

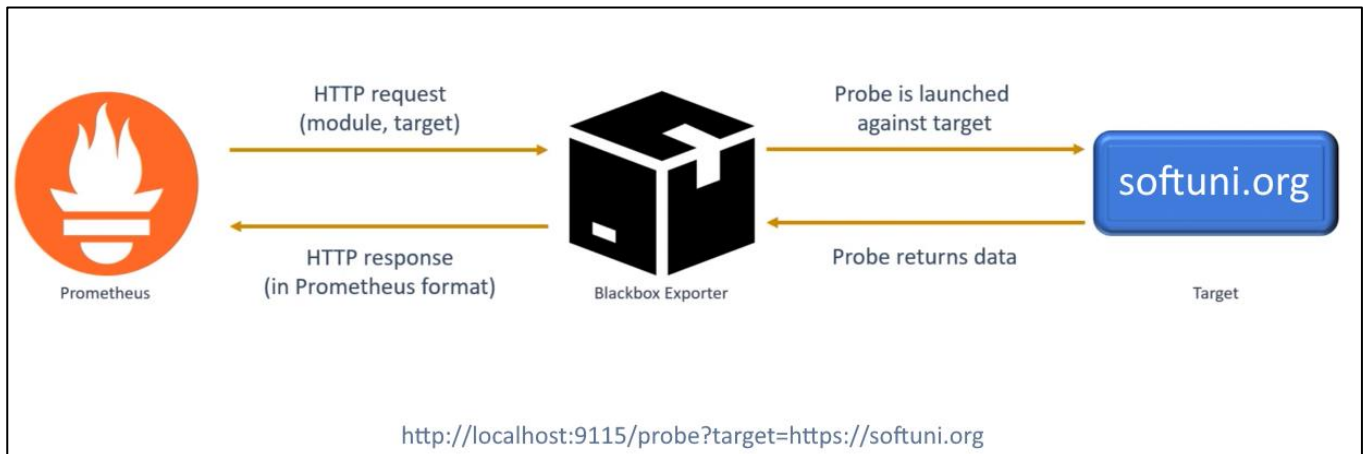
# Lab: App Monitoring

Lab assignment for the ["Containers and Clouds" course @ SoftUni](#).

## 1. Prometheus and Blackbox Exporter

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**:

```
Windows PowerShell
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:

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

# 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:

The screenshot shows the SoftUni Global website in a web browser. The Chrome DevTools Network panel is open, displaying a list of requests. The 'softuni.org' request is selected, and the 'Explanation' tab is active, showing a waterfall chart of the request phases. The total duration of the request is highlighted as 230.87 ms.

Phase	Duration
Queued at 0	
Started at 1.80 ms	
Resource Scheduling	
Queueing	1.80 ms
Connection Start	
Stalled	1.05 ms
DNS Lookup	38 µs
Initial connection	14.61 ms
SSL	14.61 ms
Request/Response	
Request sent	0.20 ms
Waiting for server response	130.20 ms
Content Download	82.41 ms
<b>Total</b>	<b>230.87 ms</b>

The screenshot shows the localhost:9115/probe?target=https://softuni.org endpoint. The page displays the output of the Blackbox Exporter metrics, including the total duration of the request in seconds.

