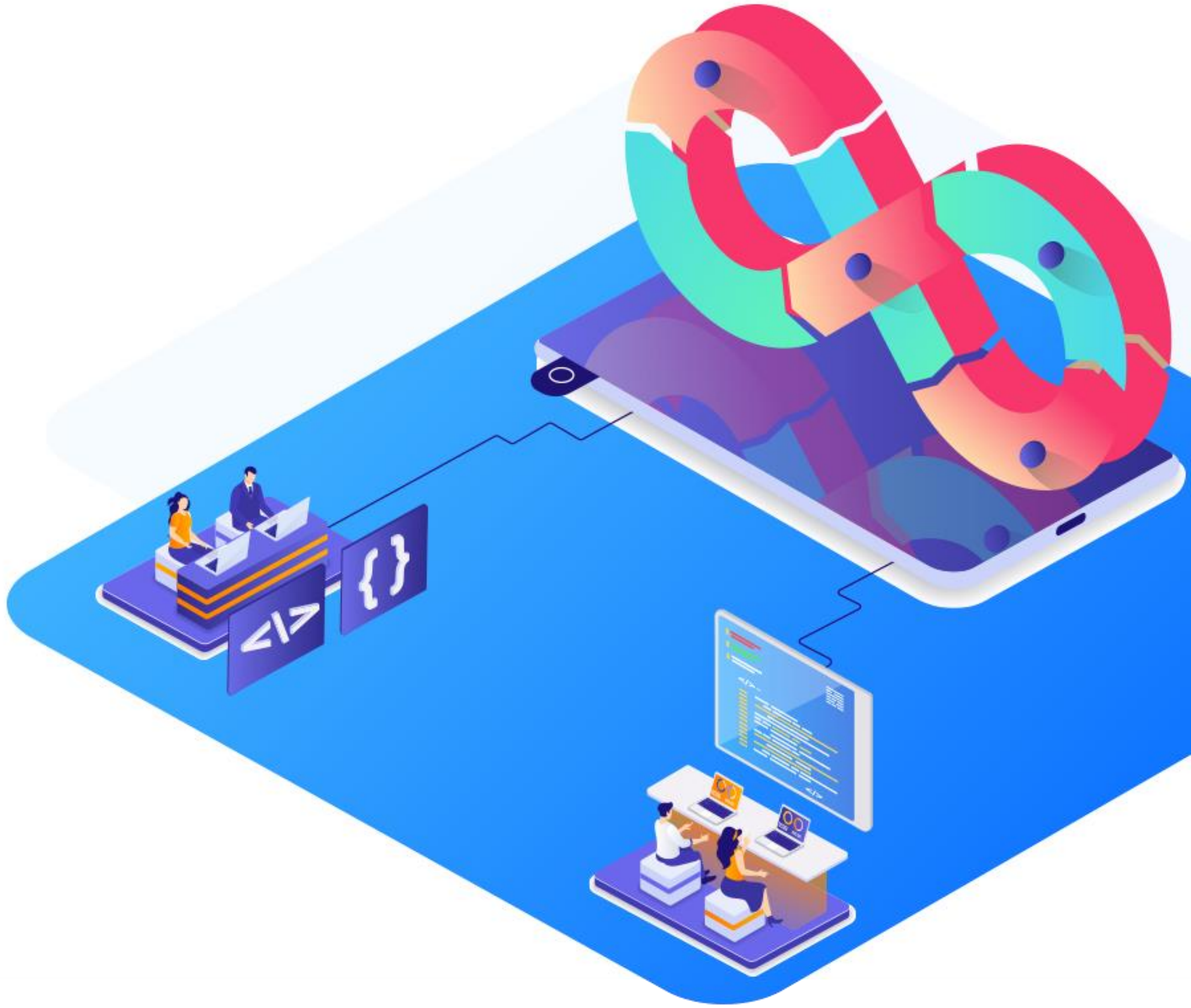
Monitoring and Logging in DevOps



Visualization in Monitoring Using Grafana

Learning Objectives

By the end of this lesson, you will be able to:

- Evaluate the core features and benefits of Grafana to enhance understanding of its capabilities for data visualization
- Configure Grafana, connect data sources, and set up dashboards and panels to effectively visualize data
- Develop and customize dashboards and panels in Grafana to visualize various data metrics
- Implement alerting and notification systems in Grafana to monitor and respond to critical metrics
- Integrate Grafana with Prometheus and utilize PromQL for performing advanced data queries

Introduction to Grafana

What Is Grafana?

Grafana is a widely used tool that helps visualize and analyze data from various sources. It allows setting alerts and exploring metrics, logs, and traces.

It offers tools to transform time series database (TSDB) data into meaningful graphs and visualizations.

Features of Grafana

Data source integration

Supports multiple data sources like Prometheus and MySQL for unified data visualization and analysis

Customizable dashboards

Offers tools to create custom dashboards with various visualizations for more tailored insights

Rich visualization options

Provides diverse charts and customizable visuals for clearer data interpretation

Features of Grafana

Dynamic queries with PromQL

Supports advanced querying with PromQL for deep data analysis and exploration

Annotations and alerting

Allows marking events on visualizations and setting alerts for proactive monitoring and faster issue resolution

Access control and permissions

Provides role-based access control for secure and controlled access to dashboards and data sources

Features of Grafana

Plugins and extensions

Offers a wide range of plugins for enhanced functionality and integration with various tools

Explore mode

Enables interactive data exploration without pre-configured dashboards for ad-hoc analysis and troubleshooting

Snapshots and annotations

Allows taking snapshots of dashboards and adding annotations for collaborative analysis and documentation

Why Is Grafana Important?

The following reasons highlight the importance of Grafana in organizations:

Scalability and performance:
Supports large-scale environments with clustering and load balancing for enhanced performance

Enterprise-grade security and access control:
Provides role-based access, audit logging, and single sign-on (SSO) support for secure and controlled access

Customizability and extensibility:
Supports customization with plugins and tailored visualizations for flexibility

Why Is Grafana Important?

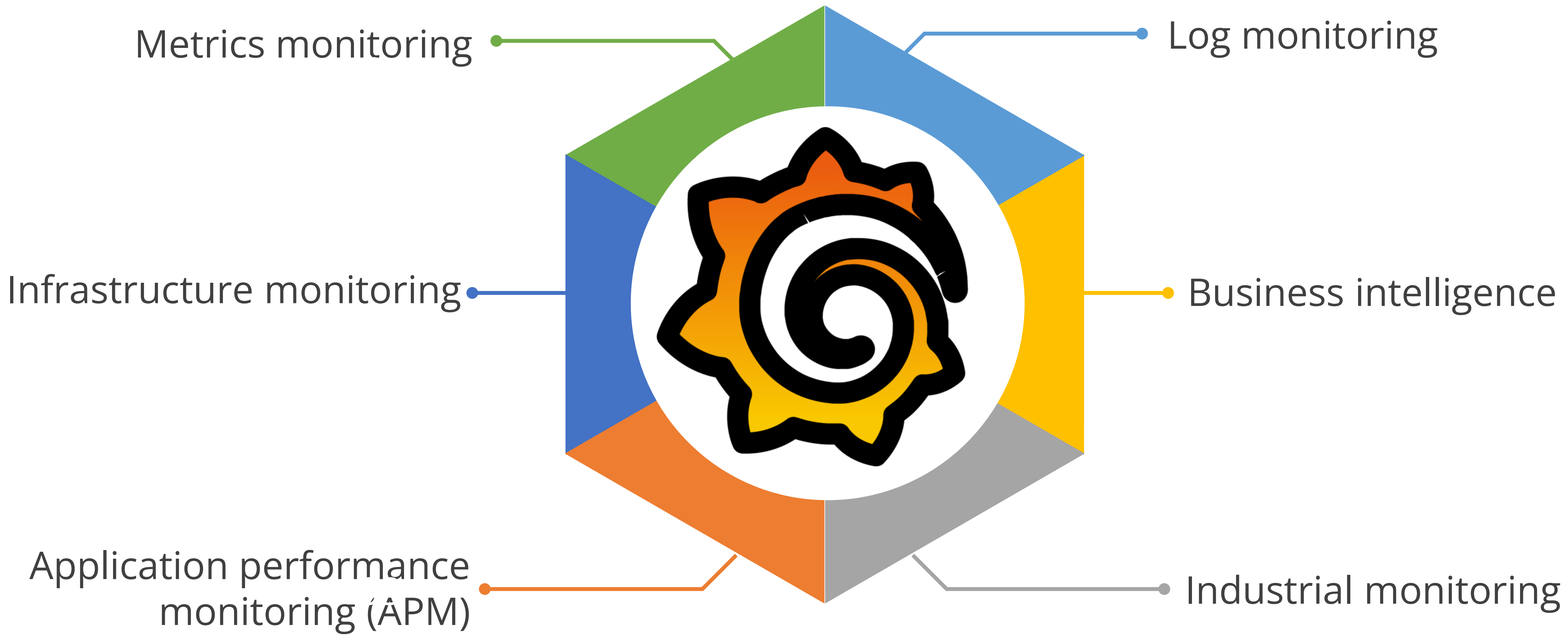
The following reasons highlight the importance of Grafana in organizations:

Seamless integration with existing tools:
Integrates with diverse monitoring and observability tools for streamlined operations

Comprehensive observability:
Unifies metrics, logs, and traces for complete system insights

Types of Monitoring with Grafana

Grafana supports various types of monitoring:



Types of Monitoring with Grafana

Metrics monitoring

- Grafana visualizes application and system metrics like CPU usage, memory usage, and network traffic using data sources such as Prometheus

- **Example:** A DevOps team monitors CPU and memory usage to optimize resources and prevent downtime.

Infrastructure monitoring

- Grafana monitors servers, containers, VMs, and cloud resources with data sources such as Prometheus and Node Exporter

- **Example:** An IT operations team tracks server health and cloud usage to manage capacity and avoid overuse.

Types of Monitoring with Grafana

Application performance monitoring

- Grafana integrates with distributed tracing systems like Jaeger and Zipkin for monitoring and analyzing performance
- **Example:** A development team identifies errors and dependencies using APM to enhance application performance.

Log monitoring

- Grafana visualizes logs from sources like Loki, Elasticsearch, and Splun alongside metrics and traces
- **Example:** A security team correlates logs and metrics to investigate and resolve security incidents.

Types of Monitoring with Grafana

Business intelligence

- Grafana connects to databases and data warehouses to visualize business data, including sales and customer metrics
- **Example:** A marketing team analyzes sales data with Grafana to improve campaign effectiveness.

Industrial monitoring

- Grafana visualizes sensor and machine data for predictive maintenance and process optimization
- **Example:** A manufacturing company uses sensor data for predictive maintenance and process optimization.

Types of Monitoring with Grafana

Type of Monitoring	Features	Description
Metrics monitoring	• Real-time tracking • custom dashboards • threshold alerts	Tracks key performance metrics like CPU usage, memory utilization, and response times.
Log monitoring	• Log analysis • Error detection • Searchable logs • Trend visualization	Analyzes system logs to identify issues, errors, and behavior patterns over time.
Infrastructure monitoring	• Server monitoring • Network status monitoring • Hardware health tracking • Uptime tracking	Monitors servers, networks, and hardware for performance and stability.
Business intelligence	• Data visualization • KPI tracking • Business analytics • Historical trend analysis	Collects and visualizes data for insights that support business decision-making.

Types of Monitoring with Grafana

Type of Monitoring	Features	Description
Application performance monitoring (APM)	• Transaction tracing • Response time monitoring • Error tracking • Latency analysis	Measures application performance to ensure smooth functionality and user experience.
Industrial monitoring	• Sensor data tracking • Machinery performance monitoring • Process optimization • Real-time alerts	Monitors industrial systems and equipment for performance and operational efficiency.

Data Sources Compatible with Grafana

Grafana supports a variety of data sources, including time series data sources.

Some such data sources are as follows:

Standard

- MySQL
- PostgreSQL
- Elasticsearch

Time series

- Prometheus
- InfluxDB
- Graphite

Grafana integrates with both standard and time series databases to deliver comprehensive data visualization and monitoring.

Metric Collection

Grafana lacks built-in metric collection and integrates with external data sources to visualize and analyze data. It supports various data sources, including:

Prometheus	Collects and stores time-series data using an open-source toolkit, ideal for dynamic environments
InfluxDB	Stores and queries large volumes of time-based data, suitable for high-availability systems
Graphite	Collects and stores time-series data with a hierarchical namespace, perfect for simple, scalable setups

Metric Collection

Grafana lacks built-in metric collection and integrates with external data sources to visualize and analyze data. It supports various data sources, including:

AWS CloudWatch	Monitors and analyzes metrics from AWS resources, optimizing cloud performance
Azure Monitor	Collects and analyzes metrics from Azure and on-premises systems, enhancing hybrid cloud monitoring

Metric Collection Approaches with Grafana

Grafana does not have native metric collection capabilities, so organizations must choose one of the following approaches to gather and store metrics:

Dedicated metric collection system

- Uses specialized systems to collect metrics and configure
- **Example:** Prometheus collecting metrics from a web server

Cloud monitoring service

- Utilizes built-in cloud provider monitoring tools
- **Example:** AWS CloudWatch monitoring EC2 instances

Metric Collection Approaches with Grafana

Grafana does not have native metric collection capabilities, so organizations must choose one of the following approaches to gather and store metrics:

Application instrumentation

- Instruments applications to expose metrics directly
- **Example:** A Java app using micrometer to expose JVM metrics

Agent-based collection

- Deploys agents on nodes to collect and forward metrics
- **Example:** Grafana Agent collecting system metrics from servers

Metrics and Visualizations

The following are two critical components of Grafana that provide insights into system performance:

Metrics

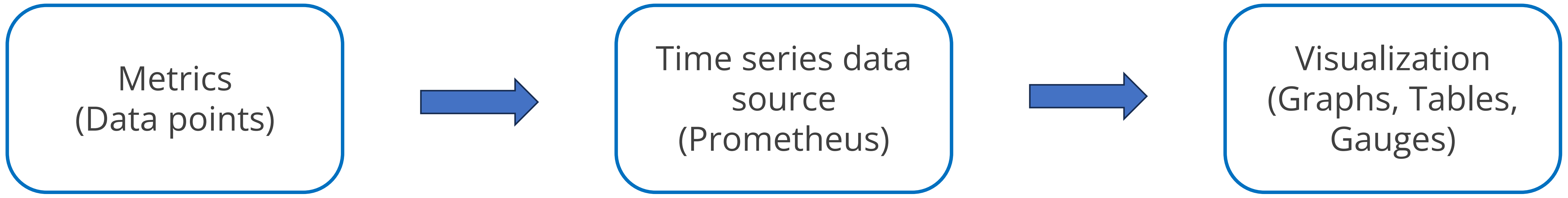
Numerical data points collected over time to measure system performance

Visualizations

Graphical representations of metrics for intuitive data analysis

Relationship between Metrics and Visualizations

Grafana uses metrics as the foundational data, with visualizations presenting this data in a readily interpretable manner. It involves three key components:



Grafana transforms performance data into clear, insightful visual formats using time series data source.

Differences between Grafana and Prometheus

Both are widely used tools for monitoring and visualization; the following outlines their distinct roles in the monitoring stack:

Features	Grafana	Prometheus
Primary function	Provides visualization and alerting tools	Focuses on monitoring and time-series data collection
Data collection	Visualizes data from various sources	Collects and stores time-series data
Query language	Supports multiple query languages, based on data source	Supports PromQL for real-time analysis
Visualization	Offers extensive options for charts and graphs	Offers limited visualization options
Alerting	Provides advanced alerting with custom rules	Provides basic alerting with simple rules
Ecosystem	Integrates with multiple databases and data sources	Supports a variety of exporters and integrations for data

Quick Check

An IT team wants to monitor system performance metrics and send notifications to specific team members according to the type of issue detected. Which feature of Grafana best supports this functionality?

