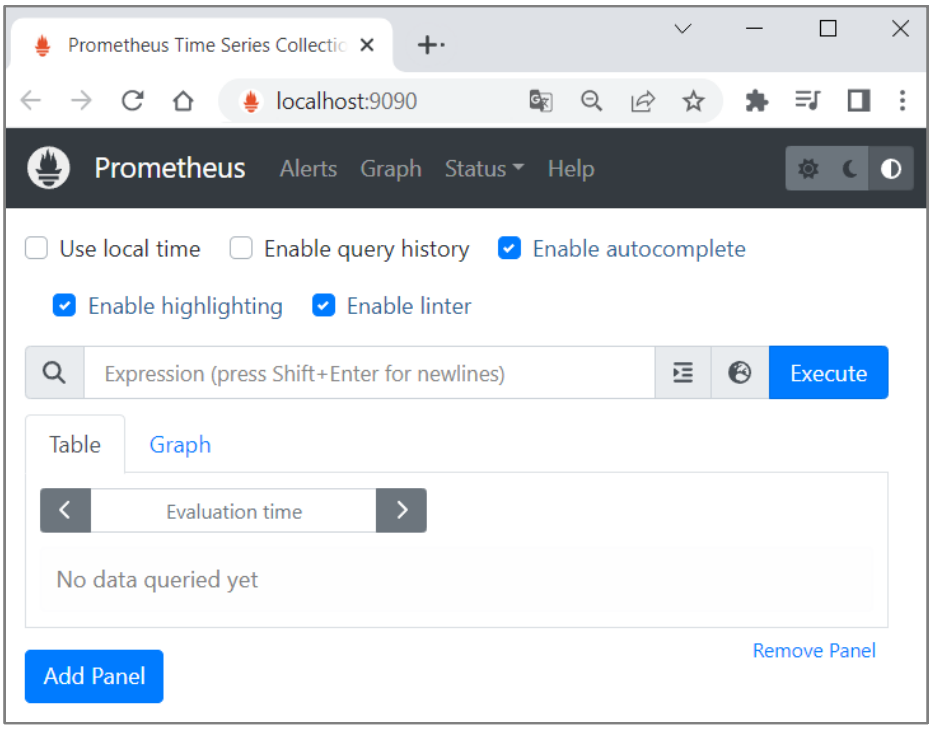
No alert groups found

Next, start Prometheus, too:



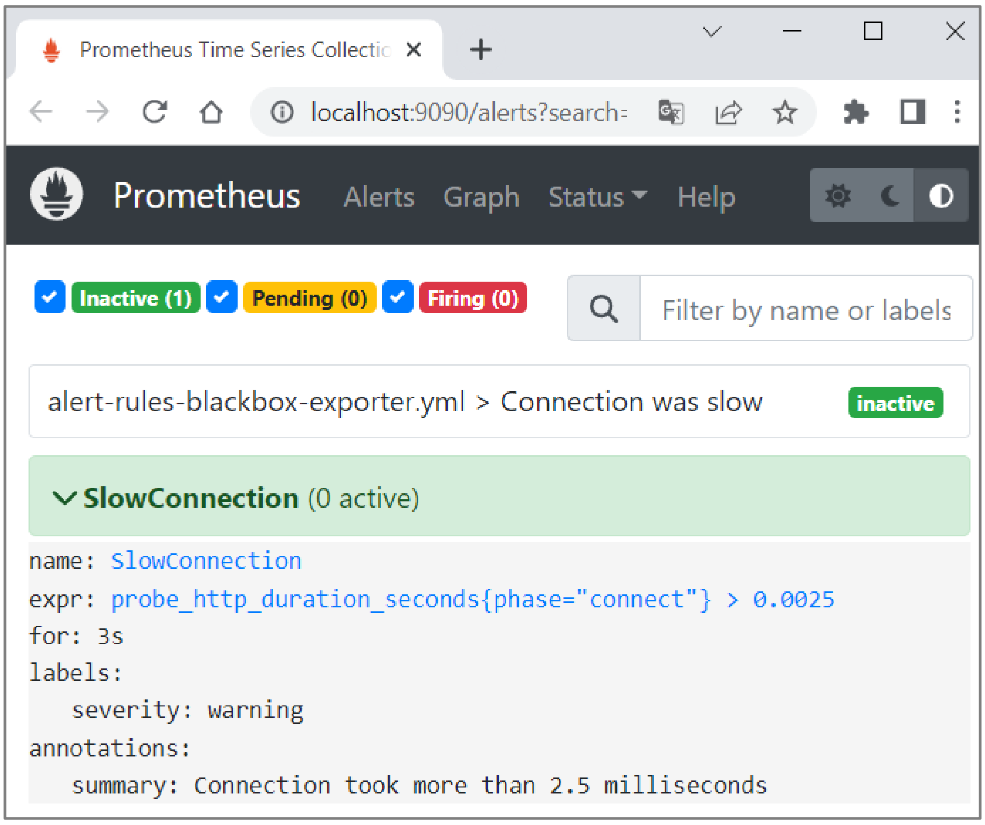
```
PS D:\Program Files\Prometheus-2.44.0> .\prometheus --config.file .\prometheus-blackbox-alertmanager.yml
```

You can access it on **http://localhost:9090**:



Step 6: Fire and Examine Alert in Prometheus

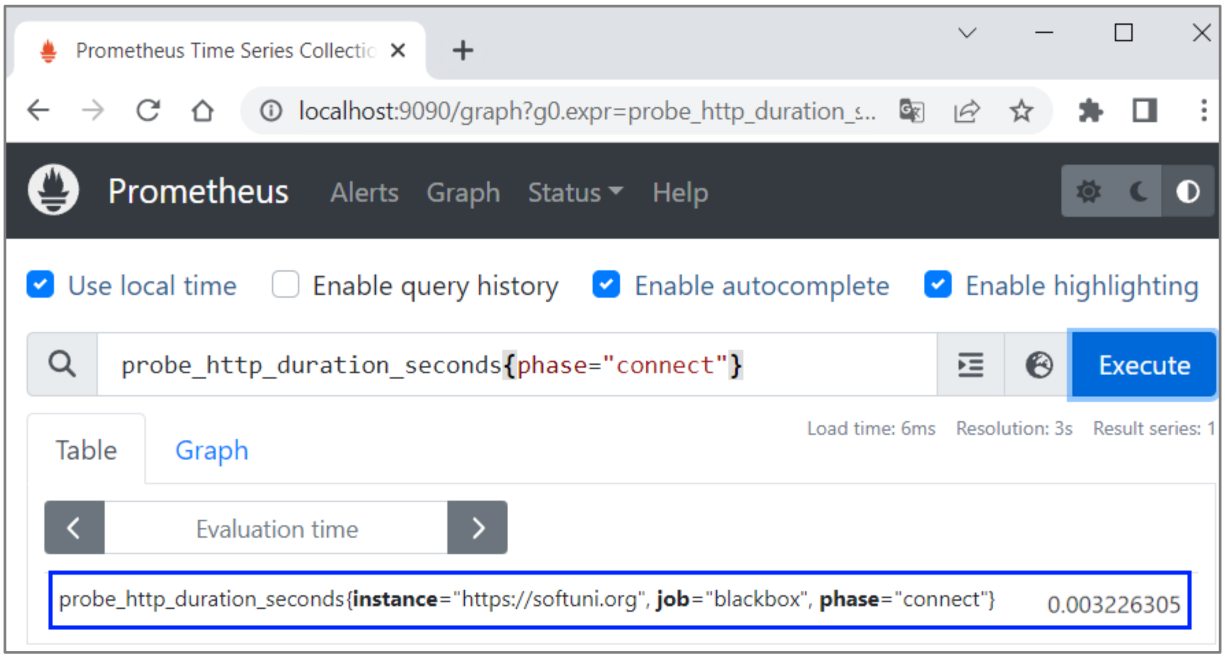
Navigate to **`[Alerts]`** in **`Prometheus`** and look at the inactive alert:



Prometheus Time Series Collectic
localhost:9090/alerts?search=
Prometheus
Alerts
Graph
Status
Help
Inactive (1)
Pending (0)
Firing (0)
Filter by name or labels
alert-rules-blackbox-exporter.yml > Connection was slow
inactive
SlowConnection (0 active)

```
name: SlowConnection
expr: probe_http_duration_seconds{phase="connect"} > 0.0025
for: 3s
labels:
  severity: warning
annotations:
  summary: Connection took more than 2.5 milliseconds
```

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


Prometheus Time Series Collectic
localhost:9090/graph?g0.expr=probe_http_duration_s...
Prometheus
Alerts
Graph
Status
Help
Table
Graph
Evaluation time
Execute

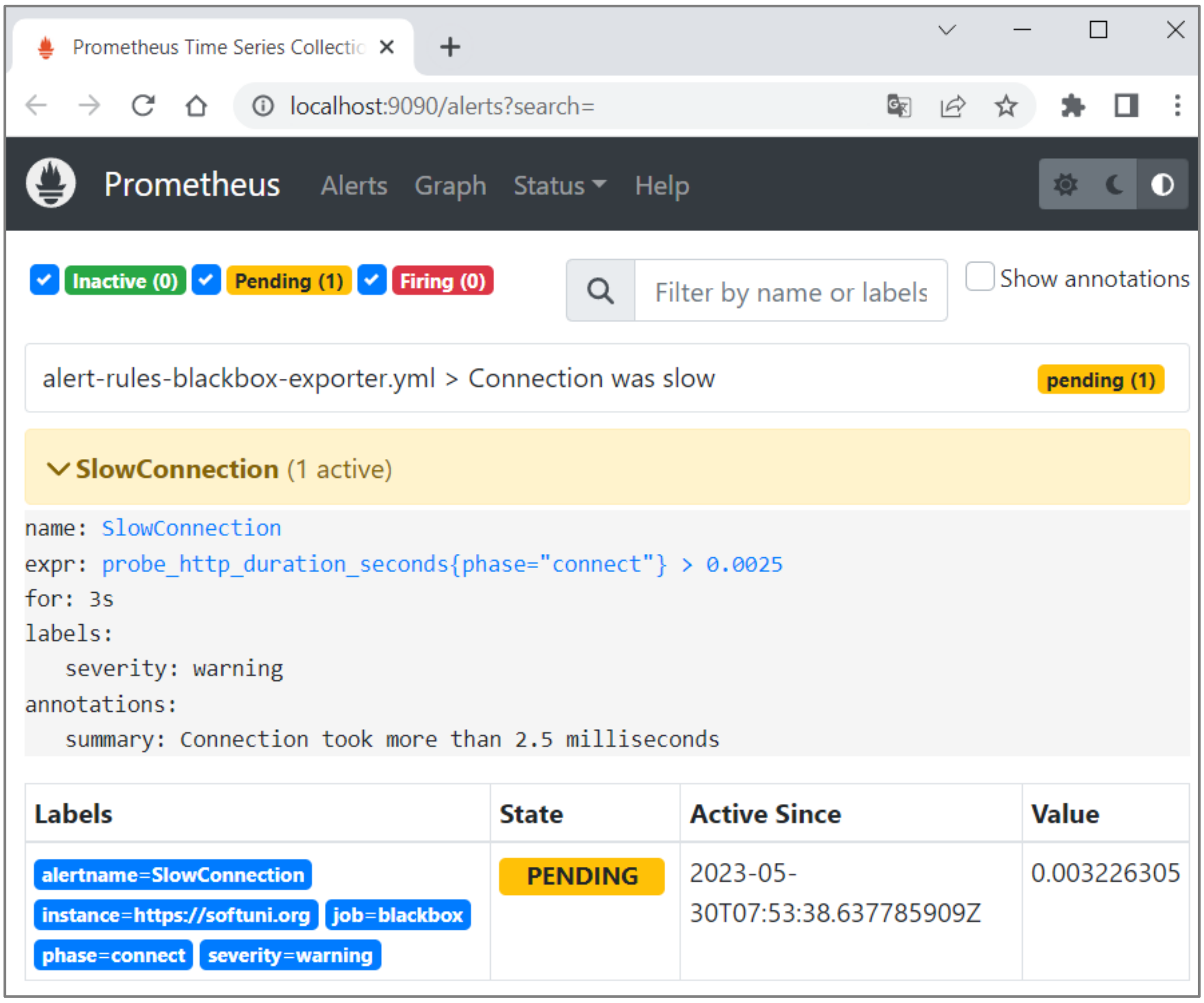
Use local time [✓] · Enable query history [ ] · Enable autocomplete [✓] · Enable highlighting [✓]

```
probe_http_duration_seconds{phase="connect"}
```

Load time: 6ms Resolution: 3s Result series: 1

```
probe_http_duration_seconds{instance="https://softuni.org", job="blackbox", phase="connect"}    0.003226305
```

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



Inactive (0) [✓] · Pending (1) [✓] · Firing (0) [✓]

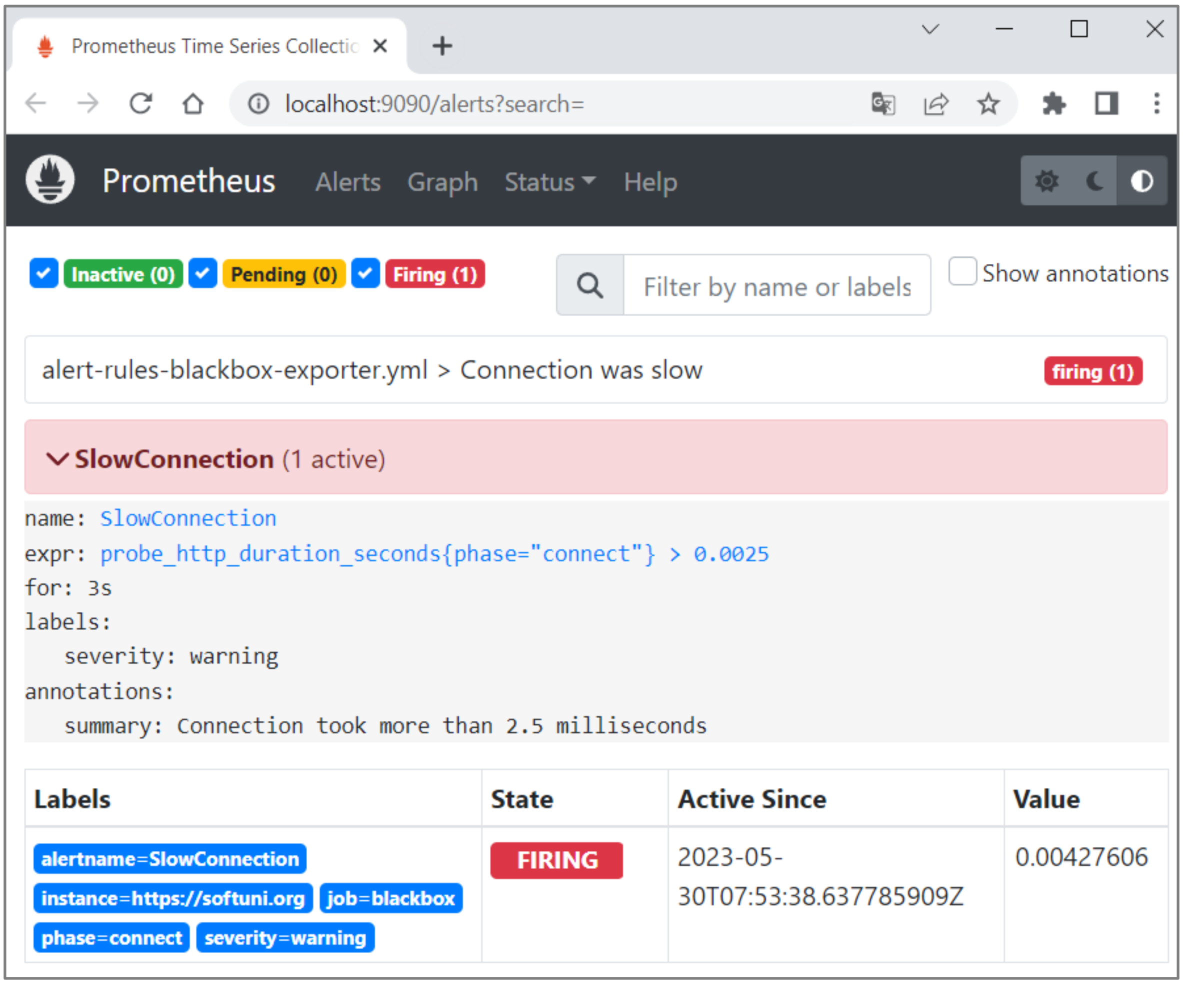
alert-rules-blackbox-exporter.yml > Connection was slow pending (1)