```
# 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
1  global:
2    scrape_interval: 15s
3
4  scrape_configs:
5    - job_name: 'blackbox'
6      metrics_path: /probe
7      static_configs:
8        - targets:
9          - https://softuni.org
10     relabel_configs:
11       - source_labels: [__address__]
12         target_label: __param_target
13       - source_labels: [__param_target]
14         target_label: instance
15       - target_label: __address__
16         replacement: 127.0.0.1:9115
```

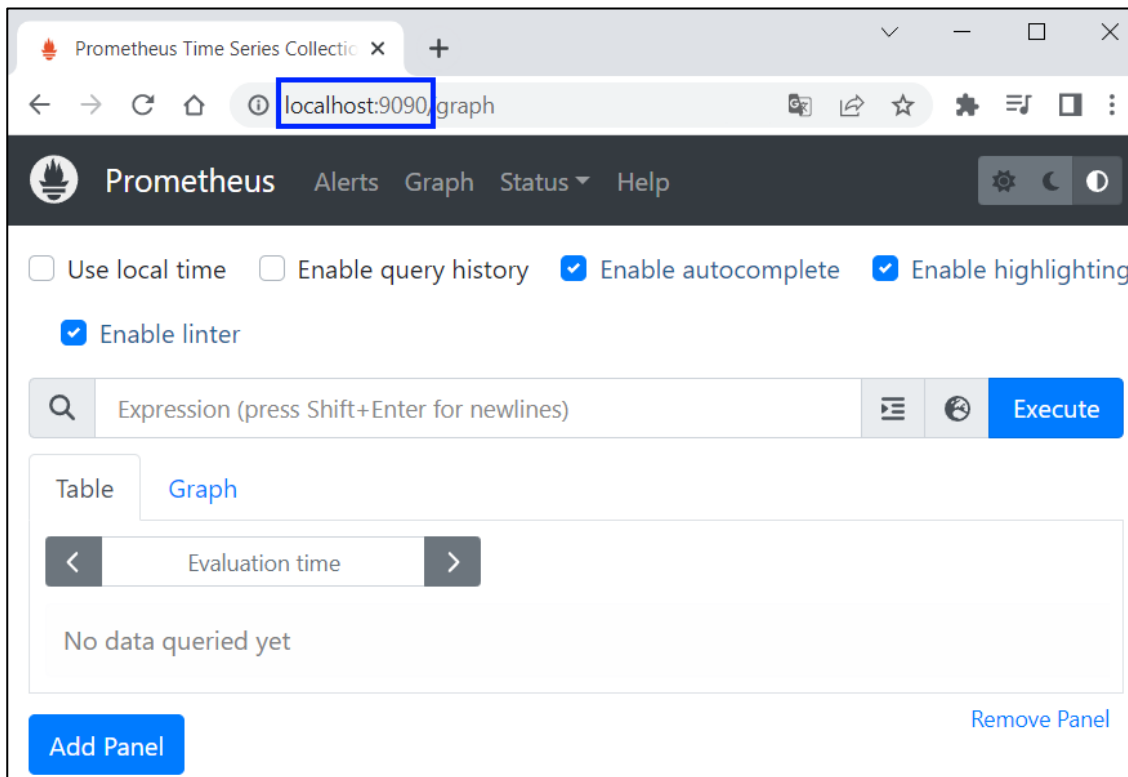
- **scrape\_interval: 15s**
  - Target is being scraped each 15 seconds
- **metrics\_path: /probe**
  - Metrics can be accessed on /probe
- **- targets:**
  - <https://softuni.org>
    - Define the targeted site URL
- **Replacement: 127.0.0.1:9115**
  - Blackbox exporter's **hostname:port**

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

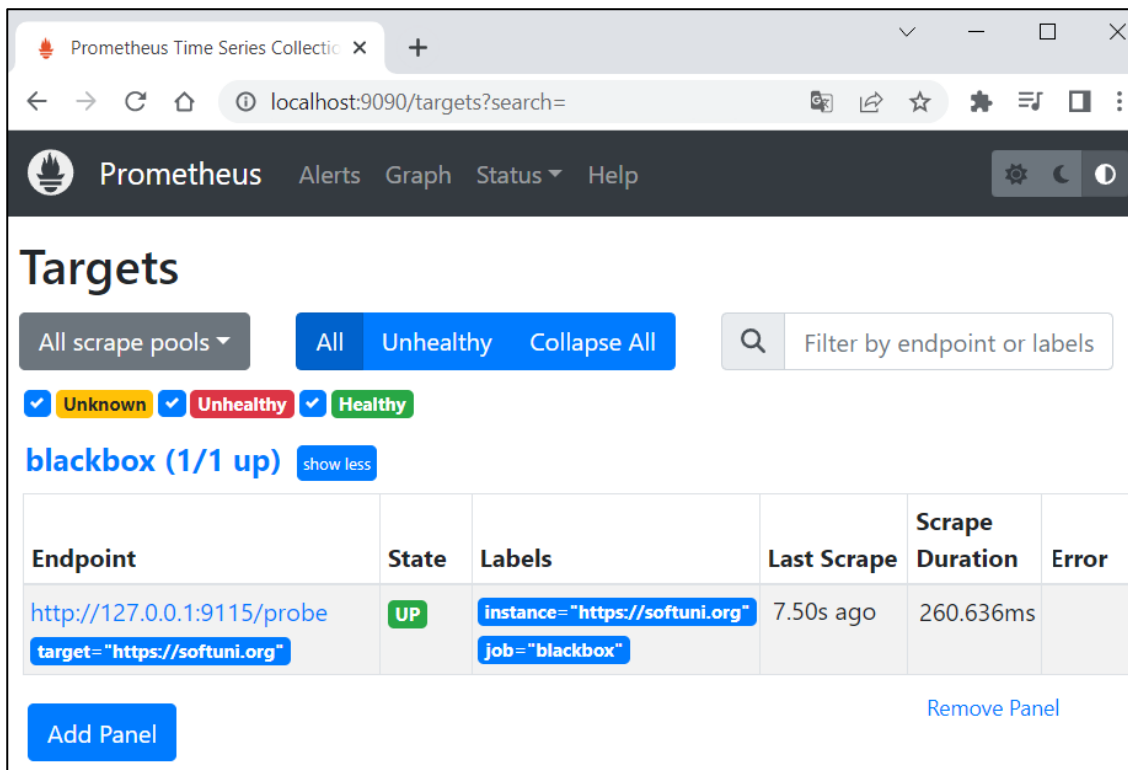
```
Windows PowerShell
PS C:\Users\PC> .\prometheus --config.file prometheus-blackbox.yaml
```

## Step 4: Access Prometheus

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



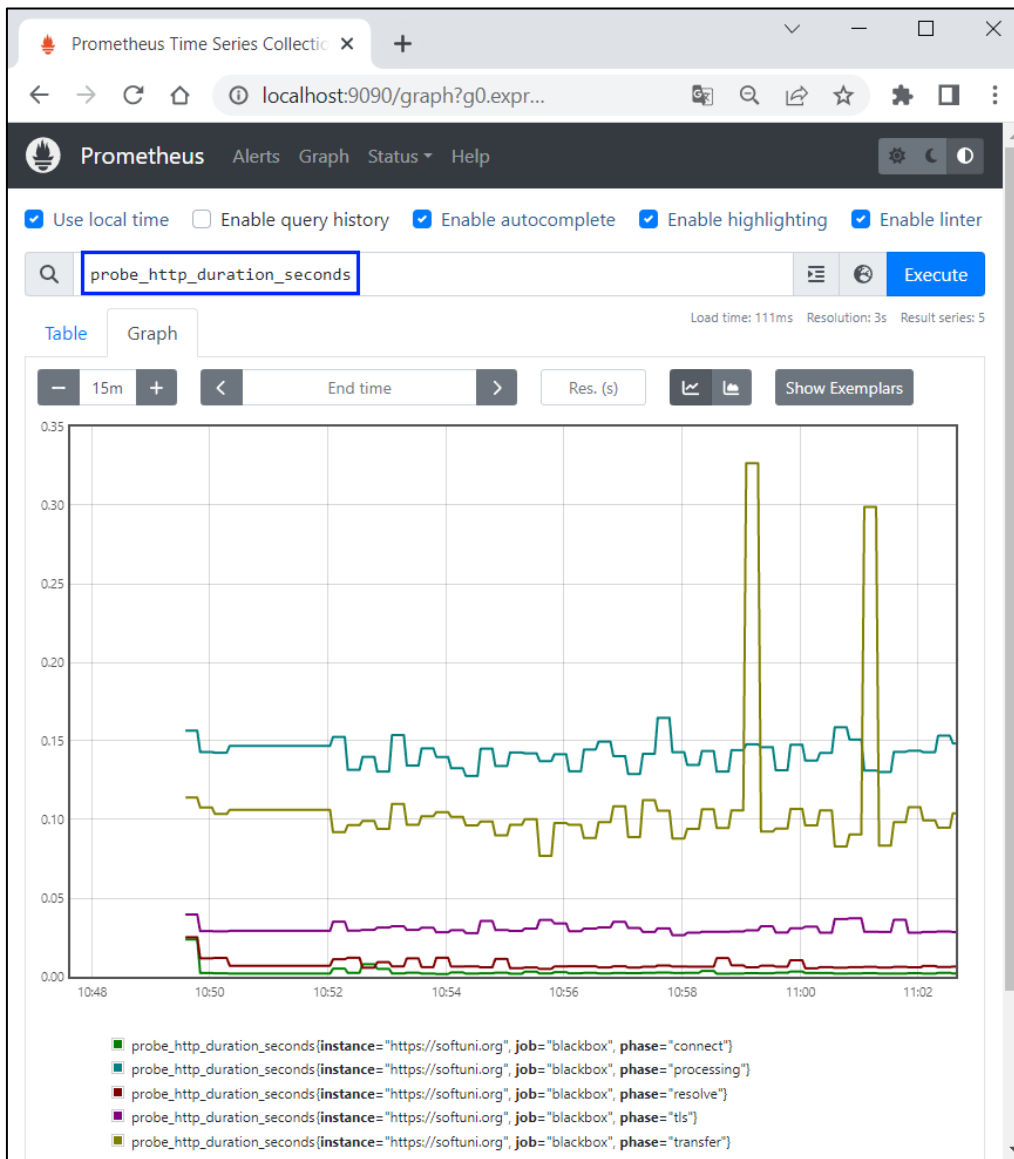
We can look at the target site status from **[Status]** → **[Targets]**:



## 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:



## 2. Prometheus and Alert Manager

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 Docs & API Custom Actions WebhookScript Terms & Privacy Support

Password Alias Schedule CSV Export Custom Actions Settings... Run Now XHR Redirect

REQUESTS (0/500)  
Newest First  
Search Query

Waiting for first request...

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!

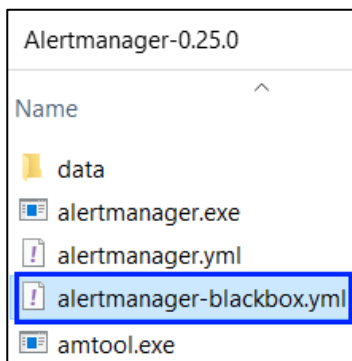
Your unique URL (Please copy it from here, *not* from the address bar!)

`https://webhook.site/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX`

Copy to clipboard Open in new tab

## Step 2: Configure Alertmanager

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



The configuration should:

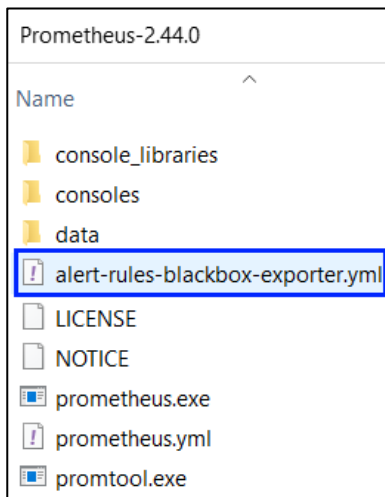
- Sets the timeout for alert resolution to 5 minutes
- Specifies that alerts are sent to the "webhook\_receiver" receiver
- Configures the "webhook\_receiver" receiver
  - It should send requests to the URL that **Webhook.site** provided

```
! alertmanager-blackbox.yml ●
D: > Program Files > Alertmanager-0.25.0 > ! alertmanager-blackbox.yml
1 route:
2   group_by: ['alertname']
3   group_wait: 30s
4   group_interval: 5m
5   repeat_interval: 1h
6   receiver: 'web.hook'
7
8 receivers:
9   - name: 'web.hook'
10     webhook_configs:
11       - url: 'https://webhook.site/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX'
```

## 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:

```
! alert-rules-blackbox-exporter.yml
D: > Program Files > Prometheus-2.44.0 > ! alert-rules-blackbox-exporter.yml
1  groups:
2    - name: Connection was slow
3      rules:
4        - alert: SlowConnection
5          expr: probe_http_duration_seconds{phase="connect"} > 0.0025
6          for: 3s
7          labels:
8            severity: warning
9          annotations:
10         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).