A. Annotations and alerting

B. Explore mode

C. Plugins and extensions

D. Rich visualization options

Setting up Grafana

Grafana Installation Platforms

Grafana can be installed on the following operating systems:

Windows

Mac

Ubuntu

Debain

Red Hat

Fedora

OpenSUSE

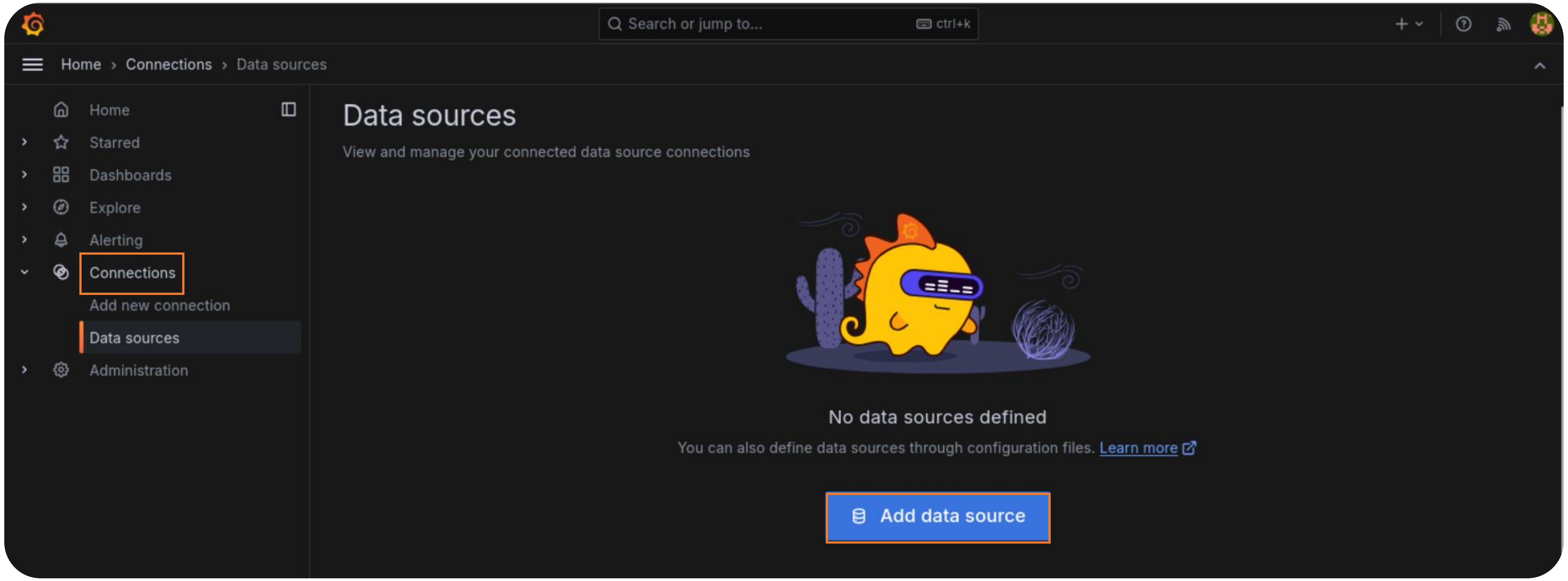
Connecting Data Sources

It is the process of integrating various databases, cloud services, or other data repositories into the Grafana platform. This consists of the following steps:

1. Access the configuration menu
2. Add a data source
3. Configure the data source
4. Save and test the connection

Access the Configuration Menu

Open Grafana; in the configuration menu on the left side, click on **Connections**, then click on **Add data sources** as shown below:



Data sources

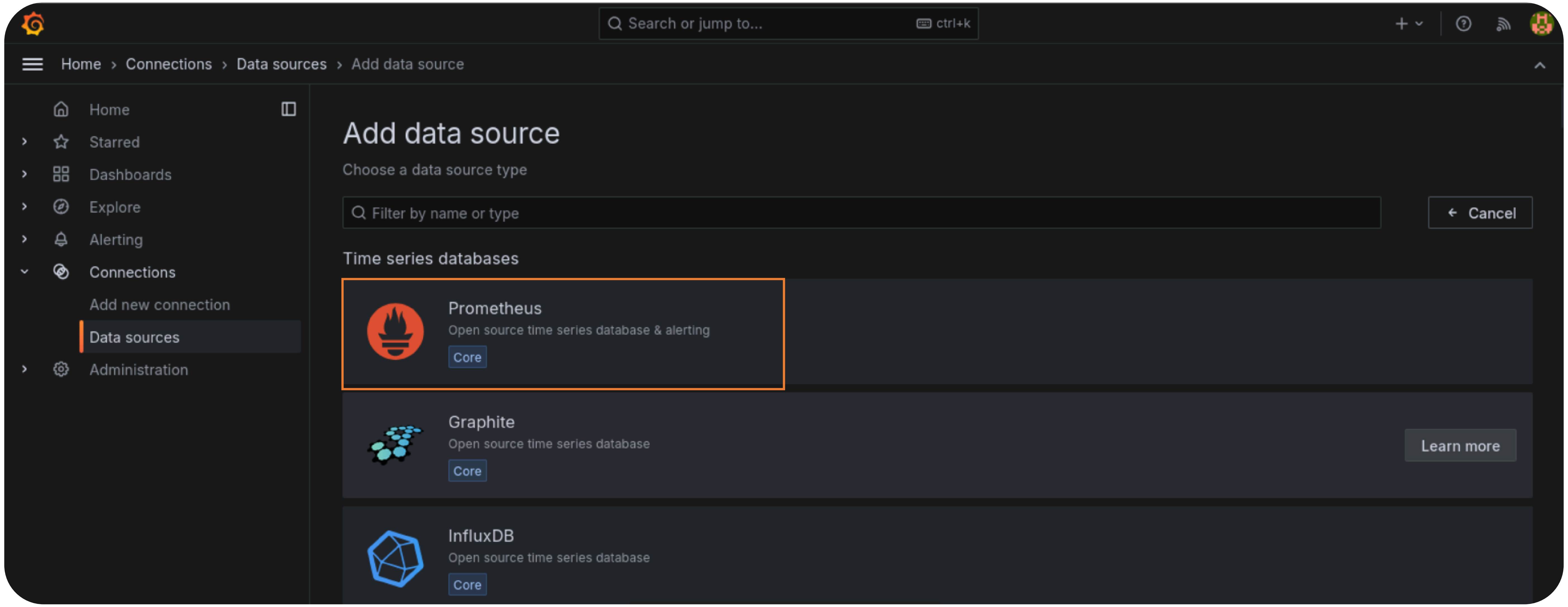
View and manage your connected data source connections

No data sources defined

You can also define data sources through configuration files. Learn more

Add a Data Source

Select a data source from the list of officially supported options that appear. In this case, the data source is **Prometheus** as shown below:



Add data source

Choose a data source type

Filter by name or type

Time series databases

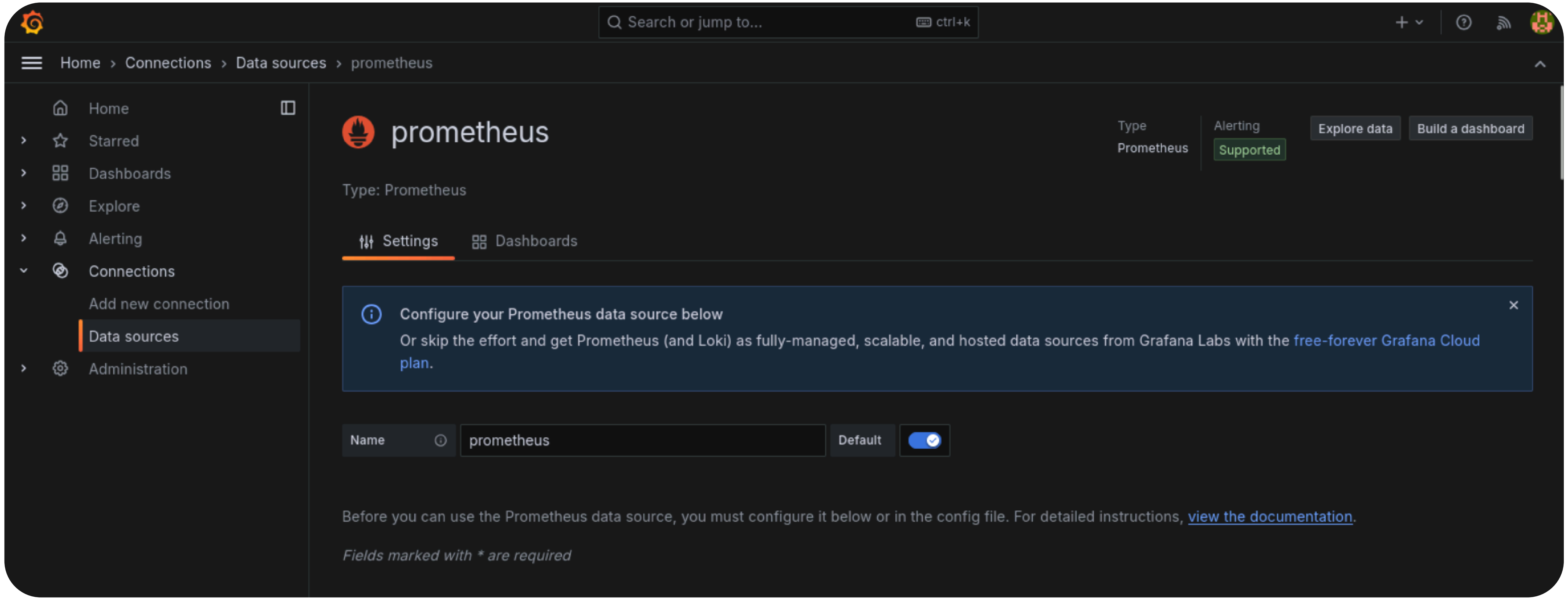
Prometheus
Open source time series database & alerting
Core

Graphite
Open source time series database
Core

InfluxDB
Open source time series database
Core

Configure the Data Source

The data source in Grafana appears as shown below:



prometheus

Type: Prometheus

Configure your Prometheus data source below

Or skip the effort and get Prometheus (and Loki) as fully-managed, scalable, and hosted data sources from Grafana Labs with the free-forever Grafana Cloud plan.

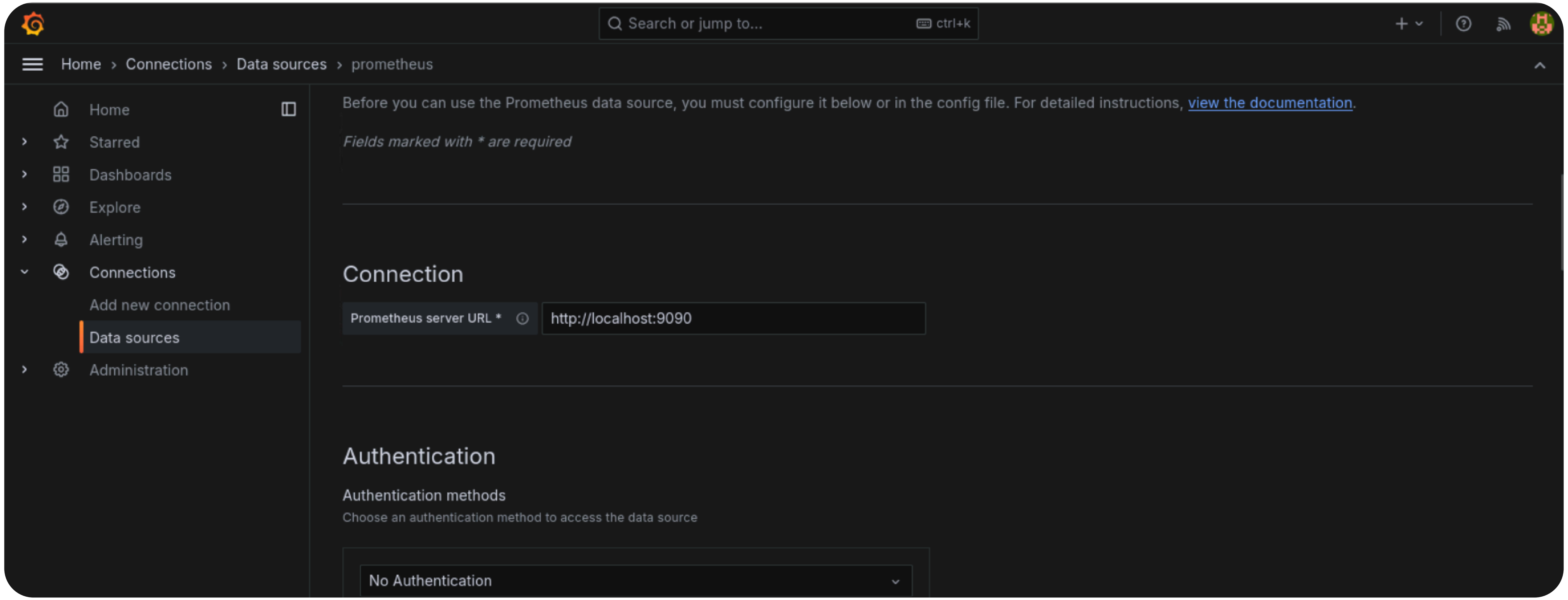
Name	prometheus	Default	✓

Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, view the documentation.

*Fields marked with * are required*

Configure the Data Source

Fill in the required parameters for the data source, including the URL, authorization details, and name, as shown below:



Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, view the documentation.

*Fields marked with * are required*

Connection

Prometheus server URL * http://localhost:9090

Authentication

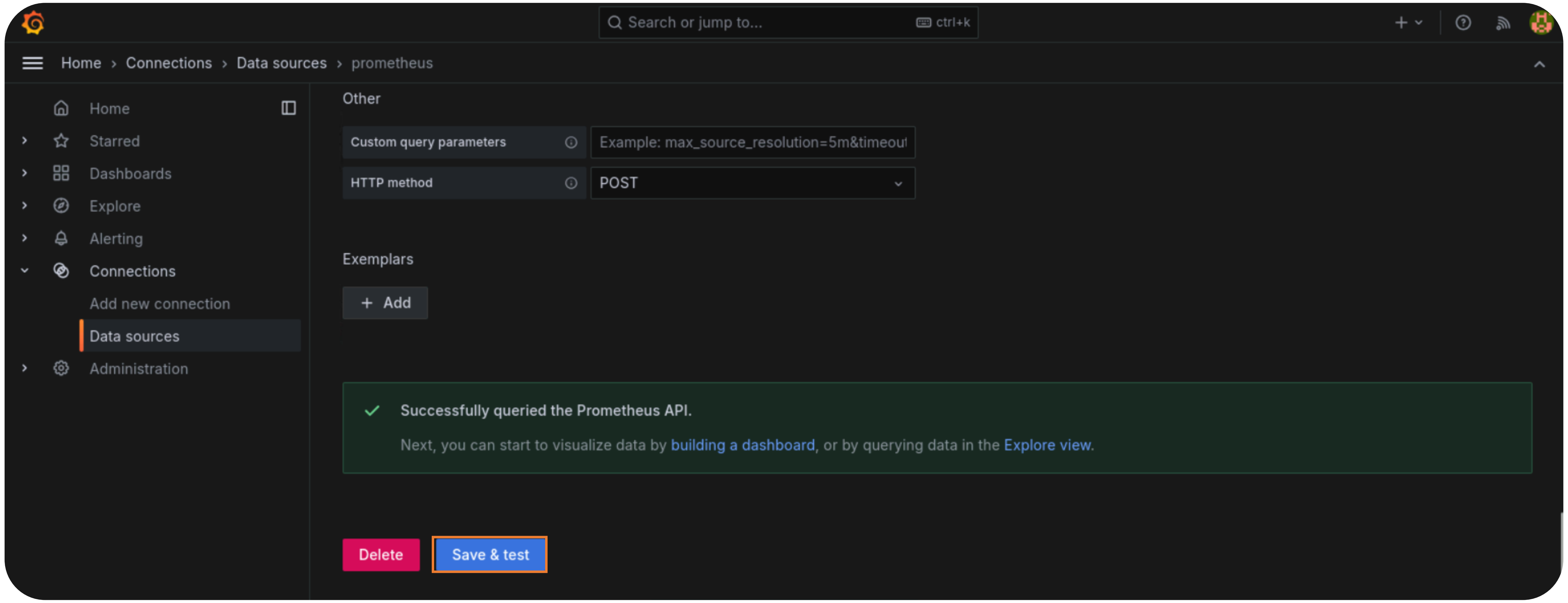
Authentication methods

Choose an authentication method to access the data source

No Authentication

Save and Test the Connection

After filling in the parameters, click on the **Save & Test** button, and Grafana will confirm the connection with a success message as shown below:



Successfully queried the Prometheus API.

Next, you can start to visualize data by building a dashboard, or by querying data in the Explore view.

NOTE

After completing these steps, navigate to the dashboards section to create interactive and real-time visualizations using the connected data sources.

Creating Dashboards and Panels in Grafana

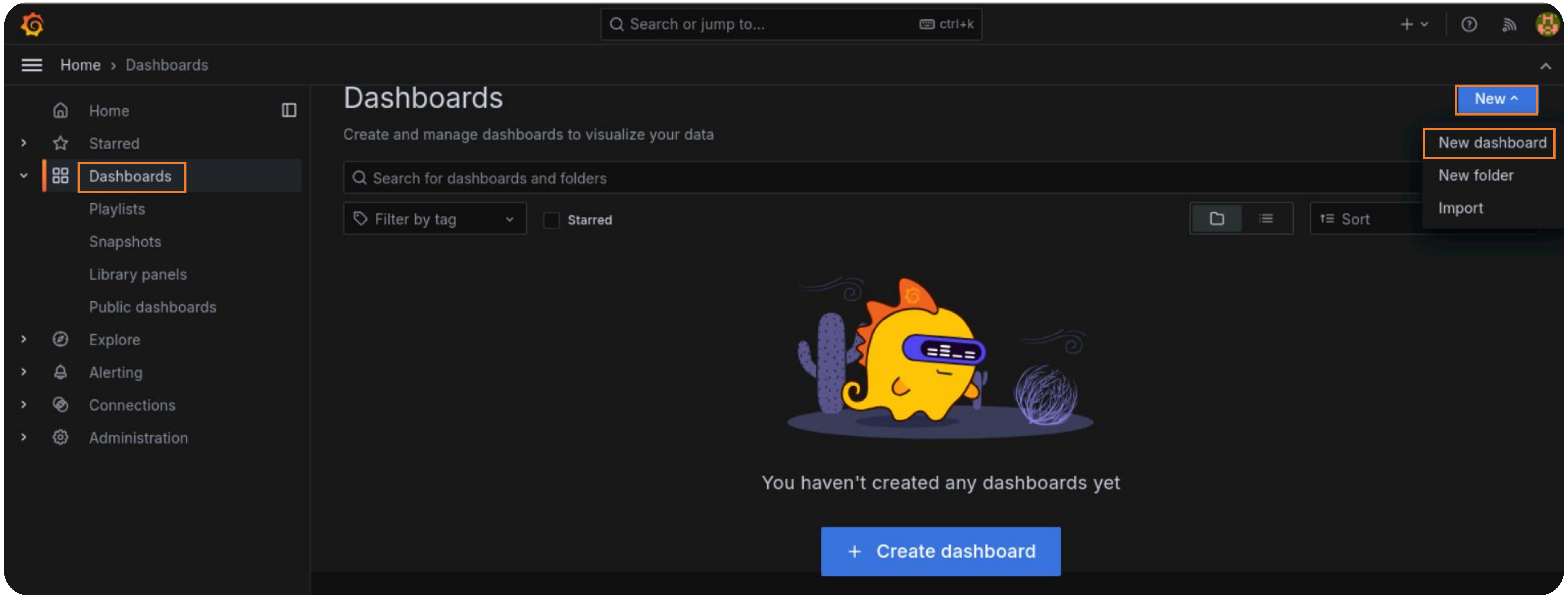
Grafana allows for the creation of dynamic dashboards with multiple panels for data visualization.