SlowConnection (1 active)

```
name: SlowConnection
expr: probe_http_duration_seconds{phase="connect"} > 0.0025
for: 3s
labels:
   severity: warning
annotations:
   summary: Connection took more than 2.5 milliseconds
```

Labels	State	Active Since	Value
alertname=SlowConnection instance=https://softuni.org job=blackbox phase=connect severity=warning	PENDING	2023-05-30T07:53:38.637785909Z	0.003226305

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



alert-rules-blackbox-exporter.yml > Connection was slow firing (1)

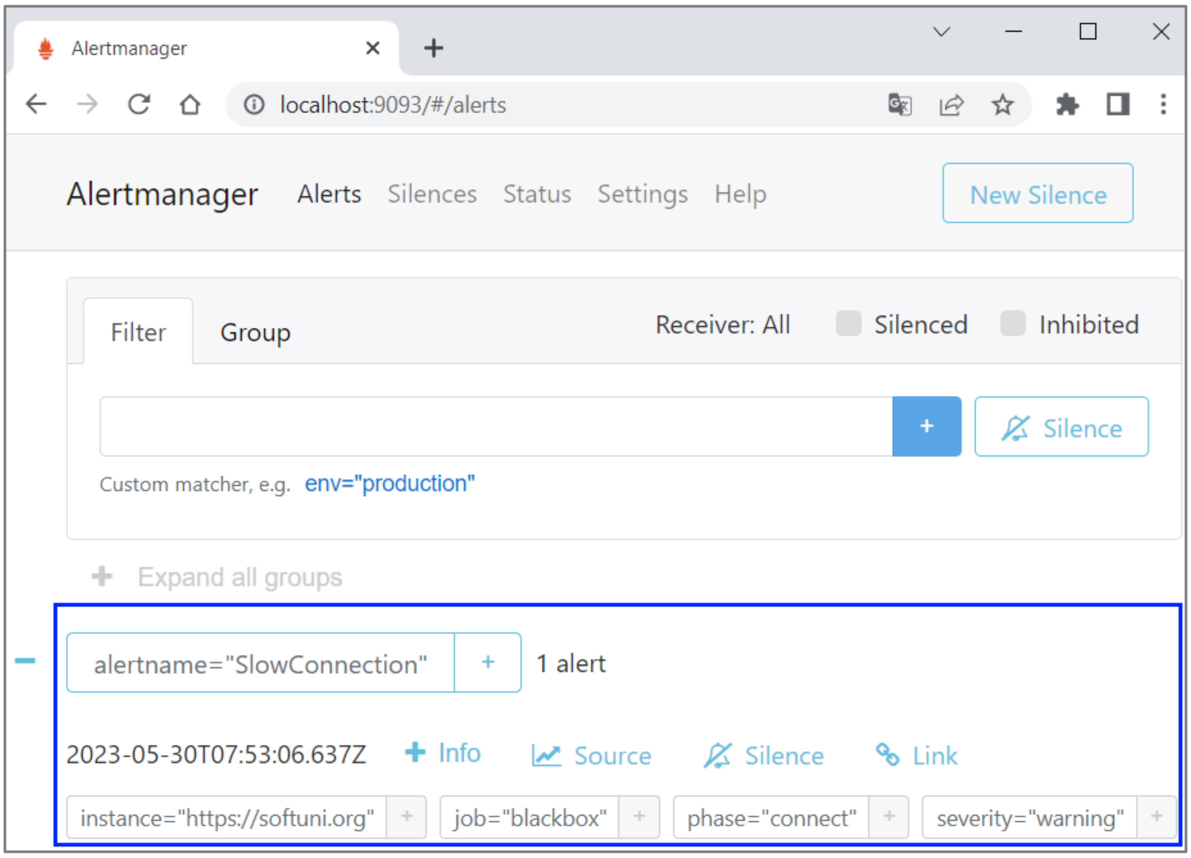
SlowConnection (1 active)

```
name: SlowConnection
expr: probe_http_duration_seconds{phase="connect"} > 0.0025
for: 3s
labels:
   severity: warning
annotations:
   summary: Connection took more than 2.5 milliseconds
```

Labels	State	Active Since	Value
alertname=SlowConnection instance=https://softuni.org job=blackbox phase=connect severity=warning	FIRING	2023-05-30T07:53:38.637785909Z	0.00427606

Step 7: Examine Alert in Alertmanager and Webhook

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

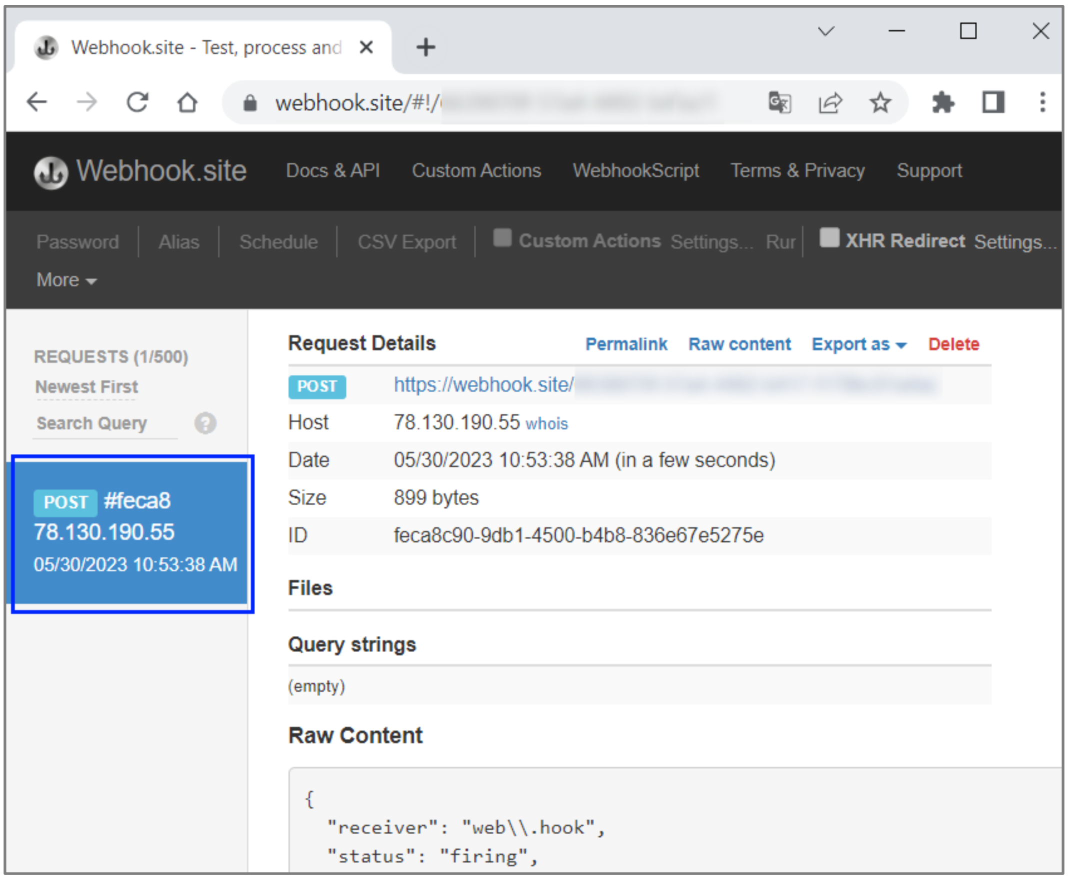


alertname="SlowConnection" 1 alert

2023-05-30T07:53:06.637Z

instance="https://softuni.org" job="blackbox" phase="connect" severity="warning"

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



POST #feca8
78.130.190.55
05/30/2023 10:53:38 AM

Request Details

POST	https://webhook.site/[redacted]
Host	78.130.190.55 whois
Date	05/30/2023 10:53:38 AM (in a few seconds)
Size	899 bytes
ID	feca8c90-9db1-4500-b4b8-836e67e5275e

Files

Query strings

(empty)

Raw Content

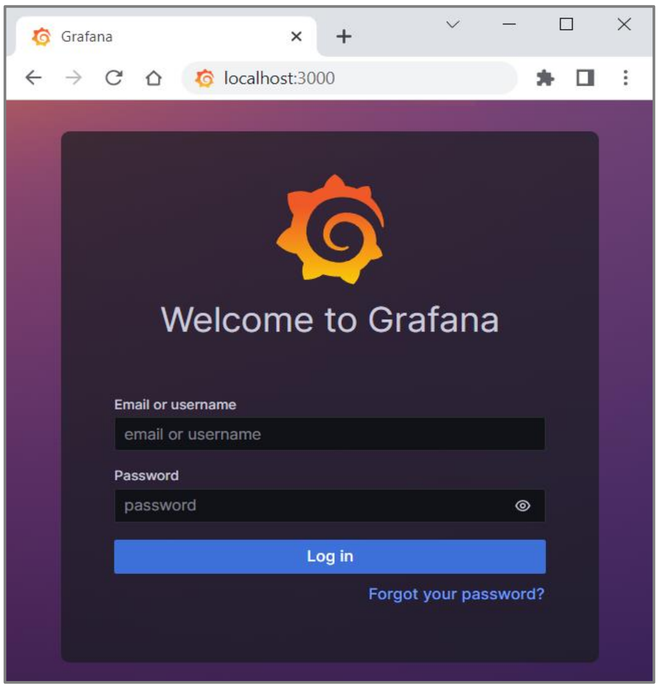
```
{
  "receiver": "web\\.hook",
  "status": "firing",
```

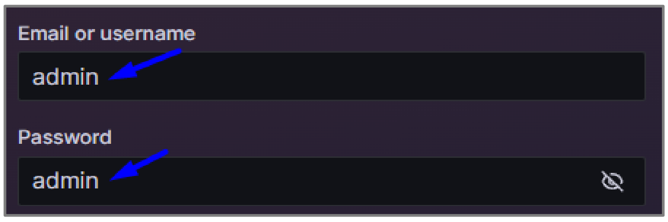
3. Grafana and Prometheus

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:

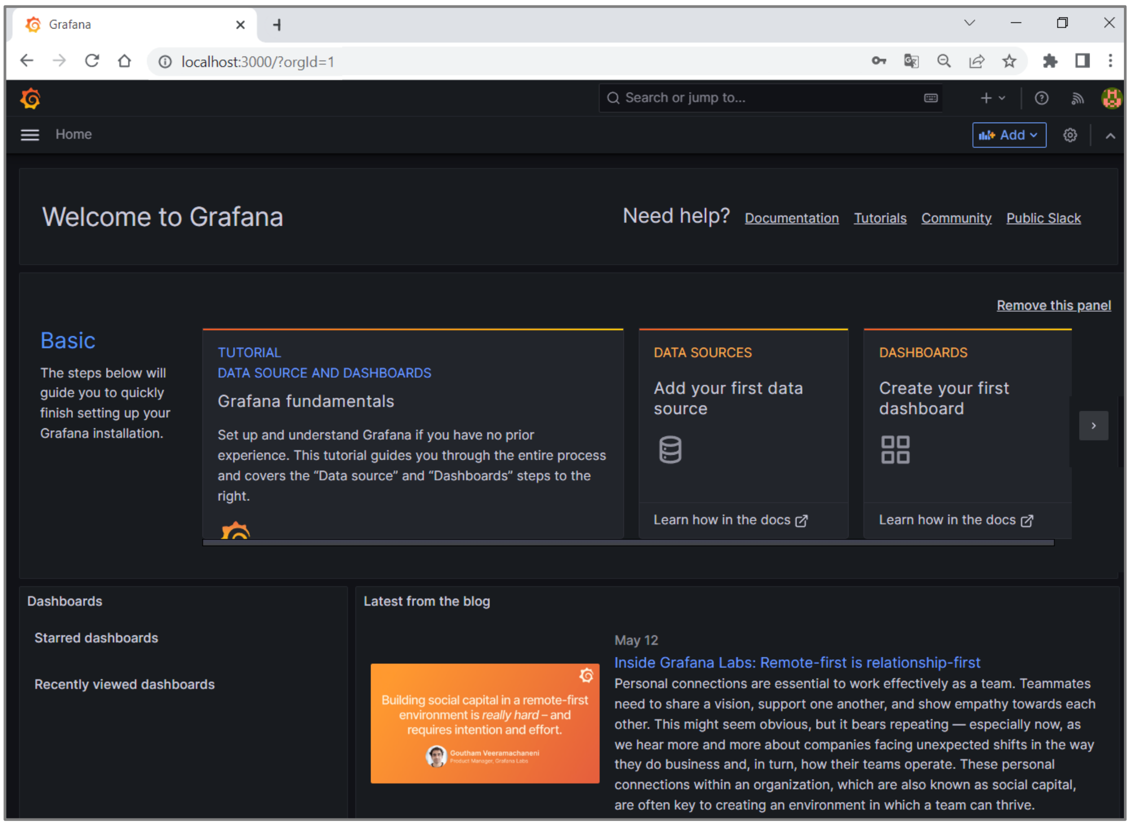
- Email or username: **`admin`**
- Password: **`admin`**




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:



Welcome to Grafana

Need help? Documentation Tutorials Community Public Slack

Remove this panel

Basic

The steps below will guide you to quickly finish setting up your Grafana installation.

TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals

Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.

DATA SOURCES
Add your first data source
Learn how in the docs

DASHBOARDS
Create your first dashboard
Learn how in the docs

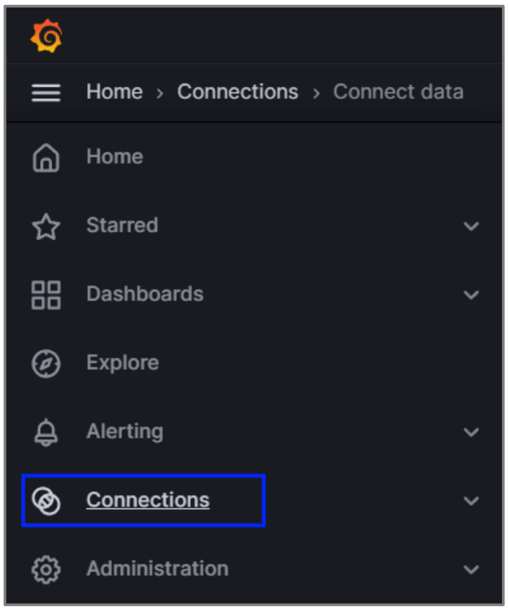
Dashboards
Starred dashboards
Recently viewed dashboards

Latest from the blog

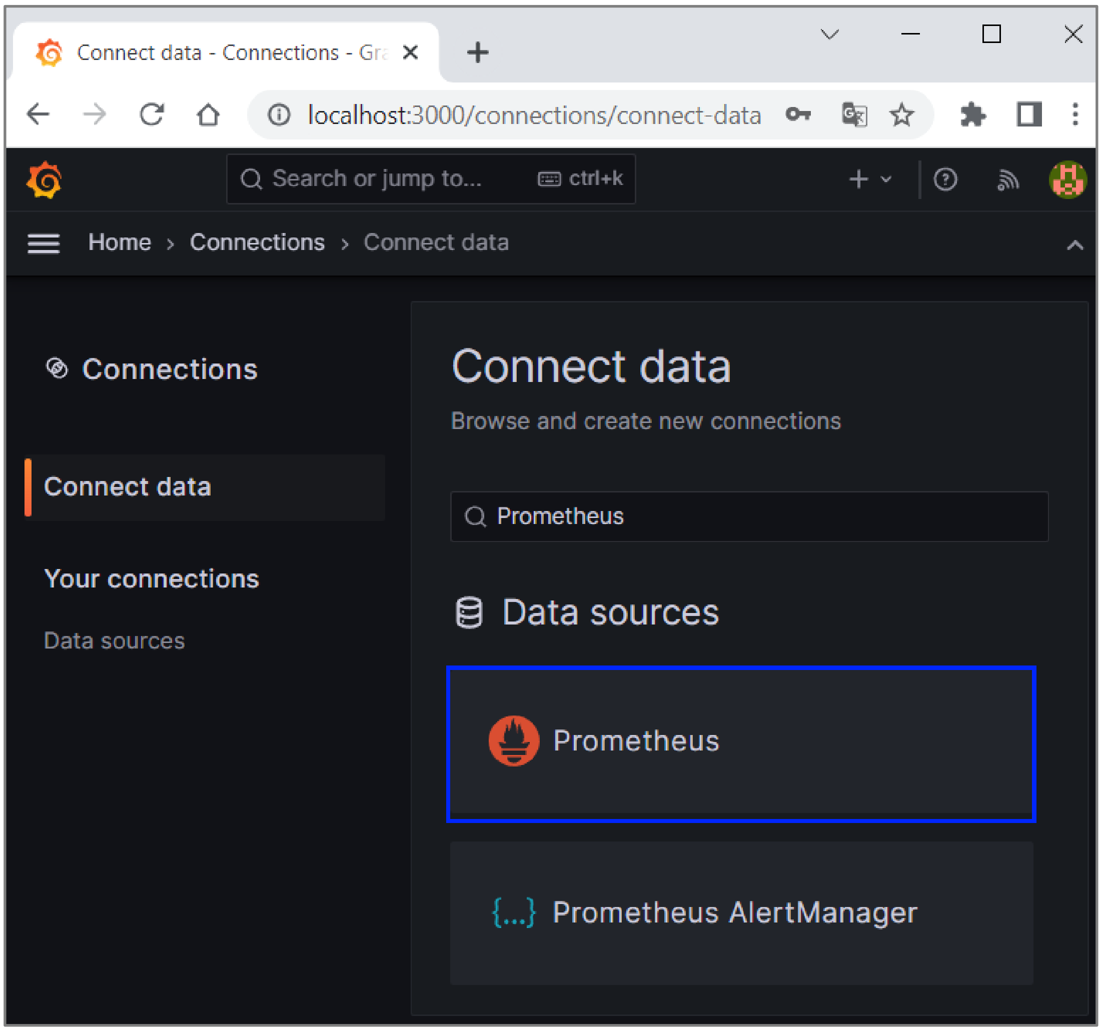
May 12
Inside Grafana Labs: Remote-first is relationship-first
Personal connections are essential to work effectively as a team. Teammates need to share a vision, support one another, and show empathy towards each other. This might seem obvious, but it bears repeating — especially now, as we hear more and more about companies facing unexpected shifts in the way they do business and, in turn, how their teams operate. These personal connections within an organization, which are also known as social capital, are often key to creating an environment in which a team can thrive.

Step 2: Create Prometheus Data Source in Grafana

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

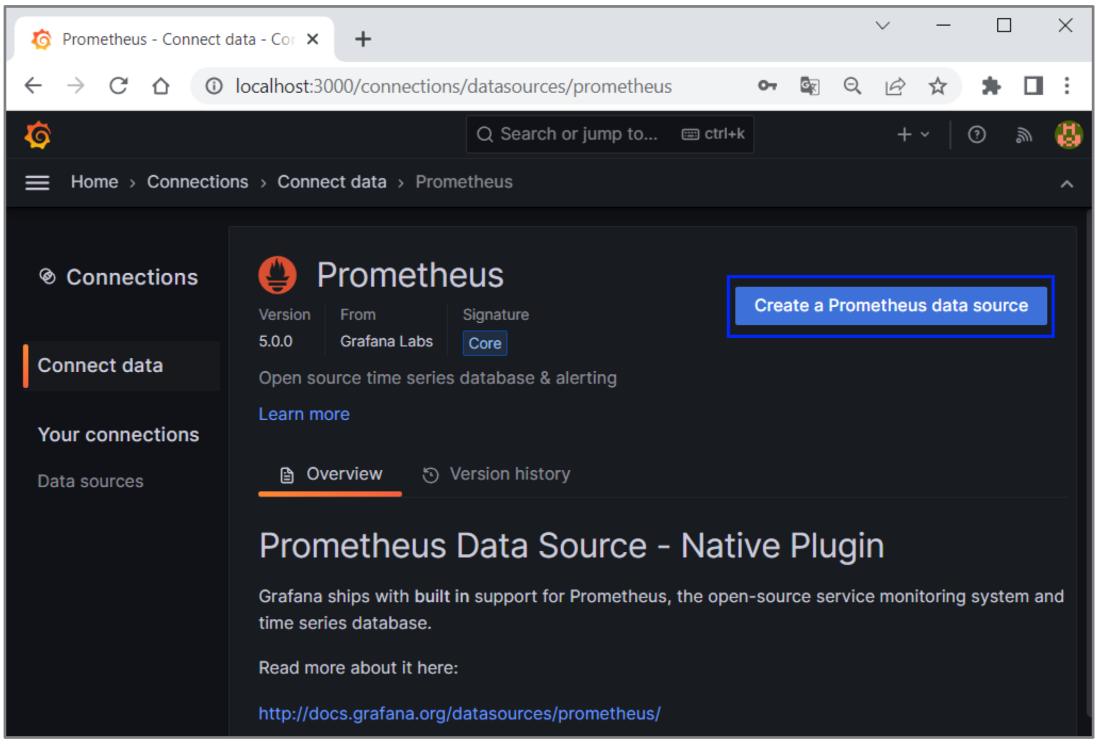


Then, search for "**`Prometheus`**" data source and click on it:



Step 3: Configure Prometheus Data Source

Now, let's create a Prometheus data source:


Prometheus - Connect data - Co
localhost:3000/connections/datasources/prometheus
Search or jump to...
ctrl+k
Home › Connections › Connect data › Prometheus
Connections
Connect data
Your connections
Data sources
Create a Prometheus data source
Learn more
Overview
Version history

Prometheus

Version	From	Signature
5.0.0	Grafana Labs	Core

Open source time series database & alerting

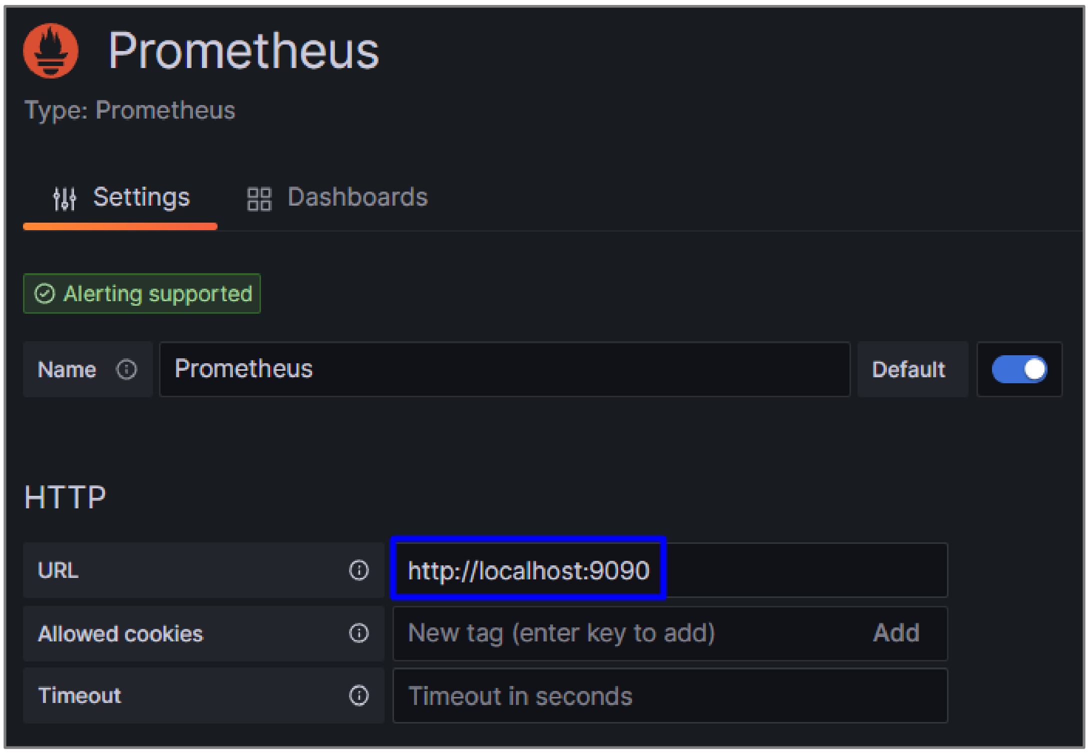
Prometheus Data Source - Native Plugin

Grafana ships with built in support for Prometheus, the open-source service monitoring system and time series database.

Read more about it here:

http://docs.grafana.org/datasources/prometheus/

We have to set the Prometheus server URL:

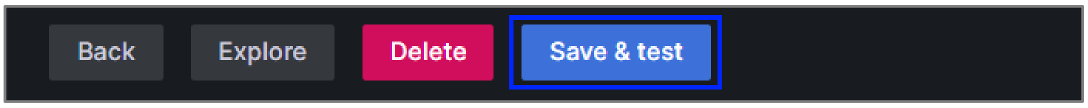

Prometheus
Type: Prometheus

Name: Prometheus
Default: on

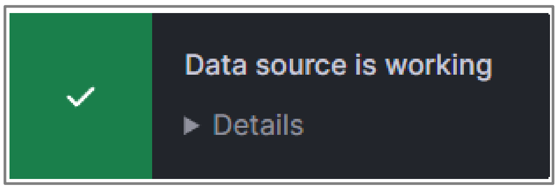
HTTP

URL: http://localhost:9090
Allowed cookies: New tag (enter key to add) Add
Timeout: Timeout in seconds

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

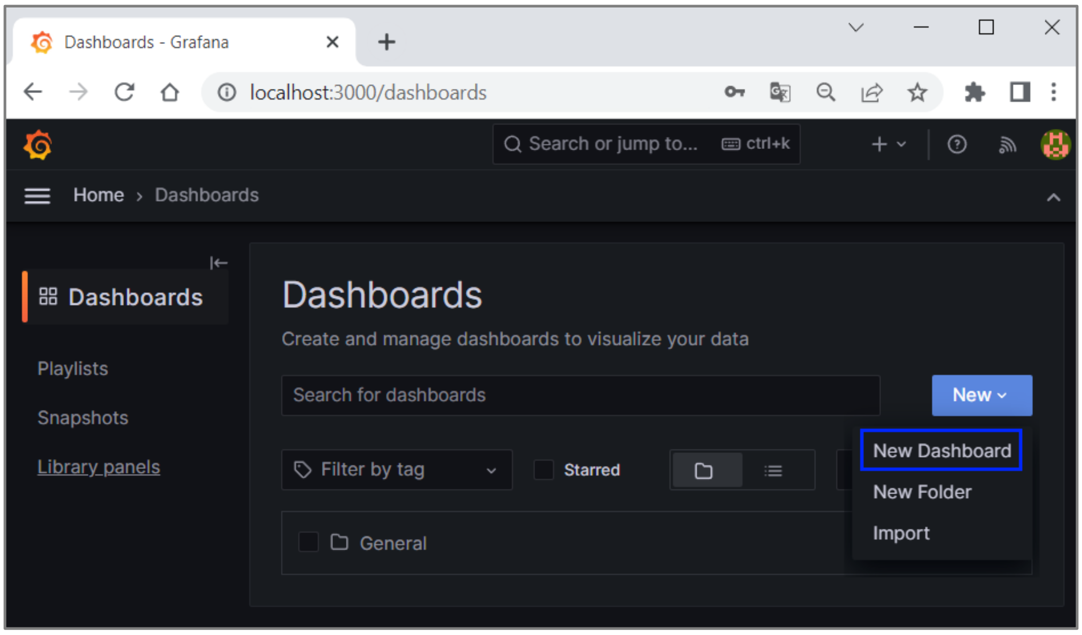


You should see the success message in a few moments:

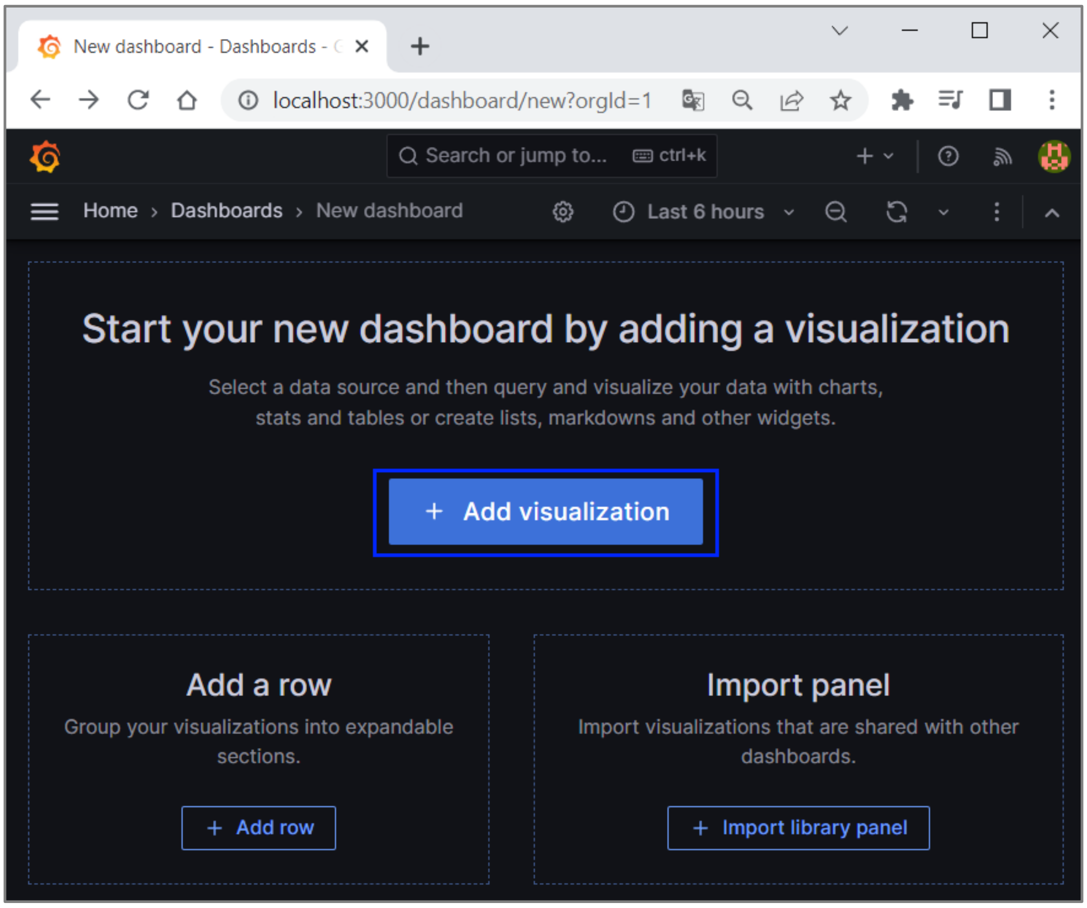

Data source is working

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**]:

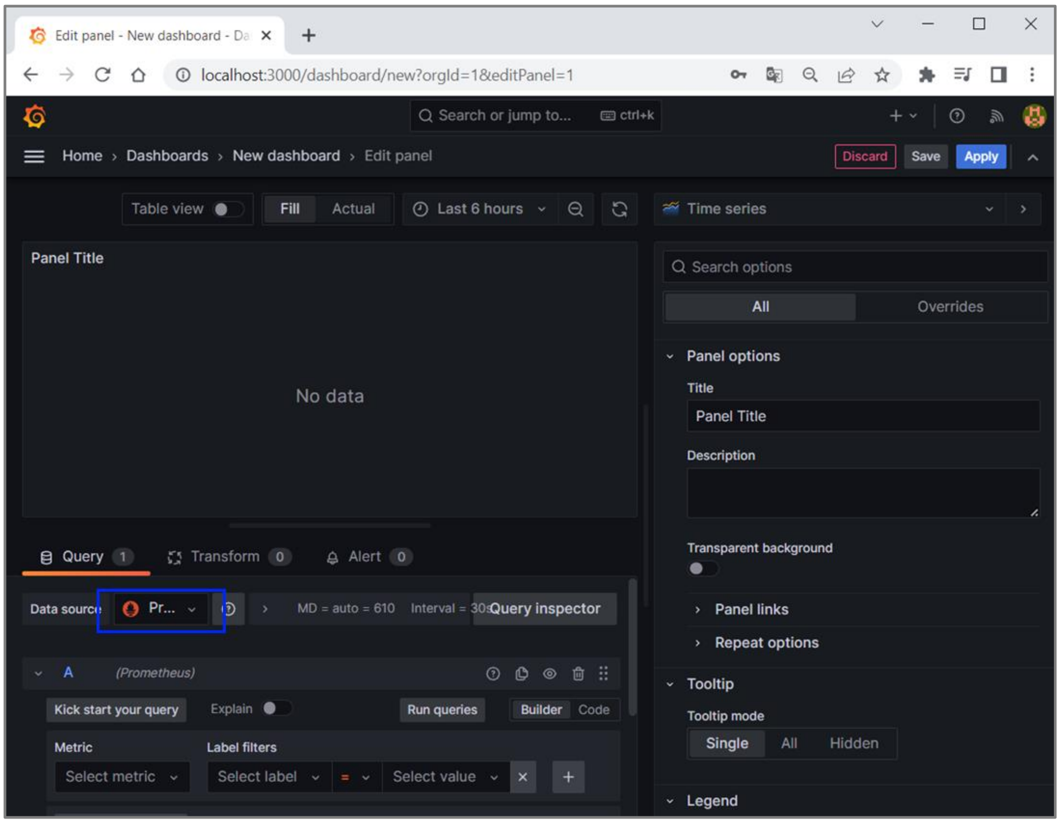


Now, click on the [**`+ Add visualization`**] button to create a visualization (panel):

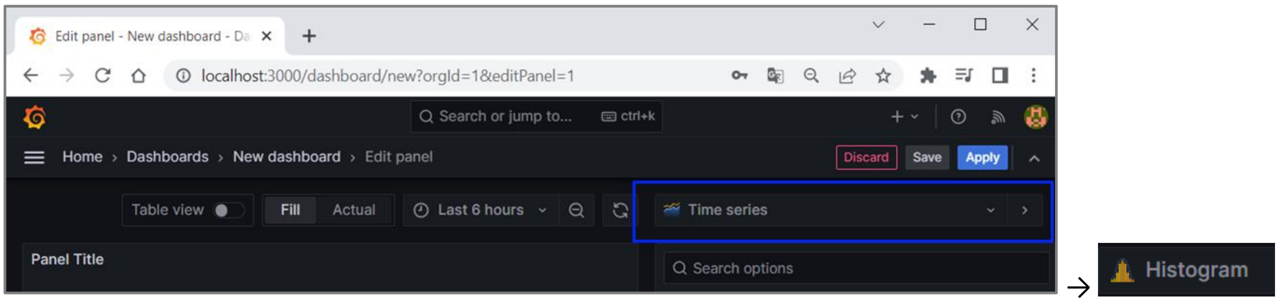

New dashboard - Dashboards - G
localhost:3000/dashboard/new?orgId=1
Search or jump to...
ctrl+k
Home › Dashboards › New dashboard
Last 6 hours
Start your new dashboard by adding a visualization
Select a data source and then query and visualize your data with charts, stats and tables or create lists, markdowns and other widgets.
+ Add visualization
Add a row
Group your visualizations into expandable sections.
+ Add row
Import panel
Import visualizations that are shared with other dashboards.
+ Import library panel

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

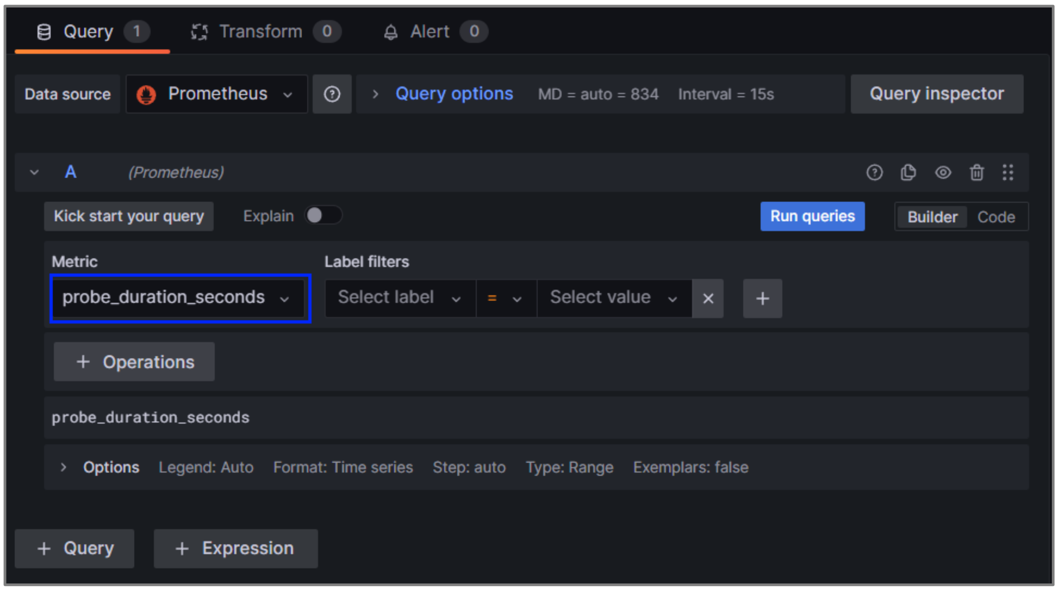
First, select the **`Prometheus`** data source from the Data source dropdown menu:



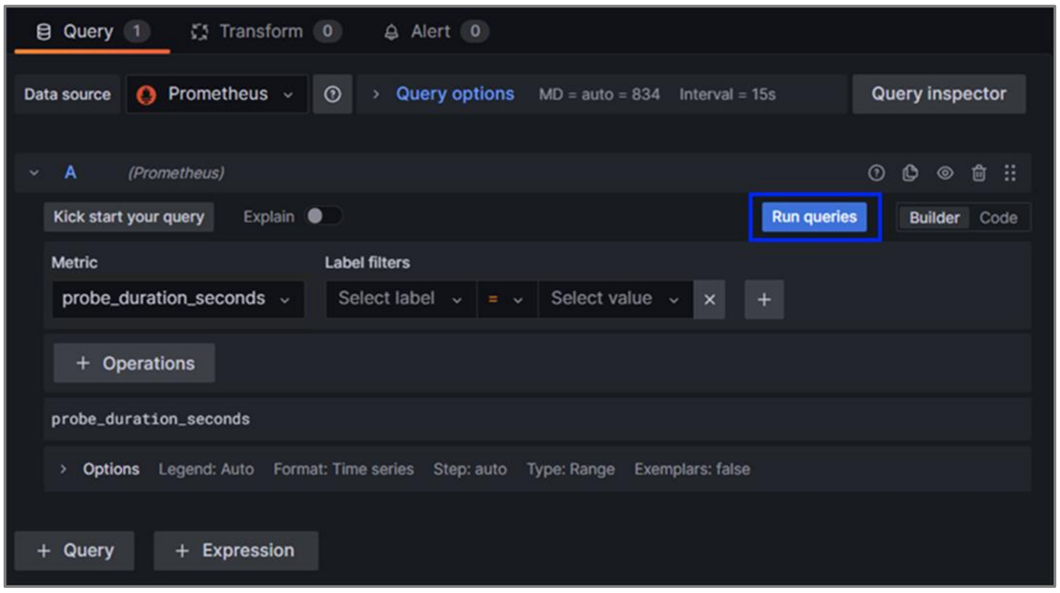
Next, click on the **`Visualization`** menu and select **`Histogram`**:


Edit panel - New dashboard - Da
localhost:3000/dashboard/new?orgId=1&editPanel=1
Search or jump to... ctrl+k
Home › Dashboards › New dashboard › Edit panel
Discard
Save
Apply
Table view
Fill
Actual
Last 6 hours
Time series
Panel Title
Search options
Histogram

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



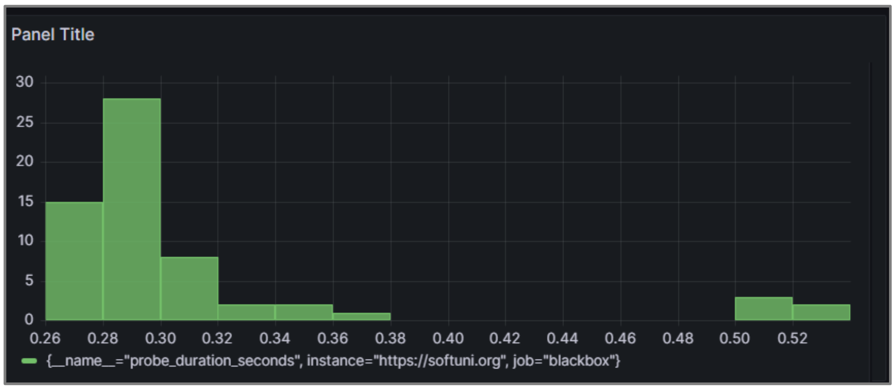
After that, click on the [**`Run queries`**] button:



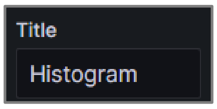
```
probe_duration_seconds
```

Options Legend: Auto Format: Time series Step: auto Type: Range Exemplars: false

And you should be able to see the panel:



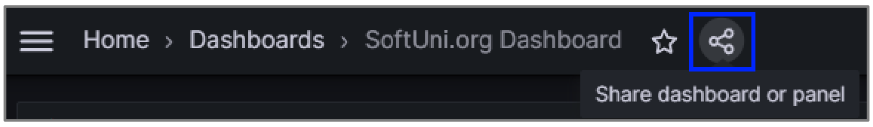
In order to change the panel title, click on the "**`Title`**" input field in the **`Panel options`** section:



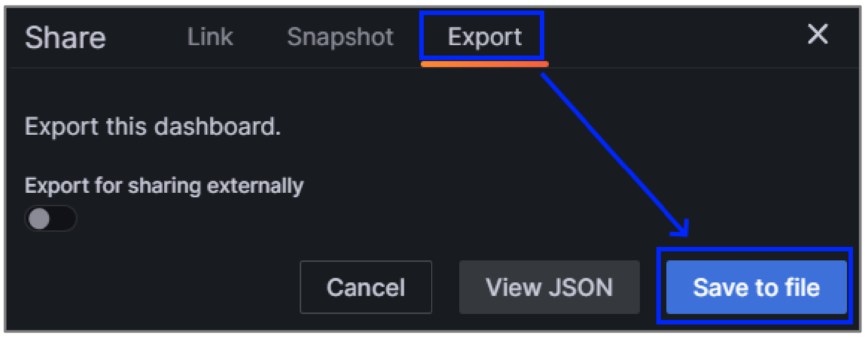
Step 5: Import and Export Dashboards

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

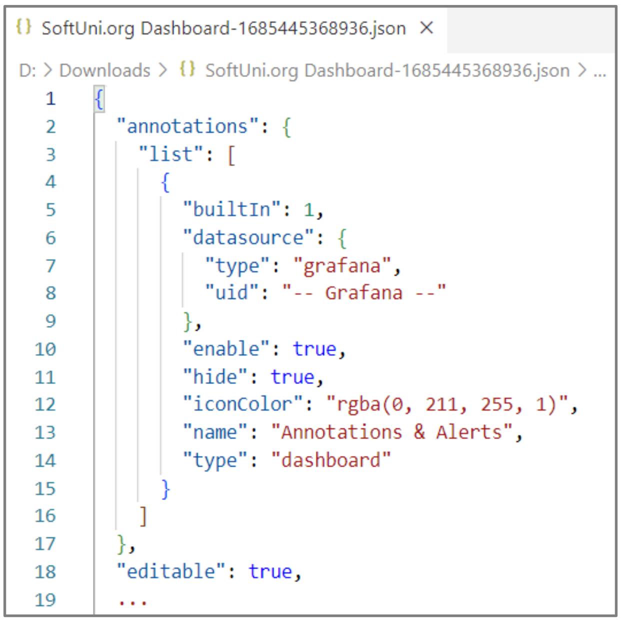
In order to **export a dashboard, open `Dashboards`** and then **click on [`Share`]:**



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

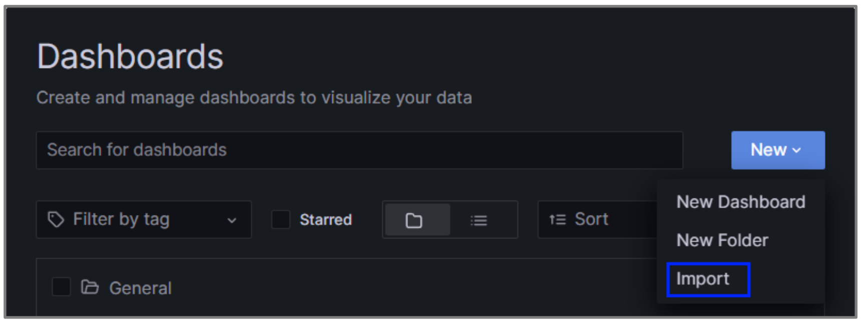


You should be able to **save** the dashboard as a **JSON** file

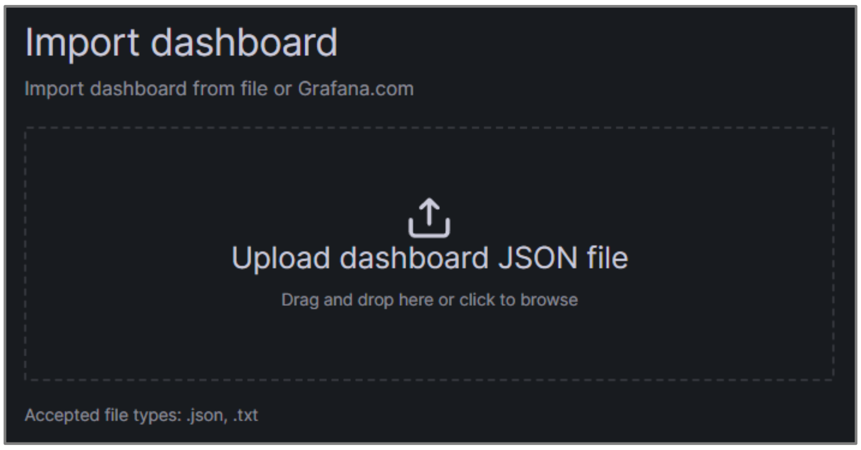

SoftUni.org Dashboard-1685445368936.json
D: > Downloads > SoftUni.org Dashboard-1685445368936.json > ...

```
{
  "annotations": {
    "list": [
      {
        "builtIn": 1,
        "datasource": {
          "type": "grafana",
          "uid": "-- Grafana --"
        },
        "enable": true,
        "hide": true,
        "iconColor": "rgba(0, 211, 255, 1)",
        "name": "Annotations & Alerts",
        "type": "dashboard"
      }
    ]
  },
  "editable": true,
  ...
```

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:


Import dashboard
Import dashboard from file or Grafana.com
Upload dashboard JSON file
Drag and drop here or click to browse
Accepted file types: .json, .txt