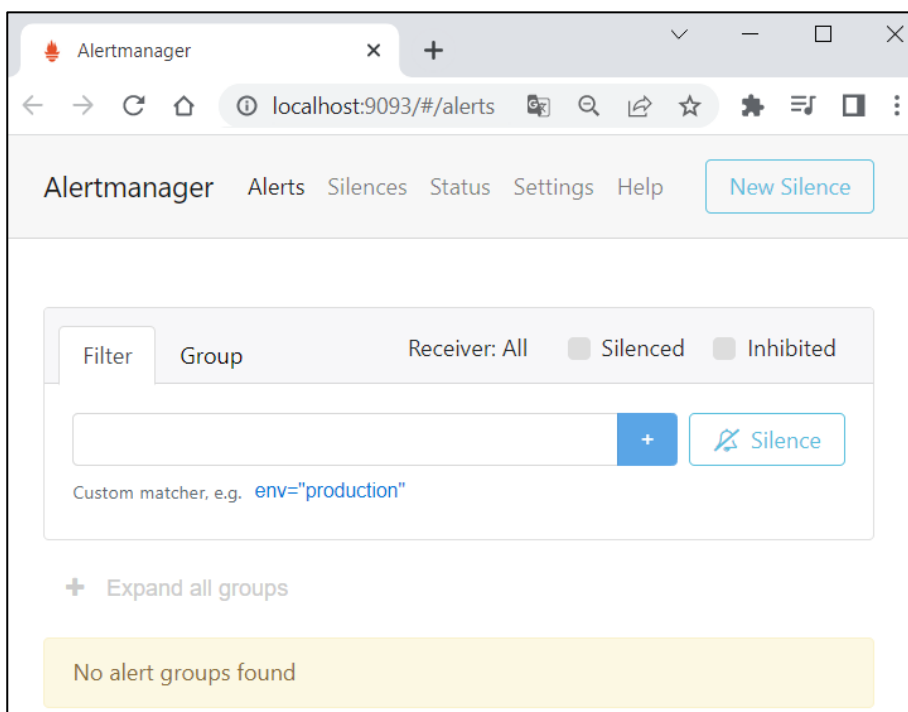
```
! prometheus-blackbox-alertmanager.yml
D: > Program Files > Prometheus-2.44.0 > ! prometheus-blackbox-alertmanager.yml
1  global:
2    scrape_interval: 15s
3    evaluation_interval: 10s
4
5  rule_files:
6    - alert-rules-blackbox-exporter.yml
7
8  alerting:
9    alertmanagers:
10   - static_configs:
11     - targets:
12       - localhost:9093
13
14  scrape_configs:
15   - job_name: 'blackbox'
16     metrics_path: /probe
17     static_configs:
18       - targets:
19         - https://softuni.org
20     relabel_configs:
21       - source_labels: [__address__]
22         target_label: __param_target
23       - source_labels: [__param_target]
24         target_label: instance
25       - target_label: __address__
26         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>:

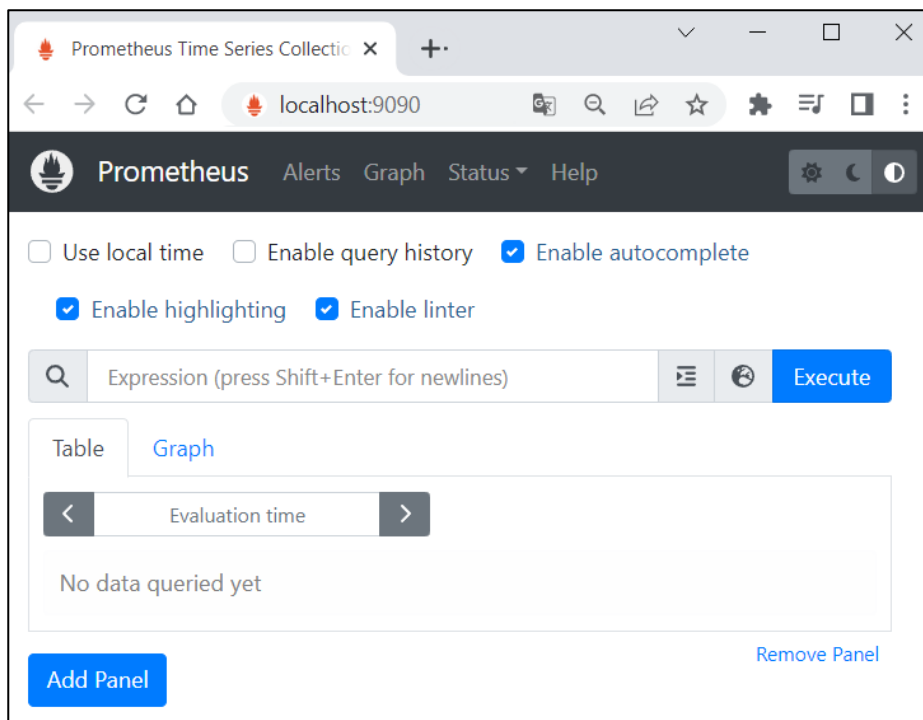




Next, start Prometheus, too:

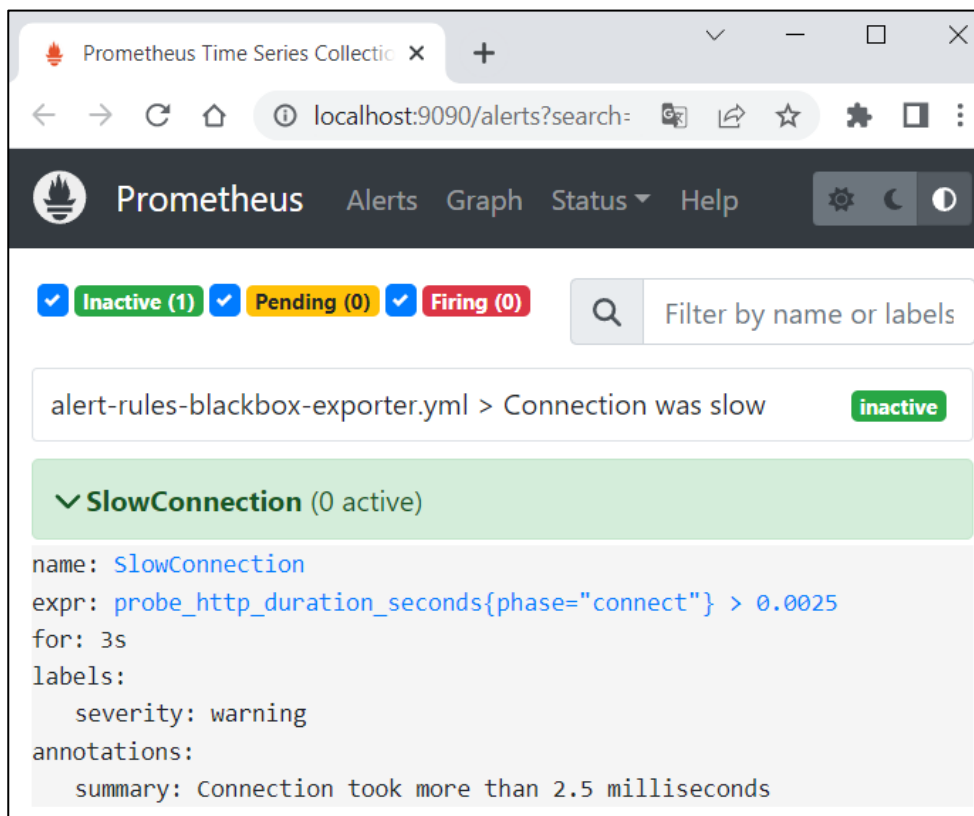
```
Windows PowerShell
PS D:\Program Files\Prometheus-2.44.0> .\prometheus --config.file .\prometheus-blackbox-alertmanager.yml
```

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:



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

The screenshot shows the Prometheus web interface at localhost:9090/graph. The query bar contains `probe_http_duration_seconds{phase="connect"}`. The 'Execute' button is highlighted. Below the query, the 'Graph' tab is selected, and the evaluation time is shown as 'Evaluation time'. The result series is displayed as a table with one row: `probe_http_duration_seconds(instance="https://softuni.org", job="blackbox", phase="connect")` with a value of `0.003226305`.

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

The screenshot shows the Prometheus web interface at localhost:9090/alerts. The alert rules are filtered by name or labels. The alert rule `alert-rules-blackbox-exporter.yml > Connection was slow` is shown in a pending state. The alert rule details are displayed below the title:

```

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

```

The alert rule is active, and the value is `0.003226305`. The alert rule is shown in a pending state.

Labels	State	Active Since	Value
<b>alertname=SlowConnection</b> <b>instance=https://softuni.org</b> <b>job=blackbox</b> <b>phase=connect</b> <b>severity=warning</b>	<b>PENDING</b>	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**:

The screenshot shows the Prometheus Alerts interface. At the top, there are filters for Inactive (0), Pending (0), and Firing (1). A search bar is present with the text "Filter by name or labels". Below this, the alert "alert-rules-blackbox-exporter.yml > Connection was slow" is shown with a "firing (1)" status. A red bar indicates "SlowConnection (1 active)". The alert details are as follows:

```

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

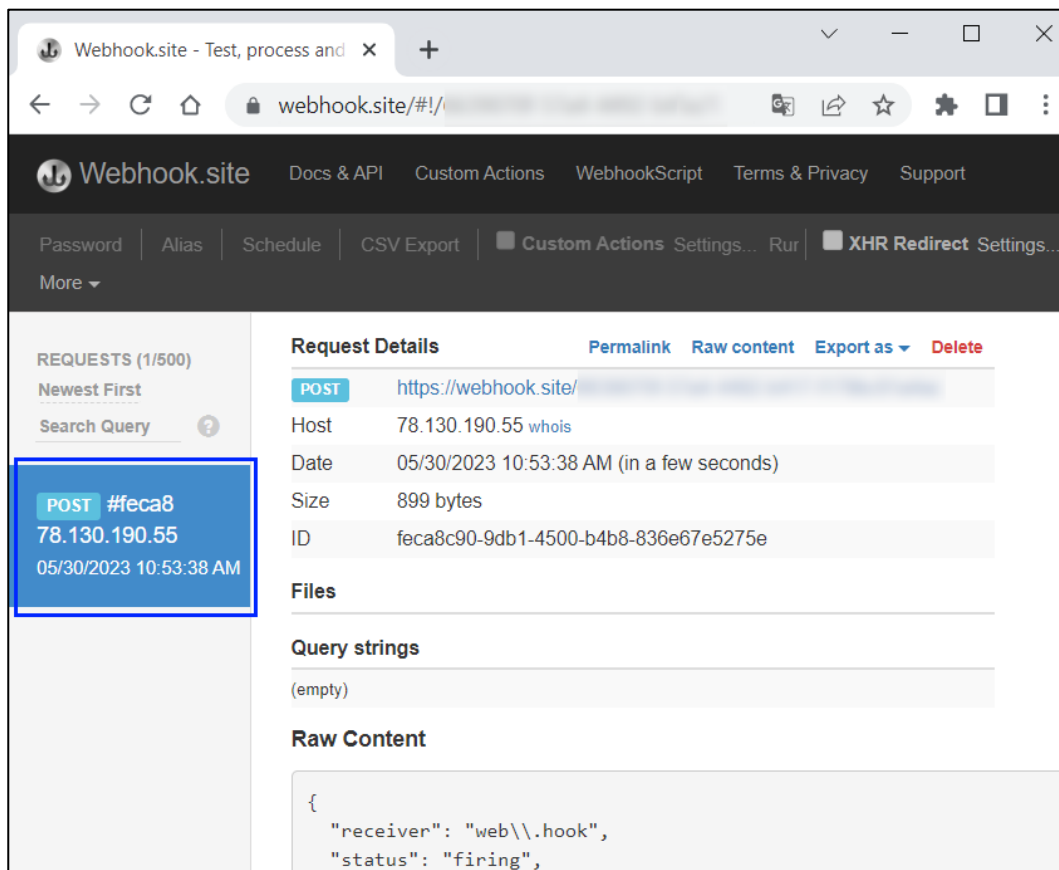
Now, let's go to Alertmanager and we should see the fired alert:

The screenshot shows the Alertmanager Alerts interface. The top navigation bar includes "Alertmanager", "Alerts", "Silences", "Status", "Settings", and "Help". A "New Silence" button is on the right. Below the navigation bar, there are tabs for "Filter" and "Group". The "Receiver: All" tab is selected. A search bar is present with a "+" button and a "Silence" button. Below the search bar, there is a text input field with the placeholder "Custom matcher, e.g. env='production'". A "+ Expand all groups" button is visible. The alert list shows one alert with the label "alertname='SlowConnection'" and a status of "1 alert". The alert details are as follows:

```

2023-05-30T07:53:06.637Z + Info Source Silence Link
instance="https://softuni.org" job="blackbox" phase="connect" severity="warning"
  
```

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



The screenshot shows the Webhook.site web application. The browser address bar displays 'webhook.site/#!/'. The navigation bar includes links for 'Webhook.site', 'Docs & API', 'Custom Actions', 'WebhookScript', 'Terms & Privacy', and 'Support'. Below the navigation bar, there are tabs for 'Password', 'Alias', 'Schedule', 'CSV Export', 'Custom Actions', 'Settings...', 'Rur', 'XHR Redirect', and 'Settings...'. The main content area is divided into two panels. The left panel, titled 'REQUESTS (1/500)', shows a list of requests with a search query input. A request is highlighted with a blue box, showing 'POST #feca8', '78.130.190.55', and '05/30/2023 10:53:38 AM'. The right panel, titled 'Request Details', shows the details of the selected request. It includes a 'POST' method, the URL 'https://webhook.site/', the host '78.130.190.55', the date '05/30/2023 10:53:38 AM (in a few seconds)', the size '899 bytes', and the ID 'feca8c90-9db1-4500-b4b8-836e67e5275e'. Below these details, there are sections for 'Files', 'Query strings' (empty), and 'Raw Content'. The 'Raw Content' section shows a JSON object: 

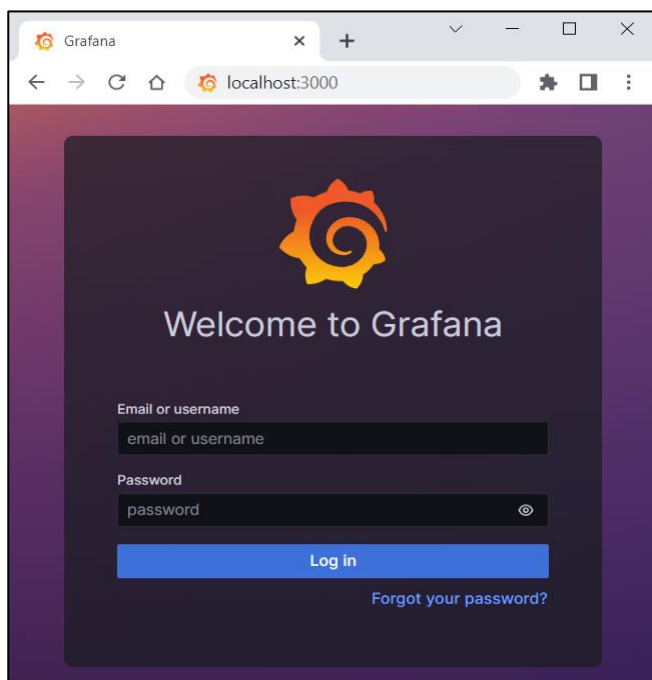
```
{  "receiver": "web\\.hook",  "status": "firing",}
```