The process involves the following steps:

1. Create a new dashboard
2. Set up data source
3. Configure visualization and panel options
4. Save and finalize the dashboard

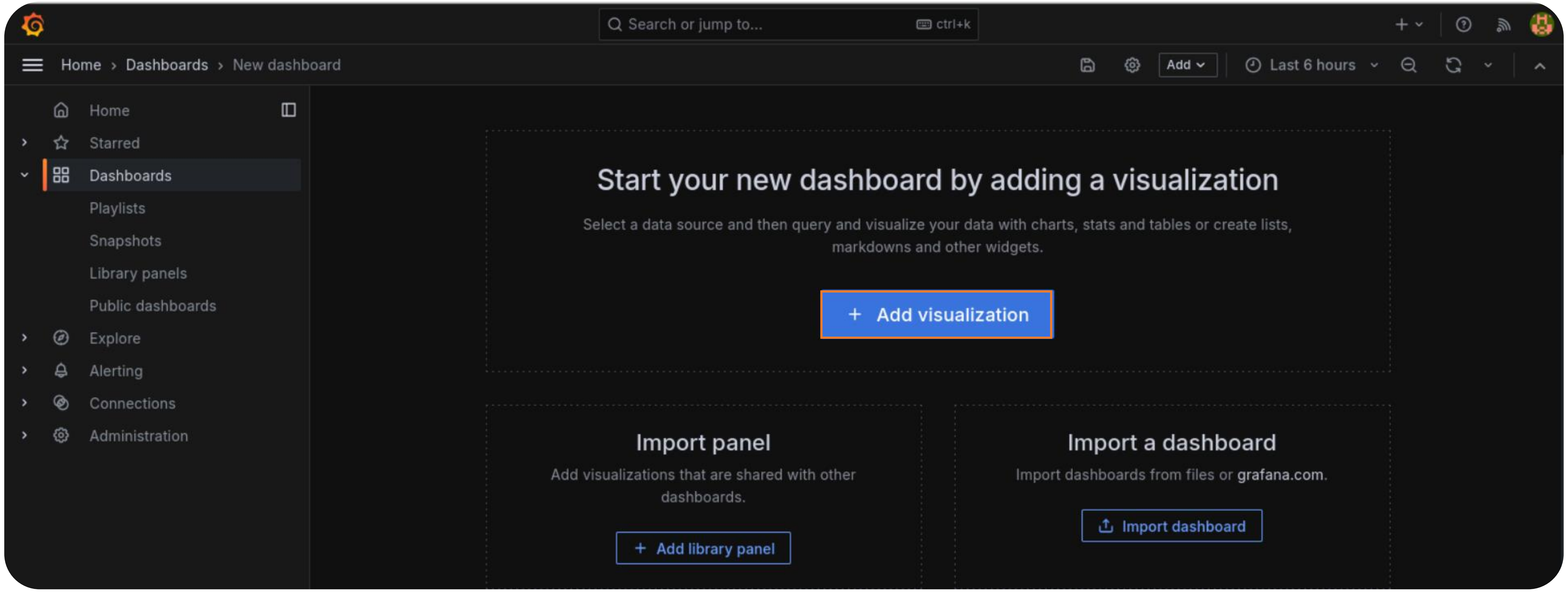
Create a New Dashboard

Click **Dashboards** on the left-side menu, and select **New Dashboard** under **New** as shown below:



Create a New Dashboard

Click on the **+ Add visualization** to start creating a new visual element as shown below:



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

Import panel

Add visualizations that are shared with other dashboards.

+ Add library panel

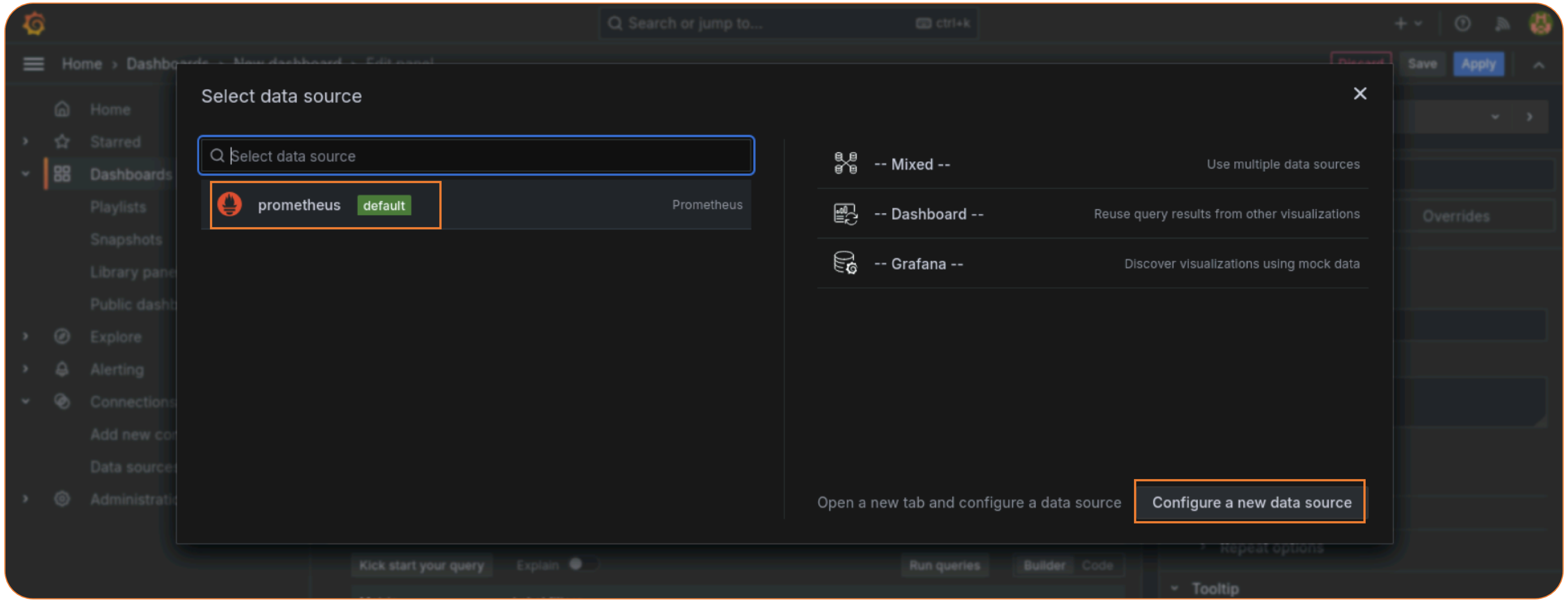
Import a dashboard

Import dashboards from files or grafana.com.

Import dashboard

Set Up Data Source

Select an existing data source. If none are available, click on **Configure a new data source** as shown below:


Select data source

Select data source

prometheus default Prometheus

-- Mixed -- Use multiple data sources

-- Dashboard -- Reuse query results from other visualizations

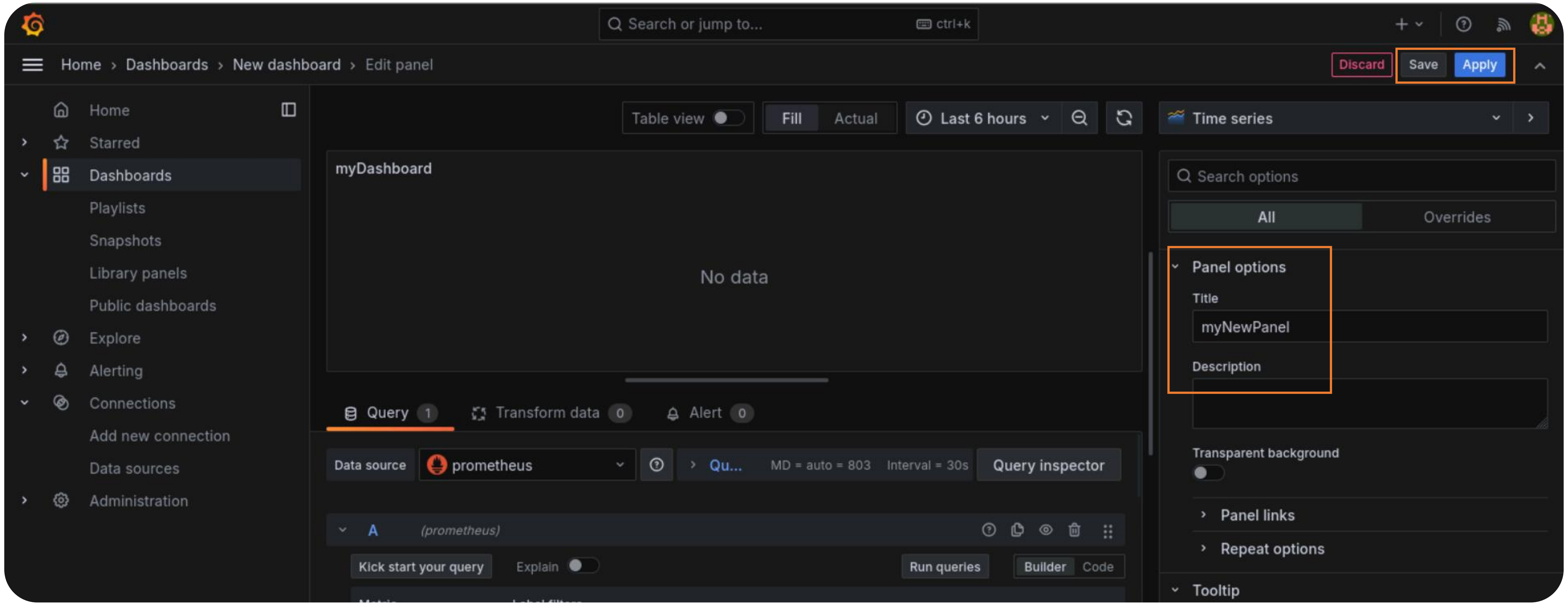
-- Grafana -- Discover visualizations using mock data

Open a new tab and configure a data source

Configure a new data source

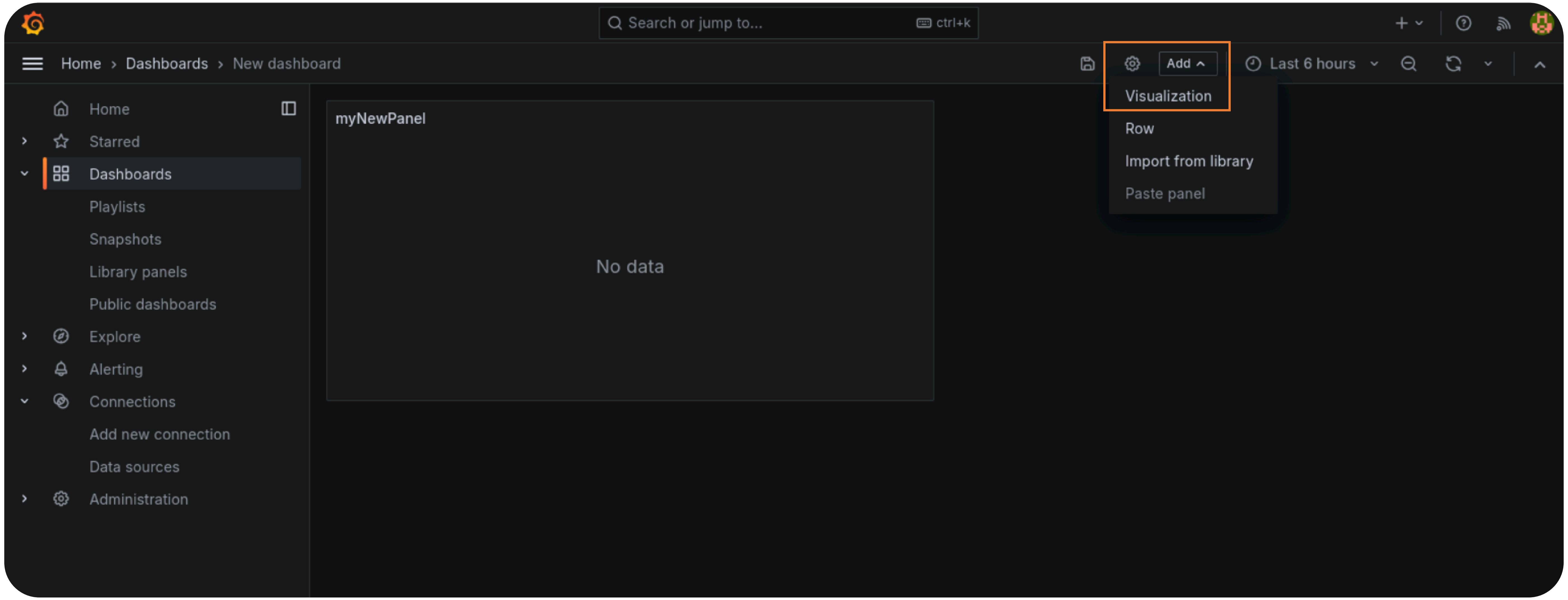
Configure Visualization and Panel Options

Enter a **Title** and **Description** in the **Panel options**, then click **Save** to save the dashboard or **Apply** to preview changes as shown below:



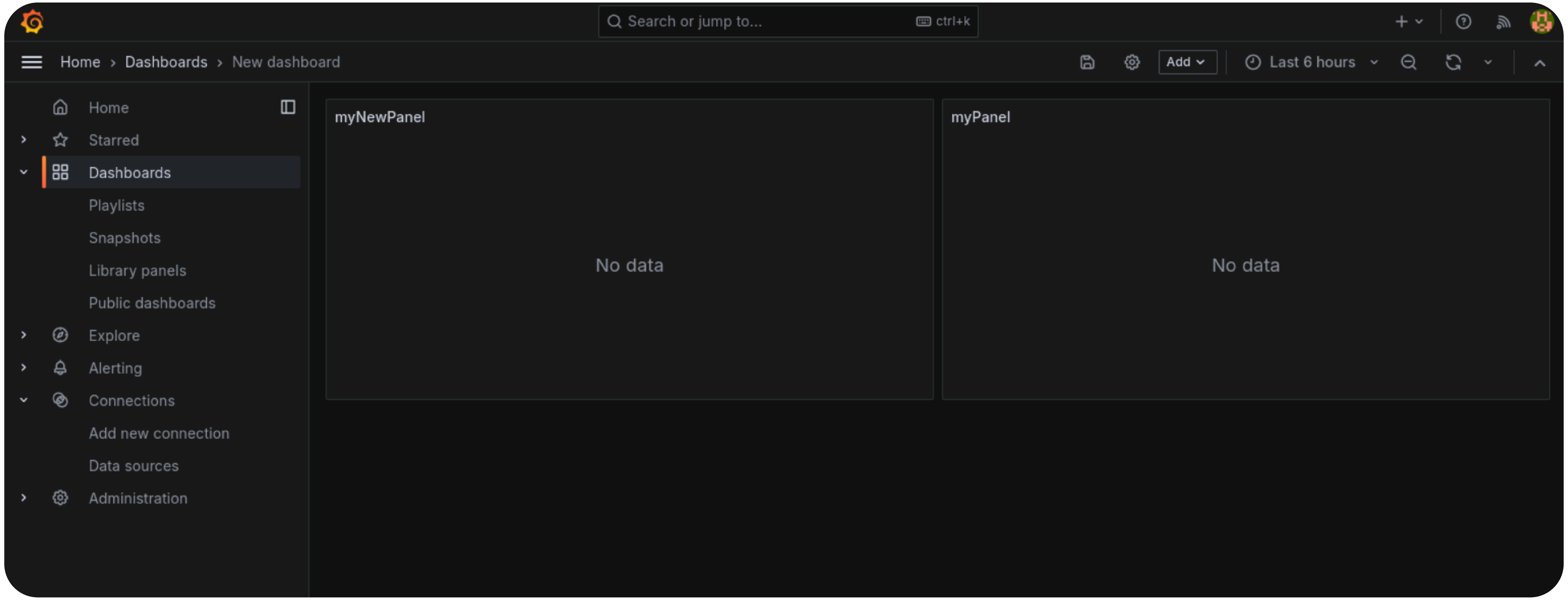
Configure Visualization and Panel Options

After saving the current configuration, the following screen will be displayed. Click on **Add**, then click **Visualization** to add more panels to the dashboard as shown below:



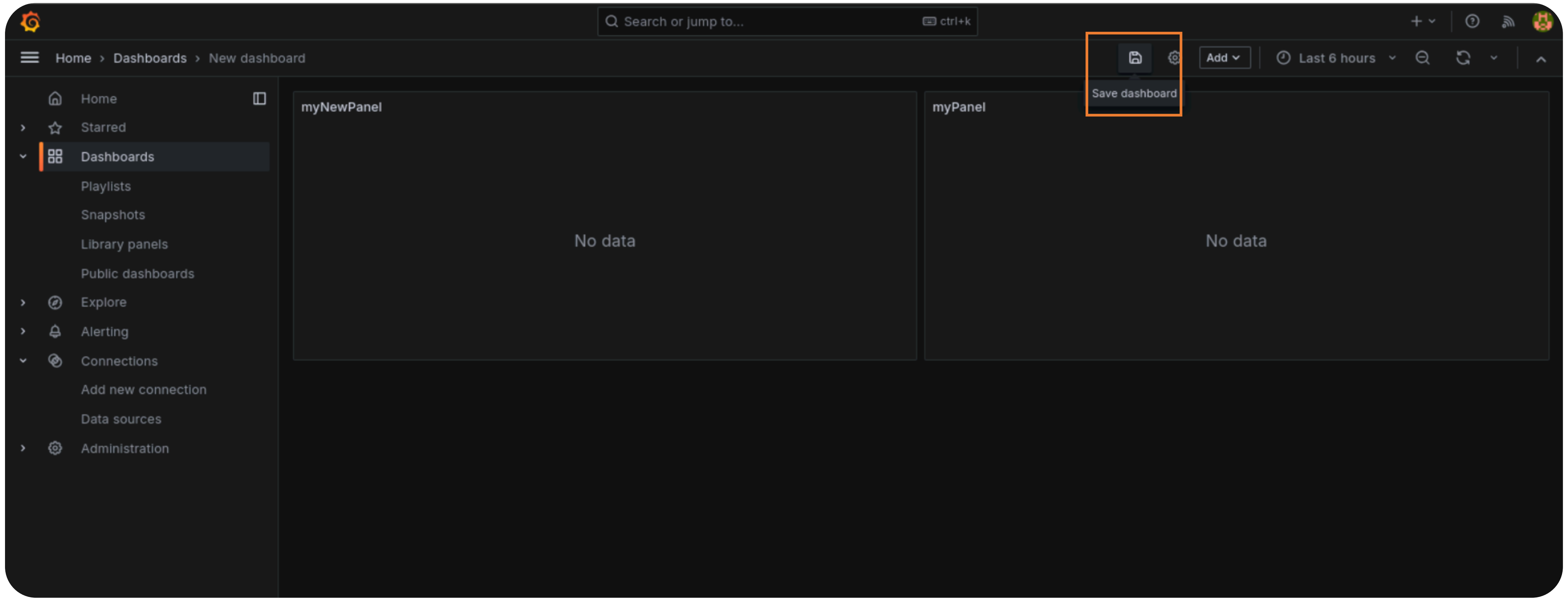
Configure Visualization and Panel Options

The dashboard consists of two panels as shown below:



Save and Finalize the Dashboard

Click on the save icon to save the dashboard as shown below:



NOTE

After saving, a prompt will appear to enter a title and description. Once saved, the dashboard will be ready to visualize data.