### 3. Grafana and Prometheus

#### Step 1: Log in to Grafana

By default, Grafana will be listening on <http://localhost:3000>. You can log in with the default credentials:

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

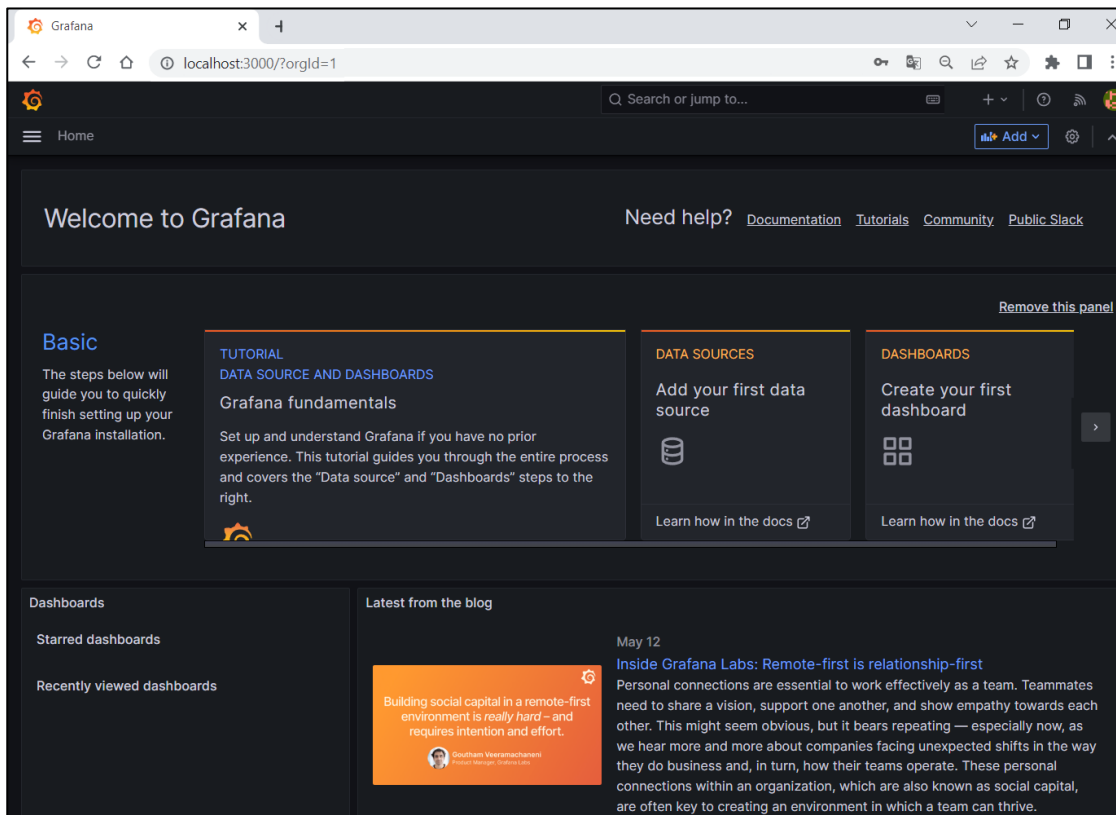


Email or username  
admin

Password  
admin

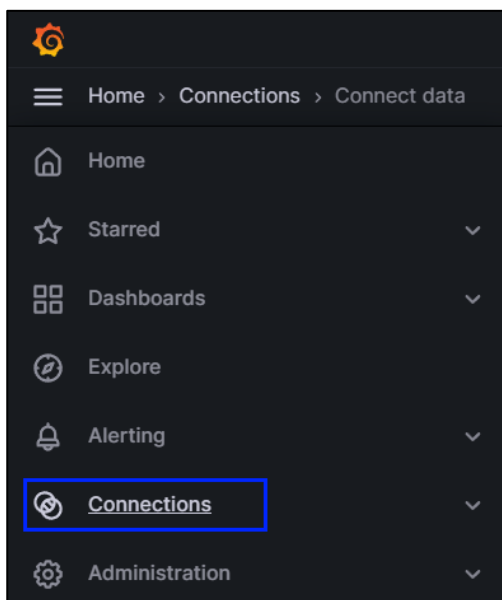
After entering the credentials, you will be asked to **set a new password**. You can do that, or you can just **skip this step**.

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

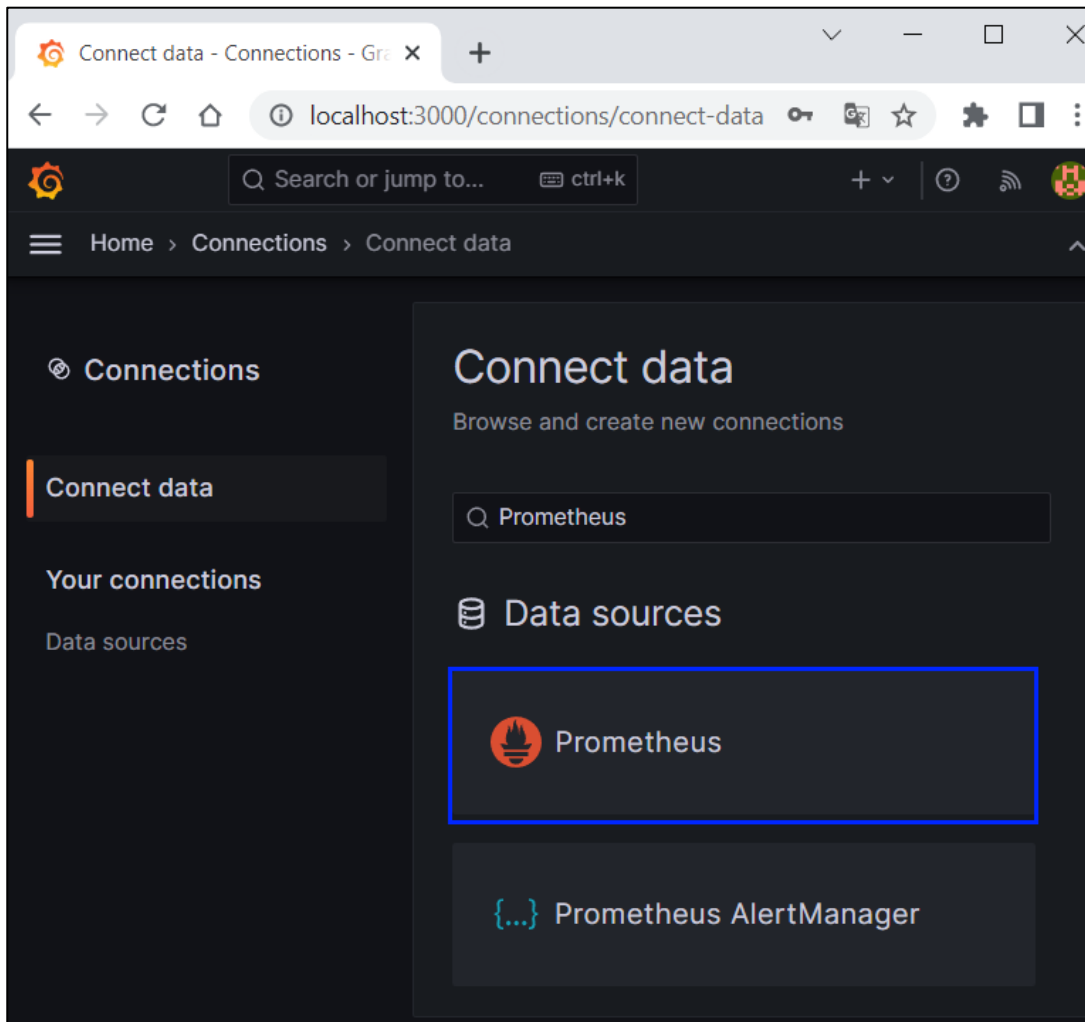


## Step 2: Create Prometheus Data Source in Grafana

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

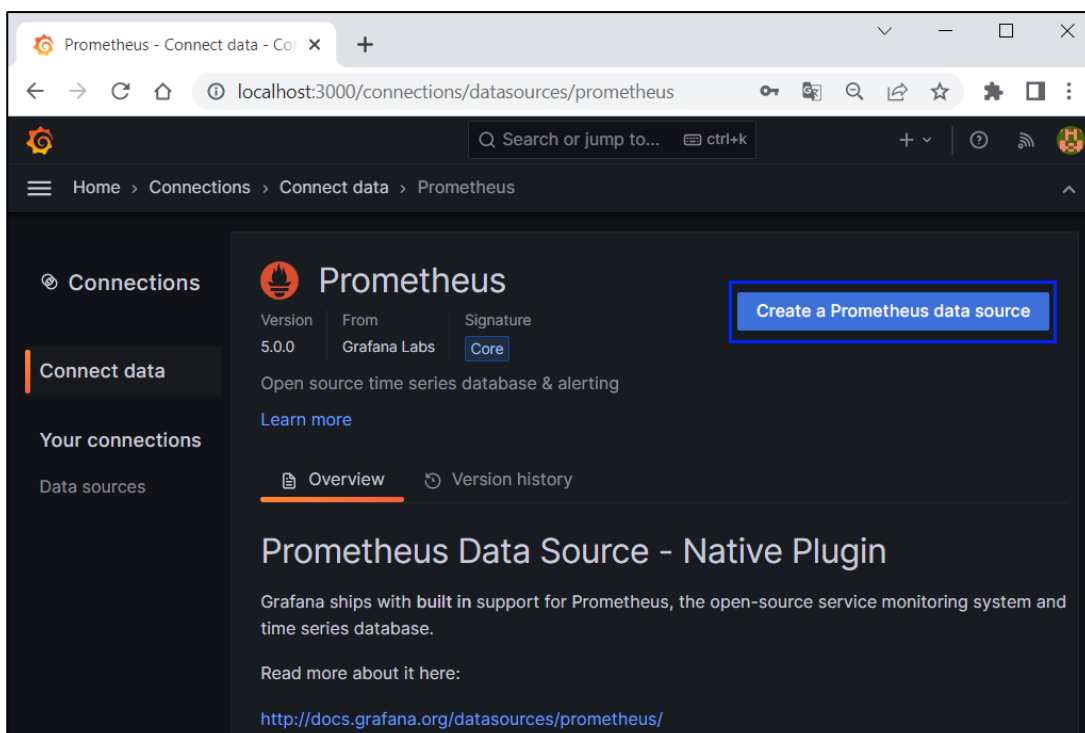


Then, search for "Prometheus" data source and click on it:

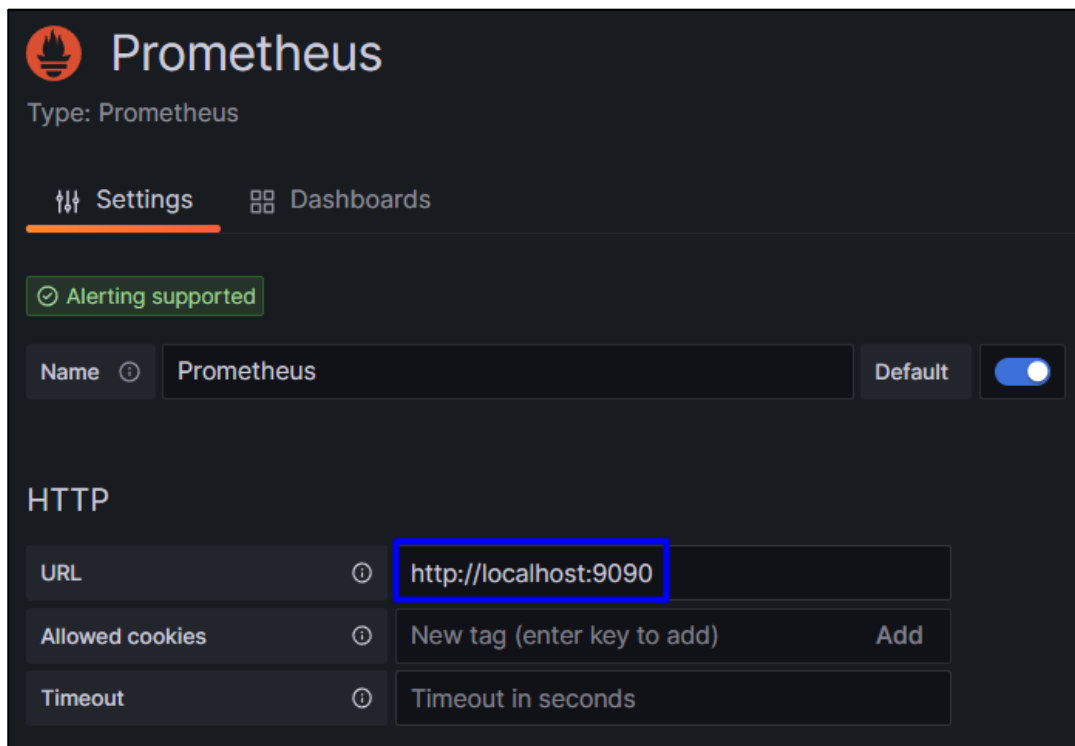


### Step 3: Configure Prometheus Data Source

Now, let's create a Prometheus data source:

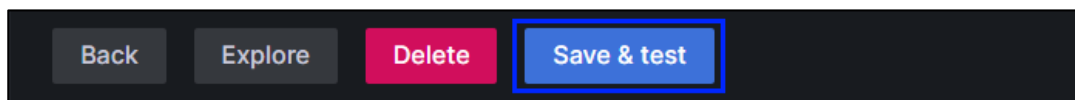


We have to set the Prometheus server URL:



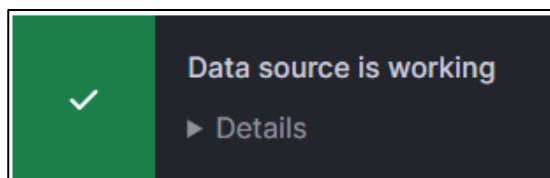
The image shows the Prometheus web interface. At the top, there's a header with the Prometheus logo and the text 'Prometheus'. Below it, 'Type: Prometheus' is displayed. There are two tabs: 'Settings' (active) and 'Dashboards'. A green box highlights 'Alerting supported'. Below that, there's a 'Name' field with 'Prometheus' and a 'Default' toggle switch. The 'HTTP' section contains three rows: 'URL' with 'http://localhost:9090' (highlighted with a blue box), 'Allowed cookies' with a 'New tag (enter key to add)' field and an 'Add' button, and 'Timeout' with a 'Timeout in seconds' field.

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



The image shows a row of four buttons: 'Back', 'Explore', 'Delete', and 'Save & test'. The 'Save & test' button is highlighted with a blue box.