Visualizing Data

Grafana offers a variety of built-in visualizations to support different use cases, including:

Graphs

Display time-series data using various graph types

Gauges

Showcase key performance indicators or critical metrics at a glance

Tables

Organize detailed data for easy comparison of specific values

Heatmaps

Identify patterns or hotspots by representing data as colors

Visualizing Data

Grafana offers a variety of built-in visualizations to support different use cases, including:

Stat panels

Visualize key metrics using single values and optional graphs

Pie charts

Visualize data distributions or percentage

Histograms

Analyze data distribution over intervals

Scatter plots

Examine relationships between variables

Visualizing Data

Grafana offers a variety of built-in visualizations to support different use cases, including:

Geomaps

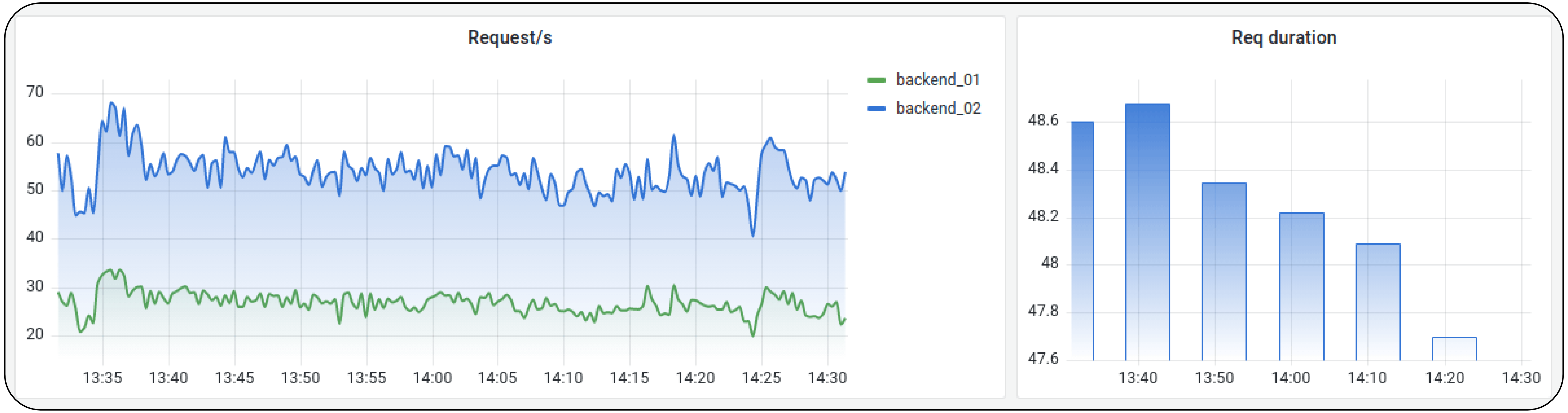
Identify spatial patterns with geographical visualizations

Plugins

Extend visualization capabilities for new or customized visualization types

Visualizing Data: Graphs

It displays time-series data in visual formats, such as line, bar, or area charts as shown below:

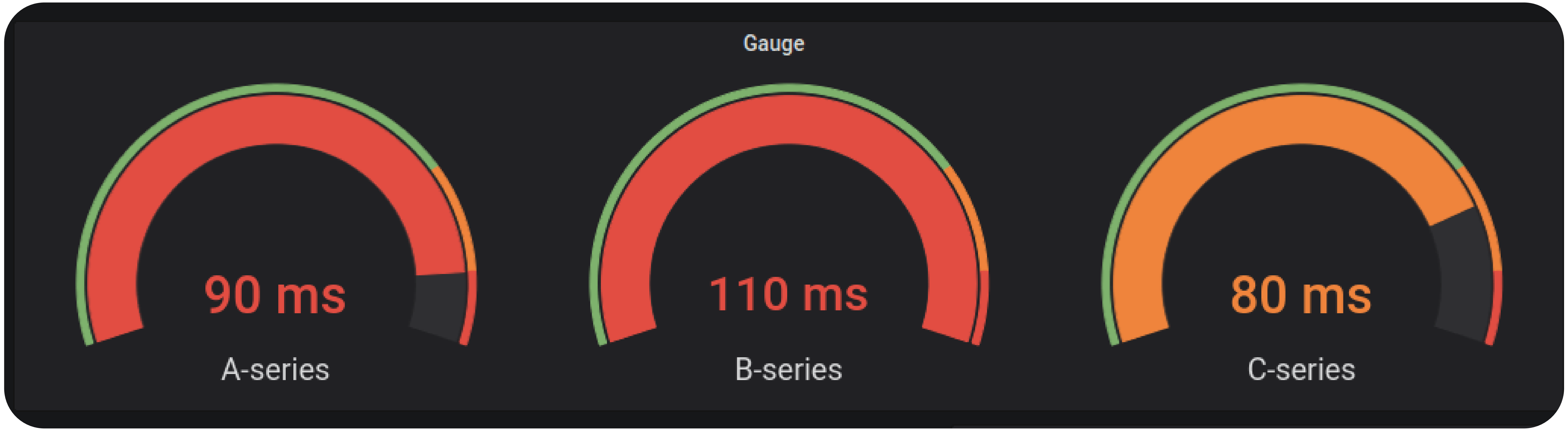


The line chart shows fluctuations over time and the bar chart displays the distribution of request duration values.

Visualizing Data: Gauge

It displays a single value that can be repeated for each series, column, or row.

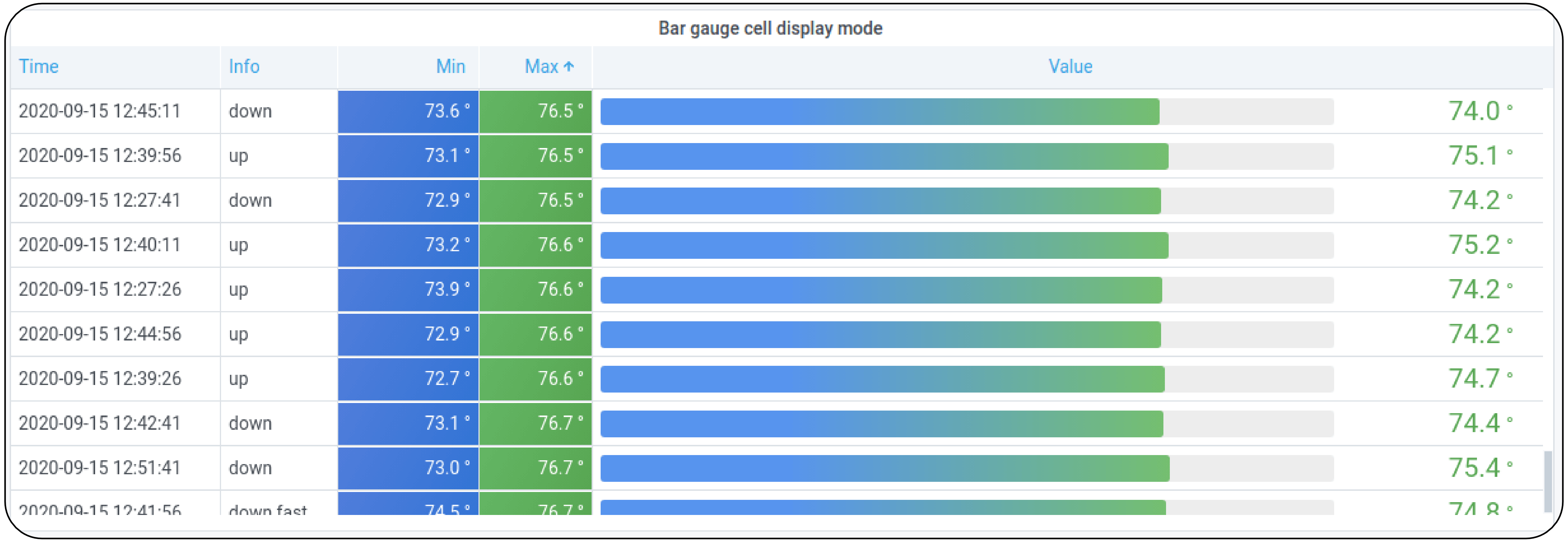
The following example shows this visualization:



This example shows the performance of different series, with values indicating response times and color-coding to reflect their status relative to defined thresholds.

Visualizing Data: Table

It displays time-series data or other types of data in a tabular format. The following example illustrates this visualization:



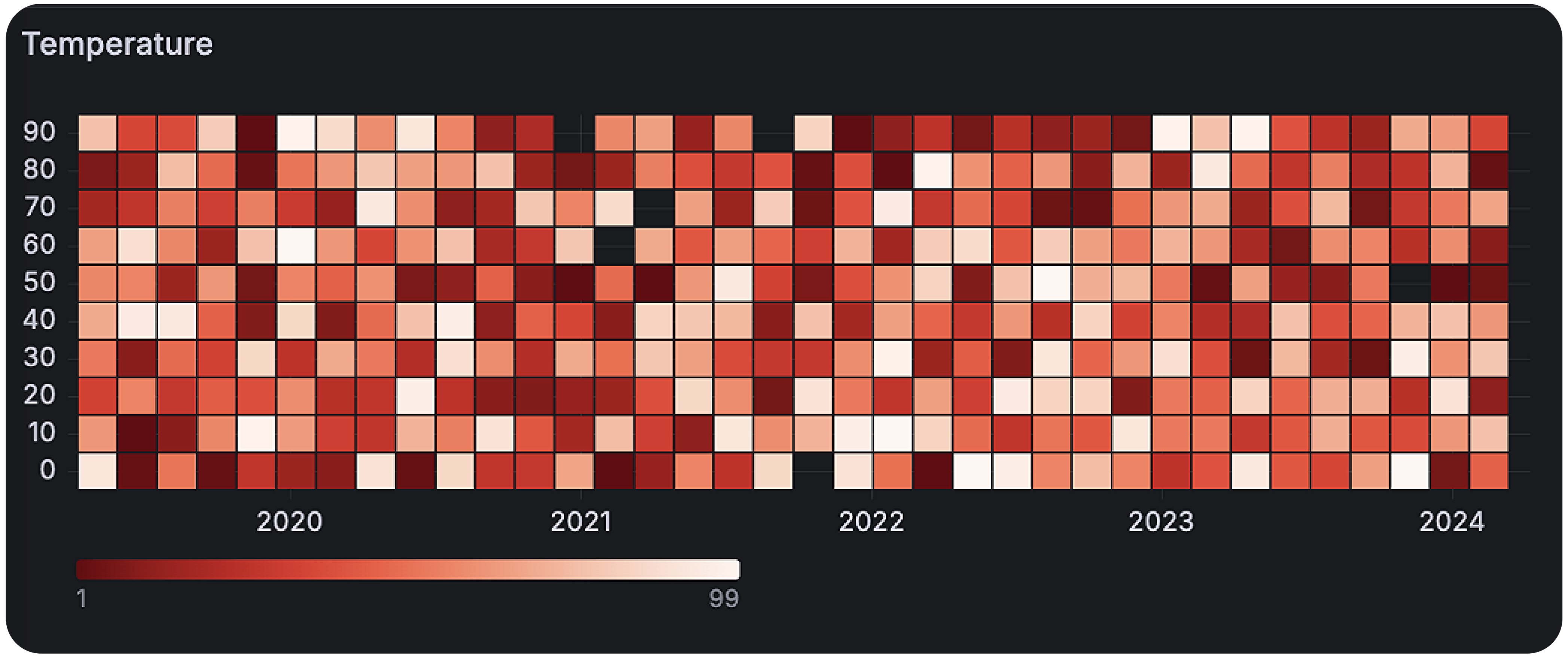
Bar gauge cell display mode

Time	Info	Min	Max ↑	Value
2020-09-15 12:45:11	down	73.6 °	76.5 °	74.0 °
2020-09-15 12:39:56	up	73.1 °	76.5 °	75.1 °
2020-09-15 12:27:41	down	72.9 °	76.5 °	74.2 °
2020-09-15 12:40:11	up	73.2 °	76.6 °	75.2 °
2020-09-15 12:27:26	up	73.9 °	76.6 °	74.2 °
2020-09-15 12:44:56	up	72.9 °	76.6 °	74.2 °
2020-09-15 12:39:26	up	72.7 °	76.6 °	74.7 °
2020-09-15 12:42:41	down	73.1 °	76.7 °	74.4 °
2020-09-15 12:51:41	down	73.0 °	76.7 °	75.4 °
2020-09-15 12:41:56	down fast	74.5 °	76.7 °	74.8 °

The above example represents temperature data across time; color-coded from blue (cooler) to green (warmer).

Visualizing Data: Heatmap

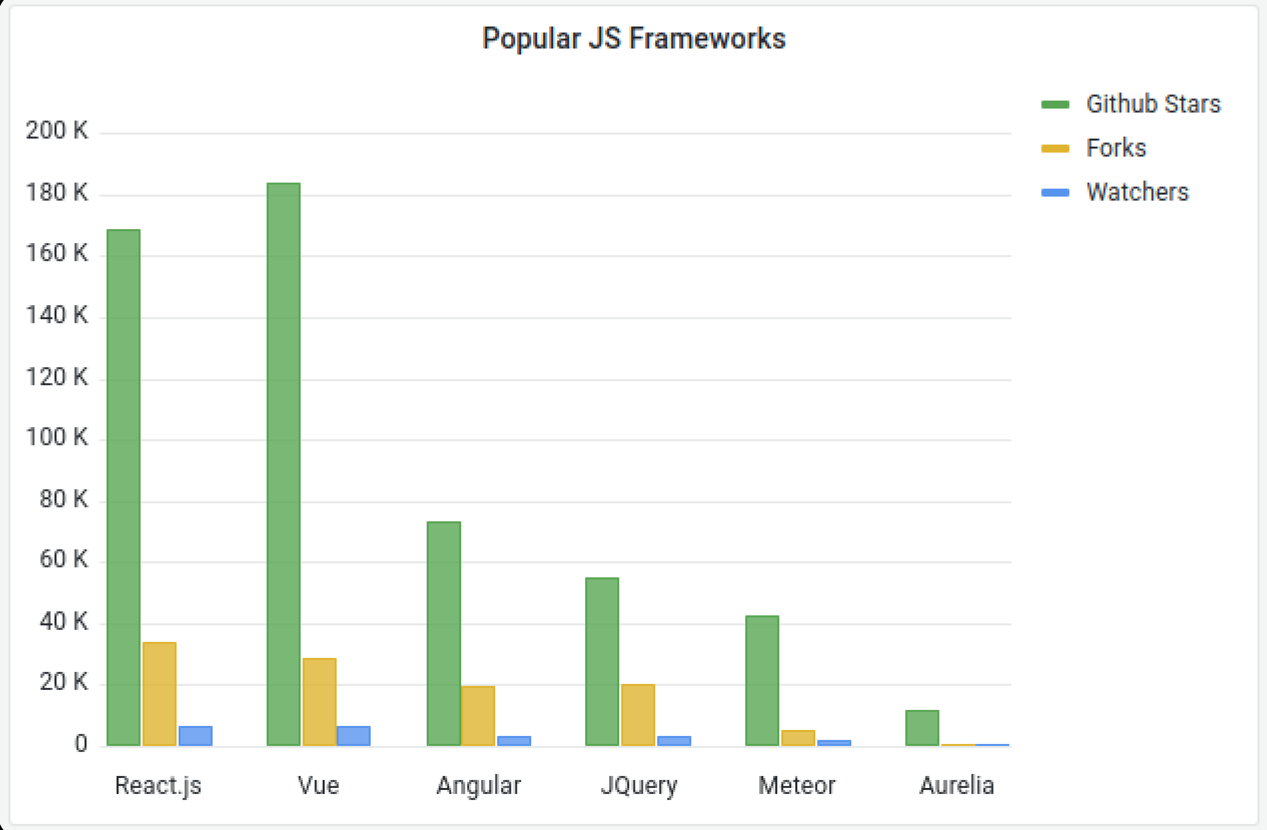
It displays data values in a two-dimensional space using color, with intensity or gradient reflecting the magnitude. The following visualization illustrates this:

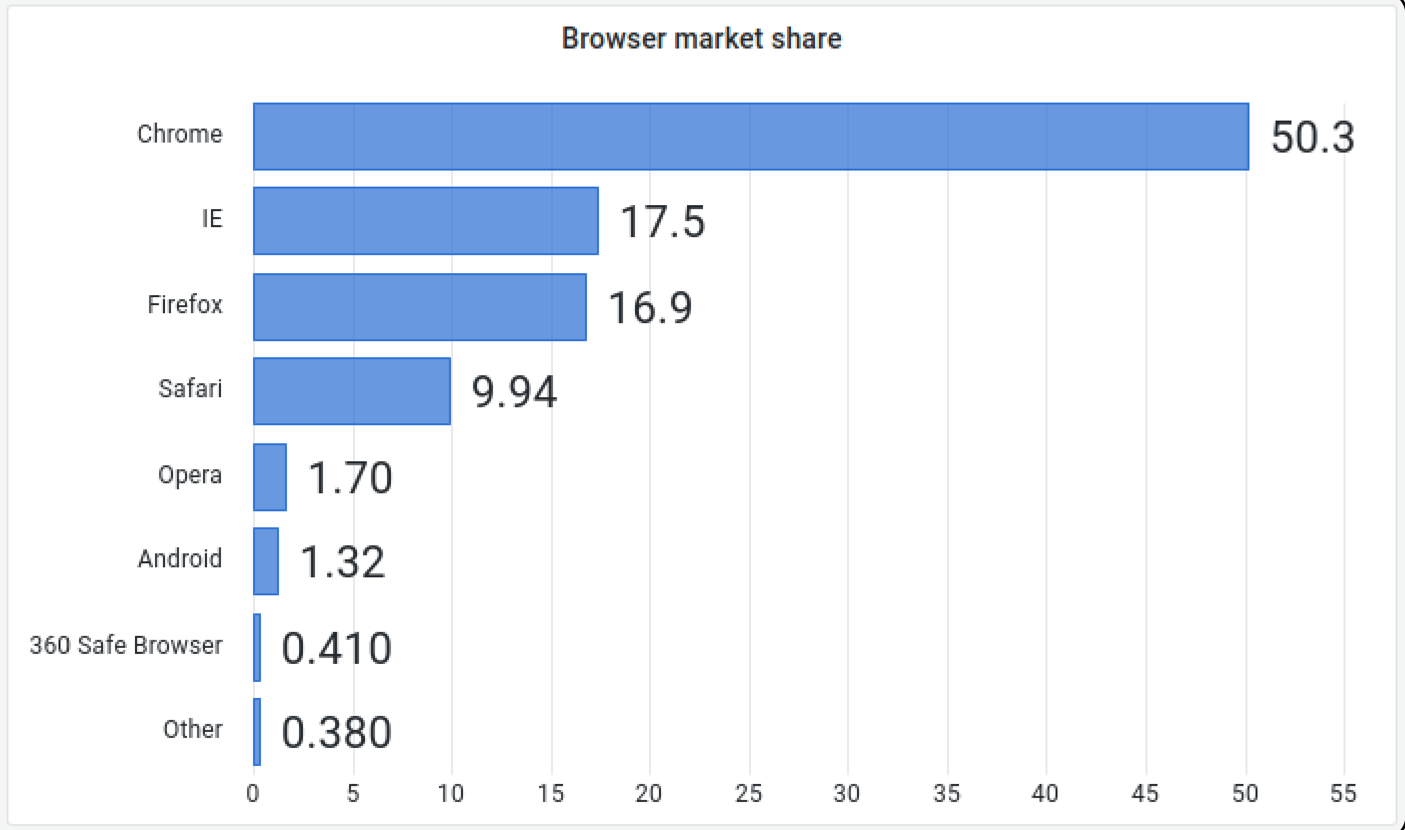


The above example shows temperature distribution over time, with color intensity indicating frequency within specific ranges from 2020 to 2024.

Visualizing Data: Bar Chart

It displays data with rectangular bars, where the length of each bar is proportional to the value as shown below:

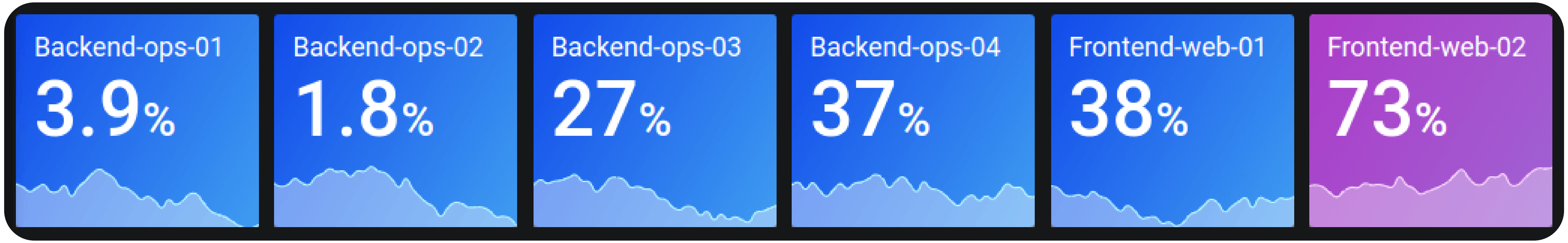




The left chart compares the popularity of JavaScript frameworks, while the right chart displays browser market share.

Visualizing Data: Big Numbers and Stats

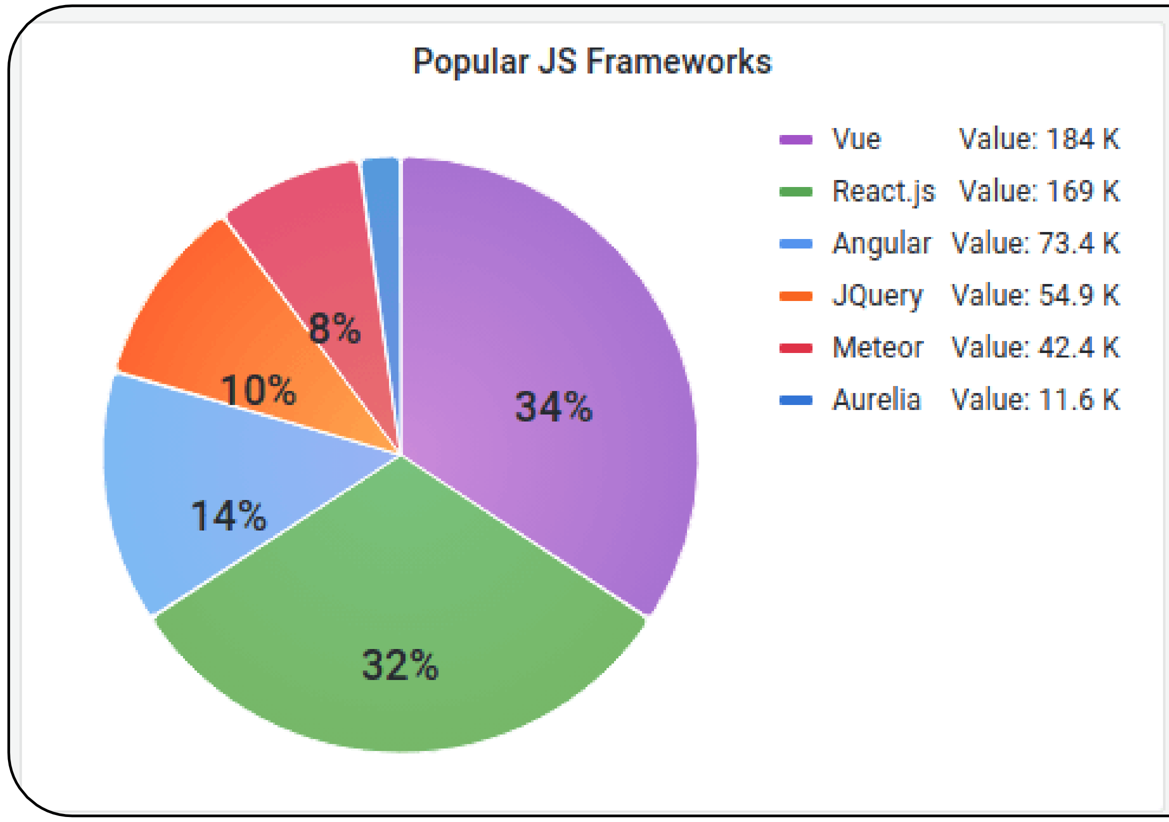
It displays a data in statistical metrics or aggregate.

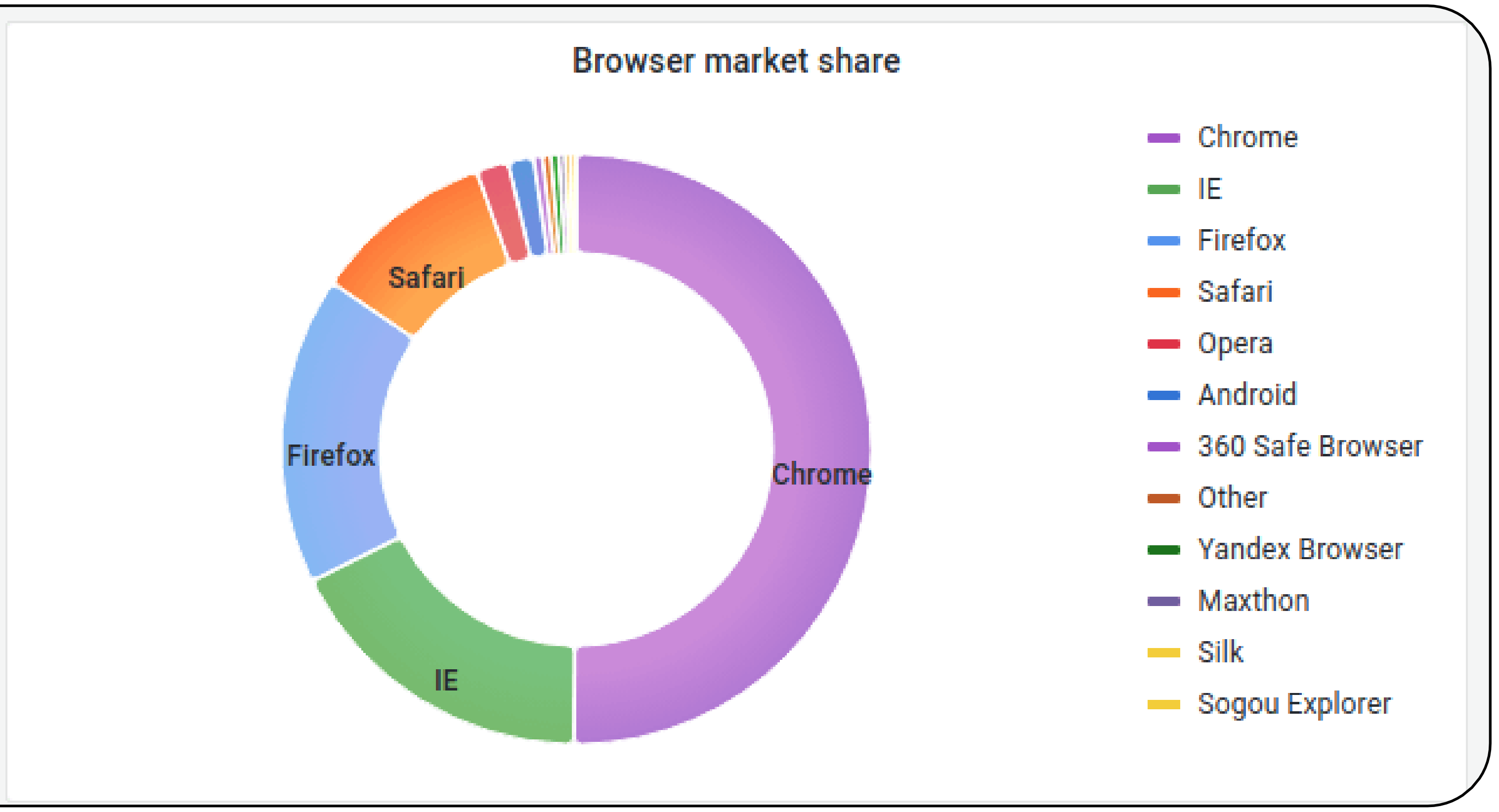


The above example shows the visualization of key metric values for different operations, with sparklines showing recent trends and colors indicating performance levels based on set thresholds.

Visualizing Data: Pie Chart

It displays data as slices of a circle, with each slice representing a proportion of the whole for easy comparison of categories. The following example illustrate this chart:

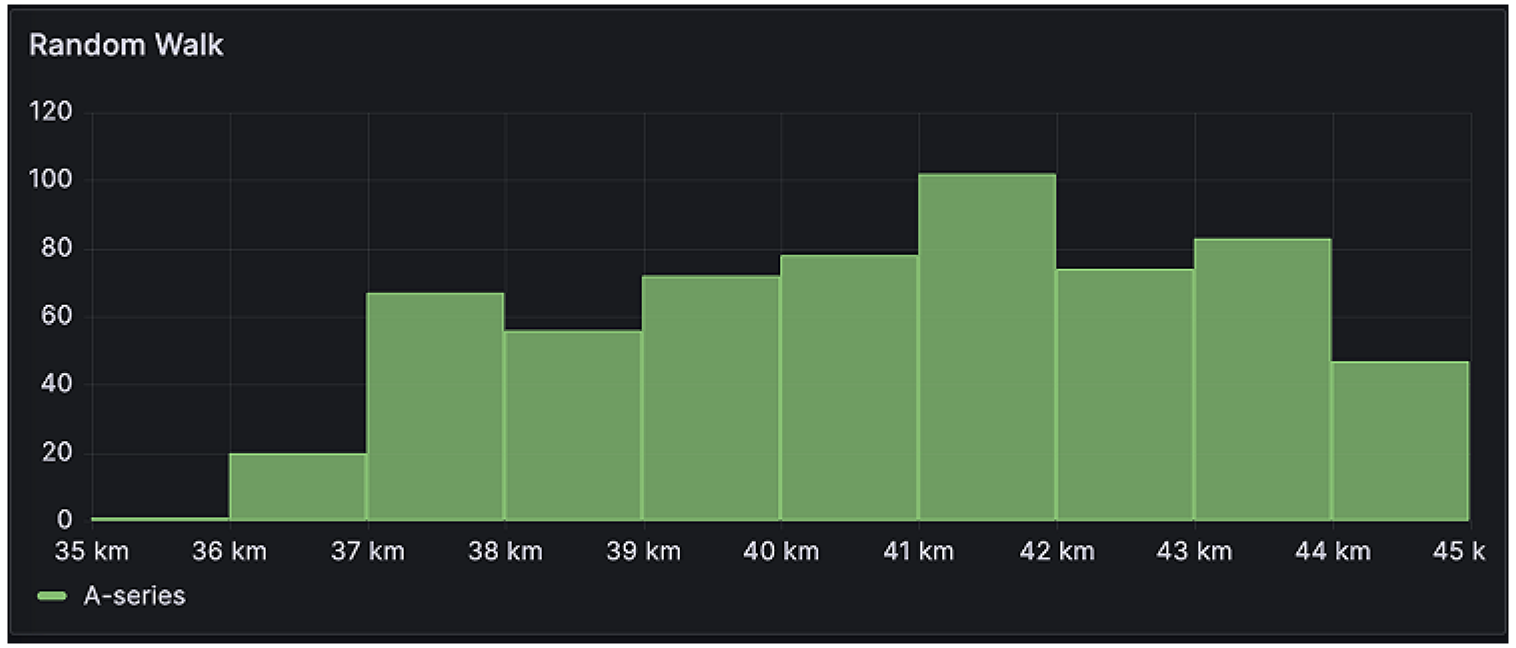




This example represents the distribution of popular JavaScript frameworks and the market share of various web browsers, showing their respective proportions within their categories.

Visualizing Data: Histogram

It is a graphical representation that organizes and displays the distribution of data into ranges or bins.

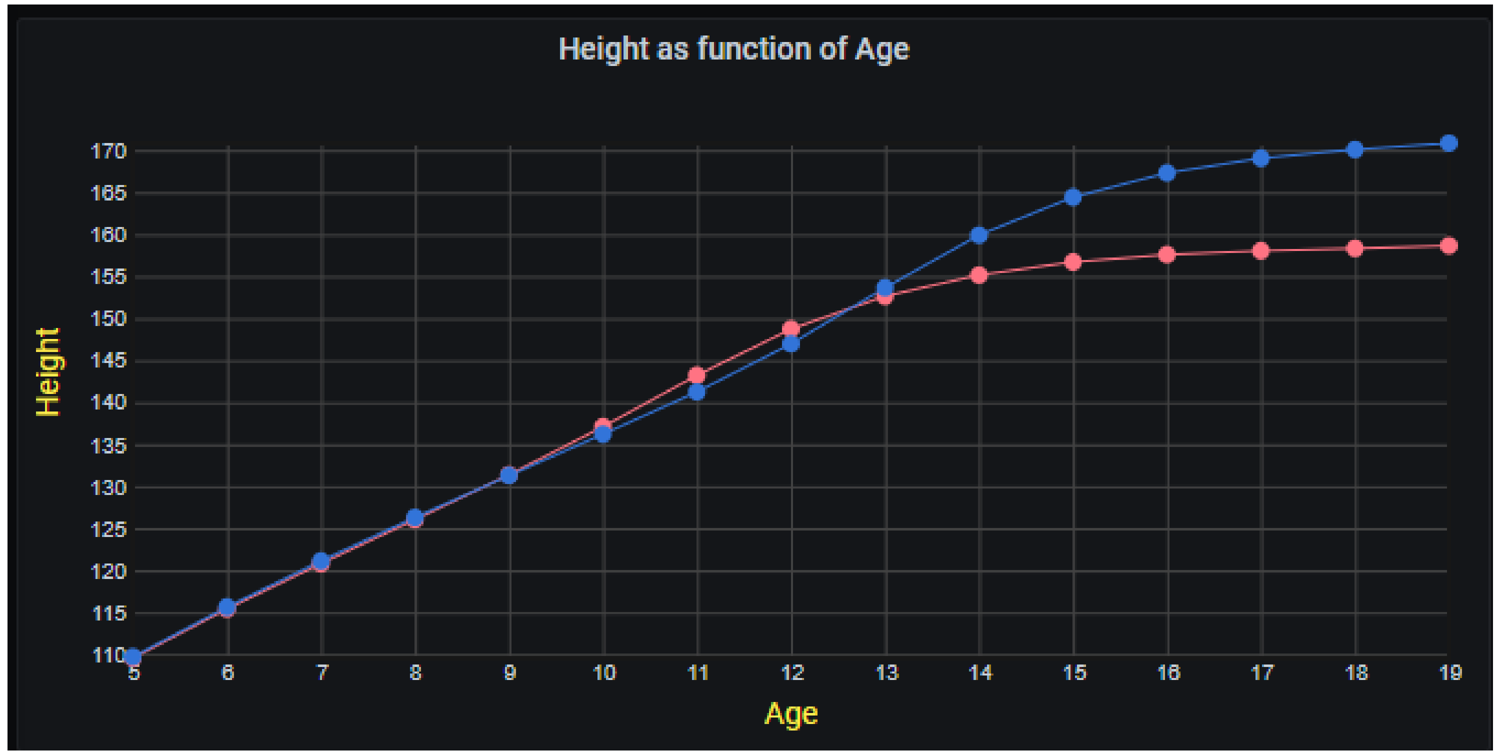


It is useful for showing the frequency of data points within specific intervals, helping to identify patterns or trends in the dataset.

Source: https://grafana.com/docs/grafana/latest/fundamentals/intro-histograms/

Visualizing Data: Scatter Plots

These are the graphs that display data points on a two-dimensional plane, showing relationships between two variables.



They help identify trends, correlations, and outliers within a dataset.

Source: https://grafana.com/grafana/plugins/michaeldmoore-scatter-panel/

Visualizing Data: Geo Maps

It represent data points on a geographical map, helping to visualize spatial relationships and trends.

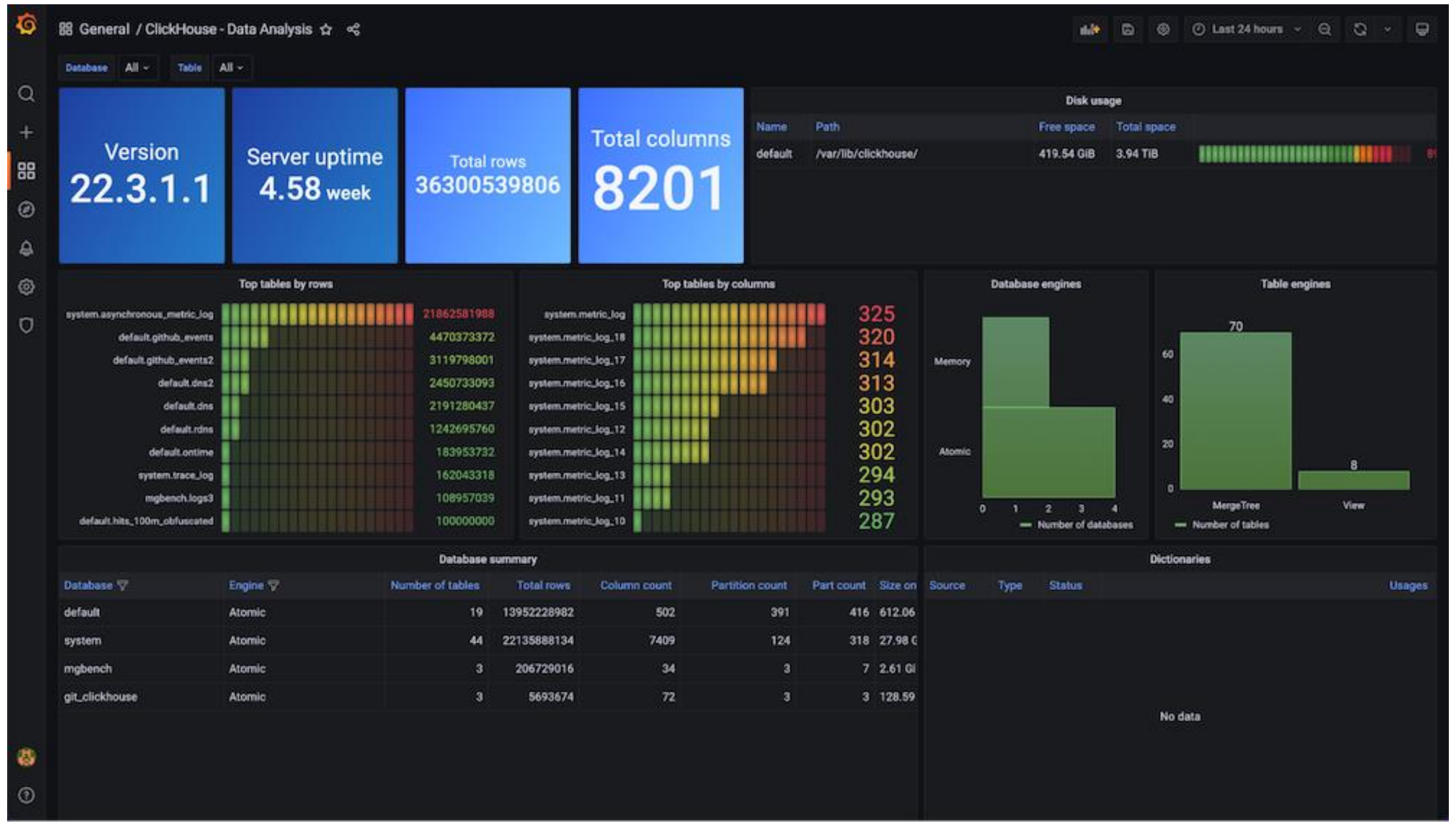
ASIA
AFRICA

They are useful for showing regional patterns, distributions, and geospatial insights across different locations.

Source: https://grafana.com/docs/grafana/latest/panels-visualizations/visualizations/geomap/

Visualizing Data: Plugins

They extend the functionality of a platform by integrating additional tools for visualizing and analyzing data.



Version 22.3.1.1

Server uptime 4.58 week

Total rows 36300539806

Total columns 8201

Disk usage

Name	Path	Free space	Total space
default	/var/lib/clickhouse/	419.54 GiB	3.94 TiB

Top tables by rows

system.asynchronous_metric_log	21862581988
default.github_events	4470373372
default.github_events2	3119798001
default.dns2	2450733093
default.dns	2191280437
default.rdns	1242695760
default.ontime	183953732
system.trace_log	162043318
mgbench.logs3	108957039
default.hits_100m_obfuscated	100000000

Top tables by columns

system.metric_log	325
system.metric_log_18	320
system.metric_log_17	314
system.metric_log_16	313
system.metric_log_15	303
system.metric_log_12	302
system.metric_log_14	302
system.metric_log_13	294
system.metric_log_11	293
system.metric_log_10	287

Database summary

Database	Engine	Number of tables	Total rows	Column count	Partition count	Part count	Size on
default	Atomic	19	13952228982	502	391	416	612.06
system	Atomic	44	22135888134	7409	124	318	27.98 (
mgbench	Atomic	3	206729016	34	3	7	2.61 Gi
git_clickhouse	Atomic	3	5693674	72	3	3	128.59

Dictionaries

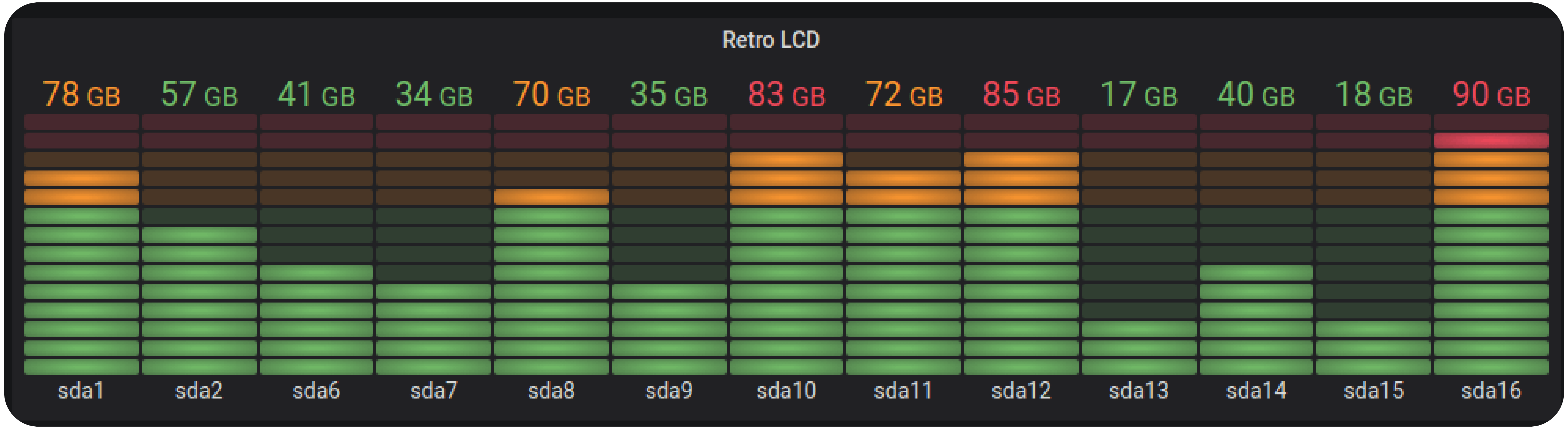
Source	Type	Status	Usages

No data

They offer custom dashboards and specialized data representations, enhancing the ability to monitor complex systems and metrics.

Visualizing Data: Bar Gauge

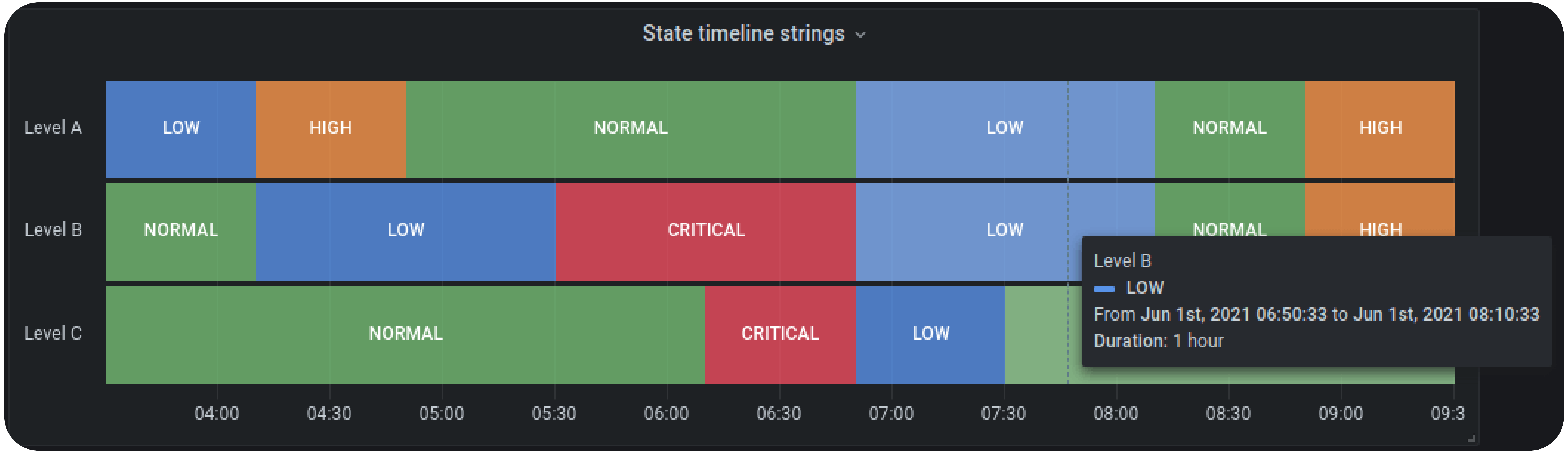
It displays data using horizontal or vertical bars, representing values within a defined range as shown below:



This visualization shows disk usage across sda1 to sda16, with color-coded bars indicating storage levels from green (low usage) to red (high usage).

Visualizing Data: State Timeline

It displays discrete state changes over time, visualizing how a system or metric transitions between different states. This is shown in the following example:



Level B
LOW
From Jun 1st, 2021 06:50:33 to Jun 1st, 2021 08:10:33
Duration: 1 hour

The above example visualizes three levels (A, B, C) with states (LOW, HIGH, NORMAL, CRITICAL) from 04:00 to 09:30, with each state color-coded and tooltips offering duration details.

Grafana Alerting

It is a feature that enables users to set up notifications based on the metrics and logs collected from various data sources.

Key features:

- **Multi-source queries:** Creates queries and expressions from various data sources
- **Flexible combinations:** Combines data to alert on metrics and logs in unique ways
- **Centralized management:** Manages and acts on alerts from a single, consolidated view

Working of Grafana Alerting

The following steps provide an overview of Grafana's alert process:

1. Grafana alerting continuously queries data sources and checks if conditions defined in the alert rules are met.
2. When a specified condition is breached (for example, disk usage exceeds 90%), an alert instance is fired.
3. The firing alert is sent to the contact point or through notification policies

Grafana Notification

It determines how, when, and where alert notifications are sent, directing them to the appropriate recipients or systems to optimize issue resolution and reduce unnecessary alerts.

Key features:

- **Customizable contact points:** Defines where alert notifications are sent to ensure they reach the right recipients
- **Flexible routing:** Routes alerts to different contact points using the notification policy tree based on alert rules
- **Impact reduction:** Configures notifications to reduce noise and focus on critical issues

Working of Grafana Notification

The following steps provide an overview of Grafana's notification process:

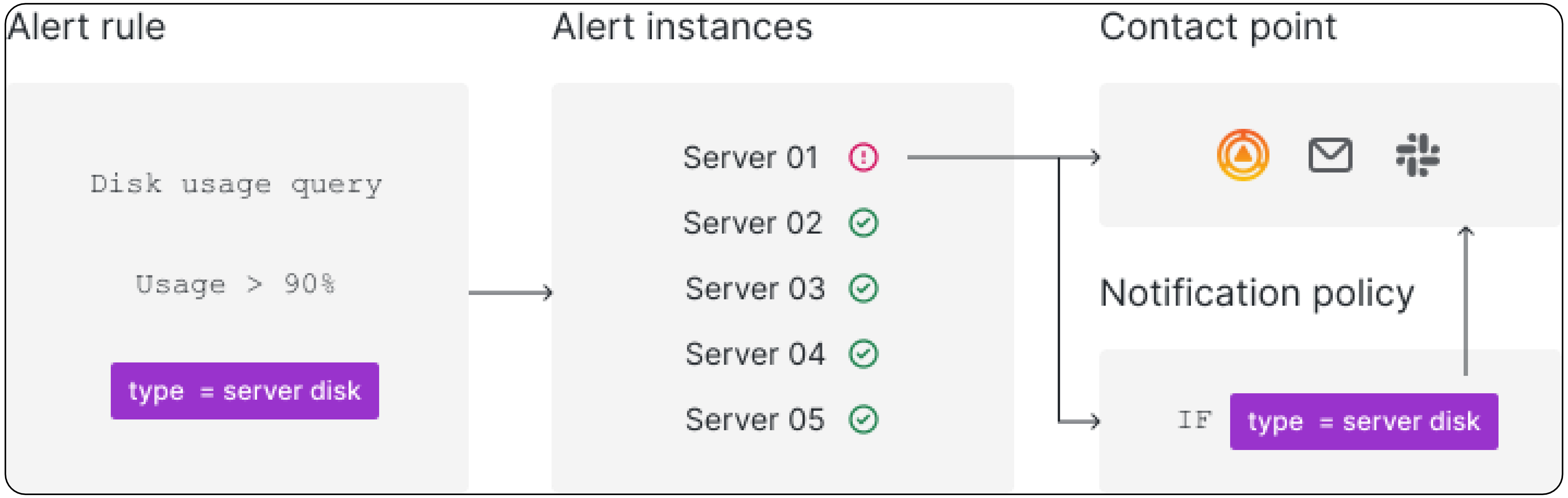
1. Grafana evaluates alert rules and triggers notifications for firing or resolved instances.

2. Triggered notifications are then sent directly to a contact point or routed via notification policies

3. Notifications are grouped to reduce alert noise and minimize alert fatigue, improving overall clarity

Grafana Alerting and Notifications

The following diagram provides an overview of Grafana alerting and notification:



The above diagram periodically queries data sources to evaluate alert conditions. When conditions are met, alerts are triggered, and notifications are sent based on predefined policies.

Grafana Alerting and Notifications

The following key elements provide an overview of Grafana's approach to managing alerting and notifications:

Alert rule	Set the initial monitoring condition, such as a disk usage query, for the specified server disk
Alert instances	Monitor multiple servers (Server01, Server02, and more) and display alerts as red dots when conditions are met
Contact point	Send alerts through channels like email and Slack when a condition triggers
Notification policy	Define rules for handling notifications based on the server disk type

Assisted Practice

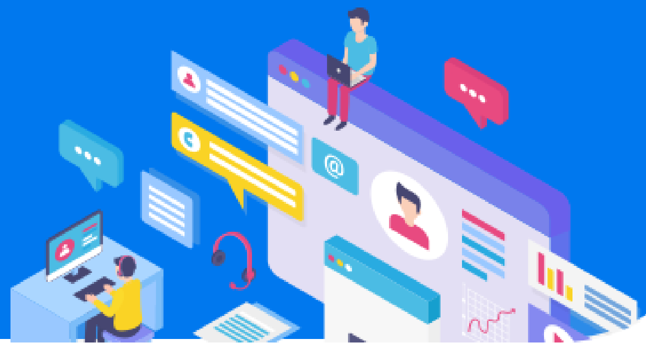
Setting up a Grafana Instance **Duration: 10 Min.**

Problem statement:
You have been assigned a task to demonstrate the process of installing Grafana, starting the server, and configuring it to monitor and visualize system metrics through a dashboard.

Outcome:
By the end of this demo, you will be able to install Grafana, start the server, and configure it to monitor and visualize system metrics through a dashboard.

Note: Refer to the demo document for detailed steps:

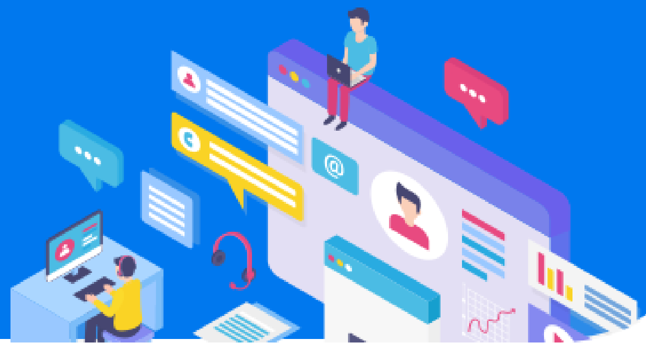
Assisted Practice: Guidelines



Steps to be followed:

1. Install Grafana from the APT repository
2. Start the Grafana server
3. Access the Grafana UI

Assisted Practice



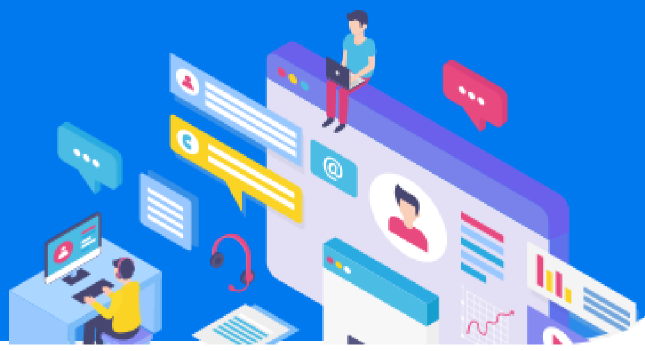
Building a Multi-Panel Dashboard in Grafana **Duration: 20 Min.**

Problem statement:
You have been assigned a task to demonstrate the process of configuring Prometheus in Grafana and creating a dashboard for comparing resource utilization across different servers using Node Exporter metrics.

Outcome:
By the end of this demo, you will be able to configure Prometheus in Grafana and create a multi-panel dashboard for comparing resource utilization across different servers using Node Exporter metrics.

Note: Refer to the demo document for detailed steps:

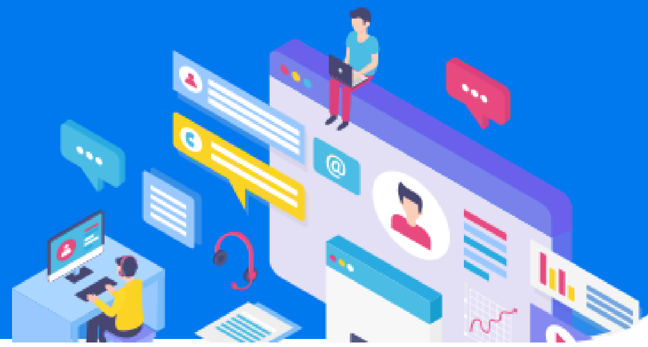
Assisted Practice: Guidelines



Steps to be followed:

1. Configure Prometheus as a data source in Grafana
2. Create a dashboard in Grafana for visualizing Prometheus metrics

Assisted Practice



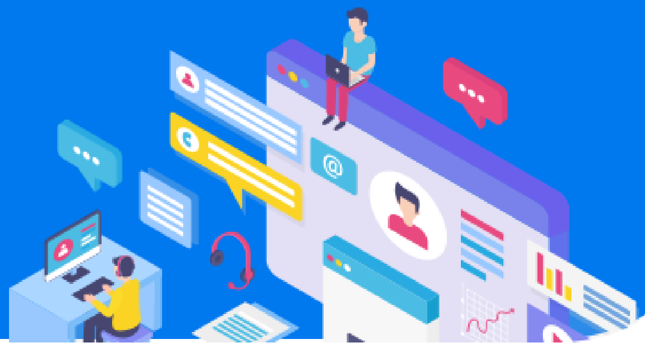
Configuring Email Alerts for Critical System Thresholds **Duration: 10 Min.**

Problem statement:
You have been assigned a task to establish an email notification system in Grafana for alerting engineers when critical system metrics exceed predefined thresholds by configuring the necessary Grafana settings.

Outcome:
By the end of this demo, you will be able to configure email alerts in Grafana to notify engineers when critical system metrics exceed predefined thresholds.

Note: Refer to the demo document for detailed steps:

Assisted Practice: Guidelines



Steps to be followed:

1. Set an app password through your Gmail account
2. Configure SMTP settings in the Grafana configuration file
3. Configure a contact point in the Grafana dashboard
4. Configure Notification policies
5. Configure alert rules and verify the email alert notifications

Quick Check

A Grafana user needs to set up an alert for a critical metric and ensure that the appropriate team members are informed based on their responsibilities. Which Grafana feature is best suited for configuring this setup?

A. Visualization types

B. Data source integration

C. Notification policies

D. Query language

Integrating Grafana with Prometheus

Configuring Grafana to Connect to a Prometheus Instance

The following are the steps to create a Prometheus data source in Grafana:

1. Navigate to the configuration menu by clicking the **cogwheel** icon in the sidebar

2. Select **Data Sources** from the menu

3. Click on the **Add data source** option

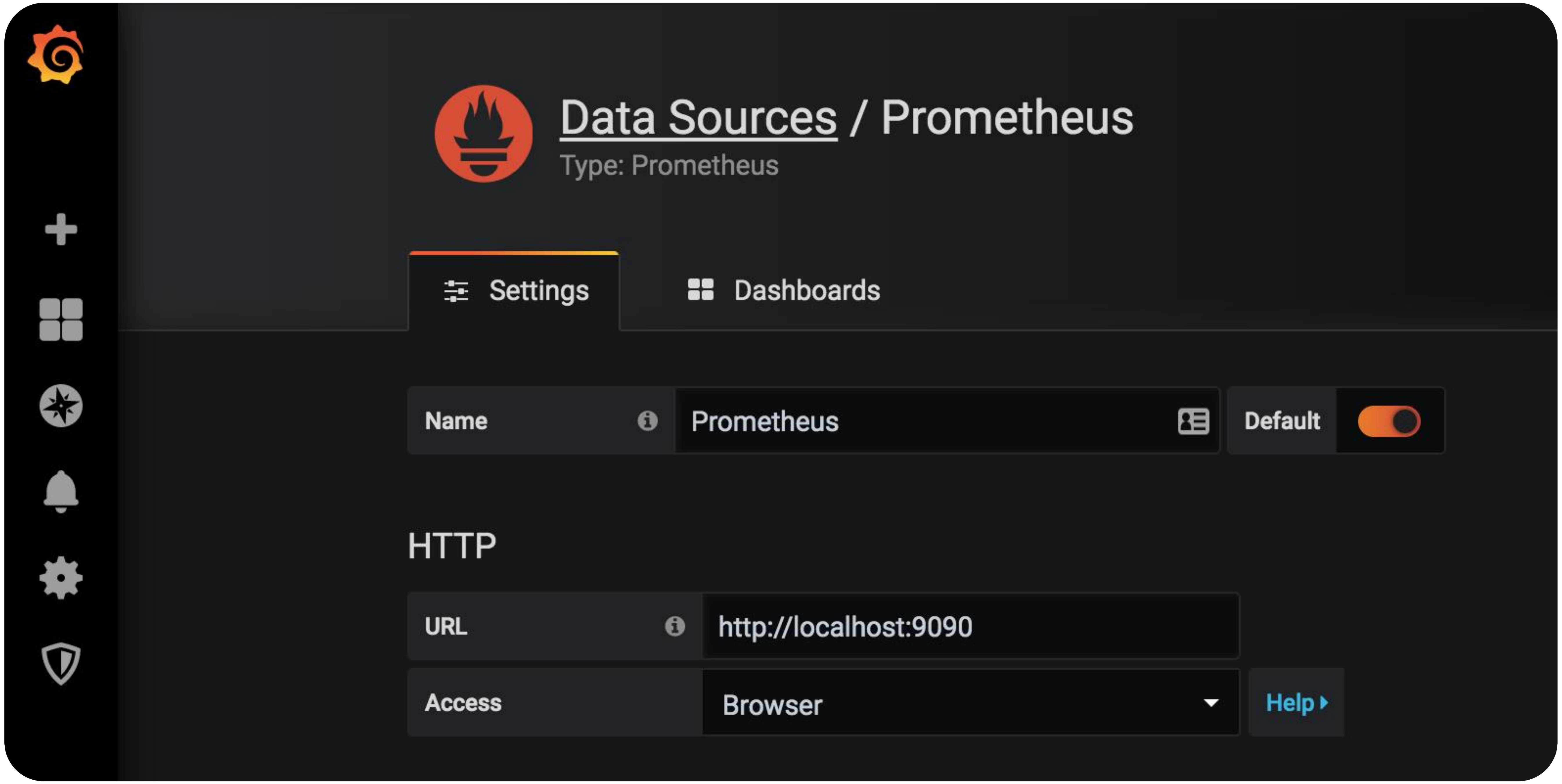
4. Choose **Prometheus** as the data source type

5. Enter the Prometheus server URL (for example, **http://localhost:9090/**)

6. Adjust additional settings if needed, then click **Save & Test**

Configuring Grafana to Connect to a Prometheus Instance

An example of a data source configuration is illustrated as follows:


Data Sources / Prometheus
Type: Prometheus

Name: Prometheus
Default: on

HTTP

URL: http://localhost:9090
Access: Browser

Utilizing PromQL Directly within Grafana Panels

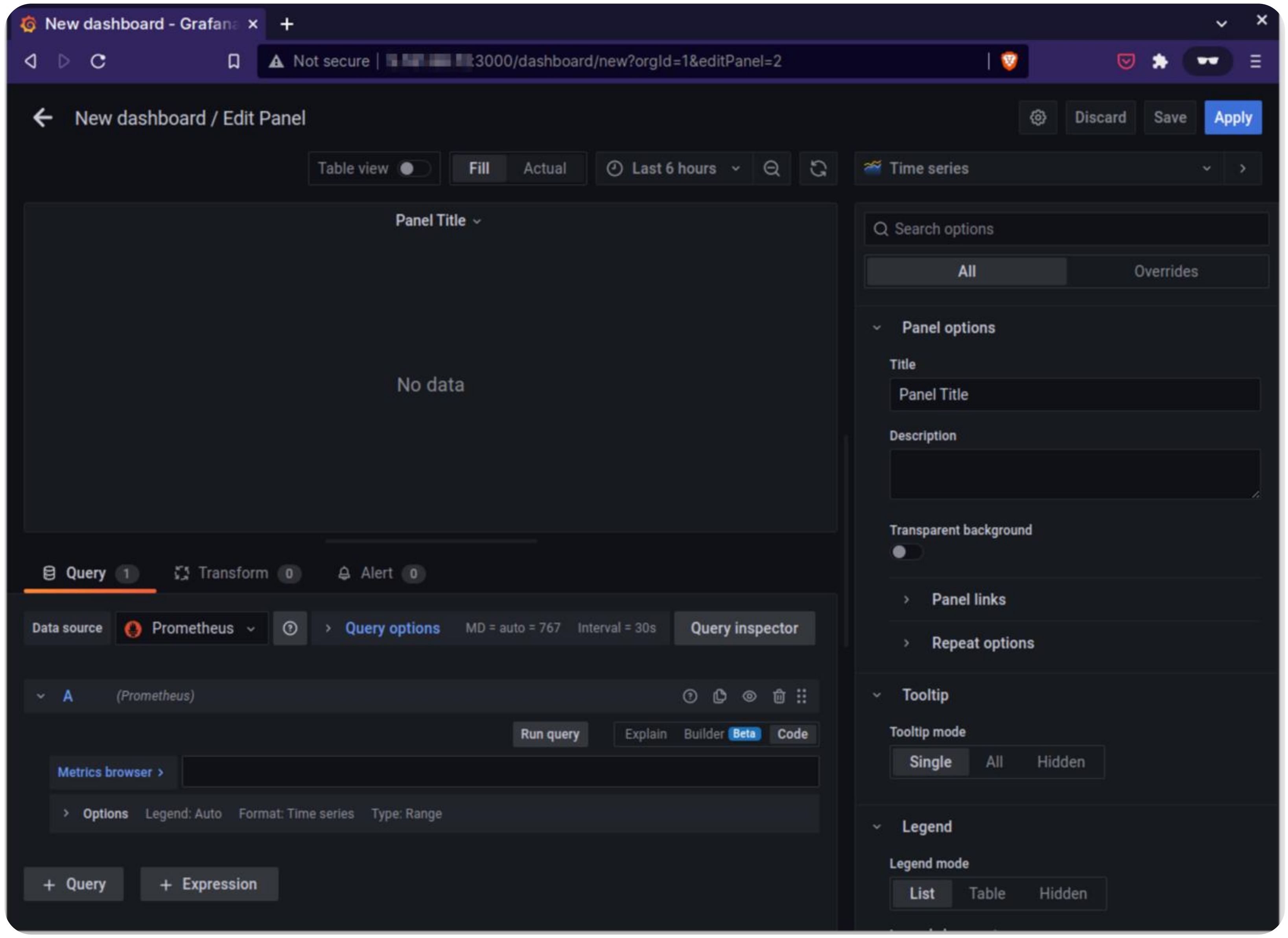
Here are the steps to use PromQL in Grafana panels to query and display Prometheus metrics:

- Create a new panel
- Enter and run PromQL query
- Apply and save the panel

Utilizing PromQL Directly within Grafana Panels

Create a new panel

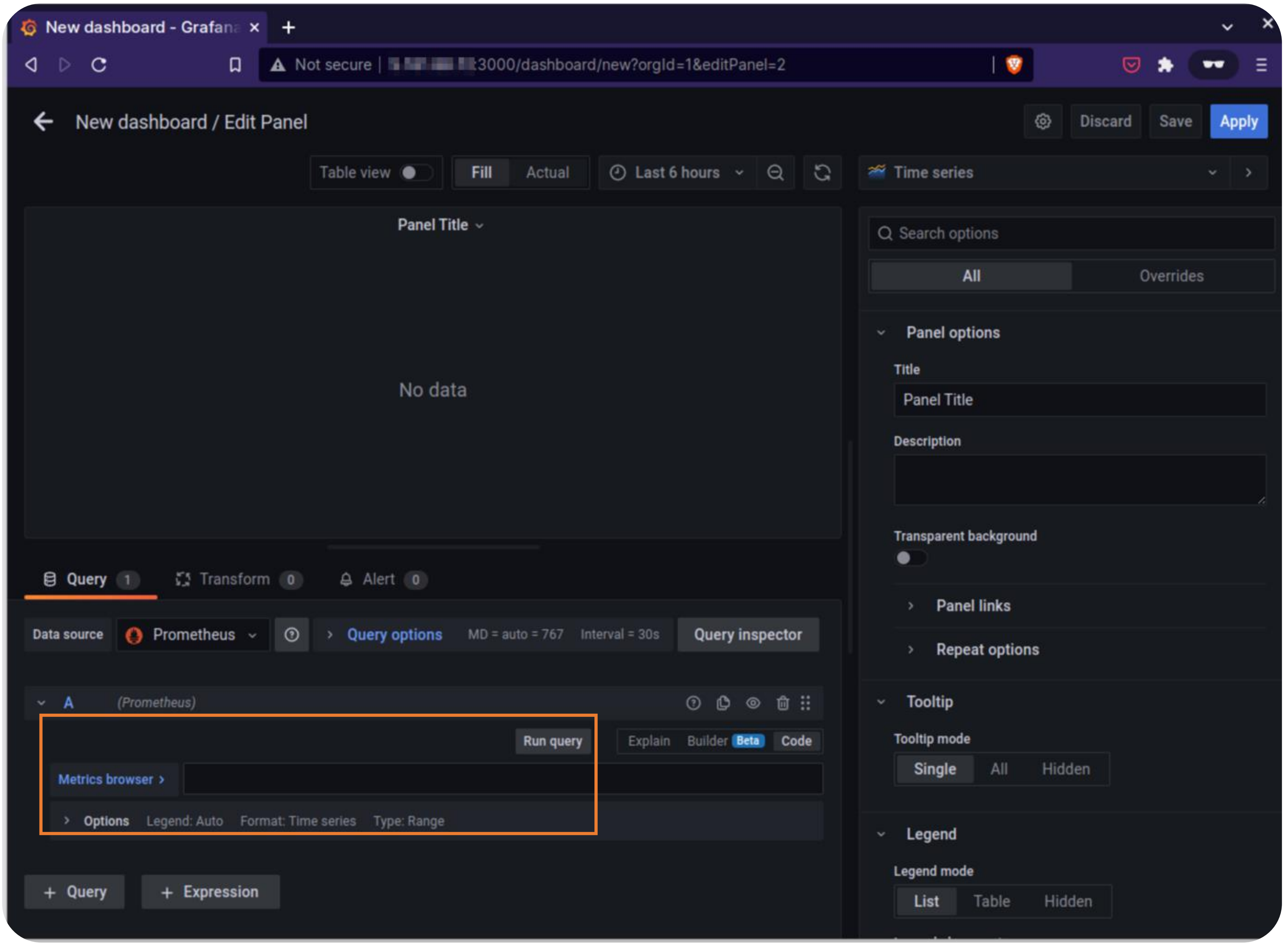
Select data source to create a new panel or select a panel from library



Utilizing PromQL Directly within Grafana Panels

Enter and run PromQL query

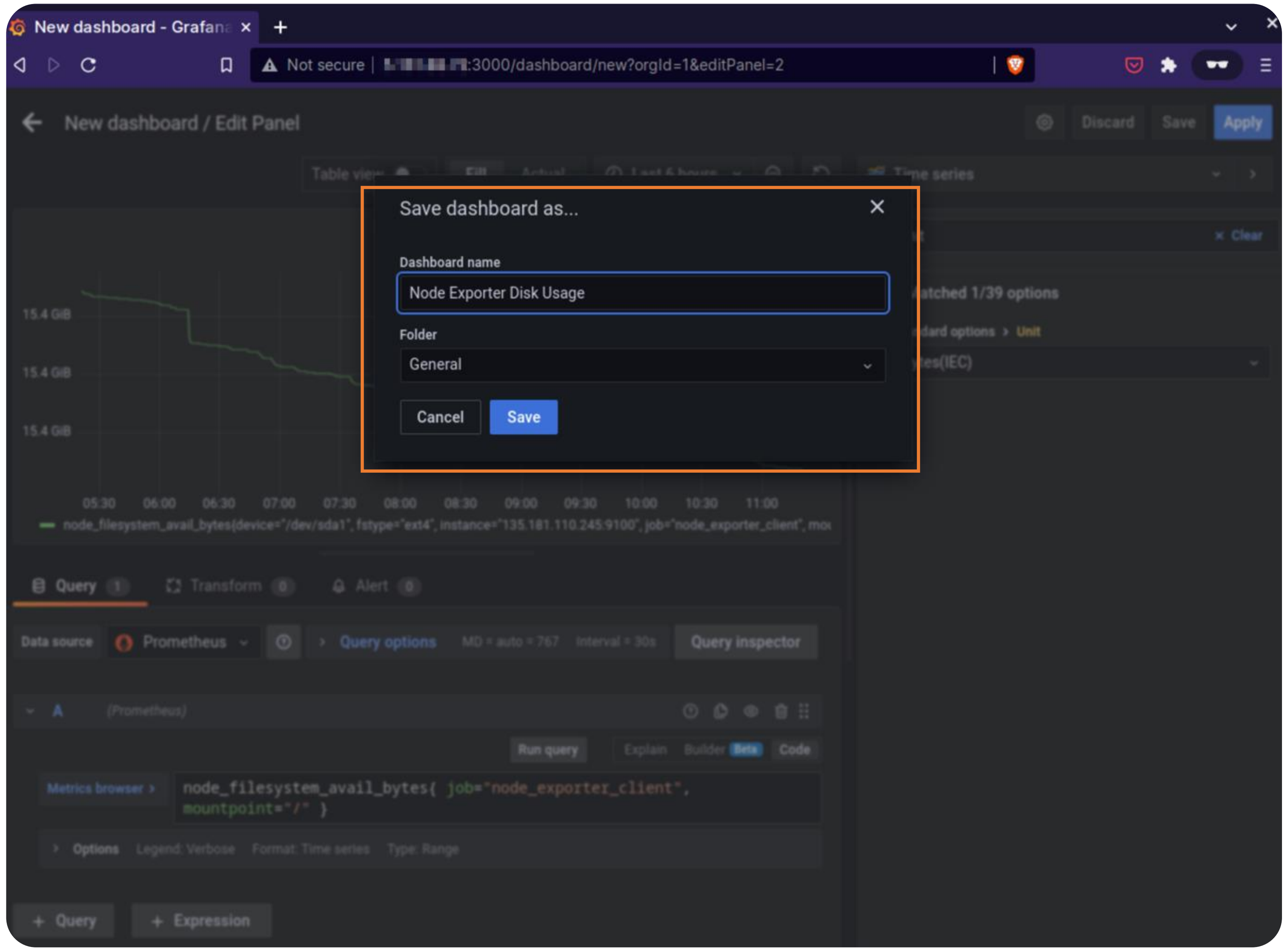
Input the PromQL query in the metrics browser field, then click **Run query** to visualize the data



Utilizing PromQL Directly within Grafana Panels

Apply and save the panel

Click **Apply**, then click **Save** to store the panel. Then, enter the dashboard name and confirm by clicking **Save**.



[redacted]:3000/dashboard/new?orgId=1&editPanel=2

Dashboard name: Node Exporter Disk Usage

Folder: General

```
node_filesystem_avail_bytes{ job="node_exporter_client",
mountpoint="/" }
```

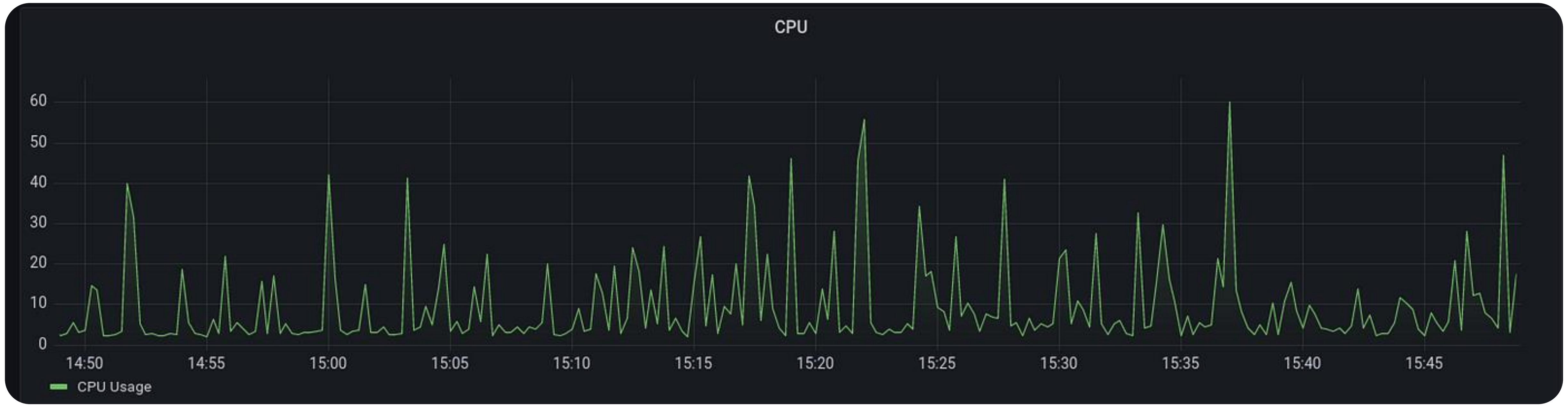
Grafana Panels Using PromQL: CPU Panel

The following example demonstrates how to visualize CPU usage data in Grafana:

Example

```
100 * (1 - avg by(instance)

(irate(node_cpu_seconds_total{mode='idle',instance=~'$instance.*'}[1m])))
```



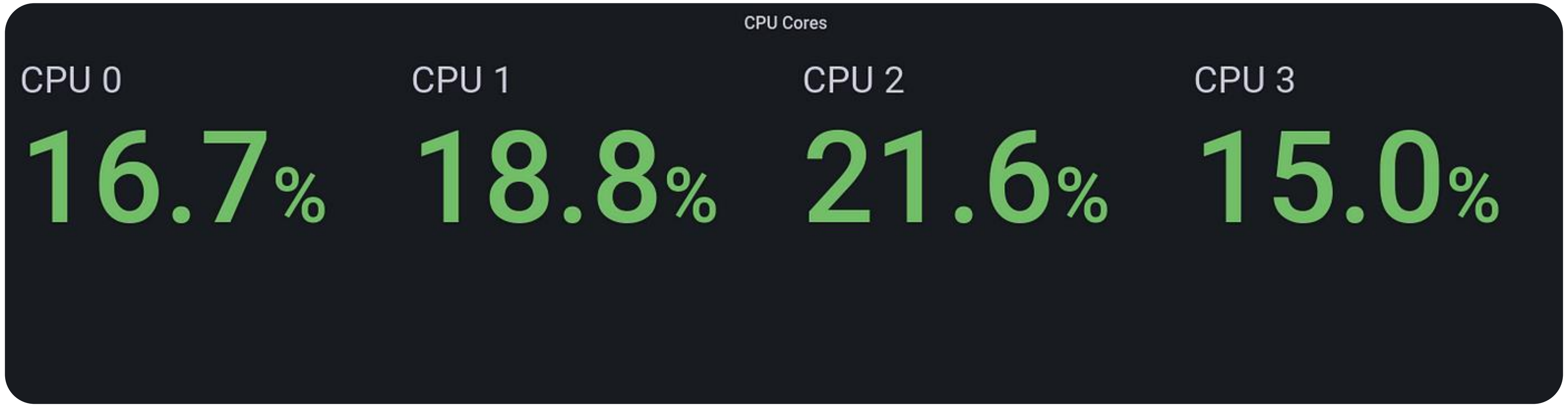
Grafana Panels Using PromQL: CPU Cores Panel

The below example shows a Grafana panel displaying the percentage usage of each CPU cores:

Example

```
100 * (1 -

(irate(node_cpu_seconds_total{mode='idle',instance=~'$instance.*'}[1m])))
```


CPU Cores

CPU 0 16.7%

CPU 1 18.8%

CPU 2 21.6%

CPU 3 15.0%

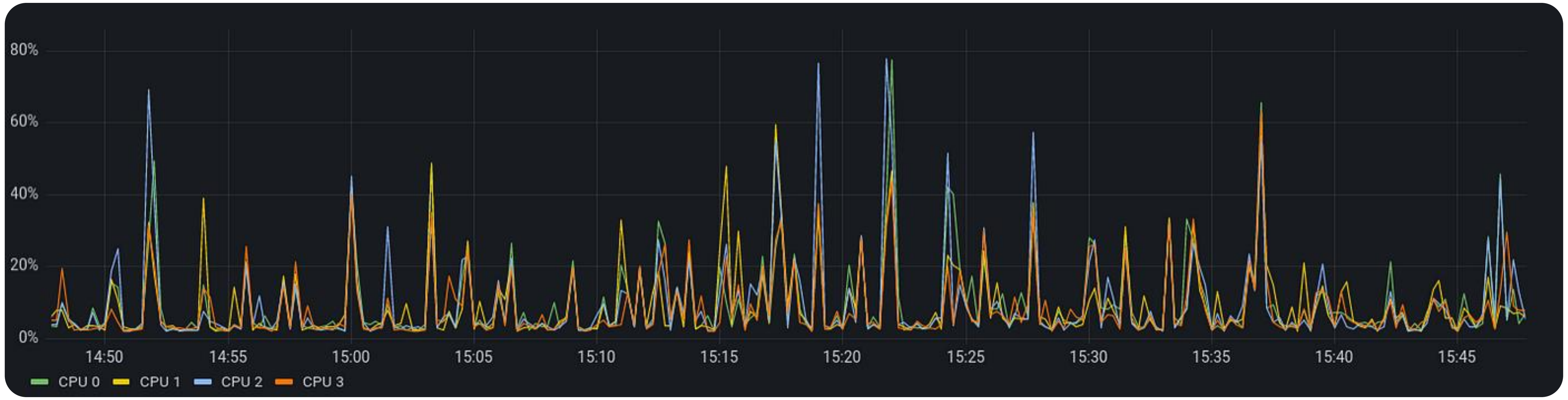
Grafana Panels Using PromQL: CPU Cores Usage Panel

The following example shows a Grafana panel visualizing CPU usage over time:

Example

```
100 * (1 -

(irate(node_cpu_seconds_total{mode='idle',instance=~'$instance.*'}[1m])))
```



Benefits of Using PromQL Directly within Grafana Panels

The following are the benefits:

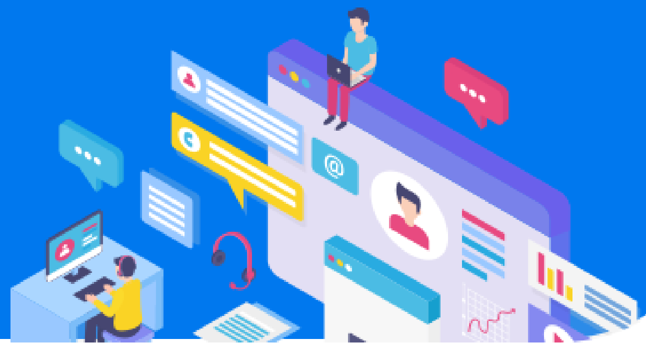
Direct access to Prometheus data
Enables querying of Prometheus data directly in Grafana for real-time insights

Advanced data manipulation
Utilizes PromQL functions and operators for complex data analysis

Seamless integration
Streamlines visualizations by integrating PromQL into Grafana panels

Enhanced customization
Allows creation of tailored visualizations with PromQL for specific metrics

Assisted Practice



Creating a Grafana Dashboard Using PromQL Queries to Visualize Specific Application Metrics

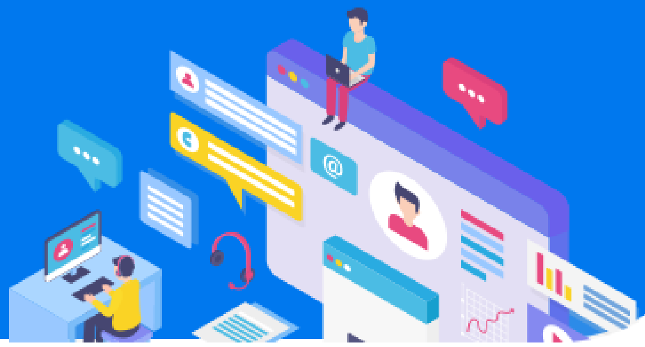
Duration: 10 Min.

Problem statement:
You have been assigned a task to demonstrate the process of creating a Grafana dashboard using PromQL queries to visualize specific application metrics.

Outcome:
By the end of this demo, you will be able to create a Grafana dashboard using PromQL queries to visualize specific application metrics effectively.

Note: Refer to the demo document for detailed steps:

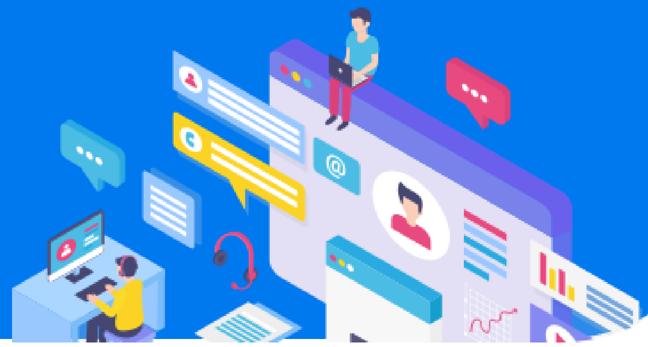
Assisted Practice: Guidelines



Steps to be followed:

1. Create a new dashboard in Grafana to add visualizations
2. Use PromQL queries to visualize the metrics collected

Assisted Practice



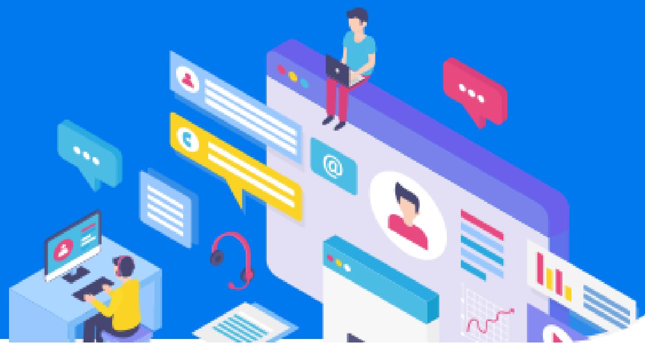
Setting up Docker Monitoring Using Prometheus and Grafana **Duration: 10 Min.**

Problem statement:
You have been assigned a task to demonstrate the process of setting up Docker monitoring using Prometheus and Grafana for visualizing system and container metrics through a Grafana dashboard.

Outcome:
By the end of this demo, you will be able to set up Docker monitoring using Prometheus and Grafana to visualize system and container metrics through a comprehensive dashboard.

Note: Refer to the demo document for detailed steps:

Assisted Practice: Guidelines



Steps to be followed:

1. Configure Prometheus monitoring stack with Docker
2. Visualize Docker and host metrics in Grafana
3. Create custom panels for Docker metrics

Quick Check

An organization wants to visualize server metrics in Grafana. After installing Prometheus and Grafana, what is the first action required to start visualizing data from Prometheus in Grafana?

A. Create a new dashboard in Grafana

B. Configure Prometheus as a data source in Grafana

C. Install a Grafana plugin for Prometheus

D. Write PromQL queries in Grafana panels

Key Takeaways

- Grafana provides a unified platform for visualizing diverse data sources and creating insightful dashboards.
- Grafana's ability to integrate multiple data sources enhances its utility for varied monitoring needs.
- Effective dashboards in Grafana combine different visualizations to convey complex data in an accessible manner.
- Grafana's alerting system enables real-time notifications based on defined conditions, thereby improving response times.
- PromQL enhances data analysis in Grafana by allowing precise queries and aggregations of time-series data from Prometheus.

Monitoring Python Web Application Using Prometheus and Grafana

Duration: 25 Min.

Project agenda: To create a Jenkinsfile pipeline script in GitHub and use it to set up a Jenkins pipeline job for cloning, compiling, testing, and packaging the codebase

Description: You are a DevOps engineer managing a web application on GitHub. To enhance the efficiency of the deployment process, you have taken the initiative to set up a Jenkins server. As part of this setup, a Jenkinsfile must be integrated into your project's GitHub repository. This Jenkinsfile is responsible for orchestrating essential tasks such as code checkout, Maven-based building, and testing. Whenever you push updates to GitHub, Jenkins automatically triggers the pipeline, ensuring that your changes are seamlessly integrated and deployed.

Monitoring Python Web Application Using Prometheus and Grafana

Duration: 25 Min.

Perform the following:

1. Create a Jenkinsfile pipeline script file in a GitHub repository
2. Create the Jenkins pipeline job
3. Execute the Jenkins job

Expected deliverables: A Jenkins pipeline job set up to perform tasks such as code checkout, Maven-based building, and testing whenever updates are pushed in GitHub

Thank You