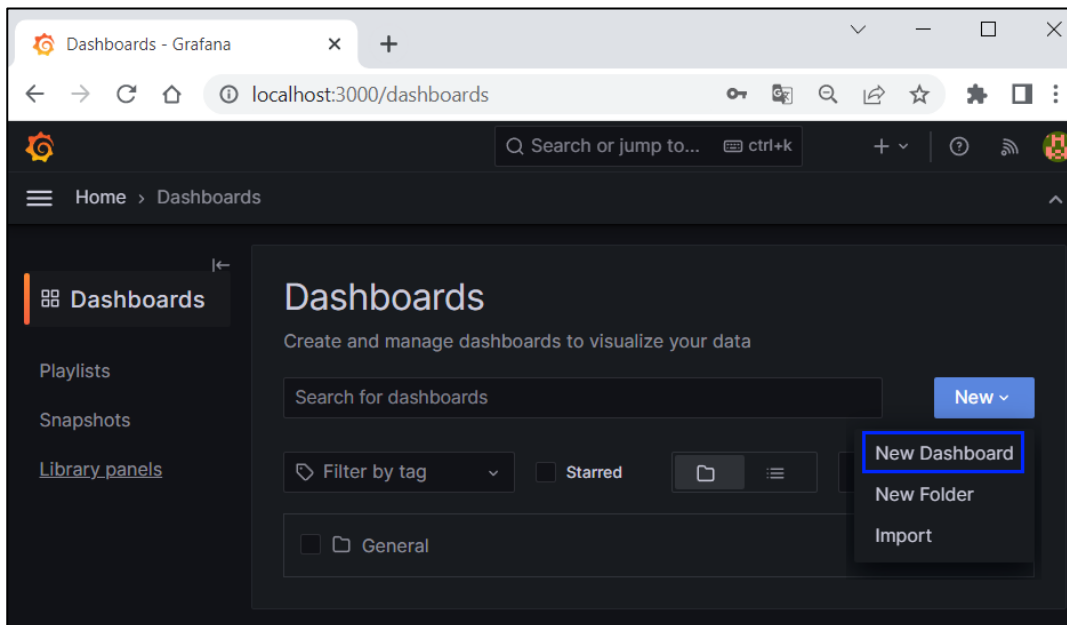
You should see the success message in a few moments:



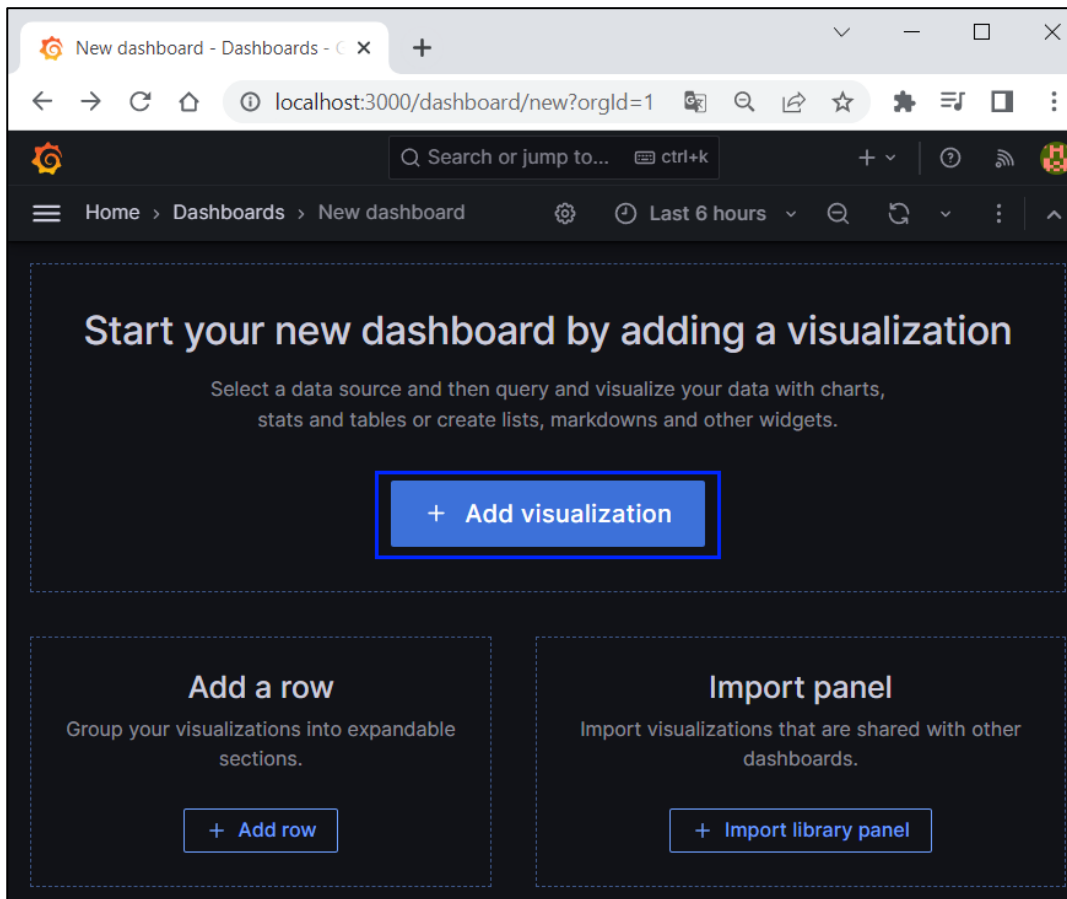
## Step 4: Create a Grafana Dashboard

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**]:



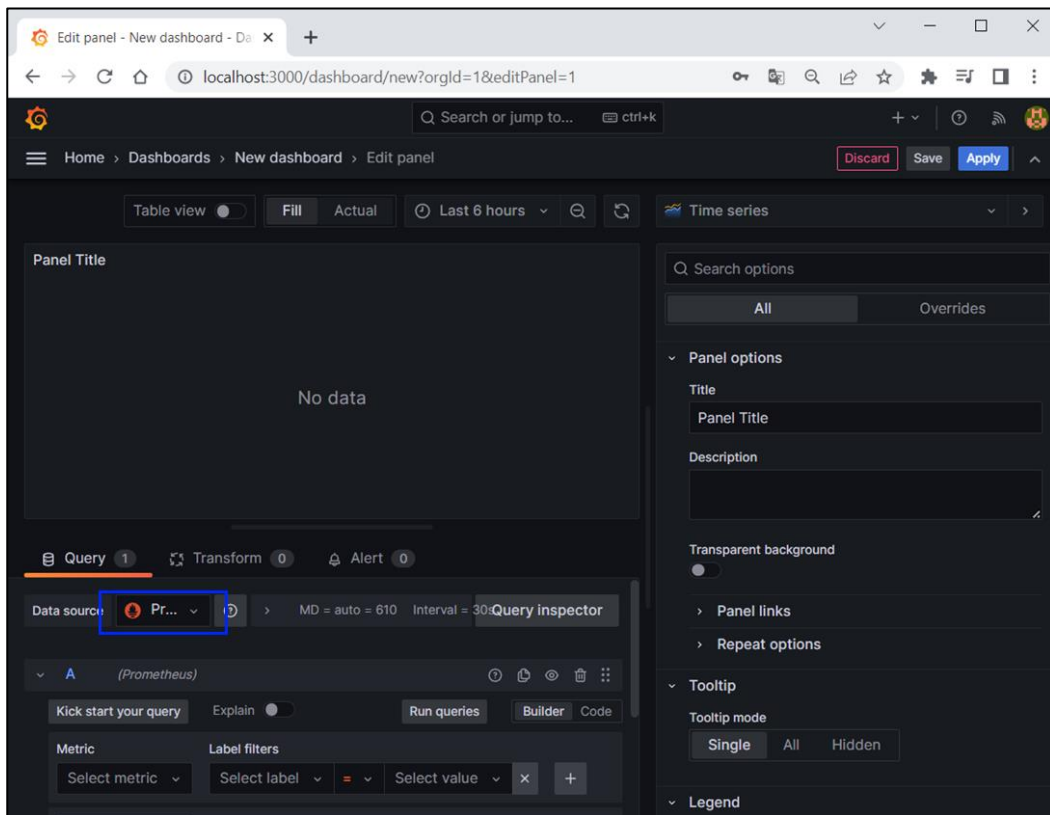


Now, click on the [+ **Add visualization**] button to create a visualization (panel):

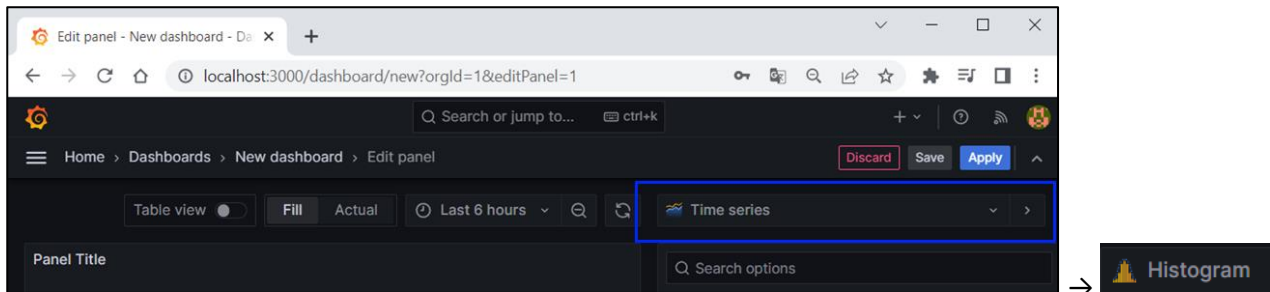


For example, you can create a histogram for the HTTP probe duration metric.

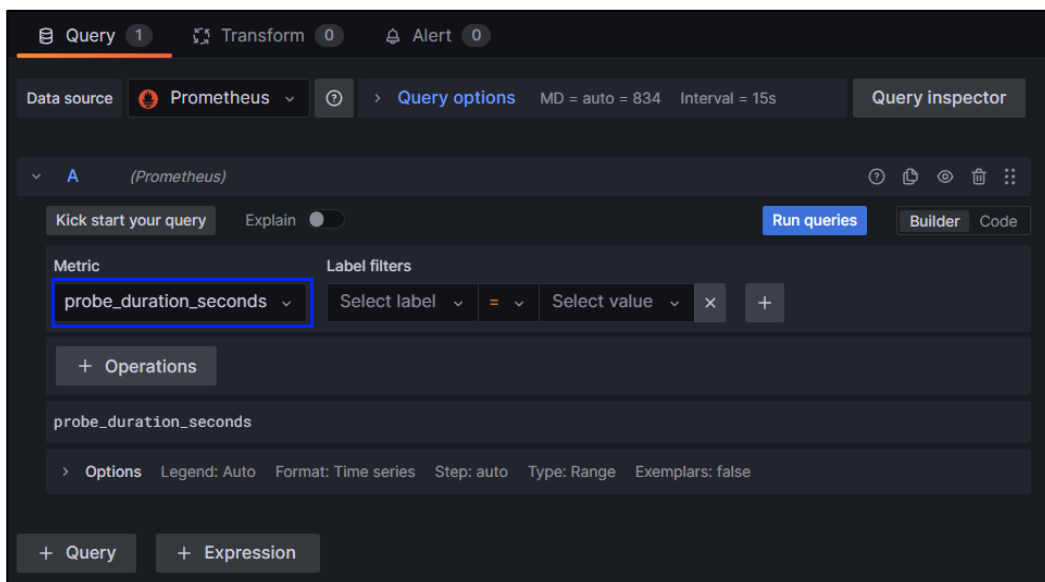
First, select the **Prometheus** data source from the Data source dropdown menu:



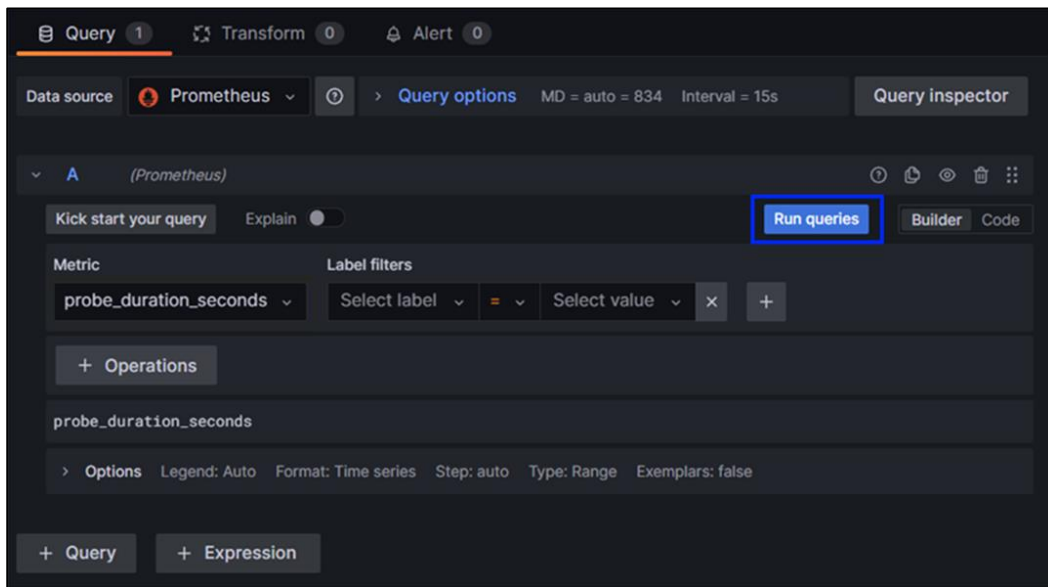
Next, click on the **Visualization** menu and select **Histogram**:



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



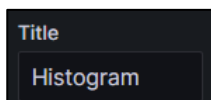
After that, click on the **[Run queries]** button:



And you should be able to see the panel:



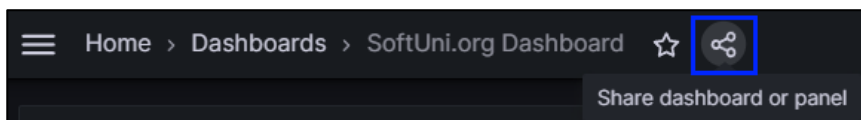
In order to change the panel title, click on the "Title" input field in the **Panel options** section:



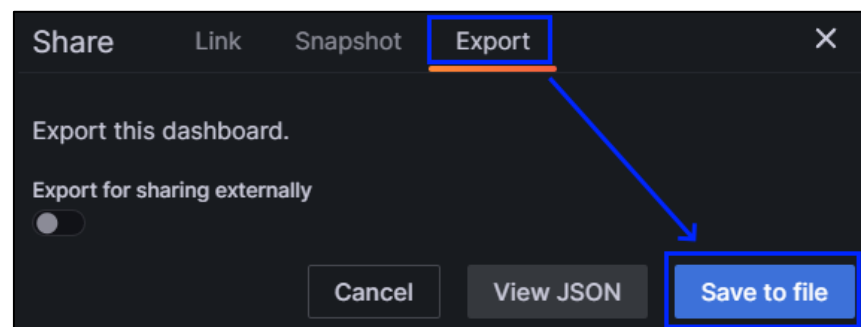
## Step 5: Import and Export Dashboards

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

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



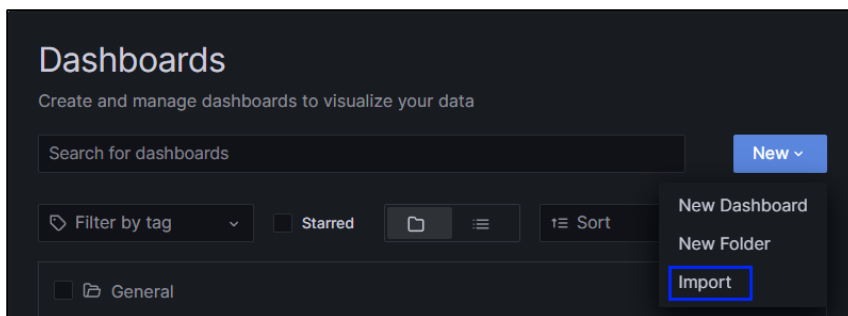
Then, click on **[Export]** → **[Save to file]**:



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:

