aws

User Guide

Amazon Managed Grafana



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Managed Grafana: User Guide

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is Amazon Managed Grafana? 1
Supported Regions 1
Getting started 4
User authentication 4
Necessary permissions 5
Create your first workspace 6
Set up AWS 10
Sign up for an AWS account 10
Manage workspaces 11
Grafana version differences 12
Grafana version 12 12
Grafana version 10 21
Grafana version 9 23
Create a workspace 27
Creating a workspace 27
User authentication 35
SAML 36
IAM Identity Center 49
Grafana version 51
Troubleshooting issues with updated workspaces 53
Enterprise plugins 54
Managing access to Enterprise plugins 55
Link with Grafana Labs 56
FAQ for AWS Marketplace Enterprise users 57
Migrate content between workspaces 60
Workspace user access 61
Grant permissions to a user or group 61
Permission mismatch errors 64
Permission mismatch frequently asked questions 64
Permissions for data sources and notifications 66
Creating resources with AWS CloudFormation 67
Amazon Managed Grafana and CloudFormation templates 67
Learn more about CloudFormation 67
Network access control 68

Configuring network access control ... 69
Encryption at rest ... 72
How Amazon Managed Grafana uses grants in AWS KMS ... 73
Step 1: Create a customer managed key ... 75
Step 2: Specifying a customer managed key for Amazon Managed Grafana ... 77
Monitoring your encryption keys for Amazon Managed Grafana ... 78
Learn more ... 84
Connect to data in Amazon VPC ... 84
How VPC connectivity works ... 85
Create a connection to a VPC ... 86
Troubleshoot VPC ... 89
Configure a workspace ... 93
Setting configuration with API or AWS CLI ... 96
Degraded workspaces ... 97
KMS key disabled (not recoverable) ... 98
KMS key disabled (recoverable) ... 98
Deleted security group ... 99
Deleted subnet ... 99
IP address exhaustion ... 100
Missing DHCP option set ... 101
Delete a workspace ... 101
Use your Grafana workspace ... 102
What is Grafana? ... 103
Explore metrics and logs ... 103
Alerts ... 103
Annotations ... 103
Dashboard variables ... 104
Connect to a workspace ... 104
Users, teams, and permissions ... 105
Users ... 106
User roles ... 106
Managing teams ... 107
Using permissions ... 108
Your first dashboard ... 114
Creating a dashboard ... 114
Grafana plugins ... 122

Plugin catalog 123
Manage plugins 124
Install or remove a plugin 125
Update a plugin 126
AWS Data Sources plugin 126
Data sources 129
Special data sources 129
Working with AWS Organizations 130
Built-in data sources 132
Enterprise data sources 323
Using Grafana version 12 403
Dashboards 404
Panels and visualizations 489
Explore 686
Drilldown apps 712
Correlations 714
Alerts 720
Using Grafana version 10 839
Dashboards 840
Panels and visualizations 923
Explore 1121
Correlations 1147
Alerts 1153
Using Grafana version 9 1275
Dashboards 1275
Panels and visualizations 1341
Explore 1484
Alerts 1500
Using Grafana version 8 1601
Panels 1601
Dashboards 1695
Explore 1723
Linking 1732
Templates and variables 1741
Grafana alerting 1766
Change your preferences 1815

Edit your Amazon Managed Grafana profile ... 1815
Edit your preferences ... 1815
View your Amazon Managed Grafana sessions ... 1816
Support bundles ... 1816
Support bundle components ... 1817
Creating a support bundle ... 1817
Classic alerts ... 1818
Alert configuration ... 1819
Clustering ... 1819
Notifications ... 1819
Alert execution ... 1819
Alert notifications ... 1820
Creating alerts ... 1825
Pausing an alert rule ... 1829
Viewing existing alert rules ... 1830
Notification templating ... 1831
Troubleshooting alerts ... 1831
Grafana API Reference ... 1833
Grafana version 12 ... 1833
Authenticate with service accounts ... 1834
New API structure ... 1839
Annotations API ... 1840
Alerting Notification Channels API ... 1846
Dashboard API ... 1855
Dashboard Permissions API ... 1865
Dashboard Versions API ... 1868
Data Source API ... 1876
Data Source Permissions API ... 1891
External Group Synchronization API ... 1896
Folder API ... 1899
Folder/Dashboard Search API ... 1906
Folder Permissions API ... 1908
Organization API ... 1912
Playlist API ... 1916
Plugin API ... 1927
Preferences API ... 1935

Snapshot API ... 1938
Short URL API ... 1943
Team API ... 1945
User API ... 1954
Grafana version 10 ... 1957
Authenticate with tokens ... 1958
Alerting API ... 1967
Alerting Notification Channels API ... 1970
Annotations API ... 1980
Authentication API ... 1986
Dashboard API ... 1988
Dashboard Permissions API ... 1998
Dashboard Versions API ... 2002
Data Source API ... 2009
Data Source Permissions API ... 2026
External Group Synchronization API ... 2031
Folder API ... 2034
Folder/Dashboard Search API ... 2041
Folder Permissions API ... 2043
Organization API ... 2047
Playlist API ... 2051
Plugin API ... 2057
Preferences API ... 2066
Snapshot API ... 2068
Team API ... 2074
User API ... 2082
Observability solutions ... 2087
Monitoring Amazon EKS ... 2087
About this solution ... 2088
Costs ... 2091
Prerequisites ... 2093
Using this solution ... 2095
List of metrics tracked ... 2100
List of alerts created ... 2107
Troubleshooting ... 2116
Monitoring JVM application ... 2119

About this solution ... 2120
Costs ... 2121
Prerequisites ... 2122
Using this solution ... 2124
List of metrics tracked ... 2126
Troubleshooting ... 2127
Monitoring Kafka application ... 2129
About this solution ... 2130
Costs ... 2131
Prerequisites ... 2132
Using this solution ... 2134
List of metrics tracked ... 2137
Troubleshooting ... 2139
Tagging ... 2142
Tagging workspaces ... 2143
Security ... 2149
Data protection ... 2150
Data protection in Amazon Managed Grafana ... 2151
Identity and Access Management ... 2151
Audience ... 2151
Authenticating with identities ... 2152
Managing access using policies ... 2153
How Amazon Managed Grafana works with IAM ... 2155
Identity-based policy examples ... 2160
AWS managed policies ... 2165
Troubleshooting ... 2180
Cross-service confused deputy prevention ... 2181
Using service-linked roles ... 2183
Permissions and policies for other AWS services ... 2187
Service-managed permissions for a single account ... 2188
Service-managed permissions for an organization ... 2190
Customer-managed permissions ... 2196
IAM permissions ... 2197
Amazon Managed Grafana permissions ... 2198
Compliance Validation ... 2199
Resilience ... 2199

Infrastructure Security 2199
CloudTrail logs 2200
Amazon Managed Grafana management events in CloudTrail 2201
Amazon Managed Grafana event examples 2202
Grafana API event examples 2206
Security best practices 2224
Use short-lived API keys 2224
Migrating from self-managed Grafana 2224
Interface VPC endpoints 2225
Using Amazon Managed Grafana with interface VPC endpoints 2225
Creating a VPC endpoint to make an AWS PrivateLink connection to Amazon Managed Grafana 2226
Using network access control to limit access to your Grafana workspace 2226
Controlling access to your Amazon Managed Grafana API VPC endpoint with an endpoint policy 2227
Service quotas 2229
Document history 2234

What is Amazon Managed Grafana?

Amazon Managed Grafana is a fully managed and secure data visualization service that you can use to instantly query, correlate, and visualize operational metrics, logs, and traces from multiple sources. Amazon Managed Grafana makes it easy to deploy, operate, and scale Grafana, a widely deployed data visualization tool that is popular for its extensible data support.

With Amazon Managed Grafana, you create logically isolated Grafana servers called *workspaces*. Then, you can create Grafana dashboards and visualizations to analyze your metrics, logs, and traces without having to build, package, or deploy any hardware to run your Grafana servers.

Amazon Managed Grafana manages the provisioning, setup, scaling, and maintenance of your logical Grafana servers so that you don't have to do these tasks yourself. Amazon Managed Grafana also provides built-in security features for compliance with corporate governance requirements, including single sign-on, data access control, and audit reporting.

Amazon Managed Grafana is integrated with AWS data sources that collect operational data, such as Amazon CloudWatch, Amazon OpenSearch Service, AWS X-Ray, AWS IoT SiteWise, Amazon Timestream, and Amazon Managed Service for Prometheus. Amazon Managed Grafana includes a permission provisioning feature for adding supported AWS services as data sources. Amazon Managed Grafana also supports many popular open-source, third-party, and other cloud data sources.

For user authentication and authorization, Amazon Managed Grafana can integrate with identity providers (IdPs) that support SAML 2.0 and also can integrate with AWS IAM Identity Center.

Amazon Managed Grafana is priced per active user in a workspace. For information about pricing, see Amazon Managed Grafana Pricing.

Supported Regions

Amazon Managed Grafana currently supports the following Regions:

Region Name	Region	Endpoint	Protocol	
US East (Ohio)	us-east-2	grafana.us-east-2.amazonaws.com	HTTPS	

Region Name	Region	Endpoint	Protocol
		grafana.us-east-2.api.aws	HTTPS
US East (N. Virginia)	us-east-1	grafana.us-east-1.amazonaws.com	HTTPS
		grafana.us-east-1.api.aws	HTTPS
US West (Oregon)	us-west-2	grafana.us-west-2.amazonaws.com	HTTPS
		grafana.us-west-2.api.aws	HTTPS
Asia Pacific (Seoul)	ap-northeast-2	grafana.ap-northeast-2.amazonaws.com	HTTPS
		grafana.ap-northeast-2.api.aws	HTTPS
Asia Pacific (Singapore)	ap-southeast-1	grafana.ap-southeast-1.amazonaws.com	HTTPS
		grafana.ap-southeast-1.api.aws	HTTPS
Asia Pacific (Sydney)	ap-southeast-2	grafana.ap-southeast-2.amazonaws.com	HTTPS
		grafana.ap-southeast-2.api.aws	HTTPS
Asia Pacific (Tokyo)	ap-northeast-1	grafana.ap-northeast-1.amazonaws.com	HTTPS
		grafana.ap-northeast-1.api.aws	HTTPS
Europe (Frankfurt)	eu-central-1	grafana.eu-central-1.amazonaws.com	HTTPS
		grafana.eu-central-1.api.aws	HTTPS
Europe (Ireland)	eu-west-1	grafana.eu-west-1.amazonaws.com	HTTPS
		grafana.eu-west-1.api.aws	HTTPS

Region Name	Region	Endpoint	Protocol
Europe (London)	eu-west-2	grafana.eu-west-2.amazonaws.com	HTTPS
		grafana.eu-west-2.api.aws	HTTPS
AWS GovCloud (US-East)	us-gov-east-1	grafana.us-gov-east-1.amazonaws.com	HTTPS
		grafana-fips.us-gov-east-1.api.aws	HTTPS
		grafana-fips.us-gov-east-1.amazonaws.com	HTTPS
		grafana.us-gov-east-1.api.aws	HTTPS
AWS GovCloud (US-West)	us-gov-west-1	grafana.us-gov-west-1.amazonaws.com	HTTPS
		grafana-fips.us-gov-west-1.api.aws	HTTPS
		grafana-fips.us-gov-west-1.amazonaws.com	HTTPS
		grafana.us-gov-west-1.api.aws	HTTPS

Learn how to create and use Amazon Managed Grafana resources

This tutorial helps you get started with Amazon Managed Grafana. Create your first workspace, and then connect to the Grafana console in that workspace.

A *workspace* is a logical Grafana server. You can have as many as five workspaces in each Region in your account.

> **Note**
>
> If you do not have an AWS account, start by learning how to Set up AWS to use Amazon Managed Grafana.

Topics

- User authentication
- Necessary permissions
- Create your first workspace
- Set up AWS to use Amazon Managed Grafana

User authentication

For user authentication within your workspaces, Amazon Managed Grafana supports the following options:

- User credentials stored in identity providers (IdPs), with authentication by Security Assertion Markup Language 2.0 (SAML 2.0)
- AWS IAM Identity Center

SAML

If you use SAML, your users must already be created in an identity provider. Amazon Managed Grafana supports identity providers that support SAML 2.0. For more information, see Use SAML with your Amazon Managed Grafana workspace.

AWS IAM Identity Center

When you create a workspace and choose to use AWS IAM Identity Center for authentication, Amazon Managed Grafana activates IAM Identity Center in your account if you are not already using it. For more information about IAM Identity Center, see What is AWS IAM Identity Center.

To use IAM Identity Center with Amazon Managed Grafana, you must also have AWS Organizations activated in your account. If you don't have it activated already, Amazon Managed Grafana activates it when it activates IAM Identity Center. If Amazon Managed Grafana enables Organizations, it also creates an organization for you. For more information about Organizations, see What is AWS Organizations.

> **Note**
>
> To create a workspace in an account that is already a member of an AWS organization, IAM Identity Center must be enabled in the management account of the organization. If you enabled IAM Identity Center in the management account before November 25, 2019, you must also enable IAM Identity Center-integrated applications in the management account. For more information, see IAM Identity Center-integrated applications.

Necessary permissions

To create a workspace that uses an IdP and SAML for authorization, you must be signed on to an IAM principal that has the **AWSGrafanaAccountAdministrator** policy attached.

To create your first workspace that uses AWS IAM Identity Center for authorization, you must be signed on to an IAM principal that has at least the following policies attached:

- **AWSGrafanaAccountAdministrator**
- **AWSSSOMemberAccountAdministrator**
- **AWSSSODirectoryAdministrator**

For more information, see Create and manage Amazon Managed Grafana workspaces and users in a single standalone account using IAM Identity Center.

Create your first workspace

Use the following steps to create your first workspace.

To create a workspace in Amazon Managed Grafana

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. Choose **Create workspace**.
3. For **Workspace name**, enter a name for the workspace.

   Optionally, enter a description for the workspace.
4. Choose **Next**.
5. For **Authentication access**, select **AWS IAM Identity Center** , **Security Assertion Markup Language (SAML)**, or both.

   - **AWS IAM Identity Center**— If you select IAM Identity Center and you have not already enabled IAM Identity Center in your account, you are prompted to enable it by creating your first IAM Identity Center user. IAM Identity Center handles user management for access to Amazon Managed Grafana workspaces.

     To enable IAM Identity Center, follow these steps:

     a. Choose **Create user**.

     b. Enter an email address, first name, and last name for the user, and choose **Create user**. For this tutorial, use the name and email address of the account that you want to use to try out Amazon Managed Grafana. An email message is sent, prompting you to create a password for this account for IAM Identity Center.

     > **Important**
     >
     > The user that you create does not automatically have access to your Amazon Managed Grafana workspace. You provide the user with access to the workspace in the workspace details page in a later step.

   - **SAML**— If you select **SAML**, you complete the SAML setup after the workspace is created.

6. Choose **Next**.
7. For this first workspace, confirm that **Service managed** is selected for **Permission type**. This selection enables Amazon Managed Grafana to automatically provision the permissions you need for the AWS data sources that you choose to use for this workspace.
8. For this tutorial, choose **Current account**.
9. (Optional) Select the data sources that you want to query in this workspace. For this getting started tutorial, you do not need to select any data sources. However, if you plan to use this workspace with any of the listed data sources, select them here.

   Selecting data sources enables Amazon Managed Grafana to create AWS Identity and Access Management (IAM) policies for each of the data sources so that Amazon Managed Grafana has permission to read their data. This does not completely set up these services as data sources for the Grafana workspace. You can do that within the Grafana workspace console.
10. (Optional) If you want Grafana alerts from this workspace to be sent to an Amazon Simple Notification Service (Amazon SNS) notification channel, select **Amazon SNS**. This enables Amazon Managed Grafana to create an IAM policy to publish to the Amazon SNS topics in your account with `TopicName` values that start with `grafana`. This does not completely set up Amazon SNS as a notification channel for the workspace. You can do that within the Grafana console in the workspace.
11. (Optional) By default, Amazon Managed Grafana automatically provides you with encryption at rest and does this using AWS-owned encryption keys. But you have the option to use a customer managed key that you create, own, and manage as an alternative. For more information, see Encryption at rest.
12. Choose **Next**.
13. Confirm the workspace details, and choose **Create workspace**.

    The workspace details page appears.

    Initially, the **Status** is **CREATING**.

    > **Important**
    >
    > Wait until the status is **ACTIVE** before doing either of the following:
    >
    > - Completing the SAML setup, if you are using SAML.
    > - Assigning your IAM Identity Center users access to the workspace, if you are using IAM Identity Center.

You might need to refresh your browser to see the current status.

14. If you are using IAM Identity Center, do the following:

    a. In the **Authentication** tab, choose **Assign new user or group**.

    b. Select the check box next to the user that you want to grant workspace access to, and choose **Assign user**.

    c. Select the check box next to the user, and choose **Make admin** action from the Actions dropdown list.

    > **Important**
    >
    > Assign at least one user as `Admin` for each workspace, in order to sign in to the Grafana workspace console to manage the workspace.

15. If you are using SAML, do the following:

    a. In the **Authentication** tab, under **Security Assertion Markup Language (SAML)**, choose **Complete setup**.

    b. For **Import method**, do one of the following:

    - Choose **URL** and enter the URL of the IdP metadata.
    - Choose **Upload or copy/paste**. If you are uploading the metadata, choose **Choose file** and select the metadata file. Or, if you are using copy and paste, copy the metadata into **Import the metadata**.

    c. For **Assertion attribute role**, enter the name of the SAML assertion attribute from which to extract role information.

    d. For **Admin role values**, either enter the user roles from your IdP who should all be granted the `Admin` role in the Amazon Managed Grafana workspace, or select **I want to opt-out of assigning admins to my workspace.**

    > **Note**
    >
    > If you choose **I want to opt-out of assigning admins to my workspace.**, you won't be able to use the Grafana workspace console to administer the workspace, including tasks such as managing data sources, users, and dashboard permissions.

You can make administrative changes to the workspace only by using Grafana APIs.

e. (Optional) To enter additional SAML settings, choose **Additional settings** and do one or more the following. All of these fields are optional.

- For **Assertion attribute name**, specify the name of the attribute within the SAML assertion to use for the user full "friendly" names for SAML users.
- For **Assertion attribute login**, specify the name of the attribute within the SAML assertion to use for the user sign-in names for SAML users.
- For **Assertion attribute email**, specify the name of the attribute within the SAML assertion to use for the user email names for SAML users.
- For **Login validity duration (in minutes)**, specify how long a SAML user's sign-in is valid before the user must sign in again.
- For **Assertion attribute organization**, specify the name of the attribute within the SAML assertion to use for the "friendly" name for user organizations.
- For **Assertion attribute groups**, specify the name of the attribute within the SAML assertion to use for the "friendly" name for user groups.
- For **Allowed organizations**, you can limit user access to only the users who are members of certain organizations in the IdP. Enter one or more organizations to allow, separating them with commas.
- For **Editor role values**, enter the user roles from your IdP who should all be granted the `Editor` role in the Amazon Managed Grafana workspace. Enter one or more roles, separated by commas.

Note

Any users that are not specifically assigned an Admin or Editor role are assigned as Viewers.

f. Choose **Save SAML configuration**.

16. In the workspace details page, choose the URL displayed under **Grafana workspace URL**.
17. Choosing the workspace URL takes you to the landing page for the Grafana workspace console. Do one of the following:

- Choose **Sign in with SAML**, and enter the name and password.
- Choose **Sign in with AWS IAM Identity Center**, and enter the email address and password of the user that you created earlier in this procedure. These credentials only work if you have responded to the email from Amazon Managed Grafana that prompted you to create a password for IAM Identity Center.

  You are now in your Grafana workspace, or logical Grafana server. You can start adding data sources to query, visualize, and analyze data. For more information, see Use your Grafana workspace.

Set up AWS to use Amazon Managed Grafana

Sign up for an AWS account

To get started with AWS, you need an AWS account. For information about creating an AWS account, see Getting started with an AWS account in the *AWS Account Management Reference Guide*.

Manage workspaces, users, and policies in Amazon Managed Grafana

To use Amazon Managed Grafana you create Grafana workspaces. A Grafana workspace is a logical Grafana server, where you can create Grafana dashboards and visualizations to analyze your metrics, logs, and traces. You add users and manage their permissions to administer, edit, or view the workspaces.

You can upgrade your workspace to newer versions of Grafana, or update to add support for Enterprise plugins, giving your workspaces access more types of data sources. You can also manage the network access to your workspace. You can create and manage your Amazon Managed Grafana workspaces using CloudFormation.

The topics in this section explain how to manage your workspaces, users, and policies in Amazon Managed Grafana.

Topics

- Differences between Grafana versions
- Create an Amazon Managed Grafana workspace
- Authenticate users in Amazon Managed Grafana workspaces
- Update your workspace version
- Manage access to Enterprise plugins
- Migrate content between Amazon Managed Grafana workspaces
- Manage user and group access to Amazon Managed Grafana workspaces
- Manage permissions for data sources and notification channels
- Creating Amazon Managed Grafana resources with AWS CloudFormation
- Configure network access to your Amazon Managed Grafana workspace
- Encryption at rest
- Connect to data sources or notification channels in Amazon VPC from Amazon Managed Grafana
- Configure a Amazon Managed Grafana workspace
- Troubleshoot degraded workspaces in Amazon Managed Grafana
- Delete a Amazon Managed Grafana workspace

Differences between Grafana versions

When creating a Grafana workspace, you must choose a Grafana version to create. You can choose between versions compatible with Grafana versions 8, 9, 10, and 12. Each of these has added functionality from the previous version. The following topics describe the changes in versions 9, 10, and 12, including changes that might break functionality that you use in previous versions.

> **Note**
>
> You can read version-specific documentation for using your Grafana workspace in the Working in Grafana version 12, Working in Grafana version 10, Working in Grafana version 9, and Working in Grafana version 8 topics.

For detailed notes by version, and more information from Grafana Labs, see What's new in Grafana in the *Grafana Labs documentation*.

Grafana version 12

The following features were added in Grafana version 12.

Drilldown apps

- **Metrics Drilldown** – A queryless, point-and-click experience for exploring Prometheus metric data. See Metrics Drilldown.
- **Logs Drilldown** – A queryless experience for browsing Loki logs with volume and text patterns. See Logs Drilldown.
- **Traces Drilldown** – A queryless experience for exploring distributed Tempo traces. See Traces Drilldown.
- **Profiles Drilldown** – A queryless experience for browsing Pyroscope profiling data. See Profiles Drilldown.

Dashboards and visualizations

- **Scenes-powered dashboards** – The dashboard rendering engine has been rebuilt using the Scenes framework, providing improved performance, better template variable support, and more flexible layouts.

- **New table visualization** – The table panel has been completely rebuilt for improved performance and new capabilities, including CSS cell styling, tooltip generation from table fields, an improved footer, a new Actions cell type for interactive buttons, and auto-formatted cell values in Cell Inspect.
- **Canvas visualization improvements** – Canvas panels now support one-click data links and actions, dynamic connection directions, the ability to disable tooltips, and pan and zoom improvements, giving you more control over interactive data-driven layouts.
- **Visualization actions** – You can now add interactive actions to visualizations that trigger API calls or navigate to URLs, with support for custom variables, one-click data links, and a new Actions cell type for table visualizations.
- **Colored table rows with conditional formatting** – Table visualizations now support row-level conditional formatting, allowing you to color entire rows based on data values.
- **Stat visualization percent change** – Stat panels now display percent change with configurable color mode options, making it easier to see trends at a glance.
- **Legend support in bar gauge** – Bar gauge visualizations now support legends, improving readability when displaying multiple series.
- **State timeline pagination** – State timeline visualizations now support pagination, making it easier to navigate large datasets.
- **Redesigned dashboard filters** – Dashboard list filters have been redesigned for a more intuitive browsing experience.
- **Switch variable type** – A new Switch variable type lets you quickly toggle between values in queries, simplifying dashboard interactions.
- **Static options for query variables** – Query variables now support static options, giving you more control over variable values without requiring a data source query.
- **Enhanced ad hoc filters** – Ad hoc filters now provide improved support across data sources, making it easier to dynamically filter dashboard data.
- **Better time region control with Cron syntax** – Annotations now support Cron syntax for time regions, providing more precise control over time-based annotations.
- **Improved geomap performance** – Geomap visualizations now render significantly faster with improved performance optimizations.
- **Navigation bookmarks** – You can now bookmark frequently used pages in the navigation menu for quick access.
- **Set threshold colors from query** – The Config from query transformation now supports setting threshold colors dynamically based on query results.

- **Enhanced custom currency format** – A new custom currency format option lets you display exact financial values with precise decimal control.
- **Server-configurable quick time ranges** – Administrators can now configure the quick time range options available in dashboard time pickers.
- **Announcement banner** – Administrators can now display announcement banners to communicate important information to workspace users.

Transformations

- **Substring matcher in Filter by value** – The Filter by value transformation now includes a substring matcher, making it easier to filter data based on partial text matches.
- **Trendline transformation** – A new Trendline transformation lets you spot patterns in your data by adding trend lines and moving averages to visualizations.
- **Binary transformation for all number fields** – You can now apply the same binary transformation to all number fields in a table at once, reducing repetitive configuration.
- **New regex option for Extract fields** – The Extract fields transformation now supports regular expressions, providing more powerful field extraction from text data.
- **Transformation updates** – Multiple transformation improvements have been added, including new modes and usability enhancements across the transformation pipeline.
- **Dashboard variables in all transformations** – Dashboard variables are now supported in all transformation types, expanding the flexibility of data processing pipelines.

Alerting

- **Keep Last State for Grafana Managed Alerting** – Alert rules can now be configured to keep their last state when a query returns no data or an error, preventing unnecessary alert state changes.
- **Alert detail view redesign** – The alert rule detail view has been redesigned with a clearer layout and more actionable information.
- **Alerting template selector** – A new template selector in the alerting UI makes it easier to choose and apply notification templates.
- **Improved paused alert visibility** – Paused alert rules are now more clearly indicated in the UI, making it easier to identify which rules are not actively evaluating.

- **Only available for Grafana-managed alerts: rule-specific silences** – You can now create silences that apply to specific alert rules with granular permissions, providing more targeted alert suppression.
- **Recording rules for Grafana-managed alerts** – Recording rules are now available for Grafana-managed alerts, allowing you to precompute frequently used queries for better performance. This replaces the deprecated recorded queries feature. See the section called “Recording rules”.
- **Alert rule version history** – Alert rules now maintain a version history, allowing you to track changes and revert to previous configurations.
- **Grafana-managed alert rule "Recovering" state** – Alert rules now support a "Recovering" state that indicates when an alert condition is no longer firing but has not yet returned to normal.
- **Alert rule improvements** – Grafana-managed alert rules include multiple usability improvements, including updated list pages, active time intervals, the ability to import rules from Prometheus YAML, and a redesigned settings page.

Explore and logs

- **New logs visualization** – A new logs visualization panel provides improved log rendering with a new field selector component for customizing which fields are displayed.
- **JSON Log Line Viewer** – The Logs Drilldown app now includes a JSON viewer for structured log lines, making it easier to inspect JSON-formatted log data.
- **Forward direction search for Loki** – Loki queries in Explore now support forward direction search, allowing you to search logs from oldest to newest.
- **Correlations in Explore** – You can now add correlations to external URLs directly from Explore, enabling seamless navigation between data sources and external systems. See Correlations in Grafana version 12.

Data sources

- **Amazon Managed Service for Prometheus data source migration** – Starting with this release, Sigv4 authentication support in the Core Prometheus plugin has been deprecated. When you upgrade to Grafana version 12, all Amazon Managed Service for Prometheus data sources that were previously using the Core Prometheus plugin are automatically migrated to the Amazon Managed Service for Prometheus plugin. Any dashboards using these data sources are automatically updated to reflect this change.

- **Amazon CloudWatch Logs Insights with PPL and SQL** – You can now query Amazon CloudWatch Logs Insights using PPL (Piped Processing Language) and SQL syntax in addition to the standard query language.
- **Amazon CloudWatch Metric Insights cross-account support** – Amazon CloudWatch Metric Insights now supports cross-account observability, allowing you to query metrics across multiple AWS accounts.
- **Amazon CloudWatch Logs Anomaly Detection and Pattern Analysis** – The Amazon CloudWatch data source now supports log anomaly detection and pattern analysis for identifying unusual patterns in your log data.
- **OpenSearch PPL and Sample Queries** – The OpenSearch data source now supports PPL language and sample queries for easier query building.
- **MSSQL Windows Active Directory authentication** – The Microsoft SQL Server data source now supports Windows Active Directory (Kerberos) authentication.
- **Removal of old Tempo and Loki Search** – The legacy Tempo Search and Loki Search interfaces have been removed in favor of the TraceQL-powered search experience.
- **BigQuery Service Account Impersonation** – The BigQuery data source now supports service account impersonation for more flexible authentication.
- **Google Sheets template variables** – The Google Sheets data source now supports template variables for dynamic data queries.
- **Time series macro support for SQL data sources** – The visual query builder for SQL data sources now supports time series macros, simplifying time-based queries.
- **Unity Catalog support in Databricks** – The Grafana Databricks plugin now supports Unity Catalog for unified data governance.
- **Honeycomb raw query support** – The Honeycomb data source now supports raw queries for advanced query capabilities.
- **GitHub App authentication** – The GitHub data source now supports GitHub App authentication as an alternative to personal access tokens.

Authentication and security

- **API keys migrated to service accounts** – API keys have been fully deprecated and automatically migrated to service accounts, providing improved security and management capabilities. See the section called “Using service accounts”.

- **OAuth and SAML session handling improvements** – Session handling for OAuth and SAML authentication providers has been improved for better reliability and security.
- **Entra Workload Identity support** – Grafana now supports Microsoft Entra Workload Identity for authentication in Azure environments.
- **OAuth2 for Alertmanager and Mimir** – You can now configure OAuth2 authentication in HTTP settings for vanilla Alertmanager and Mimir data sources.

Accessibility

- **GeoMap keyboard support** – GeoMap visualizations now support keyboard navigation for improved accessibility.
- **Panel shortcut keyboard support** – Dashboard panels now support keyboard shortcuts for common actions.
- **Heading and reduced motion improvements** – Heading structure has been improved for screen readers, and reduced motion support has been added for users who prefer minimal animation.

Breaking changes

Important

Legacy alerting is entirely removed – Legacy alerting is completely removed in Amazon Managed Grafana v12, and workspaces with legacy alerting settings will fail to start. You must migrate from legacy alerting to Grafana Alerting before upgrading to v12. See Migrating classic dashboard alerts to Grafana alerting and Classic dashboard alerts.

Important

AngularJS support is removed – AngularJS support is completely removed in Amazon Managed Grafana v12, so Angular-based plugins will not load and dashboards using them will display errors. You must migrate all Angular-based plugins to React alternatives before upgrading. See Find plugins with the plugin catalog for managing plugins in your workspace.

Important

API keys are removed – API keys are fully removed in Amazon Managed Grafana v12 in favor of service accounts, and existing API keys will no longer function. You must create service accounts to replace all API keys and update automation to use service account tokens before upgrading. See the section called “Using service accounts”.

Important

Alert rule evaluation result limit – The number of query evaluation results per alert rule is now limited to 500. If the condition query of an alert rule produces more results than this limit, the evaluation results in an error. To receive alerts when this error occurs, configure `Alert state if execution error or timeout` to `Error` under the alert rule's settings. See the section called “State and health”.

Important

Annotations limited to 3 million – The number of annotations is now limited to 3 million per workspace. If more than 3 million annotations are created, the oldest annotations will be deleted first.

- **Input data source is removed** – The Input data source plugin has been completely removed, so you must migrate affected dashboards to the TestData data source before upgrading. See TestData Documentation in the *Grafana Labs documentation*.
- **Query filtering behavior changes** – The "Disable query" button is now "Hide response/Show response" and hidden queries' responses are no longer returned to panels, so review your query configurations and update panels if data is missing after upgrading. See Query Filtering Changes in the *Grafana Labs documentation*.
- **Data source UIDs must follow stricter format** – Data source UIDs must now use only alphanumeric characters, hyphens, and underscores, so you should update any non-compliant UIDs and their dashboard references before upgrading. See Stricter Data Source UID Format in the *Grafana Labs documentation*.

- **Panel view URLs changed for repeated panels** – The URL format for viewing individual repeated panels has changed and old URLs will return "Panel not found" errors, so update any bookmarks or automation by opening panels in view mode to get the new URLs. See Panel View URL Changes in the *Grafana Labs documentation*.
- **Folders with forward slashes require escaped matchers** – The introduction of subfolders can break alert rules tied to folders containing forward slashes, so update notification policies with escaped matchers (for example, `grafana_folder=MyFolder\/sub-folder`) or rename folders to remove forward slashes before upgrading. See Creating dashboard folders and the section called “Configure notification policies”.
- **Alert template $value variable behavior changes** – The `$value` variable in alert notification templates now returns the query value when querying a single data source, so review alert templates that use `$value` and update formatting if needed. See the section called “Label templates”.
- **Alert rule access permissions relaxed** – Permission requirements for accessing alert rules have been relaxed. See the section called “Using permissions” and What's new in Grafana v11.1 in the *Grafana Labs documentation*.
- **Recording rules enabled by default** – Recording rules are now enabled by default in Grafana Alerting. See the section called “Recording rules”.
- **Internal Alertmanager config POST endpoint removed** – The POST endpoint for the internal Grafana Alertmanager config has been removed, so you must update API integrations to use the new provisioning endpoints (for example, `POST /api/v1/provisioning/alert-rules`) before upgrading.
- **Viewer and Editor permissions expanded** – Viewers can now read and write annotations and Editors can create and delete annotations. See the section called “Using permissions”.
- **Creator permissions changed** – The permissions granted to the creator of a resource (such as a dashboard or folder) have changed, so review your access control settings after upgrading. See Change to creator permissions in the *Grafana Labs documentation*.
- **API key associated permissions removed** – Permissions previously associated with API keys have been removed as part of the full migration to service accounts. See Removal of API key associated permissions in the *Grafana Labs documentation*.
- **SNS contact point migration** – SNS contact points are automatically migrated to a newer SNS notifier. This migration does not require any manual action. However, the contact point settings schema for SNS has changed. See Create notification channel for an example of the updated request format.

- **Alert rule evaluation result limit** – The number of query evaluation results per alert rule is now limited to 500. If the condition query of an alert rule produces more results than this limit, the evaluation results in an error. To receive alerts when this error occurs, Configure `Alert state if execution error or timeout` to `Error` under the alert rule's settings.
- **SigV4 authentication removed from Core Prometheus plugin** – SigV4 authentication support in the Core Prometheus plugin has been removed. When you upgrade to Grafana version 12, all Amazon Managed Service for Prometheus data sources that were previously using the Core Prometheus plugin are automatically migrated to the Amazon Managed Service for Prometheus plugin. Any dashboards using these data sources are automatically updated to reflect this change. See Connect to an Amazon Managed Service for Prometheus data source.
- **OpsGenie contact point removed** – The OpsGenie contact point is removed in Grafana version 12.

Amazon Managed Grafana v12 includes features from open source Grafana v11.0 through v12.4. For AWS-specific features, see the Amazon Managed Grafana User Guide. For detailed information about what's new, see What's new in Grafana for v11.x and v12.x in the *Grafana Labs documentation*. Versions 11.x and 12.x include changes that might break functionality, so test in a non-production environment before updating production workspaces.

For more details on breaking changes, see the following topics in the *Grafana Labs documentation*:

Grafana 12

- What's new in Grafana v12.4
- What's new in Grafana v12.3
- What's new in Grafana v12.2
- What's new in Grafana v12.1
- What's new in Grafana v12.0

Grafana 11

- What's new in Grafana v11.6
- What's new in Grafana v11.5
- What's new in Grafana v11.4
- What's new in Grafana v11.3

- What's new in Grafana v11.2
- What's new in Grafana v11.1
- What's new in Grafana v11.0

Grafana version 10

The following features were added in Grafana version 10.

- **Correlations** – Correlations define how data in one data source is used to query data in another data source, and allow the Explore visualization to easily run queries related to the shown data. For more details, see Correlations in Grafana version 10.
- **Subfolders** – When organizing your dashboards, you can now use subfolders to create a nested hierarchy. For more details, see Creating dashboard folders.
- **Alerts** – Grafana alerting now supports silencing alerts. Additionally, Grafana alerting no longer sends notifications 3 times.
- **Alerting upgrade preview** – Before upgrading from classic dashboard alerts to Grafana alerts, you can see what your alerts will look like, and even make changes that are applied when migrating. For more details, see Migrating classic dashboard alerts to Grafana alerting. Grafana Labs has announced that Grafana version 11 and beyond will no longer support classic dashboard alerts.
- **Support bundles** – Support bundles provide a simple way to collect information about your Grafana workspace to share with product support. You can quickly create a support bundle containing data about migrations, plugins, settings, and more. For more details, see Gather information for support.
- **New visualizations** – Three new visualizations are available. XY Chart, Datagrid, and Trend panel are all available for workspaces compatible with version 10. Version 9 workspaces can also use XY Charts.
- **PagerDuty** – The Enterprise plugins now include a plugin for PagerDuty.
- **Transformations redesign** – The transformations tab has an improved user experience and visual design. Transformations are categorized, and each transformation type has an illustration to help you choose the right one.
- **Prometheus metric encyclopedia** – The metrics dropdown for Prometheus metrics in the Prometheus query builder has been replaced with a paginated and searchable metric *encyclopedia*.

- **API key UI discontinued** – Service accounts are the recommended way to authenticate calls to the Grafana HTTP APIs. As part of Grafana Labs work toward discontinuing API keys, you can no longer create API keys through the workspace user interface. You can only create API keys through the AWS APIs.

  For more information about the discontinuation of API keys by Grafana Labs, see APIKeys: Sunsetting of API keys in the Grafana GitHub issues list.

Breaking changes

The Grafana version 10.4 release includes changes from Grafana versions 9.5 through 10.4. Grafana versions 10.0 and 10.3 had some changes that might break functionality in some cases. When updating to a new version, it is recommended to test in a non-production environment before updating your production workspaces.

The following changes might affect some users updating to Grafana version 10.

- **Angular discontinued** – Plugins that use Angular will no longer be supported in future releases of Grafana. In version 10, panels that use angular will show a banner stating that they use a discontinued feature, to give a notice that they won't work in future versions.
- **Alias in CloudWatch removed** – Alias patterns in the CloudWatch query editor were replaced by Label (dynamic labels).

  Open any dashboard that uses the Alias field, and save it. Alias is migrated to Label automatically.
- **Older plugins need to be upgraded** – The plugins for Athena and Amazon Redshift data source must be updated in Grafana v10 workspaces. The Athena data source plugin must be version 2.9.3 or newer; the Amazon Redshift data source plugin must be version 1.8.3 or newer.

  For information on installing or upgrading plugins, see Find plugins with the plugin catalog.
- **DoiT BigQuery plugin no longer supported** – The DoiT BigQuery data source plugin is no longer supported. Use the official Grafana Labs BigQuery data source plugin instead.
- **Transformation changes** – Grafana version 10 has made a few bug fix changes to field names and keys. For full details, see Transformation breaking changes in the Grafana Labs documentation.

- **Data source permissions APIs** – The endpoints for accessing data source permissions have changed. For full details, see Data source permissions changes in the Grafana Labs documentation.

For more details on breaking changes, see the following topics in the *Grafana Labs documentation*:

- Breaking changes in Grafana v10.0
- Breaking changes in Grafana v10.3

Grafana version 9

The following features were added in Grafana v9.

- **Alerting**: Grafana-managed alert rules now supports group names.
- **Explore**: Create a dashboard from within Explore view.
- **Prometheus queries**: A new query builder for Prometheus queries (using PromQL) makes writing queries easier.
- **Loki queries**: A new query builder for Loki queries (using LogQL) makes writing queries easier.
- **API tokens / Service accounts**: Service accounts simplify machine access in Grafana, helping you to manage API tokens.
- **Plugin management**: You can enable plugin management to install, remove, or update community plugins in your workspace. This gives you access to more data sources and visualizations, and gives you control over the version of each plugin that you use.
- **Trace to metrics**: Configure a tracing data source to add links to metrics with queries and tags.
- **Canvas panel**: A new panel visualization with static and dynamic elements to create data-driven, custom panels with images and overlain text.
- **Reorganized interface**: Updated UI with easier navigation in the Grafana console.
- **CloudWatch**: The Amazon CloudWatch data source can now monitor metrics across AWS accounts and across AWS Regions.
- **Logs**: The interface for log details has been improved.
- **General**: Bug fixes and minor improvements throughout.

Breaking changes

The Grafana version 9.4 release includes a range of new features and improvements, building upon previous versions. This version had some changes that might break functionality in some cases. When updating to a new version, we recommend you test in a non-production environment before updating your production workspaces.

The following changes might affect some users updating to Grafana version 9.4. For a detailed list of these changes, see the Grafana 9.4 changelog on *GitHub*.

- **API discontinued** – The `/api/tsdb/query` API has been removed.

  Action required: Use `/api/ds/query` instead. See Query a data source in the *Grafana public documentation* and Issue #49916 on *GitHub*.
- **API endpoint changes** – Several alerting API endpoints now require data source UID instead of numeric ID.

  Affected endpoints: `api/v1/rule/test`, `api/prometheus/`, `api/ruler/`, `api/alertmanager/`

  Action required: Update API calls to use data source UID as path parameter. See Issues #48070, #48052, #48046, and #47978 on *GitHub*.
- **Azure Monitor queries removed** – Application Insights and Insight Analytics queries are no longer supported.

  Deprecated in Grafana 8.0, removed in 9.0. Deprecated queries will not execute.

  Action required: See Azure Monitor data source in the *Grafana public documentation* for migration guidance.
- **Browser access mode removed** – Browser access mode is no longer available for InfluxDB and Prometheus data sources.

  Action required: Switch to server access mode in your data source configuration. InfluxDB: Deprecated in 8.0.0, removed in 9.2.0. See Issue #53529 on *GitHub*. Prometheus: Deprecated in 7.4.0, removed in 9.2.0. See Issue #50162 on *GitHub*.
- **Dashboard settings access restricted** – You can no longer open dashboard settings while editing panels.

  Dashboard settings are locked when panel edit mode is active. Close panel edit mode before accessing dashboard settings. See Issue #54746 on *GitHub*.
- **Data source password encryption** – Unencrypted passwords are no longer supported.

Action required: Use `secureJsonData.password` and `secureJsonData.basicAuthPassword`. Previously discontinued in v8.1.0. See Issue #49987 on *GitHub*.

- **Default data source behavior** – Default data source selection no longer affects existing panels.

  Default data source only applies to new panels. Changing the default won't update existing dashboards. Previously saved panels retain their data source configuration. See Issue #45132 on *GitHub*.

- **Elasticsearch interval property changed** – Query interval specification updated for Elasticsearch 7.x.

  Changed from `interval` to `fixed_interval` property. Provides consistency with Elasticsearch 8.x. Most queries won't show visible changes. See Issue #50297 on *GitHub*.

- **Elasticsearch Raw document mode discontinued** – Display mode changes in Elasticsearch data source.

  Action required: Use **Raw Data** mode instead. See Issue #62236 on *GitHub*.

- **Elasticsearch version support** – Older Elasticsearch versions are no longer supported.

  Action required: Upgrade Elasticsearch to version 7.10.0 or later. Versions below 7.10.0 are past end-of-life. See Issue #48715 on *GitHub*.

- **Explore URL format discontinued** – Compact Explore URLs will be removed in a future release.

  Action required: Update hard coded links to use standard URL format. Compact URLs: `&left=["now-1h","now"...]`. Standard URLs: `&left={"datasource":"test"...}`. See Issue #50873 on *GitHub*.

- **GitHub OAuth display changes** – GitHub name and login display updated.

  GitHub name appears as Grafana name. GitHub login appears as Grafana login. Improves user identification clarity. See Issue #45438 on *GitHub*.

- **Heatmap panel implementation updated** – Heatmap panels use a new implementation starting in 9.1.0.

  Significantly improved rendering performance. Buckets are placed on reasonable borders (1m, 5m, 30s). Round cells are no longer supported.

Action required: Test your heatmap panels after upgrade. Disable new implementation by setting `useLegacyHeatmapPanel` feature flag to true if needed. Add `?__feature.useLegacyHeatmapPanel=true` to dashboard URLs for testing. See Issue #50229 on *GitHub*.

- **InfluxDB backend migration** – InfluxDB data parsing behavior has changed.

  The InfluxDB backend migration feature toggle (`influxdbBackendMigration`) is reintroduced due to backend processing issues. By default, InfluxDB data is parsed in the frontend. If you upgraded to 9.4.4 and added transformations on InfluxDB data, those panels will fail to render.

  Action required: Remove affected panels and recreate them, or edit the `time` field as `Time` in `panel.json` or `dashboard.json`. See Issue #64842 on *GitHub*.

- **Log message format updated** – Log message structure has changed.

  `lvl` is now `level`. `eror` and dbug are now `error` and `debug`. Increased timestamp precision. Opt-out available with `oldlog` feature toggle (temporary). See Issue #47584 on *GitHub*.

- **Loki data format optimization** – Loki logs data uses a more efficient dataframe format.

  Single dataframe with **labels** column instead of separate dataframes. Explore and logs panels work without changes. Other panels or transforms may need adjustment.

  Action required: Replace **labels to fields** transformation with **extract fields** transformation. See Issue #47153 on *GitHub*.

- **NaN value handling** – Consistent `NaN` representation across Prometheus and Loki data sources.

  `NaN` values remain as `NaN` instead of converting to `null`. Change should be mostly invisible to users. Affects both dashboard and alerting paths. See Issues #49475 and #45389 on *GitHub*.

- **Password reset links invalidated** – Existing password reset links won't work after upgrade.

  Password reset links sent before upgrade are invalid. Users must request new password reset links. Links expire after 2 hours. See Issue #42334 on *GitHub*.

- **Reserved label prefix** – Labels starting with `grafana_` are reserved.

  Manually configured labels beginning with `grafana_` may be overwritten. Current reserved labels: `grafana_folder` (Title of the folder containing the alert). See Issue #50262 on *GitHub*.

- **Transformation improvements** – **Rename by regex** transformation now supports global patterns.

Global patterns use the format `/<stringToReplace>/g`. Some transformations may behave differently. Wrap match strings in forward slashes for previous behavior: `(.*)` becomes `/(.*)/`. See Issue #48179 on *GitHub*.

Create an Amazon Managed Grafana workspace

A *workspace* is a logical Grafana server. You can have as many as five workspaces in each AWS Region in your account.

Necessary permissions

To create a workspace, you must be signed on to an AWS Identity and Access Management (IAM) principal that has the **AWSGrafanaAccountAdministrator** policy attached.

To create your first workspace that uses IAM Identity Center for authorization, your IAM principal must also have these additional policies (or equivalent permissions) attached:

- **AWSSSOMemberAccountAdministrator**
- **AWSSSODirectoryAdministrator**

For more information, see Create and manage Amazon Managed Grafana workspaces and users in a single standalone account using IAM Identity Center.

Creating a workspace

The following steps take you through the process of creating a new Amazon Managed Grafana workspace.

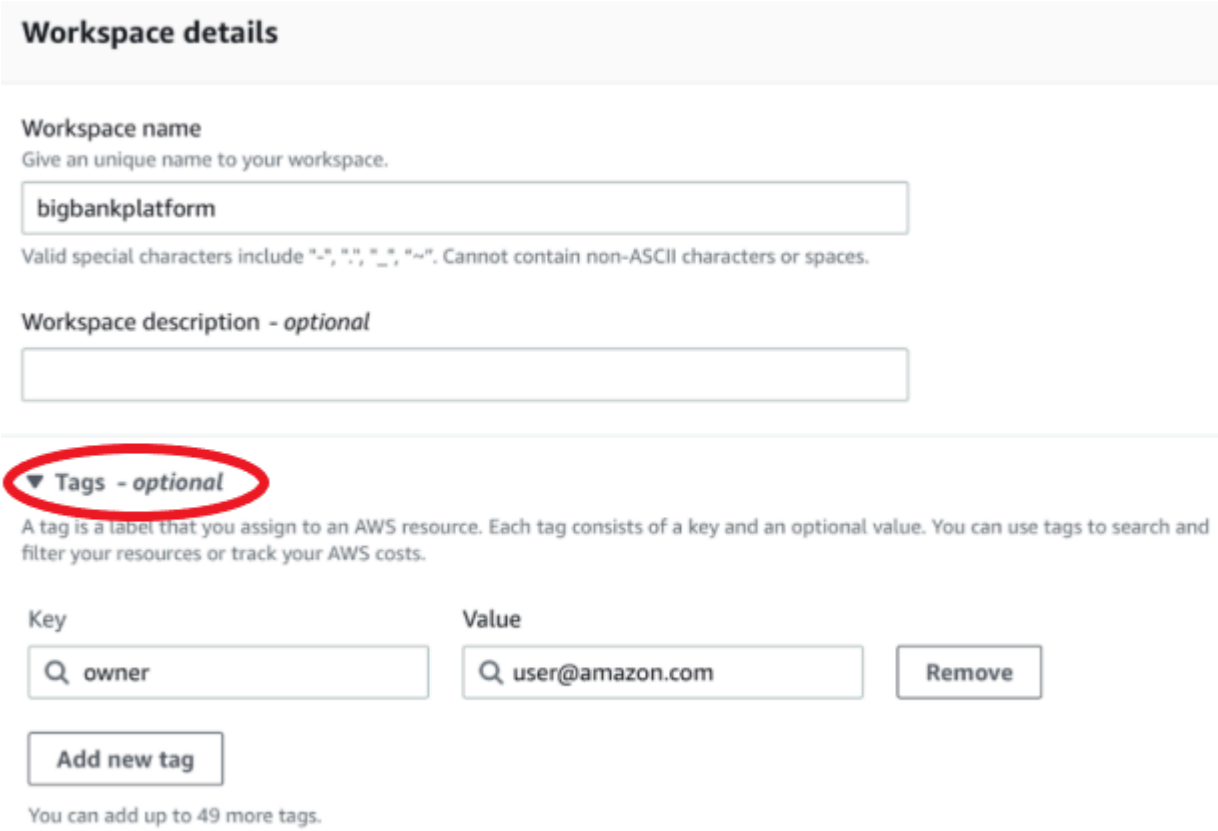
To create a workspace in Amazon Managed Grafana

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. Choose **Create workspace**.
3. In the **Workspace details** window, for **Workspace name**, enter a name for the workspace.

   Optionally, enter a description for the workspace.

   Optionally, add the tags you want to associate with this workspace. Tags help identify and organize workspaces and also can be used for controlling access to AWS resources. For

example, you can assign a tag to the workspace and only a limited groups or roles can have the permission to access the workspace using the tag. For more information on tag-based access control, see Controlling access to AWS resources using tags in IAM User Guide.



4. Choose a **Grafana version** for the workspace. You can choose version 9, 10, or 12. To understand the differences between the versions, see Differences between Grafana versions.
5. Choose **Next**.
6. For **Authentication access**, select **AWS IAM Identity Center** , **Security Assertion Markup Language (SAML)**, or both. For more information, see Authenticate users in Amazon Managed Grafana workspaces.

   - **IAM Identity Center** — If you select IAM Identity Center and you have not already enabled AWS IAM Identity Center in your account, you are prompted to enable it by creating your first IAM Identity Center user. IAM Identity Center handles user management for access to Amazon Managed Grafana workspaces.

     To enable IAM Identity Center, follow these steps:

a. Choose **Create user**.

b. Enter an email address, first name, and last name for the user, and choose **Create user**. For this tutorial, use the name and email address of the account that you want to use to try out Amazon Managed Grafana. You will receive an email message prompting you to create a password for this account for IAM Identity Center.

> **Important**
>
> The user that you create does not automatically have access to your Amazon Managed Grafana workspace. You provide the user with access to the workspace in the workspace details page in a later step.

- **SAML** — If you select **SAML**, you complete the SAML setup after the workspace is created.

7. Choose **Service managed** or **Customer managed**.

   If you choose **Service managed**, Amazon Managed Grafana automatically creates the IAM roles and provisions the permissions that you need for the AWS data sources in this account that you choose to use for this workspace.

   If you want to manage these roles and permissions yourself, choose **Customer managed**.

   If you are creating a workspace in a member account of an organization, to be able to choose **Service managed** the member account must be a delegated administrator account in an organization. For more information about delegated administrator accounts, see Register a delegated administrator.

8. (Optional) You can choose to connect to an Amazon Virtual Private Cloud (VPC) on this page, or you can connect to a VPC later. To learn more, see Connect to data sources or notification channels in Amazon VPC from Amazon Managed Grafana.

9. (Optional) You can choose other workspace configuration options on this page, including the following:

   - Enable Grafana alerting. Grafana alerting allows you to view Grafana alerts and alerts defined in Prometheus within a single alerts interface within your Grafana workspace.

In workspaces running version 8 or 9, this will send multiple notifications for your Grafana alerts. If you use alerts defined in Grafana, we recommend creating your workspace as version 10.4 or later.

- Allow Grafana admins to manage plugins for this workspace. If you don't enable plugin management, your admins will not be able to install, uninstall, or remove plugins for your workspace. You might be limited to the types of data sources and visualization panels you can use with Amazon Managed Grafana.

You can also make these configuration changes after creating your workspace. To learn more about configuring your workspace, see Configure a Amazon Managed Grafana workspace.

10. (Optional) You can choose to add **Network access control** for your workspace. To add network access control, choose **Restricted access**. You can also enable network access control after you have created your workspace.

    For more information about network access control, see Configure network access to your Amazon Managed Grafana workspace.

11. Choose the IP addressing mode for your workspace:

    - **IPv4 only** – Resources in your workspace use IPv4 addressing exclusively.
    - **Dual-stack (IPv4 + IPv6)** – Resources use both IPv4 and IPv6 addressing. Choose this if you have workloads that require IPv6 connectivity.

    > **Note**
    >
    > Dual-stack mode requires Grafana version 10 or later.

    > **Important**
    >
    > Before enabling dual-stack mode, verify the following:
    >
    > - The `sso.signin.aws` domain is allowed in your network firewall and identity provider configurations.
    > - Your network allows IPv6 traffic.
    > - Any prefix lists used for network access control include the appropriate IP ranges.

- If your identity provider uses CORS validation, the new domain is added to the CORS allowlist.
- If your workspace uses a VPC configuration, your subnets must have IPv6 CIDR blocks assigned.

Workspaces using direct SAML authentication or AWS Identity and Access Management Identity Center's built-in directory are not affected by the sign-in URL change. For more information, see Assertion Consumer Service (ACS) endpoints in the *AWS Identity and Access Management Identity Center User Guide*.
If you later switch from dual-stack back to IPv4 only, any clients or resources currently connecting over IPv6 will lose access to your workspace.

12. (Optional) By default, Amazon Managed Grafana automatically provides you with encryption at rest and does this using AWS-owned encryption keys. But you have the option to use a customer managed key that you create, own, and manage as an alternative. For more information, see Encryption at rest.
13. Choose **Next**.
14. If you chose **Service managed**, choose **Current account** to have Amazon Managed Grafana automatically create policies and permissions that allow it to read AWS data only in the current account.

    If you are creating a workspace in the management account or a delegated administrator account in an organization, you can choose **Organization** to have Amazon Managed Grafana automatically create policies and permissions that allow it to read AWS data in other accounts in the organizational units that you specify. For more information about delegated administrator accounts, see Register a delegated administrator.

    > **Note**
    >
    > Creating resources such as Amazon Managed Grafana workspaces in the management account of an organization is against AWS security best practices.

    a. If you chose **Organization**, and you are prompted to enable AWS CloudFormation StackSets, choose **Enable trusted access**. Then, add the AWS Organizations organizational units (OUs) that you want Amazon Managed Grafana to read data from. Amazon Managed Grafana can then read data from all accounts in each OU that you choose.

b. If you chose **Organization**, choose **Data sources and notification channels - optional**.

15. Select the AWS data sources that you want to query in this workspace. Selecting data sources enables Amazon Managed Grafana to create IAM roles and permissions that allow Amazon Managed Grafana to read data from these sources. You must still add the data sources in the Grafana workspace console.
16. (Optional) If you want Grafana alerts from this workspace to be sent to an Amazon Simple Notification Service (Amazon SNS) notification channel, select **Amazon SNS**. This enables Amazon Managed Grafana to create an IAM policy to publish to the Amazon SNS topics in your account with `TopicName` values that start with `grafana`. This does not completely set up Amazon SNS as a notification channel for the workspace. You can do that within the Grafana console in the workspace.
17. Choose **Next**.
18. Confirm the workspace details, and choose **Create workspace**.

    The workspace details page appears.

    Initially, the **Status** is **CREATING**.

    > **Important**
    >
    > Wait until the status is **ACTIVE** before doing either of the following:
    >
    > - Completing the SAML setup, if you are using SAML.
    > - Assigning your IAM Identity Center users access to the workspace, if you are using IAM Identity Center.
    >
    > You might need to refresh your browser to see the current status.

19. If you are using IAM Identity Center, do the following:

    a. In the **Authentication** tab, choose **Assign new user or group**.

    b. Select the check box next to the user that you want to grant workspace access to, and choose **Assign user**.

    c. Select the check box next to the user, and choose **Make admin**.

> **Important**
>
> Assign at least one user as `Admin` for each workspace, in order to sign in to the Grafana workspace console to manage the workspace.

20. If you are using SAML, do the following:

    a. In the **Authentication** tab, under **Security Assertion Markup Language (SAML)**, choose **Complete setup**.

    b. For **Import method**, do one of the following:

    - Choose **URL** and enter the URL of the IdP metadata.
    - Choose **Upload or copy/paste**. If you are uploading the metadata, choose **Choose file** and select the metadata file. Or, if you are using copy and paste, copy the metadata into **Import the metadata**.

    c. For **Assertion attribute role**, enter the name of the SAML assertion attribute from which to extract role information.

    d. For **Admin role values**, either enter the user roles from your IdP who should all be granted the `Admin` role in the Amazon Managed Grafana workspace, or select **I want to opt-out of assigning admins to my workspace.**

> **Note**
>
> If you choose **I want to opt-out of assigning admins to my workspace.**, you won't be able to use the console to administer the workspace, including tasks such as managing data sources, users, and dashboard permissions. You can make administrative changes to the workspace only by using Amazon Managed Grafana APIs.

    e. (Optional) To enter additional SAML settings, choose **Additional settings** and do one or more the following. All of these fields are optional.

    - For **Assertion attribute name**, specify the name of the attribute within the SAML assertion to use for the user full "friendly" names for SAML users.
    - For **Assertion attribute login**, specify the name of the attribute within the SAML assertion to use for the user sign-in names for SAML users.

- For **Assertion attribute email**, specify the name of the attribute within the SAML assertion to use for the user email names for SAML users.
- For **Login validity duration (in minutes)**, specify how long a SAML user's sign-in is valid before the user must sign in again. The default is 1 day, and the maximum is 30 days.
- For **Assertion attribute organization**, specify the name of the attribute within the SAML assertion to use for the "friendly" name for user organizations.
- For **Assertion attribute groups**, specify the name of the attribute within the SAML assertion to use for the "friendly" name for user groups.
- For **Allowed organizations**, you can limit user access to only the users who are members of certain organizations in the IdP. Enter one or more organizations to allow, separating them with commas.
- For **Editor role values**, enter the user roles from your IdP who should all be granted the `Editor` role in the Amazon Managed Grafana workspace. Enter one or more roles, separated by commas.

f. Choose **Save SAML configuration**.

21. In the workspace details page, choose the URL displayed under **Grafana workspace URL**.
22. Choosing the workspace URL takes you to the landing page for the Grafana workspace console. Do one of the following:

    - Choose **Sign in with SAML**, and enter the name and password.
    - Choose **Sign in with AWS IAM Identity Center**, and enter the email address and password of the user that you created earlier in this procedure. These credentials only work if you have responded to the email from Amazon Managed Grafana that prompted you to create a password for IAM Identity Center.

      You are now in your Grafana workspace, or logical Grafana server. You can start adding data sources to query, visualize, and analyze data. For more information, see Use your Grafana workspace.

For more information on

Tip

You can automate the creation of Amazon Managed Grafana workspaces by using CloudFormation. For more detailed information see Creating Amazon Managed Grafana resources with AWS CloudFormation.

Authenticate users in Amazon Managed Grafana workspaces

Individual users sign into your workspaces, to edit and view your dashboards. You can assign users to your workspaces and give them user, editor, or administrator permissions. To get started, you create (or use an existing) identity provider to authenticate users.

Users are authenticated to use the Grafana console in an Amazon Managed Grafana workspace by single sign-on using your organization's identity provider, instead of by using IAM. Each workspace can use one or both of the following authentication methods:

- User credentials stored in identity providers (IdPs) that support Security Assertion Markup Language 2.0 (SAML 2.0)
- AWS IAM Identity Center. AWS Single-sign-on (**AWS SSO**) was rebranded to **IAM Identity Center**.

For each of your workspaces, you can use SAML, IAM Identity Center, or both. If you begin by using one method, you can switch to using the other.

You must give your users (or groups that they belong to) permissions to the workspace before they can access functionality within the workspace. For more information about giving permissions to your users, see Manage user and group access to Amazon Managed Grafana workspaces.

Topics

- Use SAML with your Amazon Managed Grafana workspace
- Use AWS IAM Identity Center with your Amazon Managed Grafana workspace

Use SAML with your Amazon Managed Grafana workspace

> **Note**
>
> Amazon Managed Grafana does not currently support IdP initiated login for workspaces. You should set up your SAML applications with a blank Relay State.

You can use SAML authentication to use your existing identity provider and offer single sign-on for logging into the Grafana console of your Amazon Managed Grafana workspaces. Rather than authenticating through IAM, SAML authentication for Amazon Managed Grafana lets you use third-party identity providers to log in, manage access control, search your data, and build visualizations. Amazon Managed Grafana supports identity providers that use the SAML 2.0 standard and have built and tested integration applications with Azure AD, CyberArk, Okta, OneLogin, and Ping Identity.

For details about how to set up SAML authentication during workspace creation, see Creating a workspace.

In the SAML authentication flow, an Amazon Managed Grafana workspace acts as the service provider (SP), and interacts with the IdP to obtain user information. For more information about SAML, see Security Assertion Markup Language.

You can map groups in your IdP to teams in the Amazon Managed Grafana workspace, and set fine-grained access permissions on those teams. You can also map organization roles that are defined in the IdP to roles in the Amazon Managed Grafana workspace. For example, if you have a **Developer** role defined in the IdP, you can map that role to the **Grafana Admin** role in the Amazon Managed Grafana workspace.

> **Note**
>
> When you create an Amazon Managed Grafana workspace that uses an IdP and SAML for authorization, you must be signed on to an IAM principal that has the **AWSGrafanaAccountAdministrator** policy attached.

To sign in to the Amazon Managed Grafana workspace, a user visits the workspace's Grafana console home page and chooses **Log in using SAML**. The workspace reads the SAML configuration and redirects the user to the IdP for authentication. The user enters their sign-in credentials in the

IdP portal, and if they are a valid user, the IdP issues a SAML assertion and redirects the user back to the Amazon Managed Grafana workspace. Amazon Managed Grafana verifies that the SAML assertion is valid, and the user is signed in and can use the workspace.

Amazon Managed Grafana supports the following SAML 2.0 bindings:

- From the service provider (SP) to the identity provider (IdP):
  - HTTP-POST binding
  - HTTP-Redirect binding
- From the identity provider (IdP) to the service provider (SP):
  - HTTP-POST binding

Amazon Managed Grafana supports signed and encrypted assertions, but does not support signed or encrypted requests.

Amazon Managed Grafana supports SP-initiated requests, and does not support IdP-initiated requests.

Assertion mapping

During the SAML authentication flow, Amazon Managed Grafana receives the assertion consumer service (ACS) callback. The callback contains all relevant information for the user being authenticated, embedded in the SAML response. Amazon Managed Grafana parses the response to create (or update) the user within its internal database.

When Amazon Managed Grafana maps the user information, it looks at the individual attributes within the assertion. You can think of these attributes as key-value pairs, although they contain more information than that.

Amazon Managed Grafana provides configuration options so that you can modify which keys to look at for these values.

You can use the Amazon Managed Grafana console to map the following SAML assertion attributes to values in Amazon Managed Grafana:

- For **Assertion attribute role**, specify the name of the attribute within the SAML assertion to use as the user roles.
- For **Assertion attribute name**, specify the name of the attribute within the SAML assertion to use for the user full "friendly" names for SAML users.

- For **Assertion attribute login**, specify the name of the attribute within the SAML assertion to use for the user sign-in names for SAML users.
- For **Assertion attribute email**, specify the name of the attribute within the SAML assertion to use for the user email names for SAML users.
- For **Assertion attribute organization**, specify the name of the attribute within the SAML assertion to use for the "friendly" name for user organizations.
- For **Assertion attribute groups**, specify the name of the attribute within the SAML assertion to use for the "friendly" name for user groups.
- For **Allowed organizations**, you can limit user access to only the users who are members of certain organizations in the IdP.
- For **Editor role values**, specify the user roles from your IdP who should all be granted the `Editor` role in the Amazon Managed Grafana workspace.

Connecting to your identity provider

The following external identity providers have been tested with Amazon Managed Grafana and provide applications directly in their app directories or galleries to help you configure Amazon Managed Grafana with SAML.

Topics

- Configure Amazon Managed Grafana to use Azure AD
- Configure Amazon Managed Grafana to use CyberArk
- Configure Amazon Managed Grafana to use Okta
- Configure Amazon Managed Grafana to use OneLogin
- Configure Amazon Managed Grafana to use Ping Identity

Configure Amazon Managed Grafana to use Azure AD

Use the following steps to configure Amazon Managed Grafana to use Azure Active Directory as an identity provider. These steps assume that you have already created your Amazon Managed Grafana workspace and you have made a note of the workspace *ID*, *URLs*, and *AWS Region*.

Step 1: Steps to complete in Azure Active Directory

Complete the following steps in Azure Active Directory.

To set up Azure Active Directory as an identity provider for Amazon Managed Grafana

1. Sign in to the Azure console as an admin.
2. Choose **Azure Active Directory**.
3. Choose **Enterprise Applications**.
4. Search for **Amazon Managed Grafana SAML2.0**, and select it.
5. Select the application and choose **Setup**.
6. In the Azure Active Directory application configuration, choose **Users and groups**.
7. Assign the application to the users and groups that you want.
8. Choose **Single sign-on**.
9. Choose **Next** to get to the SAML configuration page.
10. Specify your SAML settings:
    - For **Identifier (Entity ID)**, paste in your **Service provider identifier** URL from the Amazon Managed Grafana workspace.
    - For **Reply URL (Assertion Consumer Service URL)**, paste in your **Service provider reply** from the Amazon Managed Grafana workspace.
    - Make sure that **Sign Assertion** is selected and that **Encrypt Assertion** is not selected.
11. In the **User Attributes & Claims** section, make sure that these attributes are mapped. They are case sensitive.
    - **mail** is set with **user.userprincipalname**.
    - **displayName** is set with **user.displayname**.
    - **Unique User Identifier** is set with **user.userprincipalname**.
    - Add any other attributes that you would to pass. For more information about the attributes that you can pass to Amazon Managed Grafana in the assertion mapping, see Assertion mapping.
12. Copy the **SAML Metadata URL** for use in the Amazon Managed Grafana workspace configuration.

Step 2: Steps to complete in Amazon Managed Grafana

Complete the following steps in the Amazon Managed Grafana console.

To finish setting up Azure Active Directory as an identity provider for Amazon Managed Grafana

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the name of the workspace.
5. In the **Authentication** tab, choose **Setup SAML configuration**.
6. Under **Import the metadata**, choose **Upload or copy/paste** and paste the Azure Active Directory URL that you copied from **SAML Metadata URL** in the previous section.
7. Under **Assertion mapping**, do the following:

   - Make sure that **I want to opt-out of assigning admins to my workspace** is not selected.

     > **Note**
     >
     > If you choose **I want to opt-out of assigning admins to my workspace**, you won't be able to use the Amazon Managed Grafana workspace console to administer the workspace, including tasks such as managing data sources, users, and dashboard permissions. You can make administrative changes to the workspace only by using Grafana APIs.

   - Set **Assertion attribute role** to the attribute name that you chose.
   - Set **Admin role values** to value corresponding to your admin users' roles.
   - (Optional) If you changed the default attributes in your Azure Active Directory application, expand **Additional settings - optional** and then set the new attribute names.

     By default, the Azure **displayName** attribute is passed as the **Name** attribute and the Ping Identity **mail** attribute is passed to both the **email** and **login** attributes.
8. Choose **Save SAML Configuration**.

Configure Amazon Managed Grafana to use CyberArk

Use the following steps to configure Amazon Managed Grafana to use CyberArk as an identity provider. These steps assume that you have already created your Amazon Managed Grafana workspace and you have made a note of the workspace's ID, URLs, and Region.

Step 1: Steps to complete in CyberArk

Complete the following steps in CyberArk.

To set up CyberArk as an identity provider for Amazon Managed Grafana

1. Sign in to the CyberArk Identity Admin Portal.
2. Choose **Apps**, **Web Apps**.
3. Choose **Add Web App**.
4. Search for **Amazon Managed Grafana for SAML2.0**, and choose **Add**.
5. In the CyberArk application configuration, go to the **Trust** section.
6. Under **Identity Provider Configuration**, choose **Metadata**.
7. Choose **Copy URL** and save the URL to use later in these steps.
8. Under **Service Provider Configuration**, choose **Manual Configuration**.
9. Specify your SAML settings:
   - For **SP Entity ID**, paste in your **Service provider identifier** URL from the Amazon Managed Grafana workspace.
   - For **Assertion Consumer Service (ACS) URL**, paste in your **Service provider reply** from the Amazon Managed Grafana workspace.
   - Set **Sign Response Assertion** to **Assertion**.
   - Make sure that **NameID Format** is **emailAddress**.
10. Choose **Save**.
11. In the **SAML Response** section, make sure that the Amazon Managed Grafana attribute is in **Application Name** and that the CyberArk attribute is in **Attribute Value**. Then make sure that the following attributes are mapped. They are case sensitive.
    - **displayName** is set with **LoginUser.DisplayName**.
    - **mail** is set with **LoginUser.Email**.
    - Add any other attributes that you would to pass. For more information about the attributes that you can pass to Amazon Managed Grafana in the assertion mapping, see Assertion mapping.
12. Choose **Save**.
13. In the **Permissions** section, choose which users and groups to assign this application to, and then choose **Save**.

Step 2: Steps to complete in Amazon Managed Grafana

Complete the following steps in the Amazon Managed Grafana console.

To finishg setting up CyberArk as an identity provider for Amazon Managed Grafana

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the name of the workspace.
5. In the **Authentication** tab, choose **Setup SAML configuration**.
6. Under **Import the metadata**, choose **Upload or copy/paste** and paste the CyberArk URL that you copied in the previous procedure.
7. Under **Assertion mapping**, do the following:

   - Make sure that **I want to opt-out of assigning admins to my workspace** is not selected.

     > **Note**
     >
     > If you choose **I want to opt-out of assigning admins to my workspace**, you won't be able to use the Amazon Managed Grafana workspace console to administer the workspace, including tasks such as managing data sources, users, and dashboard permissions. You can make administrative changes to the workspace only by using Grafana APIs.

   - Set **Assertion attribute role** to the attribute name that you chose.
   - Set **Admin role values** to value corresponding to your admin users' roles.
   - (Optional) If you changed the default attributes in your CyberArk application, expand **Additional settings - optional** and then set the new attribute names.

     By default, the CyberA **displayName** attribute is passed to the **name** attribute and the CyberArk **mail** attribute is passed to both the **email** and **login** attributes.
8. Choose **Save SAML Configuration**.

Configure Amazon Managed Grafana to use Okta

Use the following steps to configure Amazon Managed Grafana to use Okta as an identity provider. These steps assume that you have already created your Amazon Managed Grafana workspace and you have made a note of the workspace's ID, URLs, and Region.

Step 1: Steps to complete in Okta

Complete the following steps in Okta.

To set up Okta as an identity provider for Amazon Managed Grafana

1. Sign in to the Okta console as an admin.
2. In the left panel, choose **Applications**, **Applications**.
3. Choose **Browse App Catalog** and search for **Amazon Managed Grafana**.
4. Choose **Amazon Managed Grafana** and choose **Add**, **Done**.
5. Choose the application to start setting it up.
6. In the **Sign On** tab, choose **Edit**.
7. Under **Advanced Sign-on Settings**, enter your Amazon Managed Grafana workspace id and your Region in the **Name Space** and **Region** fields respectively. Your Amazon Managed Grafana workspace id and Region can be found in your Amazon Managed Grafana workspace url which is of the format ***`workspace-id`*.grafana-workspace.*`Region`*.amazonaws.com**.
8. Choose **Save**.
9. Under **SAML 2.0**, copy the URL for **Identity Provider metadata**. You use this later in this procedure in the Amazon Managed Grafana console.
10. In the **Assignments** tab, choose the **People** and **Groups** that you want to be able to use Amazon Managed Grafana.

Step 2: Steps to complete in Amazon Managed Grafana

Complete the following steps in the Amazon Managed Grafana console.

To finish setting up Okta as an identity provider for Amazon Managed Grafana

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the navigation pane, choose the menu icon.
3. Choose **All workspaces**.

4. Choose the name of the workspace.
5. In the **Authentication** tab, choose **Complete Setup**.
6. Under **Import the meta data**, choose **Upload or copy/paste** and paste the Okta URL that you copied in the previous procedure.
7. Under **Assertion mapping**, do the following:

   - Make sure that **I want to opt-out of assigning admins to my workspace** is not selected.

     > **Note**
     >
     > If you choose **I want to opt-out of assigning admins to my workspace**, you won't be able to use the Amazon Managed Grafana workspace console to administer the workspace, including tasks such as managing data sources, users, and dashboard permissions. You can make administrative changes to the workspace only by using Grafana APIs.

   - Set **Assertion attribute role** to the attribute name that you chose.
   - Set **Admin role values** to value corresponding to your admin users' roles.
   - (Optional) If you changed the default attributes in your Okta application, expand **Additional settings - optional** and then set the new attribute names.

     By default, the Okta **displayName** attribute is passed to the **name** attribute and the Okta **mail** attribute is passed to both the **email** and **login** attributes.
8. Choose **Save SAML Configuration**.

Configure Amazon Managed Grafana to use OneLogin

Use the following steps to configure Amazon Managed Grafana to use OneLogin as an identity provider. These steps assume that you have already created your Amazon Managed Grafana workspace and you have made a note of the workspace's ID, URLs, and Region.

Step 1: Steps to complete in OneLogin

Complete the following steps in OneLogin.

To set up OneLogin as an identity provider for Amazon Managed Grafana

1. Sign in to the OneLogin portal as an administrator.

2. Choose **Applications**, **Applications**, **Add app**.
3. Search for **Amazon Managed Service for Grafana**.
4. Assign a **Display name** of your choice and choose **Save**.
5. Navigate to **Configuration** and enter the Amazon Managed Grafana workspace ID in **Namespace**, and enter the Region of your Amazon Managed Grafana workspace.
6. In the **Configuration** tab, enter your Amazon Managed Grafana workspace URL.
7. You can leave the **adminRole** parameter as the default **No Default** and populate it using the **Rules** tab, if an admin requires a corresponding value in Amazon Managed Grafana. In this example, the **Assertion attribute role** would be set to **adminRole** in Amazon Managed Grafana, with a value of true. You can point this value to any attribute in your tenant. Click the **+** to add and configure parameters to meet your organization's requirements.
8. Choose the **Rules** tab, choose **Add Rule**, and give your Rule a name. In the **Conditions** field (the if statement), we add **Email contains [email address]**. In the **Actions** field (the then statement), we select **Set AdminRole in Amazon Managed Service** and we select **Macro** in the **Set adminRole** to dropdown, with a value of **true**. Your organization can choose different rules to resolve different use cases.
9. Choose **Save**. Go to **More Actions** and choose **Reapply entitlement mappings**. You must reappy mappings any time that you create or update rules.
10. Make a note of the **Issuer URL**, which you use later in the configuration in the Amazon Managed Grafana console. Then choose **Save**.
11. Choose the **Access** tab to assign the OneLogin roles that are to access Amazon Managed Grafana and select an app security policy.

Step 2: Steps to complete in Amazon Managed Grafana

Complete the following steps in the Amazon Managed Grafana console.

To finish setting up OneLogin as an identity provider for Amazon Managed Grafana

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the name of the workspace.
5. In the **Authentication** tab, choose **Setup SAML configuration**.

6. Under **Import the metadata**, choose **Upload or copy/paste** and paste the OneLogin Issuer URL that you copied from the OneLogin console in the previous procedure.
7. Under **Assertion mapping**, do the following:

   - Make sure that **I want to opt-out of assigning admins to my workspace** is not selected.

     > **Note**
     >
     > If you choose **I want to opt-out of assigning admins to my workspace**, you won't be able to use the Amazon Managed Grafana workspace console to administer the workspace, including tasks such as managing data sources, users, and dashboard permissions. You can make administrative changes to the workspace only by using Grafana APIs.

   - Set **Assertion attribute role** to the attribute name that you chose. The default value for OneLogin is **adminRole**.
   - Set **Admin role values** to value corresponding to your admin users' roles.
   - (Optional) If you changed the default attributes in your OneLogin application, expand **Additional settings - optional** and then set the new attribute names.

     By default, the OneLogin **displayName** attribute is passed to the **name** attribute and the OneLogin **mail** attribute is passed to both the **email** and **login** attributes.
8. Choose **Save SAML Configuration**.

Configure Amazon Managed Grafana to use Ping Identity

Use the following steps to configure Amazon Managed Grafana to use Ping Identity as an identity provider. These steps assume that you have already created your Amazon Managed Grafana workspace and you have made a note of the workspace's ID, URLs, and Region.

Step 1: Steps to complete in Ping Identity

Complete the following steps in Ping Identity.

To set up Ping Identity as an identity provider for Amazon Managed Grafana

1. Sign in to the Ping Identity console as an admin.
2. Choose **Applications**.

3. Choose **Add Application**, **Search Application Catalog**.
4. Search for the **Amazon Managed Grafana for SAML** application, then choose it and choose **Setup**.
5. In the Ping Identity application, choose **Next** to get to the SAML configuration page. Then make the following SAML settings:
   - For **Assertion Consumer Service**, paste in your **Service provider reply URL** from the Amazon Managed Grafana workspace.
   - For **Entity ID**, paste in your **Service provider identifier** from the Amazon Managed Grafana workspace.
   - Make sure that **Sign Assertion** is selected and that **Encrypt Assertion** is not selected.
6. Choose **Continue to Next Step**.
7. In **SSO Attribute Mapping**, make sure that the Amazon Managed Grafana attribute is in **Application Attribute** and that the Ping Identity attribute is in the **Identity Bridge Attribute**. Then make the following settings:
   - **mail** must be **Email (Work)**.
   - **displayName** must be **Display Name**.
   - **SAML_SUBJECT** must be **Email (Work)**. And then for this attribute, choose **Advanced**, set the **Name ID Format to send to SP** to **urn:oasis:names:tc:SAML:2.0:nameid-format:transient** and choose **Save**.
   - Add in any other attribute that you would like to pass.
   - Add any other attributes that you would like to pass. For more information about the attributes that you can pass to Amazon Managed Grafana in the assertion mapping, see Assertion mapping.
8. Choose **Continue to Next Step**.
9. In **Group Access**, choose which groups to assign this application to.
10. Choose **Continue to Next Step**.
11. Copy the **SAML Metadata URL** which starts with `https://admin- api.pingone.com/latest/metadata/`. You use this later in the configuration.
12. Choose **Finish**.

Step 2: Steps to complete in Amazon Managed Grafana

Complete the following steps in the Amazon Managed Grafana console.

To finish setting up Ping Identity as an identity provider for Amazon Managed Grafana

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the name of the workspace.
5. In the **Authentication** tab, choose **Setup SAML configuration**.
6. Under **Import the metadata**, choose **Upload or copy/paste** and paste the Ping URL that you copied in the previous procedure.
7. Under **Assertion mapping**, do the following:

   - Make sure that **I want to opt-out of assigning admins to my workspace** is not selected.

     > **Note**
     >
     > If you choose **I want to opt-out of assigning admins to my workspace**, you won't be able to use the Amazon Managed Grafana workspace console to administer the workspace, including tasks such as managing data sources, users, and dashboard permissions. You can make administrative changes to the workspace only by using Grafana APIs.

   - Set **Assertion attribute role** to the attribute name that you chose.
   - Set **Admin role values** to value corresponding to your admin users' roles.
   - (Optional) If you changed the default attributes in your Ping Identity application, expand **Additional settings - optional** and then set the new attribute names.

     By default, the Ping Identity **displayName** attribute is passed to the **name** attribute and the Ping Identity **mail** attribute is passed to both the **email** and **login** attributes.
8. Choose **Save SAML Configuration**.

Use AWS IAM Identity Center with your Amazon Managed Grafana workspace

Amazon Managed Grafana integrates with AWS IAM Identity Center to provide identity federation for your workforce. Using Amazon Managed Grafana and IAM Identity Center, users are redirected to their existing company directory to sign in with their existing credentials. Then, they are seamlessly signed in to their Amazon Managed Grafana workspace. This ensures that security settings such as password policies and two-factor authentication are enforced. Using IAM Identity Center does not impact your existing IAM configuration.

If you do not have an existing user directory or prefer not to federate, IAM Identity Center offers an integrated user directory that you can use to create users and groups for Amazon Managed Grafana. Amazon Managed Grafana does not support the use of IAM users and roles to assign permissions within an Amazon Managed Grafana workspace.

For more information about IAM Identity Center, see What is AWS IAM Identity Center. For more information about getting started with IAM Identity Center, see Getting started.

To use IAM Identity Center, you must also have AWS Organizations activated for the account. If needed, Amazon Managed Grafana can activate Organizations for you when you create your first workspace that is configured to use IAM Identity Center.

Required permissions for scenarios using IAM Identity Center

This section explains the policies that are required for using Amazon Managed Grafana with IAM Identity Center. The policies needed to administer Amazon Managed Grafana differ based on whether your AWS account is part of an organization or not.

Create a Grafana administrator in AWS Organizations accounts

To grant permissions to create and manage Amazon Managed Grafana workspaces in an organization, and to allow dependencies such as AWS IAM Identity Center, assign the following policies to a role.

- Assign the **AWSGrafanaAccountAdministrator** IAM policy to allow administering Amazon Managed Grafana workspaces.
- **AWSSSODirectoryAdministrator** allows the role to use IAM Identity Center when setting up Amazon Managed Grafana workspaces.

- To allow creating and managing Amazon Managed Grafana workspaces across the entire organization, give the role the **AWSSSOMasterAccountAdministrator** IAM policy. Alternately, give the role the **AWSSSOMemberAccountAdministrator** IAM policy to allow creating and managing workspaces within a single member account of the organization.
- You can also optionally give the role the **AWSMarketplaceManageSubscriptions** IAM policy (or equivalent permissions) if you want to allow the role to upgrade an Amazon Managed Grafana workspace to Grafana enterprise.

If you want to use service-managed permissions when you create an Amazon Managed Grafana workspace, the role that creates the workspace must also have the `iam:CreateRole`, `iam:CreatePolicy`, and `iam:AttachRolePolicy` permissions. These are required to use CloudFormation StackSets to deploy policies that allow you to read data sources in the organization's accounts.

> **Important**
>
> Granting a user the `iam:CreateRole`, `iam:CreatePolicy`, and `iam:AttachRolePolicy` permissions gives that user full administrative access to your AWS account. For example, a user with these permissions can create a policy that has full permissions for all resources, and attach that policy to any role. Be very careful about who you grant these permissions to.

To see the permissions granted to **AWSGrafanaAccountAdministrator**, see AWS managed policy: AWSGrafanaAccountAdministrator

Create and manage Amazon Managed Grafana workspaces and users in a single standalone account

A standalone AWS account is an account that is not a member of an organization. For more information about AWS Organizations, see What is AWS Organizations?

To grant permission to create and manage Amazon Managed Grafana workspaces and users in a standalone account, assign the following IAM policies to a role:

- **AWSGrafanaAccountAdministrator**
- **AWSSSOMasterAccountAdministrator**

- **AWSOrganizationsFullAccess**
- **AWSSSODirectoryAdministrator**

> **Important**
>
> Granting a role the **AWSOrganizationsFullAccess** policy gives that role full administrative access to your AWS account. Be very careful about who you grant these permissions to.

To see the permissions granted to **AWSGrafanaAccountAdministrator**, see AWS managed policy: AWSGrafanaAccountAdministrator

Update your workspace version

You can update your Amazon Managed Grafana workspace to a newer version of Grafana in the Amazon Managed Grafana console in two ways.

> **Note**
>
> You can only update the version to a newer version of Grafana. You can't downgrade to a previously released version of Grafana.
> Updating your version of Grafana will not update the plugins that are installed in your workspace. You might need to individually update any plugins that are not compatible with the new version of Grafana. For details on viewing and managing plugins, see Find plugins with the plugin catalog. For a list of changes in each version, see Differences between Grafana versions.

Option 1 - Update the version from the list of workspaces

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana.
2. In the left navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. In the row containing the details for the workspace you want to update, choose **Update version**. Only workspaces that are eligible to be updated will include this option.

> **Warning**
>
> The update process is irreversible and can't be paused or canceled. We recommend testing the newer version in a non-production environment before updating a production workspace. During an update, you can't make changes to the workspace.

5. Choose a version number from the dropdown on the **Update version** screen and click **Update** to confirm.
6. Periodically check the status of your update on the **Workspaces** tab. The update process could take up to 10 minutes. During this process, the workspace will be in 'read only' mode. A banner update will display to indicate if your workspace update succeeded or failed. If your update failed, follow the action items outlined in the banner and try again.

Option 2 - Update the version from the workspace summary page

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana.
2. In the left navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the hyperlinked **Workspace name** of the workspace you want to update. Only workspaces that are eligible to be updated will include this option.
5. Choose the **Update version** prompt in the **Summary** block.

> **Warning**
>
> The update process is irreversible and can't be paused or canceled. We recommend testing the newer version in a non-production environment before updating a production workspace. During an update, you can't make changes to the workspace.

6. Choose a version number from the dropdown on the **Update version** screen and click **Update** to confirm.
7. Periodically check the status of your update on the **Workspaces** tab. The update process could take up to 10 minutes. During this process, the workspace will be in 'read only' mode. A banner update will display to indicate if your workspace update succeeded or failed. If your update failed, follow the action items outlined in the banner and try again.

> **Note**
>
> You can also update the version using the UpdateWorkspaceConfiguration operation in the Amazon Managed Grafana API.

If you run into issues with your updated workspace, see Troubleshooting issues with updated workspaces.

Troubleshooting issues with updated workspaces

Your updated workspace should continue to work after updating. This section can help you track down possible issues after you update.

- **Differences between versions.**

  Some functionality has changed between versions.

  - For a list of major changes between versions, including changes that may cause issues in functionality, see Differences between Grafana versions.
  - For documentation of version 9 specific functionality, see Working in Grafana version 9. For version 10, see Working in Grafana version 10. For version 12, see Working in Grafana version 12.

- **PostgreSQL TLS issue**

  If your **TLS/SSL Mode** was set to `require` in version 8, and you were only using a root certificate, you could experience TLS or certificate issues with the PostgreSQL data source after updating. Modify your TLS settings for your PostgreSQL data source (available in your Grafana workspace side menu, by choosing the **Configuration** icon, then **Data Sources**).

  - Change the **TLS/SSL Mode** to `verify-ca`.
  - Set **TLS/SSL Method** to `Certificate content`.
  - Set the **Root Certificate** to the root certificate for your PostgreSQL database server. This is the only field in which you should enter a certificate.

Manage access to Enterprise plugins

You can use the Amazon Managed Grafana console to manage your workspace and gain access to Enterprise plugins. Upgrading gives you access to Enterprise plugins with support for data sources from a variety of third-party independent software vendors (ISVs), including the list below.

An Enterprise license also gives you access to Grafana Labs consulting and support services.

Enterprise data sources available with Amazon Managed Grafana Enterprise plugins include:

- AppDynamics
- Databricks
- Datadog
- Dynatrace
- GitLab
- Honeycomb
- Jira
- MongoDB
- New Relic
- Oracle Database
- Salesforce
- SAP/HANA
- ServiceNow
- Snowflake
- Splunk
- Splunk Infrastructure Monitoring (formerly SignalFx)
- Wavefront

For details about the Enterprise data source plugins available when you upgrade, see Connect to Enterprise data sources. New plugins can be added at any time. For a complete and current list, you can use the plugin catalog within your Amazon Managed Grafana workspace.

When you create a workspace, by default it does not have access to Enterprise plugins, but you can upgrade at any time. If you want to have multiple Amazon Managed Grafana workspaces with Enterprise plugins, you must upgrade each of them.

You can manage your Enterprise plugin license, including adding or removing your access through the **Manage Amazon Managed Grafana Enterprise** page.

The process of managing access to Amazon Managed Grafana Enterprise plugins has changed. If you previously used AWS Marketplace, you may be interested in the FAQ for AWS Marketplace Enterprise users topic.

Topics

- Managing your access to Amazon Managed Grafana Enterprise plugins
- Link your account with Grafana Labs
- FAQ for AWS Marketplace Enterprise users

Managing your access to Amazon Managed Grafana Enterprise plugins

To manage your access to Enterprise plugins

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana.
2. In the left navigation pane, choose the menu icon.
3. Choose **All workspaces**.

   You can see the list of workspaces. For each workspace, the **Enterprise license** columns shows the type of license the workspace has (either no license, or the **Enterprise plugins** license.
4. Select the name of the workspace whose license you want to manage. This opens the workspace details page for that workspace.
5. In the summary, under **Enterprise License**, choose either **Manage** or **Upgrade to Amazon Managed Grafana Enterprise** (only one option is available, based on the current status of the Enterprise license).

   This opens the **Manage Amazon Managed Grafana Enterprise** page. You can choose between two options. The active option is marked with **(current)**.

   - **None** – This is the option to remove, or not have an Amazon Managed Grafana Enterprise license. If you currently have an Enterprise license, selecting this option for your workspace immediately removes access to the Enterprise plugins when you save.
   - **Enterprise plugins** – This allows you to install any Enterprise plugins to your workspace, as well as giving access to Grafana Labs consulting and support services. Installing Enterprise plugins in your workspace gives you access to additional data sources.

The first time that you choose this option, you must link your AWS account with a token from Grafana Labs, and are prompted to do so. For more information, see the next section, Link your account with Grafana Labs.

Amazon Managed Grafana Enterprise plugin access includes user fees that are in addition to the prices for Amazon Managed Grafana. For detailed fee information, see the Amazon Managed Grafana Pricing page.

6. After making your selection, choose **Save** to continue.

Link your account with Grafana Labs

Workspaces upgraded to Amazon Managed Grafana Enterprise plugins get access to support and consulting from Grafana Labs. To access this feature, the AWS account must be linked with a Grafana Labs account token. You register your new or existing Grafana Labs account with AWS when you upgrade to an Enterprise license.

> **Note**
>
> You only need to register your Grafana Labs account token one time per region. If your account was previously linked (for example, when upgrading a different workspace in the region to access Enterprise plugins), you are not prompted to link again.

Linking consists of getting a token from a Grafana Labs account that is used in Amazon Managed Grafana to register the account. You can create a new account at Grafana Labs or use an existing one.

We recommend that you copy and save your Grafana Labs token in a secure, convenient location for future use.

To link your Grafana Labs account

1. Follow the instructions in Managing your access to Amazon Managed Grafana Enterprise plugins to upgrade your account with access Enterprise plugins. You are prompted to link your account by adding a token during the upgrade process.
2. If you already have a token, you can enter it directly. If you do not have a token, select **Get your token**. This opens the Grafana Labs website in a new browser tab.

From the Grafana Labs website, you can sign into your Grafana Labs account (or create a new one), then get a token.

3. After you copy the token, return to the Amazon Managed Grafana browser tab or window. Enter the token in the **Grafana Labs Token** section.
4. You are now able to choose **Save** to complete your upgrade.

Reusing your token with other workspaces

If you have previously registered your Grafana Labs account and are prompted for a Grafana Labs token (for example, when upgrading a workspace in another region), you can use the same token to register each time, so that you do not need to create a new Grafana Labs account. If you have not saved your token, you may be able to retrieve it in one of these ways:

- You can get the token by looking it up in your Grafana Labs account by going to https://grafana.com/partners/amg/support, and choosing **My Account**.
- You can get the token from an existing, already linked workspace, by using the DescribeWorkspace API to retrieve the token.
- If the token is no longer available to you via either of those methods, you must contact Grafana Labs support.

FAQ for AWS Marketplace Enterprise users

Previously, you may have purchased a license for Grafana Enterprise through AWS Marketplace. You can no longer purchase new licenses through AWS Marketplace, and you can not renew any license that was previously purchased through AWS Marketplace. The following FAQ may help you depending on the state of your AWS Marketplace license.

I subscribed to a 30-day free trial from AWS Marketplace, but I haven't associated it with my workspace. Can I apply it now?

No. The free trials are no longer supported in Amazon Managed Grafana.

I purchased a 30-day free trial from AWS Marketplace, and I already associated it with my workspace. What will happen to my trial?

Your free trial will continue until it expires. If you want to upgrade and use the Enterprise plugins, you can upgrade through the Amazon Managed Grafana console, as described in the previous section.

I have a AWS Marketplace paid license that hasn't yet expired, but I want to use Amazon Managed Grafana managed Enterprise plugins. How do I do that?

As long as you have a current AWS Marketplace license, you can only associate that license with your workspaces. You can only upgrade in the Amazon Managed Grafana console after your AWS Marketplace license expires (or you cancel it through AWS Marketplace).

The following questions and answers provide more details.

I purchased a full Grafana Enterprise license from AWS Marketplace and associated it with one or more workspaces. What will happen to those?

When your license expires (after 30 days, unless you have autorenewal turned on), any Enterprise data sources that you are using in your workspace will stop working. If you wish to continue using Enterprise data sources, you can upgrade to use Enterprise plugins directly from the Amazon Managed Grafana console.

It sounds like there will be downtime associated with my license expiring, where my workspace can't access any Enterprise plugins. How do I avoid that?

There will be some downtime associated with your license expiring, as you switch to the new Enterprise plugins license. However, you can minimize this.

> **Note**
>
> The following steps need to be performed precisely to minimize downtime. We recommend that you read them carefully before beginning.
> To get the new pricing, we recommend that you upgrade to Amazon Managed Grafana Enterprise plugins, rather than continue using the AWS Marketplace license.

To switch from AWS Marketplace Enterprise license to Amazon Managed Grafana Enterprise plugins while minimizing downtime.

1. To prepare, first go to the Grafana Labs website, and sign into your account (or create a new one). Get your Grafana Labs token that you will use later in the process.

   For more details on this part of the process, see Link your account with Grafana Labs.
2. Sign into the AWS Marketplace console, and choose **Manage subscriptions** from the left menu.
3. Find the subscription that you want to switch, and choose **Manage**. This will bring up details about your subscription.

   > **Note**
   >
   > This page shows your service end date. You can wait until you are nearing that date to continue these steps, to maximize use of your current subscription before canceling.

4. Choose **Actions**, and select **Cancel subscription**.

   This cancels your subscription in AWS Marketplace. However, you can continue to use the Enterprise data sources until Amazon Managed Grafana automatically removes your license at the end of the day (local time for your workspace).

   For more information about canceling subscriptions in AWS Marketplace, see Cancel your product subscription in the *AWS Marketplace Buyer Guide*.
5. After your subscription is canceled in AWS Marketplace, cancel it in Amazon Managed Grafana:

   1. Sign into the Amazon Managed Grafana console.
   2. From the left menu, choose **All workspaces**.
   3. Choose the name of the workspace you are switching.
   4. Under **Enterprise license**, choose **Manage**.
   5. Choose **None** and then **Save**. This will remove the AWS Marketplace license from Amazon Managed Grafana

   When the Enterprise license is removed, you will no longer be able to access Enterprise plugins in your workspace.

6. You can now upgrade in the Amazon Managed Grafana console. Follow the instructions in the Managing your access to Amazon Managed Grafana Enterprise plugins topic, using the Grafana Labs token you created in the first step.

> **Note**
>
> Your workspace is not able to access Enterprise data sources from the time you cancel the license in Amazon Managed Grafana until when you upgrade to access Enterprise plugins. This is typically around 10-15 minutes, but can take longer, depending on how quickly you can perform these steps. Making sure that you have the Grafana Labs token ready will minimize this time.

I have an AWS Marketplace license with autorenew. Will that continue?

Yes. The AWS Marketplace subscription is retired, and you can't manually renew it, but if you had autorenew set up, it will continue until you turn it off. When you do that, you can upgrade, following the instructions in the previous answers.

To get the new pricing, we recommend that you upgrade to Amazon Managed Grafana Enterprise plugins, rather than continue using the AWS Marketplace license.

I have an AWS Marketplace license that I haven't yet associated with a workspace, can I use it?

Yes, you can associate that AWS Marketplace license and use it until it expires. That will happen within 30 days, unless you turned on autorenew. See the previous questions and answers for more information.

Migrate content between Amazon Managed Grafana workspaces

There are times that you want to migrate your content (including data sources, dashboards, folder, and alert rules) from one workspace to another. For example, you are migrating from an on-premise Grafana instance to an Amazon Managed Grafana workspace, and you want to migrate your existing content to the new workspace.

Amazon Managed Grafana does not directly support migrating content between workspaces, however, AWS does provide an open-source migration utility that can handle this scenario by providing export and import functionality within a workspace or Grafana instance. This utility is called the **Amazon Managed Grafana Migrator**.

For more information, see Amazon Managed Grafana Migrator on GitHub.

Manage user and group access to Amazon Managed Grafana workspaces

You access Amazon Managed Grafana workspaces with users that are set up in your Identity provider (IdP) or AWS IAM Identity Center. You must give those users (or groups that they belong to) permissions to the workspace. You can give them `User`, `Editor`, or `Admin` permissions.

Grant permissions to a user or group

Prerequisites

- To grant a user or a user group access to Amazon Managed Grafana workspaces, the user or group must first be provisioned in an Identity provider (IdP) or in AWS IAM Identity Center. For more information, see Authenticate users in Amazon Managed Grafana workspaces.
- To manage user and group access, you must be signed in as a user that has the AWS Identity and Access Management (IAM) policy **AWSGrafanaWorkspacePermissionManagementV2**, or equivalent permissions. If you are managing users with IAM Identity Center, you must also have the **AWSSSOMemberAccountAdministrator** and **AWSSSODirectoryReadOnly** IAM policies, or equivalent permissions. For more information, see Assign and unassign users access to Amazon Managed Grafana.

To manage user access to a Grafana workspace using the Amazon Managed Grafana console

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the left navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the name of the workspace that you want to manage.
5. Choose the **Authentication** tab.

6. If you are using IAM Identity Center in this workspace, choose **Configure users and user groups** and do one or more of the following:

   - To give a user access to the Amazon Managed Grafana workspace, select the check box next to the user, and choose **Assign user**.
   - To make a user an `Admin` of the workspace, choose **Make admin**.
   - To remove workspace access for a user, choose **Unassign user**.
   - To add groups of users such as an LDAP group, choose the **Assigned user groups** tab. Then, do one of the following:
     - To give all members of a group access to the Amazon Managed Grafana workspace, select the check box next to the group, and choose **Assign group**.
     - To give all members of a group the `Admin` role in the workspace, choose **Make admin**.
     - To remove workspace access for all members of a group, choose **Unassign group**.

   > **Note**
   >
   > If you are using IAM Identity Center to manage users, use the IAM Identity Center console only to provision new users and groups. Use the Amazon Managed Grafana console or APIs to give or remove access to your Grafana workspaces.
   > If IAM Identity Center and Amazon Managed Grafana get out of sync, you are presented with an option to **Resolve** any conflicts. For more information, see Permission mismatch errors when configuring users and groups, below.

7. If you are using SAML in this workspace, choose **SAML configuration** and do one or more of the following:

   - For **Import method**, do one of the following:
     - Choose **URL** and enter the URL of the IdP metadata.
     - Choose **Upload or copy/paste**. If you are uploading the metadata, choose **Choose file** and select the metadata file. Or, if you are using copy and paste, copy the metadata into **Import the metadata**.
   - For **Assertion attribute role**, enter the name of the SAML assertion attribute from which to extract role information.

- For **Admin role values**, either enter the user roles from your IdP who should all be granted the `Admin` role in the Amazon Managed Grafana workspace, or select **I want to opt-out of assigning admins to my workspace.**

  > **Note**
  >
  > If you choose **I want to opt-out of assigning admins to my workspace.**, you won't be able to use the Amazon Managed Grafana console to administer the workspace, including tasks such as managing data sources, users, and dashboard permissions. You can make administrative changes to the workspace only by using Amazon Managed Grafana APIs.

- (Optional) To enter additional SAML settings, choose **Additional settings** and do one or more the following, and then choose **Save SAML configuration**. All of these fields are optional.
  - For **Assertion attribute name**, specify the name of the attribute within the SAML assertion to use for the user full "friendly" names for SAML users.
  - For **Assertion attribute login**, specify the name of the attribute within the SAML assertion to use for the user sign-in names for SAML users.
  - For **Assertion attribute email**, specify the name of the attribute within the SAML assertion to use for the user email names for SAML users.
  - For **Login validity duration (in minutes)**, specify how long a SAML user's sign-in is valid before the user must sign in again.
  - For **Assertion attribute organization**, specify the name of the attribute within the SAML assertion to use for the "friendly" name for user organizations.
  - For **Assertion attribute groups**, specify the name of the attribute within the SAML assertion to use for the "friendly" name for user groups.
  - For **Allowed organizations**, you can limit user access to only the users who are members of certain organizations in the IdP. Enter one or more organizations to allow, separating them with commas.
  - For **Editor role values**, enter the user roles from your IdP who should all be granted the `Editor` role in the Amazon Managed Grafana workspace. Enter one or more roles, separated by commas.

8. Alternatively, to add groups of users such as an LDAP group, choose the **User Group** tab. Then, do one of the following:

- To give all members of a group access to the Amazon Managed Grafana workspace, select the check box next to the group, and choose **Assign group**.
- To give all members of a group the `Admin` role in the workspace, choose **Make admin**.
- To remove workspace access for all members of a group, choose **Unassign group**.

Permission mismatch errors when configuring users and groups

You might run into mismatch errors when configuring users and groups in the Amazon Managed Grafana console. This indicates that Amazon Managed Grafana and IAM Identity Center are out of sync. In this case, Amazon Managed Grafana displays a warning and a choice to **Resolve** the mismatch. If you choose **Resolve**, Amazon Managed Grafana displays a dialog with a list of users that have permissions that are out of sync.

Users that have been removed from IAM Identity Center show up as `Unknown user`, with a numeric ID in the dialog. For these users, the only way to fix the mismatch is to choose **Resolve**, and remove their permissions.

Users that are still in IAM Identity Center, but no longer belong to a group with the access rights that they previously had, show up with their user name in the **Resolve** list. There are two ways to fix this issue. You can use the **Resolve** dialog to remove or reduce their access, or you can give them access by following the instructions in the previous section.

Frequently asked questions about permissions mismatches

Why am I seeing an error stating mismatch in permissions in the Configure Users and Groups section of the Amazon Managed Grafana console?

You are seeing this message because a mismatch has been identified in users and group associations in IAM Identity Center and permissions in Amazon Managed Grafana for your workspace. You can add or remove users to your Grafana workspace from the Amazon Managed Grafana console (in the **Configure Users and Groups** tab), or from the IAM Identity Center console (**Application assignments** page). However, the Grafana user permissions can only be defined from Amazon Managed Grafana (using the Amazon Managed Grafana console or APIs), by assigning **Viewer**, **Editor**, or **Admin** permissions to the user or group. A user can belong to multiple groups with varying permissions, in which case their permission is based on the highest access level across all groups and permissions the user belongs to.

Mismatched records can result from:

- A user or group is deleted from IAM Identity Center, but not in Amazon Managed Grafana. These records show as **Unknown users** in the Amazon Managed Grafana console.
- A user or group's association with Grafana is deleted in IAM Identity Center (under **Application assignments**), but not in Amazon Managed Grafana.
- User permissions were previously updated from the Grafana workspace directly. Updates from the Grafana workspace are not supported in Amazon Managed Grafana.

To avoid these mismatches, use the Amazon Managed Grafana console or Amazon Managed Grafana APIs to manage user and group permissions for your workspace.

I have previously updated the access levels for some of my team members from the Grafana workspace. Now I see that their access levels are reverted back to their older access level. Why am I seeing this and how do I resolve this?

This is most likely due to a mismatch that was identified between the user and group association in IAM Identity Center and the permission records Amazon Managed Grafana for your workspace. If your team members are experiencing different access levels, you or an admin for your Amazon Managed Grafana might have resolved the mismatch from the Amazon Managed Grafana console, removing the mismatched records. You can re-assign the required access levels from the Amazon Managed Grafana console or APIs to restore the desired permissions.

> **Note**
>
> User access management is not supported from the Grafana workspace. Use the Amazon Managed Grafana console or APIs to assign user or group permissions.

Why am I noticing changes in my access levels? For example, I previously had admin access, but now only have editor permissions.

An admin for your workspace might have changed your permissions. This can happen inadvertently in the case of a mismatch between your user and group associations in IAM Identity Center and your permissions in Amazon Managed Grafana. In this case, resolving the mismatch might have removed your higher access permissions. You can request an admin to re-assign the required access level from the Amazon Managed Grafana console.

Manage permissions for data sources and notification channels

Your Amazon Managed Grafana workspace must have permission to access AWS data sources for your metrics and notification channels for your alerts. You can use the Amazon Managed Grafana console to have Amazon Managed Grafana automatically create AWS Identity and Access Management (IAM) policies and permissions for the AWS data sources and notification channels that you want to use in the Amazon Managed Grafana workspace.

To manage permissions and policies for data sources and notification channels

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the left navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the name of the workspace that you want to manage.
5. To switch between using **Service managed** and **Customer managed** permissions, choose the edit icon for **IAM role** and then make your selection. For more information, see Amazon Managed Grafana permissions and policies for AWS data sources.

   If you change from **Service managed** permissions to **Customer managed** permissions, the roles and policies that Amazon Managed Grafana created for you are not deleted in the current account. If you were using **Service managed** permissions for an organization, the roles and policies in other accounts in the organization are deleted.
6. Choose the **Data sources** tab.
7. If you are using **Service managed** permissions, you can choose **Edit** next to **IAM permission access settings** to change whether your **Service managed** permissions apply to only the current account or to an entire organization. For more information, see Amazon Managed Grafana permissions and policies for AWS data sources.

   Under **Data sources**, select the AWS data sources that you want to query in this workspace. Selecting data sources enables Amazon Managed Grafana to create the IAM roles and permissions that allow Amazon Managed Grafana to read data from these sources. You must still add the data sources in the Grafana workspace console.

   To manage AWS services that can be used as notification channels, choose **Notification channels**.

Select the AWS notification channel that you want to use in this workspace. Selecting a notification channel enables Amazon Managed Grafana to create IAM roles and permissions that allow Amazon Managed Grafana to use these services. You must still add the notification channels in the Grafana workspace console.

> **Note**
>
> For more information about using notifications, see Manage your alert notifications.

Creating Amazon Managed Grafana resources with AWS CloudFormation

Amazon Managed Grafana is integrated with AWS CloudFormation, a service that helps you to model and set up your AWS resources so that you can spend less time creating and managing your resources and infrastructure. You create a template that describes all the AWS resources that you want (such as workspaces), and CloudFormation provisions and configures those resources for you.

When you use CloudFormation, you can reuse your template to set up your Amazon Managed Grafana resources consistently and repeatedly. Describe your resources once, and then provision the same resources over and over in multiple AWS accounts and Regions.

Amazon Managed Grafana and CloudFormation templates

To provision and configure resources for Amazon Managed Grafana and related services, you must understand CloudFormation templates. Templates are formatted text files in JSON or YAML. These templates describe the resources that you want to provision in your CloudFormation stacks. If you're unfamiliar with JSON or YAML, you can use CloudFormation Designer to help you get started with CloudFormation templates. For more information, see What is CloudFormation Designer? in the *AWS CloudFormation User Guide*.

Amazon Managed Grafana supports creating workspaces in CloudFormation. For more information, including examples of JSON and YAML templates for workspaces, see the Amazon Managed Grafana resource type reference in the *AWS CloudFormation User Guide*.

Learn more about CloudFormation

To learn more about CloudFormation, see the following resources:

- AWS CloudFormation
- AWS CloudFormation User Guide
- CloudFormation API Reference
- AWS CloudFormation Command Line Interface User Guide

Configure network access to your Amazon Managed Grafana workspace

You can control how users and hosts access your Grafana workspaces.

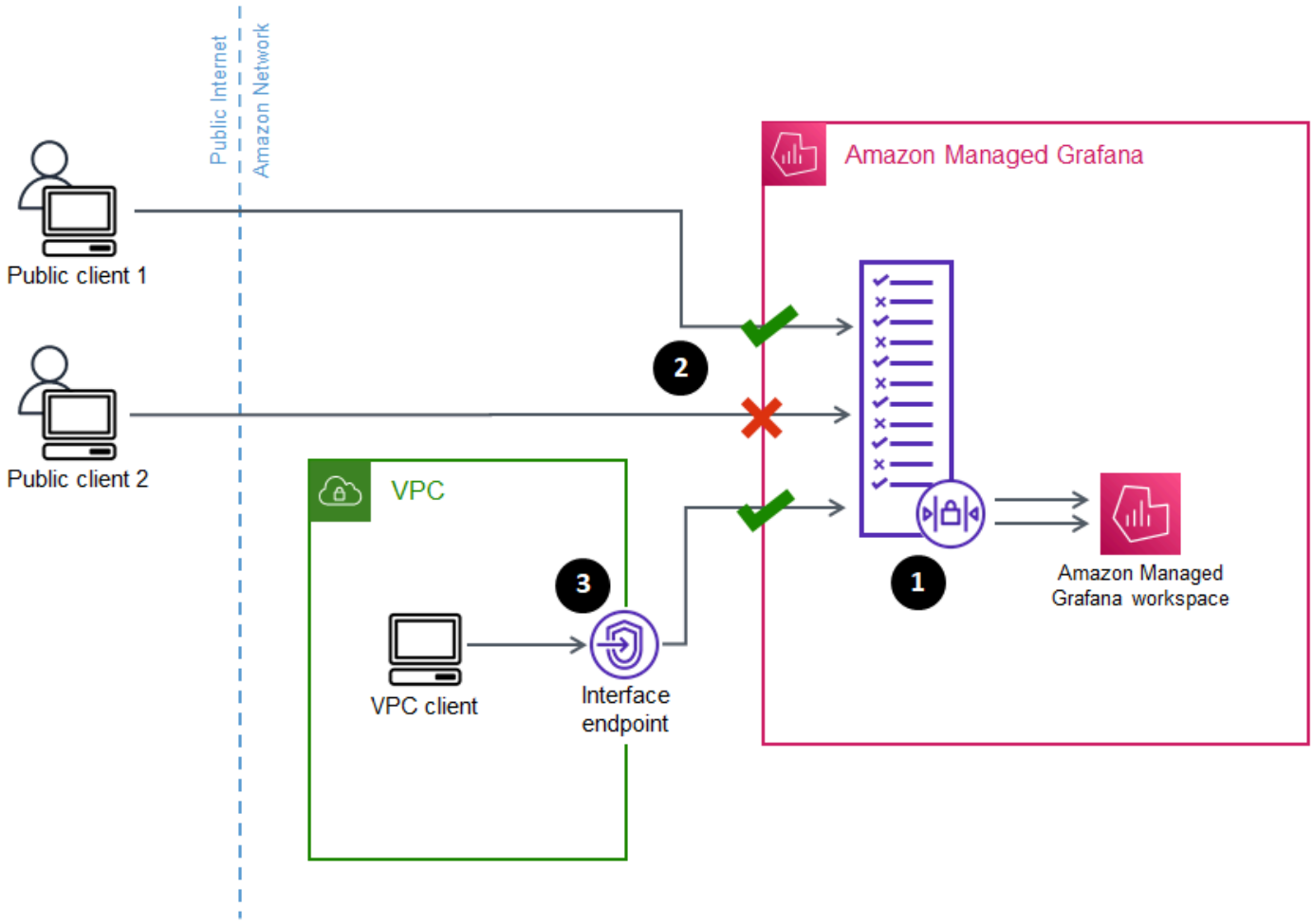
Grafana requires all users to be authenticated and authorized. However, by default, Amazon Managed Grafana workspaces are open to all network traffic. You can configure network access control for a workspace, to control what network traffic is allowed to reach it.

Control traffic to your workspace in two ways:

- **IP Addresses** (prefix lists) – Create a managed prefix list with IP ranges that are allowed to access workspaces. Prefix lists can contain IPv4 or IPv6 address ranges. To filter IPv6 traffic, use an IPv6 prefix list with a dual-stack workspace.
- **VPC endpoints** – Create a list of VPC endpoints to your workspaces that are allowed to access a specific workspace.

When you configure network access control, you must include at least one prefix list or VPC endpoint.

Amazon Managed Grafana uses the prefix lists and VPC endpoints to decide which requests to the Grafana workspace are allowed to connect. The following diagram shows this filtering.



Configuring network access control (**1**) for an Amazon Managed Grafana workspace specifies which requests should be allowed to access the workspace. Network access control can allow or block traffic by IP address (**2**), or by which interface endpoint is being used (**3**).

The following section describes how to set up network access control.

Configuring network access control

You can add network access control to an existing workspace or configure it as part of the initial creation of the workspace.

Prerequisites

To set up network access control you must first create either an interface VPC endpoint for your workspaces, or at least one IP prefix list for the IP addresses that you want to allow. You can create both or more than one of both, as well.

- **VPC endpoint** – You can create an interface VPC endpoint that gives access to all of your workspaces. After you have created the endpoint, you need the VPC endpoint ID for each endpoint you want to allow. VPC endpoint IDs have the format `vpce-1a2b3c4d`.

  For information about creating a VPC endpoint for your Grafana workspaces, see Interface VPC endpoints. To create a VPC endpoint specifically for your workspaces, use the `com.amazonaws.region.grafana-workspace` endpoint name.

  For VPC endpoints that you give access to your workspace, you can further limit their access by configuring security groups for the endpoints. To learn more, see Associate security groups and Security group rules in the *Amazon VPC documentation*.

- **Managed prefix list** (for IP address ranges) – to allow IP addresses, you must create one or more prefix lists in Amazon VPC with the list of IP ranges to allow. There are a few limitations for prefix lists when used for Amazon Managed Grafana:
  - Each prefix list can contain up to 100 IP address ranges.
  - IPv6 address ranges in prefix lists are only effective on dual-stack workspaces. On IPv4-only workspaces, IPv6 ranges are ignored.
  - Private IP address ranges (for example, `10.0.0.0/16`) are ignored. You can include private IP address ranges in a prefix list, but Amazon Managed Grafana ignores those when filtering traffic to the workspace. To allow those hosts to reach the workspace, create a VPC endpoint for your workspaces and give them access.

  You create managed prefix lists through the Amazon VPC Console. After you have created the prefix lists, you need the prefix list ID for each list you want to allow in Amazon Managed Grafana. Prefix list IDs have the format `pl-1a2b3c4d`.

  For more information about creating prefix lists, see Group CIDR blocks using managed prefix lists in the *Amazon Virtual Private Cloud User Guide*.

- You must have the necessary permissions to configure or create an Amazon Managed Grafana workspace. For example, you could use the AWS managed policy, `AWSGrafanaAccountAdministrator`.

After you have the list of IDs for the prefix lists or VPC endpoints that you want to give access to your workspace, you are ready to create the network access control configuration.

> **Note**
>
> If you enable network access control, but do not add a prefix list to the configuration, no access to your workspace is allowed, except through the allowed VPC endpoints.
> Similarly, if you enable network access control, but do not add a VPC endpoint to the configuration, no access to your workspace is allowed, except through the allowed IP addresses.
> You must include at least one prefix list or VPC endpoint in the network access control configuration, or you would not be able to access your workspace from anywhere.

To configure network access control for a workspace

1. Open the Amazon Managed Grafana console.
2. In the left navigation pane, choose **All workspaces**.
3. Select the name of the workspace that you want to configure network access control.
4. In the **Network access control** tab, under **Network access control**, choose **Restricted access** to configure network access control.

   > **Note**
   >
   > You can access these same options while creating a workspace.

5. From the drop down select whether you are adding a **Prefix list** or a **VPC endpoint**.
6. Select the VPC endpoint or Prefix list ID that you want to add (alternatively, you can type the ID that you want to use. You must choose at least one.
7. To add more endpoints or lists, select **Add new resource** for each one you want to add.

   > **Note**
   >
   > You can add up to 5 prefix lists and 5 VPC endpoints.

8. To configure IPv6 (dual-stack) connectivity, use the **Workspace configuration options** tab and choose **IP Address Type**. Dual-stack mode requires Grafana version 10 or later. For prerequisites, see Create an Amazon Managed Grafana workspace.
9. Choose **Save changes** to complete the setup.

> **Warning**
>
> If you have existing users of your workspace, include their IP ranges or VPC endpoints in the configuration, or they will lose access with a `403 Forbidden` error. It is recommended that you test existing access points after setting up or modifying the configuration of network access control.

Encryption at rest

By default, Amazon Managed Grafana automatically provides you with encryption at rest and does this using AWS owned encryption keys.

- **AWS owned keys** – Amazon Managed Grafana uses these keys to automatically encrypt data of your workspace. You can't view, manage or use AWS owned keys, or audit their use. However, you don't have to take any action or change any programs to protect the keys that encrypt your data. For more information, see AWS-owned keys in the *AWS KMS Developer Guide*.

Encryption of data at rest helps reduce the operational overhead and complexity that goes into protecting sensitive customer data, such as personally identifiable information. It allows you to build secure applications that meet strict encryption compliance and regulatory requirements.

You can alternatively choose to use a customer managed key when you create your workspace:

- **Customer managed keys** – Amazon Managed Grafana supports the use of a symmetric customer managed key that you create, own, and manage to encrypt the data in your workspace. Because you have full control of this encryption, you can perform such tasks as:
  - Establishing and maintaining key policies
  - Establishing and maintaining IAM policies and grants
  - Enabling and disabling key policies
  - Rotating key cryptographic material
  - Adding tags
  - Creating key aliases
  - Scheduling keys for deletion

For more information, see customer managed keys in the *AWS KMS Developer Guide* and What is AWS KMS?

Choose whether to use customer managed keys or AWS owned keys carefully. Workspaces created with customer managed keys can't be converted to use AWS owned keys later (and vice versa).

Note

- Amazon Managed Grafana automatically enables encryption at rest using AWS owned keys to protect your data at no charge.
- However, AWS KMS charges apply for using a customer managed key. For more information about pricing, see AWS KMS pricing.

Important

- If you disable the customer managed key or remove Amazon Managed Grafana access in the key policy, your workspace will become inaccessible. The workspace will remain in an `ACTIVE` state but will be functionally unavailable. You have 7 days to restore access by re-enabling the key or restoring the key policy. After 7 days, the workspace will transition to a `FAILED` state and can only be deleted.
- Scheduling a key for deletion in AWS KMS has a minimum waiting period of 7 days before the key is deleted. Once a key is deleted, it cannot be restored, and any workspace encrypted with that key will permanently lose access to its data.
- Customer managed key encryption is only available when creating new workspaces. Existing workspaces cannot be converted to use customer managed keys.
- You cannot modify a workspace's customer managed key after creation.

How Amazon Managed Grafana uses grants in AWS KMS

Amazon Managed Grafana requires grants to use your customer managed key.

When you create an Amazon Managed Grafana workspace encrypted with a customer managed key, Amazon Managed Grafana creates grants on your behalf by sending CreateGrant requests to AWS KMS. Grants in AWS KMS are used to give Amazon Managed Grafana access to the KMS key in

your account, even when not called directly on your behalf (for example, when storing dashboard data or user configurations).

Amazon Managed Grafana requires the grants to use your customer managed key for the following internal operations:

- Send CreateGrant requests to AWS KMS to create additional grants as needed.
- Send DescribeKey requests to AWS KMS to verify that the symmetric customer managed KMS key given when creating a workspace is valid.
- Send ReEncryptTo and ReEncryptFrom requests to AWS KMS to re-encrypt data when moving between different encryption contexts.
- Send Encrypt requests to AWS KMS to encrypt data directly with your customer managed key.
- Send Decrypt requests to AWS KMS to decrypt the encrypted data keys so that they can be used to encrypt your data.
- Send GenerateDataKey requests to AWS KMS to generate data keys encrypted by your customer managed key.
- Send GenerateDataKeyWithoutPlaintext requests to AWS KMS to generate encrypted data keys without returning the plaintext version.
- Send RetireGrant requests to AWS KMS to retire grants that are no longer needed.

Amazon Managed Grafana creates grants to the AWS KMS key that allow Amazon Managed Grafana to use the key on your behalf. You can remove access to the key by changing the key policy, by disabling the key, or by revoking the grant. You should understand the consequences of these actions before performing them. This can cause data loss in your workspace.

If you remove access to any of the grants in any way, Amazon Managed Grafana won't be able to access any of the data encrypted by the customer managed key, nor store new data sent to the workspace, which affects operations that are dependent on that data. New updates to the workspace will not be accessible and may be permanently lost.

Warning

- If you disable the key, or remove Amazon Managed Grafana access in the key policy, the workspace data is no longer accessible. The workspace will remain in an `ACTIVE` state but will be functionally unavailable. New updates being sent to the workspace will not be accessible and may be permanently lost. You can restore access to the workspace data

and resume receiving new data by re-enabling the key or restoring Amazon Managed Grafana access to the key within 7 days. After 7 days without access, the workspace will transition to a `FAILED` state.

- If you schedule the key for deletion in AWS KMS, the key will be deleted after the mandatory 7-day waiting period. Once deleted, the key cannot be restored, and the workspace data will be permanently inaccessible.
- If you *revoke* a grant, it can't be recreated, and the data in the workspace is lost permanently.
- Amazon Managed Grafana creates additional child grants through Amazon RDS due to its dependency on RDS for data storage. Revoking these RDS-related grants will have the same permanent data loss effect as revoking the primary Grafana grants.

Step 1: Create a customer managed key

You can create a symmetric customer managed key by using the AWS Management Console, or the AWS KMS APIs. The key must be in the same region as the Amazon Managed Grafana workspace and must be a symmetric key with `ENCRYPT_DECRYPT` key usage.

To create a symmetric customer managed key

- Follow the steps for Creating symmetric customer managed key in the *AWS KMS Developer Guide*.

Key policy

Key policies control access to your customer managed key. Every customer managed key must have exactly one key policy, which contains statements that determine who can use the key and how they can use it. When you create your customer managed key, you can specify a key policy. For more information, see Managing access to customer managed keys in the *AWS KMS Developer Guide*.

To use your customer managed key with your Amazon Managed Grafana workspaces, the following API operations must be permitted in the key policy:

- kms:CreateGrant – Adds a grant to a customer managed key. Grants control access to a specified KMS key, which allows access to grant operations Amazon Managed Grafana requires. For more

information, see Using Grants in the *AWS KMS Developer Guide*. This allows Amazon Managed Grafana to do the following:

- Call `GenerateDataKey` to generate an encrypted data key and store it.
- Call `Decrypt` to use the stored encrypted data key to access encrypted data.

- kms:DescribeKey – Provides the customer managed key details to allow Amazon Managed Grafana to validate the key.

The following are policy statement examples you can add for Amazon Managed Grafana:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow IAM Users and Roles to validate KMS key",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/root"
      },
      "Action": [
        "kms:DescribeKey",
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": [
            "grafana.<region>.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "Allow IAM Users and Roles to create grant on KMS key",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/root"
      },
      "Action": "kms:CreateGrant",
      "Resource": "*",
      "Condition": {
```

```
        "StringEquals": {
          "kms:ViaService": [
            "grafana.<region>.amazonaws.com"
          ],
          "kms:GrantConstraintType": "EncryptionContextSubset"
        },
        "ForAllValues:StringEquals": {
          "kms:GrantOperations": [
            "CreateGrant",
            "RetireGrant",
            "Decrypt",
            "Encrypt",
            "GenerateDataKey",
            "GenerateDataKeyWithoutPlaintext",
            "ReEncryptFrom",
            "ReEncryptTo"
          ]
        }
      }
    }
  ]
}
```

- For more information about specifying permissions in a policy, see the AWS Key Management Service Developer Guide.
- For more information about troubleshooting key access, see the AWS Key Management Service Developer Guide.

Step 2: Specifying a customer managed key for Amazon Managed Grafana

When you create a workspace, you can specify the customer managed key by entering a KMS Key ARN, which Amazon Managed Grafana uses to encrypt the data stored by the workspace.

Using the AWS Management Console

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. Choose **Create workspace**.
3. In the **Encryption** section, select **Customer managed key**.

4. Enter the ARN of your customer managed key in the **KMS Key ARN** field.
5. Complete the remaining workspace configuration and choose **Create workspace**.

Using the AWS CLI

You can specify a customer managed key when creating a workspace using the `--kms-key-id` parameter:

```
aws grafana create-workspace \
    --workspace-name "my-encrypted-workspace" \
    --workspace-description "Workspace with customer managed encryption" \
    --account-access-type "CURRENT_ACCOUNT" \
    --authentication-providers "AWS_SSO" \
    --permission-type "SERVICE_MANAGED" \
    --kms-key-id "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
```

Monitoring your encryption keys for Amazon Managed Grafana

When you use an AWS KMS customer managed key with your Amazon Managed Grafana workspaces, you can use AWS CloudTrail or Amazon CloudWatch Logs to track requests that Amazon Managed Grafana sends to AWS KMS.

The following examples are AWS CloudTrail events for `CreateGrant`, `DescribeKey`, `GenerateDataKey`, and `Decrypt` to monitor KMS operations called by Amazon Managed Grafana to access data encrypted by your customer managed key:

CreateGrant

When you use an AWS KMS customer managed key to encrypt your workspace, Amazon Managed Grafana sends `CreateGrant` requests on your behalf to access the KMS key you specified. The grants that Amazon Managed Grafana creates are specific to the resource associated with the AWS KMS customer managed key.

The following example event records a `CreateGrant` operation:

```
{
"eventVersion": "1.08",
"userIdentity": {
"type": "AssumedRole",
```

```
"principalId": "TESTANDEXAMPLE:Sampleuser01",
"arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
"accountId": "111122223333",
"accessKeyId": "EXAMPLE-KEY-ID1",
"sessionContext": {
"sessionIssuer": {
"type": "Role",
"principalId": "TESTANDEXAMPLE:Sampleuser01",
"arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
"accountId": "111122223333",
"userName": "Admin"
},
"webIdFederationData": {},
"attributes": {
"mfaAuthenticated": "false",
"creationDate": "2021-04-22T17:02:00Z"
}
},
"invokedBy": "grafana.amazonaws.com"
},
"eventTime": "2021-04-22T17:07:02Z",
"eventSource": "kms.amazonaws.com",
"eventName": "CreateGrant",
"awsRegion": "us-west-2",
"sourceIPAddress": "172.12.34.56",
"userAgent": "ExampleDesktop/1.0 (V1; OS)",
"requestParameters": {
"retiringPrincipal": "grafana.amazonaws.com",
"operations": [
"CreateGrant",
"DescribeKey",
"ReEncryptTo",
"ReEncryptFrom",
"Encrypt",
"Decrypt",
"GenerateDataKey",
"GenerateDataKeyWithoutPlaintext",
"RetireGrant"
],
"keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
"granteePrincipal": "grafana.amazonaws.com"
},
"responseElements": {
"grantId": "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE"
```

```
},
"requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
"eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
"readOnly": false,
"resources": [
{
"accountId": "111122223333",
"type": "AWS::KMS::Key",
"ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
}
],
"eventType": "AwsApiCall",
"managementEvent": true,
"eventCategory": "Management",
"recipientAccountId": "111122223333"
}
```


DescribeKey

Amazon Managed Grafana uses the `DescribeKey` operation to verify if the AWS KMS customer managed key associated with your workspace exists in the account and region.

The following example event records the `DescribeKey` operation:


```
{
"eventVersion": "1.08",
"userIdentity": {
"type": "AssumedRole",
"principalId": "TESTANDEXAMPLE:Sampleuser01",
"arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
"accountId": "111122223333",
"accessKeyId": "EXAMPLE-KEY-ID1",
"sessionContext": {
"sessionIssuer": {
"type": "Role",
"principalId": "TESTANDEXAMPLE:Sampleuser01",
"arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
"accountId": "111122223333",
"userName": "Admin"
},
"webIdFederationData": {},
"attributes": {
"mfaAuthenticated": "false",
```

```
"creationDate": "2021-04-22T17:02:00Z"
}
},
"invokedBy": "grafana.amazonaws.com"
},
"eventTime": "2021-04-22T17:07:02Z",
"eventSource": "kms.amazonaws.com",
"eventName": "DescribeKey",
"awsRegion": "us-west-2",
"sourceIPAddress": "172.12.34.56",
"userAgent": "ExampleDesktop/1.0 (V1; OS)",
"requestParameters": {
"keyId": "00dd0db0-0000-0000-ac00-b0c000SAMPLE"
},
"responseElements": null,
"requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
"eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
"readOnly": true,
"resources": [
{
"accountId": "111122223333",
"type": "AWS::KMS::Key",
"ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
}
],
"eventType": "AwsApiCall",
"managementEvent": true,
"eventCategory": "Management",
"recipientAccountId": "111122223333"
}
```


GenerateDataKey

Amazon Managed Grafana uses the `GenerateDataKey` operation to generate data keys that are used to encrypt workspace data.

The following example event records the `GenerateDataKey` operation:


```
{
"eventVersion": "1.08",
"userIdentity": {
"type": "AssumedRole",
"principalId": "TESTANDEXAMPLE:Sampleuser01",
```

```
"arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
"accountId": "111122223333",
"accessKeyId": "EXAMPLE-KEY-ID1",
"sessionContext": {
"sessionIssuer": {
"type": "Role",
"principalId": "TESTANDEXAMPLE:Sampleuser01",
"arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
"accountId": "111122223333",
"userName": "Admin"
},
"webIdFederationData": {},
"attributes": {
"mfaAuthenticated": "false",
"creationDate": "2021-04-22T17:02:00Z"
}
},
"invokedBy": "grafana.amazonaws.com"
},
"eventTime": "2021-04-22T17:07:02Z",
"eventSource": "kms.amazonaws.com",
"eventName": "GenerateDataKey",
"awsRegion": "us-west-2",
"sourceIPAddress": "172.12.34.56",
"userAgent": "ExampleDesktop/1.0 (V1; OS)",
"requestParameters": {
"keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
"keySpec": "AES_256"
},
"responseElements": null,
"requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
"eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
"readOnly": false,
"resources": [
{
"accountId": "111122223333",
"type": "AWS::KMS::Key",
"ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
}
],
"eventType": "AwsApiCall",
"managementEvent": true,
"eventCategory": "Management",
"recipientAccountId": "111122223333"
```

```
}
```

Decrypt

Amazon Managed Grafana uses the `Decrypt` operation to decrypt encrypted data keys so that they can be used to decrypt workspace data.

The following example event records the `Decrypt` operation:

```
{
"eventVersion": "1.08",
"userIdentity": {
"type": "AssumedRole",
"principalId": "TESTANDEXAMPLE:Sampleuser01",
"arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
"accountId": "111122223333",
"accessKeyId": "EXAMPLE-KEY-ID1",
"sessionContext": {
"sessionIssuer": {
"type": "Role",
"principalId": "TESTANDEXAMPLE:Sampleuser01",
"arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
"accountId": "111122223333",
"userName": "Admin"
},
"webIdFederationData": {},
"attributes": {
"mfaAuthenticated": "false",
"creationDate": "2021-04-22T17:02:00Z"
}
},
"invokedBy": "grafana.amazonaws.com"
},
"eventTime": "2021-04-22T17:07:02Z",
"eventSource": "kms.amazonaws.com",
"eventName": "Decrypt",
"awsRegion": "us-west-2",
"sourceIPAddress": "172.12.34.56",
"userAgent": "ExampleDesktop/1.0 (V1; OS)",
"requestParameters": {
"encryptionContext": {
"aws:grafana:workspace-id": "g-1234567890abcdef0"
}
```

```
},
"responseElements": null,
"requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
"eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
"readOnly": true,
"resources": [
{
"accountId": "111122223333",
"type": "AWS::KMS::Key",
"ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
}
],
"eventType": "AwsApiCall",
"managementEvent": true,
"eventCategory": "Management",
"recipientAccountId": "111122223333"
}
```


Learn more

The following resources provide more information about data encryption at rest.

- For more information about AWS KMS basic concepts, see the AWS KMS Developer Guide.
- For more information about Security best practices for AWS KMS, see the AWS KMS Developer Guide.

Connect to data sources or notification channels in Amazon VPC from Amazon Managed Grafana

By default, traffic from your Amazon Managed Grafana workspace to data sources or notification channels flows via the public Internet. This limits the connectivity from your Amazon Managed Grafana workspace to services that are publicly accessible.

> **Note**
>
> When you have not configured a private VPC, and Amazon Managed Grafana is connecting to publicly accessible data sources, it connects to some AWS services in the same region via AWS PrivateLink. This includes services such as CloudWatch, Amazon Managed Service for Prometheus and AWS X-Ray. Traffic to those services does not flow via the public Internet.

If you want to connect to private-facing data sources that are within a VPC, or keep traffic local to a VPC, you can connect your Amazon Managed Grafana workspace to the Amazon Virtual Private Cloud (Amazon VPC) hosting these data sources. After you configure the VPC data source connection, all traffic flows via your VPC.

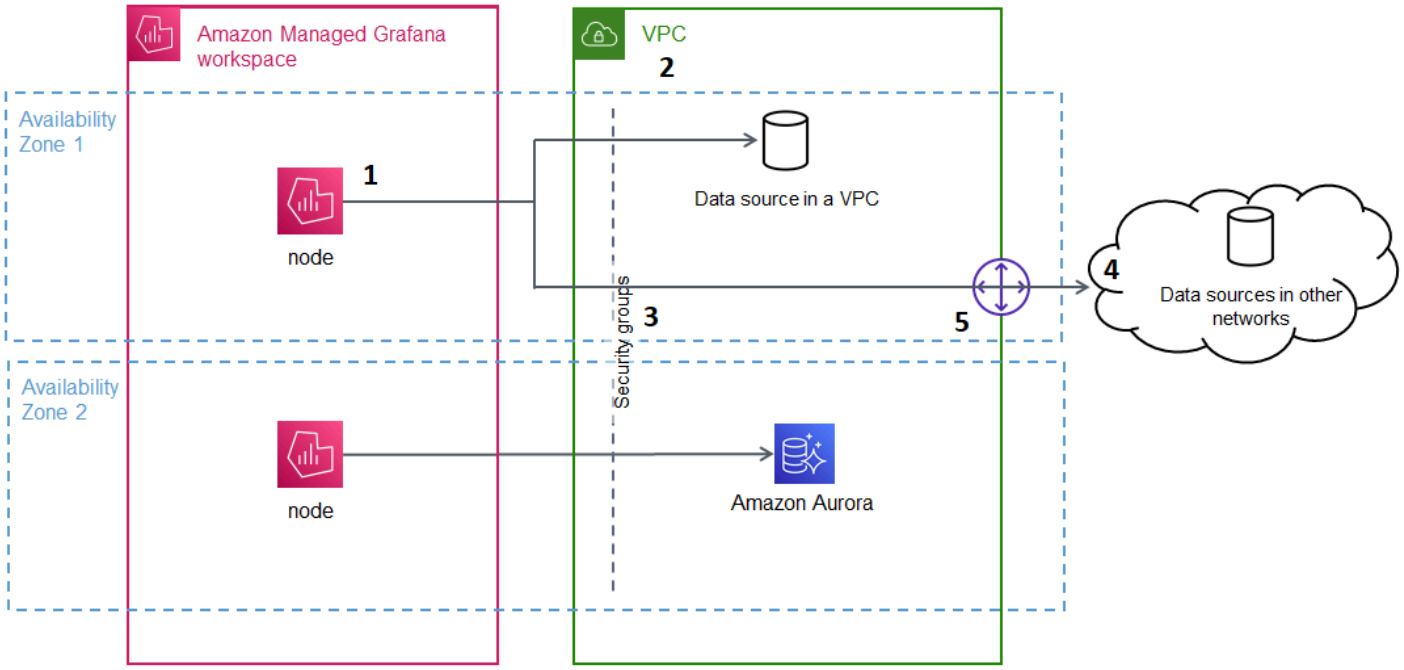
A *virtual private cloud* (VPC) is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks, including other VPCs and the public internet. Use Amazon VPC to create and manage your VPCs in the AWS Cloud. Amazon VPC gives you full control over your virtual networking environment, including resource placement, connectivity, and security. Amazon Managed Grafana data sources, and other resources, can be created in your VPC. For more information on Amazon VPC, see What is Amazon VPC? in the *Amazon Virtual Private Cloud User Guide*.

> **Note**
>
> If you want your Amazon Managed Grafana workspace to connect to data outside of the VPC, in another network or public Internet, you must add routing to the other network. For information about how to connect your VPC to another network, see Connect your VPC to other networks in the *Amazon Virtual Private Cloud User Guide*.

How VPC connectivity works

Amazon VPC gives you complete control over your virtual networking environment, including creating public-facing and private-facing *subnets* for your application to connect, and *security groups* to manage what services or resources have access to the subnets.

To use Amazon Managed Grafana with resources in a VPC, you must create a connection to that VPC for the Amazon Managed Grafana workspace. After you set up the connection, Amazon Managed Grafana connects your workspace to each provided subnet in each Availability Zone in that VPC, and all traffic to or from the Amazon Managed Grafana workspace flows through the VPC. The following diagram shows how this connectivity looks, logically.



Amazon Managed Grafana creates a connection (**1**) per subnet (using an elastic network interface, or ENI) to connect to the VPC (**2**). The Amazon Managed Grafana VPC connection is associated with a set of security groups (**3**) that control the traffic between the VPC and your Amazon Managed Grafana workspace. All traffic is routed through the configured VPC, including alert destination and data source connectivity. To connect to data sources and alert destinations in other VPCs or the public Internet (**4**), create a gateway (**5**) between the other network and your VPC.

Create a connection to a VPC

This section describes the steps to connect to a VPC from your existing Amazon Managed Grafana workspace. You can follow these same instructions when creating your workspace. For more information about creating a workspace, see Create an Amazon Managed Grafana workspace.

Prerequisites

The following are prerequisites for establishing a connection to a VPC from an existing Amazon Managed Grafana workspace.

- You must have the necessary permissions to configure or create an Amazon Managed Grafana workspace. For example, you could use the AWS managed policy, `AWSGrafanaAccountAdministrator`.

- You must have a VPC setup in your account with at least two Availability Zones configured, with one *private subnet* configured for each. You must know the subnet and security group information for your VPC.

  Note

  Local Zones and Wavelength Zones are not supported.
  VPCs configured with `Tenancy` set to `Dedicated` are not supported.

  Important

  A minimum of 15 available IP addresses must be in each subnet connected to your Amazon Managed Grafana workspace. We strongly recommend that you configure alarms to monitor IP usage in your VPC subnets. If the number of available IP addresses for a subnet falls below 15, you might experience the following issues:

  - Inability to make configuration changes to your workspace until you free up additional IP addresses or attach subnets with additional IP addresses
  - Your workspace will not be able to receive security updates or patches
  - In rare scenarios, you could experience a complete availability loss for the workspace, resulting in non-functioning alerts and inaccessible dashboards

- If you are connecting an existing Amazon Managed Grafana workspace that has data sources configured, we recommend that you have your VPC configured to connect to those data sources before connecting Amazon Managed Grafana to the VPC. This includes services such as CloudWatch that are connected through AWS PrivateLink. Otherwise, connectivity to those data sources is lost.
- If your VPC already has multiple gateways to other networks, you might need to set up DNS resolution across the multiple gateways. For more information, see Route 53 Resolver.

Connecting to a VPC from an existing Amazon Managed Grafana workspace

The following procedure describes adding an Amazon VPC data source connection to an existing Amazon Managed Grafana workspace.

> **Note**
>
> When you configure the connection to Amazon VPC, it creates an IAM role. With this role, Amazon Managed Grafana can create connections to the VPC. The IAM role uses the service-linked role policy, `AmazonGrafanaServiceLinkedRolePolicy`. To learn more about service-linked roles, see Service-linked role permissions for Amazon Managed Grafana.

To connect to a VPC from an existing Amazon Managed Grafana workspace

1. Open the Amazon Managed Grafana console.
2. In the left navigation pane, choose **All workspaces**.
3. Select the name of the workspace that you want to add a VPC data source connection.
4. In the **Network access settings** tab, next to **Outbound VPC connection**, choose **Edit** to create your VPC connection.
5. Choose the **VPC** you want to connect.
6. Under **Mappings**, select the Availability Zones you want to use. You must choose at least two.
7. Select at least one *private subnet* in each Availability Zone. The subnets must support IPv4.
8. Select at least one **Security group** for your VPC. You can specify up to 5 security groups. Alternately, you can create a security group to apply to this connection.
9. Choose **Save changes** to complete the setup.

Now that you have set up your VPC connection, you can add Connect to data sources accessible from that VPC to your Amazon Managed Grafana workspace.

Changing outbound VPC settings

To change your settings, you can return to the **Network access settings** tab of your workspace configuration, or you can use the UpdateWorkspace API.

> **Important**
>
> Amazon Managed Grafana manages your VPC configuration for you. Do not edit these VPC settings using the Amazon EC2 console or APIs, or the settings will get out of sync.

Troubleshoot using VPC with Amazon Managed Grafana

Answers to common questions regarding using Amazon Virtual Private Cloud (Amazon VPC) with Amazon Managed Grafana.

When do I need to configure a VPC in Amazon Managed Grafana?

You need to configure a VPC in Amazon Managed Grafana if you are trying to connect to a data source that is only available in a private VPC (that is not publicly accessible).

For data sources that are publicly available, or have a public-facing endpoint, you do not need to configure a VPC.

If you connect to Amazon CloudWatch, Amazon Managed Service for Prometheus, or AWS X-Ray, you do not need to configure a VPC. These data source are connected to Amazon Managed Grafana via AWS PrivateLink by default.

Why are my existing data sources failing to connect after I configured a VPC with my Amazon Managed Grafana workspace?

Your existing data sources are likely accessible through the public network and your Amazon VPC configuration does not allow access to the public network. After configuring the VPC connection in your Amazon Managed Grafana workspace, all traffic must flow through that VPC. This includes private data sources hosted within that VPC, data sources in another VPC, AWS Managed Services that are not available in the VPC, and internet-facing data sources.

To resolve this issue, you must connect the other data sources to the VPC that you have configured:

- For internet-facing data sources, connect the VPC to the internet. You can, for example, Connect to the internet or other networks using NAT devices (from the *Amazon Virtual Private Cloud User Guide*).
- For data sources in other VPCs, create a peering between the two VPCs. For more information, see Connect VPCs using VPC peering (from the *Amazon Virtual Private Cloud User Guide*).
- For AWS Managed Services that are not accessible in your VPC, such as CloudWatch, X-Ray, or Amazon Managed Service for Prometheus, you might need to create an interface VPC endpoint for that service in your VPC. For more information, see Access an AWS service using an interface VPC endpoint in the *AWS PrivateLink Guide*.

Can I use a VPC with dedicated tenancy?

No, VPCs configured with `Tenancy` set to `Dedicated` are not supported.

Can I connect both AWS Managed Services (such as Amazon Managed Service for Prometheus, CloudWatch, or X-Ray) and private data sources (including Amazon Redshift) to the same Amazon Managed Grafana workspace?

Yes. You must configure connectivity to the AWS Managed Services in the same VPC as your private data sources (for example, using an interface VPC endpoint or a NAT Gateway), and configure your Amazon Managed Grafana workspace to connect to the same VPC.

Why do I get a `502 Bad Gateway Error` when I am trying to connect to a data source after I configured the VPC in my Amazon Managed Grafana workspace?

The following are the three most common reasons why your data source connection returns a `502` error.

- **Security group error** — The security groups selected during VPC configuration in Amazon Managed Grafana must allow connectivity to the data source via inbound and outbound rules.

  To resolve this issues, make sure that the rules in both the data source security group and the Amazon Managed Grafana security group allow this connectivity.
- **User permission error** — The assigned workspace user does not have the right permissions to query the data source.

  To resolve this issue, confirm that the user has the required IAM permissions to edit the workspace, and the correct data source policy to access and query the data from the hosting service. Permissions are available in the AWS Identity and Access Management (IAM) console at https://console.aws.amazon.com/iam/.
- **Incorrect connection details provided** — The Amazon Managed Grafana workspace is unable to connect to your data source due to incorrect connection details provided.

  To resolve this issue, please confirm the information in the data source connection, including the data source authentication and endpoint URL, and retry the connection.

Can I connect to multiple VPCs from the same Amazon Managed Grafana workspace?

You can only configure a single VPC for a Amazon Managed Grafana workspace. To access data sources in a different VPC, or across regions, see the next question.

How do I connect to data sources in a different VPC? How do I connect to data sources from a VPC that's in a different AWS Region or AWS account?

You can use VPC peering or AWS Transit Gateway to connect the cross-region or cross-account VPCs, then connect the VPC that is in the same AWS account and Region as your Amazon Managed Grafana workspace. Amazon Managed Grafana connects to the outside data sources as any other connection within the VPC.

> **Note**
>
> If VPC peering isn't an option for you, share your use case with your Account Manager, or send email to aws-grafana-feedback@amazon.com.

When my Amazon Managed Grafana workspace is connected to a VPC will I still be able to connect to other public data sources?

Yes. You can connect data sources from both your VPC and public data sources to a single Amazon Managed Grafana workspace at the same time. For public data sources, you must configure VPC connectivity via a NAT Gateway, or other VPC connection. Requests to public data sources traverse your VPC, giving you additional visibility and control over those requests.

What should I do if I'm unable to update an Amazon Managed Grafana workspace due to insufficient IP addresses?

You might encounter the following error when modifying your Amazon Managed Grafana workspace configuration: All subnets in the VPC configuration must have at least 15 available IP addresses.

You will receive this error if one or more subnets connected to your workspace do not meet the minimum IP requirements. A minimum of 15 available IP addresses must be in each subnet connected to your workspace. When the number of available IP addresses for a subnet falls below 15, you might experience the following issues:

- Inability to make configuration changes to your workspace until you free up additional IP addresses or attach subnets with additional IP addresses
- Your workspace will not be able to receive security updates or patches
- In rare scenarios, you could experience a complete availability loss for the workspace, resulting in non-functioning alerts and inaccessible dashboards

Mitigate IP exhaustion

1. If a subnet has less than 15 available IP addresses, release IP addresses associated with instances or delete unused network interfaces to free up IP capacity.
2. If you are unable to free up IP addresses in the existing subnet, then you must replace the subnet with one that has at least 15 available IP addresses. We recommend using dedicated subnets for Amazon Managed Grafana.

Replace a subnet

1. Open the Amazon Managed Grafana console.
2. In the left navigation pane, choose **All workspaces**, then select the name of your workspace.
3. In the **Network access control** tab, next to **Outbound VPC connection**, choose **Edit**.
4. Under **Mappings**, select the Availability Zone which contains the subnet with insufficient IP addresses.
5. In the dropdown, deselect the subnet with insufficient IP addresses and select a subnet with at least 15 available IP addresses. If necessary, create a new subnet in your VPC. For more information, see Create a subnet in the *Amazon VPC User Guide*.
6. Choose **Save changes** to complete the setup.

Before configuring a VPC connection my Grafana alerts were successfully being sent to downstream services, such as PagerDuty and Slack. After configuring VPC, why are my Grafana alerts not being delivered to these notification destinations?

After you configure a VPC connection for an Amazon Managed Grafana workspace, all traffic to data sources in the workspace flows through the configured VPC. Make sure that the VPC has a route to reach these alert notification services. For example, alert notification destinations hosted by third parties might require connectivity to the Internet. Much like data sources, configure an Internet or AWS Transit Gateway, or other VPC connection to the external destination.

Can I edit my VPC manually? Why does modifying my security group or subnet cause my Amazon Managed Grafana workspace to become unavailable?

The Amazon Managed Grafana VPC connection uses the security groups and subnets to control the traffic allowed between the VPC and your Amazon Managed Grafana workspace. When the security group or subnet is modified or deleted from outside the Amazon Managed Grafana console (such as with the VPC console), the VPC connection in your Amazon Managed Grafana workspace stops protecting your workspace security, and the workspace becomes unreachable. To fix this issue, update the security groups configured for your Amazon Managed Grafana workspace in the Amazon Managed Grafana console. When viewing your workspace, select **Outbound VPC connection** on the **Network access control** tab to modify the subnets or security groups associated with the VPC connection.

Configure a Amazon Managed Grafana workspace

Amazon Managed Grafana configuration can be separated into configuration of the Amazon Managed Grafana authentication and permissions, and configuration of the Grafana workspace. This section includes information regarding configuration of your Grafana workspace.

For more information about configuring Amazon Managed Grafana authentication and permissions, see the following topics.

- Authenticate users in Amazon Managed Grafana workspaces
- Manage user and group access to Amazon Managed Grafana workspaces
- Users, teams, and permissions

You can modify the configuration of your Grafana workspace within Amazon Managed Grafana on the **Workspace configuration options** tab when viewing the properties of your workspace.

Making configuration changes to your Grafana instance can cause the instance to restart to reload the new settings. After configuration changes are made, your users might need to refresh any browser pages that show the Grafana workspace.

> **Note**
>
> The same options are available to you when you first create your workspace.

To change the configuration of a Grafana workspace using the Amazon Managed Grafana console

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the left navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the name of the workspace that you want to configure. This opens the details for that workspace.
5. Choose the **Workspace configuration options** tab to see the instance configuration options for your instance.
6. Select **Edit** next to either **Grafana alerting**, **Plugin management**, or **IP Address Type**.

   - **Grafana alerting**

     You can enable Grafana alerting. To view Prometheus alerts in your Grafana workspace, select the check box to **Turn Grafana alerting on**. In workspaces running version 8 or 9, this will send multiple notifications for your Grafana alerts. If you use alerts defined in Grafana, we recommend updating your workspace to version 10.4 or later.

     If you want to the classic Grafana alerts instead, *clear* the check box next to **Turn Grafana alerting on**. This turns on the classic dashboard alerts. Even if you don't turn Grafana alerting on, your existing Grafana alerts are evaluated.

     > **Note**
     >
     > Classic dashboard alerts have been removed in Grafana version 12. In version 12 workspaces, Grafana alerting is always enabled and the toggle is no longer available. For more information, see Migrating classic dashboard alerts to Grafana alerting.

   - **Plugin management**

     To turn on plugin management, select the check box to **Turn plugin management on**. Turning plugin management on allows admins in your Amazon Managed Grafana workspace to install, update, or remove plugins using the Grafana plugin catalog. This option is only available for workspaces that support Grafana version 9 or newer.

   - **IP Address type**

Choose **IPv4 only** or **Dual-stack mode**. Your selection determines the addressing protocols that resources use to communicate with your workspace. Choose dual-stack mode when you have resources that must communicate with your workspace over IPv6. Dual-stack mode is available only for workspaces that support Grafana version 10 or later.

Important

Before enabling dual-stack mode, verify the following:

- The `sso.signin.aws` domain is allowed in your network firewall and identity provider configurations.
- Your network allows IPv6 traffic.
- Any prefix lists used for network access control include the appropriate IP ranges.
- If your identity provider uses CORS validation, the new domain is added to the CORS allowlist.
- If your workspace uses a VPC configuration, your subnets must have IPv6 CIDR blocks assigned.

Workspaces using direct SAML authentication or AWS Identity and Access Management Identity Center's built-in directory are not affected by the sign-in URL change. For more information, see Assertion Consumer Service (ACS) endpoints in the *AWS Identity and Access Management Identity Center User Guide*.

If you later switch from dual-stack back to IPv4 only, any clients or resources currently connecting over IPv6 will lose access to your workspace.

Note

If you turn *off* Grafana alerting, you lose all changes made to the alerting configuration while Grafana alerting was on. This includes any new alert rules that you created.

For more information about using Grafana alerting, and the effects of turning it on or off, see Alerts in Grafana version 10.

The next section shows how to make changes to the Grafana instance configuration using the Amazon Managed Grafana API or the AWS CLI.

Setting configuration with API or AWS CLI

You can set the Grafana workspace configuration using the Amazon Managed Grafana API or the AWS CLI.

> **Note**
>
> The `configuration` is a JSON string to allow for future configuration settings which made be added later.

AWS CLI

To update Amazon Managed Grafana instance configuration using the AWS CLI

Run the following command to turn on the Grafana alerting and plugin management features for an instance. Replace the *<region>* and *<workspace-id>* strings with appropriate values for your instance.

```
aws grafana update-workspace-configuration \
    --region region \
    --workspace-id <workspace-id> \
    --configuration '{"plugins": {"pluginAdminEnabled": true}, "unifiedAlerting":
 {"enabled": true}}'
```

The configuration currently supports the following options. These turn Grafana alerting or plugin management on or off.

- To enable Grafana alerting, use this configuration option:

  ```
  --configuration '{"unifiedAlerting": { "enabled": true }}'
  ```

- To enable plugin management, use this configuration option:

  ```
  --configuration '{"plugins": {"pluginAdminEnabled": true }}'
  ```

  This option is only available in workspaces that support Grafana version 9 or newer.

Amazon Managed Grafana API

To update Amazon Managed Grafana instance configuration using the API

Use the following action to turn on the Grafana alerting and plugin management features for an instance. Replace the *<workspace-id>* string with an appropriate value for your instance.

```
PUT /workspaces/<workspace-id>/configuration HTTP/1.1
Content-type: application/json

{
   "configuration": "{ \"unifiedAlerting\": { \"enabled\": true }, \"plugins\":
 { \"pluginAdminEnabled\": true }}"
}
```

The configuration currently supports the following options. These turn Grafana alerting or plugin management on or off.

- To enable Grafana alerting, use this configuration option:

  ```
  "configuration": "{\"unifiedAlerting\": { \"enabled\": true }}"
  ```

- To enable plugin management, use this option:

  ```
  "plugins": "{\"pluginAdminEnabled\": true }"
  ```

  This option is only available in workspaces that support Grafana version 9 or newer.

Troubleshoot degraded workspaces in Amazon Managed Grafana

An Amazon Managed Grafana workspace can enter a degraded state for several reasons, including VPC configuration issues and KMS key problems. When a workspace is degraded, you may experience availability loss, inability to make configuration changes, and missed security updates. The following sections describe each degraded reason and the actions you can take to resolve it.

KMS key disabled (not recoverable)

Your workspace has failed and cannot be recovered because the KMS key used in the workspace has been disabled for more than 7 days, or the KMS grant has been revoked.

You will experience the following issues:

- A complete availability loss for the workspace, resulting in non-functioning alerts and inaccessible dashboards
- Inability to make configuration changes to your workspace
- Your workspace will not be able to receive security updates or patches
- All workspace data is permanently lost and cannot be recovered

To resolve this issue:

This workspace cannot be recovered. You must create a new workspace. For more information about encryption at rest, see Encryption at rest. For best practices on managing KMS keys, see Best practices for AWS KMS in the *AWS KMS Developer Guide*.

KMS key disabled (recoverable)

Your workspace is disabled and non-operational because the KMS key used for customer managed key encryption has been disabled.

Until you take action, you will experience the following issues:

- A complete availability loss for the workspace, resulting in non-functioning alerts and inaccessible dashboards
- Inability to make configuration changes to your workspace
- Your workspace will not be able to receive security updates or patches

To resolve this issue:

Re-enable the KMS key and restore Amazon Managed Grafana access in the key policy. For more information, see Enabling and disabling keys in the *AWS KMS Developer Guide*.

Important

You must re-enable the KMS key within 7 days. After this period, the workspace transitions to a `FAILED` state and cannot be recovered.
If you revoke KMS grants created by Amazon Managed Grafana to access your KMS key, the grants cannot be recreated, and the data in the workspace is lost permanently. For more information about grants, see Grants in AWS KMS in the *AWS KMS Developer Guide*.

Deleted security group

Your workspace is disabled and non-operational because a security group associated with the workspace VPC configuration has been deleted.

Until you take action, you will experience the following issues:

- A complete availability loss for the workspace, resulting in non-functioning alerts and inaccessible dashboards
- Inability to make configuration changes to your workspace
- Your workspace will not be able to receive security updates or patches

To resolve this issue:

1. Open the Amazon Managed Grafana console and select your workspace.
2. Update the security groups to valid security groups under the **Outbound VPC connection** setting.
3. Confirm the change and retry the VPC connection.

To avoid this issue in the future, update the security groups in your Amazon Managed Grafana console before deleting them from their VPC.

Deleted subnet

Your workspace is disabled and non-operational because a subnet has been deleted from your Elastic Network Interface (ENI).

Until you take action, you will experience the following issues:

- A complete availability loss for the workspace, resulting in non-functioning alerts and inaccessible dashboards
- Inability to make configuration changes to your workspace
- Your workspace will not be able to receive security updates or patches

To resolve this issue:

1. Open the Amazon Managed Grafana console and select your workspace.
2. Update the subnets to valid subnets under the **Outbound VPC connection** setting.
3. Confirm the change and retry the VPC connection.

To avoid this issue in the future, update the subnets in your Amazon Managed Grafana console before deleting them from their VPC.

IP address exhaustion

Your workspace is experiencing availability loss because the subnets connected to your workspace do not have enough free IP addresses.

To resolve this issue:

1. Open the Amazon Managed Grafana console. In the left navigation pane, choose **All workspaces**, then select your workspace.
2. In the **Network access control** tab, under **Outbound VPC connection**, choose each subnet to access the Subnet Details page.
3. Verify that each subnet has at least 15 available IPv4 addresses.
4. If a subnet has fewer than 15 free IP addresses, free up IP addresses by releasing addresses associated with instances or deleting unused network interfaces.
5. If you cannot free up IP addresses, replace the subnet with one that has at least 15 free IP addresses. We recommend using dedicated subnets for Amazon Managed Grafana. For step-by-step instructions, see What should I do if I'm unable to update an Amazon Managed Grafana workspace due to insufficient IP addresses?.

We strongly recommend that you configure alarms to monitor IP usage in your VPC subnets. For more information, see Track IP addresses in the *Amazon VPC IPAM Guide*.

Missing DHCP option set

Your workspace is experiencing availability loss because the VPC connected to your workspace does not have a DHCP option set configured.

To resolve this issue:

1. Open the Amazon Managed Grafana console. In the left navigation pane, choose **All workspaces**, then select your workspace.
2. In the **Network access control** tab, under **Outbound VPC connection**, open the VPC associated with your workspace.
3. In the VPC Details, choose **Actions**, then choose **Edit VPC settings**.
4. Under **DHCP settings**, change the DHCP option set from **No DHCP Option set** to a valid option set. For more information, see DHCP option sets in the *Amazon VPC User Guide*.

Delete a Amazon Managed Grafana workspace

If you delete an Amazon Managed Grafana workspace, all the configuration data for that workspace is also deleted. This includes dashboards, data source configuration, alerts, and snapshots.

To delete an Amazon Managed Grafana workspace

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the left navigation pane, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the name of the workspace that you want to delete.
5. Choose **Delete**.
6. To confirm the deletion, enter the name of the workspace and choose **Delete**.

> **Note**
>
> This procedure deletes a workspace. Other resources may not be deleted. For example, IAM roles that were in use by the workspace are not deleted (but may be unlocked if they are no longer in use).

Use your Grafana workspace

Your Grafana workspace is where you work on projects to create visualizations and explore your metrics. Set up and query data sources for your metrics. Create panels inside dashboards to view your metrics. Explore your data. Create alarms on your metrics.

The topics in this section explain how to use your Amazon Managed Grafana workspace.

> **Note**
>
> Some topics vary based on the version of Grafana that you have in your workspace. For documentation specific to each version, see Working in Grafana version 10, Working in Grafana version 9, and Working in Grafana version 8. For information about upgrading your workspace from one version to another, see Update your workspace version.

Topics

- What is Grafana?
- Connect to your workspace
- Users, teams, and permissions
- Create your first dashboard
- Extend your workspace with plugins
- Connect to data sources
- Working in Grafana version 12
- Working in Grafana version 10
- Working in Grafana version 9
- Working in Grafana version 8
- Change your preferences
- Gather information for support
- Classic dashboard alerts

What is Grafana?

Grafana is open source visualization and analytics software. You can use it to query, visualize, alert on, and explore your metrics no matter where they are stored.

For example, if you want to view the metric, log, and trace data for your application, you might create a dashboard. If you are the administrator for a corporation and you manage Grafana for multiple teams, you might need to set up provisioning and authentication.

The following sections provide an overview of things you can do with your Grafana database and links so that you can learn more.

Explore metrics and logs

Explore your data through one-time, or ad hoc, queries and dynamically drilling down. You can split the view and compare different time ranges, queries, and data sources side by side.

For more information, see Explore in Grafana version 10.

Alerts

If you're using Grafana alerting, alerts can be sent through different alert notifiers, including the following:

- Amazon SNS
- PagerDuty
- VictorOps
- OpsGenie
- Slack

For more information, see Alerts in Grafana version 10.

Annotations

Annotate graphs with rich events from different data sources. Pause on events to see the full event metadata and tags.

This feature, which shows up as a graph marker in Grafana, is useful for correlating data in case something goes wrong. You can create the annotations manually by pressing **Ctrl** while you choose a graph and then entering some text. Or you can fetch data from any data source.

For more information, see Annotate visualizations.

Dashboard variables

Use template variables to create dashboards that can be reused for many different use cases. With these templates, values aren't hardcoded. This means that you can use a dashboard for multiple servers. For example, if you have a production server and a test server, you can use the same dashboard for both.

Templating helps you drill down into your data. For example, you can drill down from all data to North America data, down to Texas data, and beyond. You can also share these dashboards across teams within your organization. If you create a great dashboard template for a popular data source, you can also contribute it to the whole community to customize and use.

For more information, see Variables.

Connect to your workspace

Before you can use your Amazon Managed Grafana workspace you must connect to it by signing in with the identity provider that you have set up. If you have not set up an authentication method via some identity provider, see Authenticate users in Amazon Managed Grafana workspaces for more information.

> **Note**
>
> If you are trying to connect to your workspace programmatically, you must use API tokens. For more information, see Authenticate with service accounts.

To sign in to your Grafana workspace

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/, and sign in.
2. In the upper left corner of the page, choose the menu icon and then choose **All workspaces**.
3. Choose the name of the workspace you want to sign into.

4. In the workspace details page, choose the URL displayed under **Grafana workspace URL**.
5. Choosing the workspace URL takes you to the landing page for the Grafana workspace console. Choose **Sign in with AWS IAM Identity Center**, and enter the email address and password.

> **Note**
>
> The sign in button will have different text and requirements if you have set up authentication with an identity provider.

Users, teams, and permissions

Permissions in Amazon Managed Grafana are managed across the Amazon Managed Grafana console and directly within the workspace.

- **Users** – Users are authenticated in IAM Identity Center or an identity provider that you set up through SAML in the Amazon Managed Grafana console.
- **Role access** – You can give your users or groups access with the `User`, `Editor`, or `Admin` roles, to give them default permissions to your workspace, using the Amazon Managed Grafana console.
- **Groups, or Teams** – You can create groups of users to give access to in two ways. You can create groups in your identity provider (or IAM Identity Center). You can then give these groups access, just like a user, in the Amazon Managed Grafana console. Or you can create Teams in the Grafana workspace, and give them the role you want them to have.
- **Specific permissions** – If you want to override the permissions granted by roles for a specific dashboard, folder, or data source, you can remove the default permissions, and assign permissions to specific users or teams. This is done within the Grafana workspace.

This section describes how to perform permissions management within the Grafana workspace.

Topics

- Users
- User roles
- Managing teams
- Using permissions

Users

In Amazon Managed Grafana, you don't add users in the Grafana workspace. Instead, you use IAM Identity Center or and identity provider to authenticate users, and then grant users access to Amazon Managed Grafana workspaces from within the Amazon Managed Grafana console. For more information, see Manage user and group access to Amazon Managed Grafana workspaces.

User roles

In Amazon Managed Grafana, each user enabled to use the Amazon Managed Grafana workspace are assigned to one of three roles in the Amazon Managed Grafana console.

- **Admin role**— Users with the Admin role can do the following:
  - Can add, edit, and delete data sources.
  - Can add and edit users and teams.
  - Can add, edit, and delete folders containing dashboards.
  - Can do everything allowed by the Editor role.
- **Editor role**— Users with the Editor role can do the following:
  - Can view, add, and edit dashboards, panels, and alert rules in dashboards they have access to. This can be disabled on specific folders and dashboards.
  - Can create, update, or delete playlists.
  - Can access Explore.
  - Can add, edit, and delete notification channels.
  - Cannot add, edit, or delete data sources.
  - Can do everything allowed by the Viewer role.
- **Viewer role**— Users with the Viewer role can do the following:
  - Can view any dashboard they have access to. This can be disabled on specific folders and dashboards.
  - Cannot create, update, or delete playlists.
  - Cannot access Explore.
  - Cannot add, edit, and delete notification channels.
  - Cannot add, edit, or delete data sources.
  - Cannot add, edit, or delete dashboards or panels.
  - Cannot manage other users or teams.

User assignment and user access management from the Grafana workspace is not supported in Amazon Managed Grafana. How you manage user and group access depends on whether you use IAM Identity Center or SAML for authentication:

- If your workspace uses IAM Identity Center for authentication, you can use Amazon Managed Grafana console or APIs to assign roles. For more information, see Manage user and group access to Amazon Managed Grafana workspaces.
- If your workspace uses SAML for authentication, user roles are defined only by assertion attributes. For more information, see Assertion mapping.

Managing teams

Using *teams* enables you to grant permissions to a group of users at the same time. You can also set up team sync to automatically synchronize team membership between your Grafana workspace and your authorization provider.

Creating or removing a team

Create teams to manage users in groups.

To create a team

1. In the sidebar, choose the **Configuration**(gear) icon, and choose **Teams**.
2. Choose **New team**.
3. For **Name**, enter a name for the new team, and then choose **Create**.

To remove a team

1. In the sidebar, choose the **Configuration** (gear) icon, and choose **Teams**.
2. To the right of the team's name, choose **X**.
3. To confirm, choose **Delete**.

Adding or removing a user from a team

Use these steps to add users to teams or remove them from teams.

To add a user to a team

1. In the sidebar, choose the **Configuration** (gear) icon, and choose **Teams**.
2. Choose the team that you want to add the user to.
3. Choose **Add member**.
4. In the **Add team member** box, select the user to add to the team, and then choose **Add to team**.

To remove a user from a team

1. In the sidebar, choose the **Configuration** (gear) icon, and choose **Teams**.
2. Choose the team that you want to remove the user from.
3. To the right of the user's name, choose **X**.
4. To confirm, choose **Delete**.

Using team sync

With *team sync*, you can set up synchronization between your authorization provider's groups and the teams in Grafana. The authorization providers currently supported are IAM Identity Center and SAML.

To synchronize a Grafana team with an external group.

1. In the Grafana console, navigate to **Configuration**, **Teams**.
2. To synchronize with an IAM Identity Center group, enter the IAM Identity Center group ID. To synchronize with a group from a SAML based identity provider, enter the value of the attribute name entered in the **Assersion attribute groups** field in SAML configuration section on the Amazon Managed Grafana workspace configuration page.
3. Choose **Add group**.

Using permissions

What you can do in a Grafana workspace in Amazon Managed Grafana is defined by the *permissions* that are associated with your user.

Amazon Managed Grafana uses three types of permissions:

- Permissions granted as a Grafana admin
- Permissions associated with your membership on a team
- Permissions granted to a specific folder or dashboard

You can be granted permissions based on your admin status, dashboard or folder permissions assigned to your user, and data source permissions.

Dashboard and folder permissions overview

By using dashboard and folder permissions, you can remove the default role-based permissions for editors and viewers. You can then assign permissions to specific users and teams. For more information, see Dashboard and folder permissions.

Data source permissions overview

By default, a data source can be queried by any user. For example, a user with the `Viewer` role can issue any possible query to a data source, not just those queries that exist on dashboards to which they have access.

Using data source permissions, you can change the default permissions for data sources and restrict query permissions to specific **Users** and **Teams**. For more information, see Data source permissions.

Dashboard and folder permissions

For dashboards and dashboard folders, you can use the **Permissions** page to remove the default role based permissions for **Editors** and **Viewers**. On this page, you can add and assign permissions to specific **Users** and **Teams**.

Amazon Managed Grafana provides the following permission levels. The permissions vary based on the version of Grafana the workspace supports.

For workspaces that support version 8:

- `Admin`: Can edit and create dashboards and edit permissions. Can also add, edit, and delete folders.
- `Edit`: Can edit and create dashboards. **Can't** edit folder or dashboard permissions, or add, edit, or delete folders.
- `View`: Can only view existing dashboards and folders.

For workspaces that support version 9 and above:

- `Admin`: Can create, edit or delete a dashboard. Can add, edit, or delete folders, and create dashboards and subfolders in a folder. Administrators can also change dashboard and folder permissions.
- `Edit`: Can create, edit, or delete a dashboard. Can edit or delete a folder, and create dashboards and subfolders in a folder. An editor **can't** change folder or dashboard permissions.
- `View`: Can only view existing dashboards and folders.

Granting folder permissions

To grant folder permissions

1. In the sidebar, hover over the **Dashboards** (squares) icon, and then choose **Manage**.
2. Hover over a folder, and then choose **Go to folder**.
3. On the **Permissions** tab, choose **Add Permission**.
4. In the **Add Permission For** dialog box, choose **User**, **Team**, or one of the role options. If your workspace uses Grafana version 10 or newer, choose **User, Team, Service account, or Role**.
5. In the second box, select the user, team, service account, or role to which you want to add permissions. If your workspace is using Grafana version 9 or earlier, and you selected a role option in the previous step, then skip this step.
6. In the third box, select the permission that you want to add.
7. Choose **Save**.

Granting dashboard permissions

To grant dashboard permissions

1. In the top right corner of your dashboard, choose the cog icon to go to **Dashboard settings**.
2. On the **Permissions** tab, choose **Add Permission**.
3. In the **Add Permission For** dialog box, choose **User**, **Team**, or one of the role options. If your workspace uses Grafana version 10 or newer, choose **User, Team, Service account, or Role**.
4. In the second box, select the user, team, service account, or role to which you want to add permissions. If your workspace is using Grafana version 9 or earlier, and you selected a role option in the previous step, then skip this step.

5. In the third box, select the permission you that want to add.
6. Choose **Save**.

Restricting access

The highest permission always wins.

- You cannot override permissions for users with the `Admin` role. Admins always have access to everything.
- A more specific permission with a lower permission level does not have any effect if a more general rule exists with a higher permission level. You need to remove or lower the permission level of the more general rule.

How Amazon Managed Grafana resolves multiple permissions – examples

The following examples show how multiple permissions are resolved.

Example 1: `user1` has the `Editor` role

Permissions for a dashboard:

- Everyone with the `Editor` role can edit.
- `user1` can view.

Result: `user1` has Edit permission because the highest permission always wins.

Example 2: `user1` has the Viewer role and is a member of `team1`

Permissions for a dashboard:

- Everyone with the `Viewer` role can view.
- `user1` has the `Editor` role and can edit.
- `team1` has the `Admin` role.

Result: `user1` has Admin permission because the highest permission always wins.

Example 3: `user1` has multiple permissions at different levels

Permissions for a dashboard:

- `user1` has the `Admin` role (inherited from parent folder).
- `user1` has the `Editor` role and can edit.

Result: You cannot override to a lower permission. `user1` has Admin permission because the highest permission always wins.

Summary

- **View**: Can only view existing dashboards or folders.
- A more specific permission with a lower permission level will not have any effect if a more general rule exists with higher permission level.

Data source permissions

By default, data sources can be queried by any user. For example, a user with the `Viewer` role can issue any possible query to a data source, not just queries that exist on dashboards to which they have access.

You can use data source permissions to restrict access for users to query a data source. For each data source, there is a permission page where you can enable or restrict query permissions to specific **Users** and **Teams**.

Enabling data source permissions

When permissions are enabled for a data source, you restrict admin and query access for that data source to Admin users by default. You can selectively add access for specific users and teams.

To enable permissions for a data source

1. Navigate to **Configuration**, **Data Sources**. For workspaces that support Grafana version 10, Navigate to **Connections**, **Data Sources**.
2. Select the data source for which you want to enable permissions.
3. On the **Permissions** tab, choose **Enable**.

Warning

If you enable permissions for the default data source, users who are not listed in the permissions are unable to invoke queries. Panels that use the default data source will return the `Access denied to data source` error for those users.

Allowing users and teams to query a data source

After you enable permissions for a data source, only admins have access to that data source by default. You can assign query permissions to users or teams. The query permissions will allow access to query the data source.

To assign query permissions to users and teams

1. Navigate to **Configuration**, **Data Sources**. For workspaces that support Grafana version 10, Navigate to **Connections**, **Data Sources**.
2. Select the data source for which you want to assign query permissions.
3. On the **Permissions** tab, choose **Add Permission**.
4. Select **Team** or **User**. For workspaces that support Grafana version 10 or newer, you can also select **Service account** or **Role**.
5. Select the team, user, service account, or role that you want to grant query access to, and then choose **Save**.

Disabling data source permissions

If you have enabled permissions for a data source and want to return data source permissions to the default, follow these steps.

Note

All existing permissions created for the data source will be deleted.

To disable permissions for a data source

1. Navigate to **Configuration**, **Data Sources**. For workspaces that support Grafana version 10, Navigate to **Connections**, **Data Sources**.

2. Select the data source for which you want to disable permissions.
3. On the **Permissions** tab, choose **Disable Permissions**.

Create your first dashboard

Creating a dashboard

Follow these steps to create a dashboard in the Grafana console.

To create your first dashboard

1. Choose the + icon on the left panel, choose **Create Dashboard**, and then choose **Add new panel**.
2. In the **New Dashboard/Edit Panel** view, choose the **Query** tab.
3. Configure your query by selecting the data source that you'd like to query. For example, if you have **TestDB** added as a data source, this generates a sample dashboard called the Random Walk dashboard.

Introduction to time series

Imagine that you wanted to know how the temperature outside changes throughout the day. Once every hour, you'd check the thermometer and write down the time along with the current temperature. After a while, you'd have something like the following data.

Time	Value
09:00	24°C
10:00	26°C
11:00	27°C

Temperature data such as this are one example of a *time series*—a sequence of measurements, ordered in time. Every row in the table represents one individual measurement at a specific time.

Tables are useful when you want to identify individual measurements, but they can make it difficult to see the big picture. A more common visualization for time series is the *graph*, which instead

places each measurement along a time axis. Visual representations such as the graph make it easier to discover patterns and features of the data that otherwise would be difficult to see.

Other examples of time series are:

- CPU and memory usage
- Sensor data
- Stock market index

While each of these examples is a sequence of chronologically ordered measurements, they also share other attributes:

- New data are appended at the end, at regular intervals—for example, hourly at 09:00, 10:00, 11:00, and so on.
- Measurements are seldom updated after they are added. For example, yesterday's temperature doesn't change.

Time series are powerful. They help you understand the past by letting you analyze the state of the system at any point in time. Time series could tell you that the server crashed moments after the free disk space went down to zero.

Time series can also help you predict the future by uncovering trends in your data. For example, if the number of registered users has been increasing monthly by 4 percent for the past few months, you can predict how large your user base will be at the end of the year.

Some time series have patterns that repeat themselves over a known period. For example, the temperature is typically higher during the day, before it dips down at night. By identifying these periodic, or *seasonal*, time series, you can make confident predictions about the next period. If you know that the system load peaks every day around 18:00, you can add more machines right before.

Aggregating time series

Depending on what you're measuring, the data can vary greatly. What if you wanted to compare periods longer than the interval between measurements? If you'd measure the temperature once every hour, you'd end up with 24 data points per day. To compare the temperature in August over the years, you'd have to combine the 31 times 24 data points into one.

Combining a collection of measurements is called *aggregation*. There are several ways to aggregate time series data. Here are some common ones:

- **Average** returns the sum of all values divided by the total number of values.
- **Min** and **Max** return the smallest, and largest value in the collection.
- **Sum** returns the sum of all values in the collection.
- **Count** returns the number of values in the collection.

For example, by aggregating the data in a month, you can determine that August 2017 was, on average, warmer than the year before. If you wanted to see which month had the highest temperature, you'd compare the maximum temperature for each month.

How you aggregate your time series data is an important decision, and it depends on the story that you want to tell with your data. It's common to use different aggregations to visualize the same time series data in different ways.

Time series and monitoring

In the IT industry, time series data are often collected to monitor things such as infrastructure, hardware, or application events. Machine-generated time series data are typically collected with short intervals, so that you can react to any unexpected changes, moments after they occur. The data accumulate at a rapid pace, making it vital to have a way to store and query data efficiently. As a result, databases that are optimized for time series data have seen a rise in popularity in recent years.

Time series databases

A time series database (TSDB) is a database explicitly designed for time series data. While it's possible to use any regular database to store measurements, a TSDB comes with some useful optimizations.

Modern TSDBs take advantage of the fact that measurements are only ever appended, and rarely updated or removed. For example, the timestamps for each measurement change little over time, which results in redundant data being stored.

The following example shows a sequence of Unix timestamps.

```
1572524345, 1572524375, 1572524404, 1572524434, 1572524464
```

Looking at these timestamps, they all start with 1572524, leading to poor use of disk space. Instead, you could store each subsequent timestamp as the difference, or *delta*, from the first one, as shown in the following example.

```
1572524345, +30, +29, +30, +30
```

You could even take it a step further by calculating the deltas of these deltas, as shown in the following example.

```
1572524345, +30, -1, +1, +0
```

If measurements are taken at regular intervals, most of these delta-of-deltas will be 0. Because of optimizations like these, TSDBs use drastically less space than other databases.

Another feature of a TSDB is the ability to filter measurements by using *tags*. Each data point is labeled with a tag that adds context information, such as where the measurement was taken.

The following TSDBs are supported by Grafana:

- Graphite
- InfluxDB
- Prometheus

```
weather,location=us-midwest temperature=82 1465839830100400200
  |    -------------------- --------------  |
  |             |             |             |
  |             |             |             |
+-----------+--------+-+---------+-+---------+
|measurement|,tag_set| |field_set| |timestamp|
+-----------+--------+-+---------+-+---------+
```

Collecting time series data

Now that you have a place to store your time series, how do you actually gather the measurements? To collect time series data, you'd typically install a *collector* on the device, machine, or instance that you want to monitor. Some collectors are made with a specific database in mind, and some support different output destinations.

Here are some examples of collectors:

- collectd
- statsd

- Prometheus exporters
- Telegraf

A collector either *pushes* data to a database or lets the database *pull* the data from the collector. Each approach comes with its own set of pros and cons.

	Pros	Cons
Pu	Easier to replicate data to multiple destinations.	The TSDB has no control over how much data gets sent.
Pu	More control over how the amount of data ingested and data authenticity.	Firewalls, VPNs, or load balancers can make it hard to access the agents.

Because it's inefficient to write every measurement to the database, collectors pre-aggregate the data and write to the TSDB at regular intervals.

Time series dimensions

With time series data, the data is often a set of multiple time series. Many Grafana data sources support this type of data.

The common case is issuing a single query for a measurement with one or more additional properties as dimensions. For example, you might query a temperature measurement along with a location property. In this case, multiple series are returned back from that single query, and each series has unique location as a dimension.

To identify unique series within a set of time series, Grafana stores dimensions in *labels*.

Labels

Each time series in Grafana optionally has labels. Labels are a set of key-value pairs for identifying dimensions. Example labels are `{location=us}` or `{country=us,state=ma,city=boston}`. Within a set of time series, the combination of its name and labels identifies each series. For example, `temperature {country=us,state=ma,city=boston}`.

Different sources of time series data have dimensions stored natively, or common storage patterns that enable the data to be extracted into dimensions.

Usually, TSDBs natively support dimensionality. Prometheus stores dimensions in *labels*. In TSDBs such as Graphite or OpenTSDB, the term *tags* is used instead.

In table databases such SQL, these dimensions are generally the `GROUP BY` parameters of a query.

Multiple dimensions in table format

In SQL or SQL-like databases that return table responses, additional dimensions usually are columns in the query response table.

Single dimension

For example, consider a query like the following example.

```
SELECT BUCKET(StartTime, 1h), AVG(Temperature) AS Temp, Location FROM T
  GROUP BY BUCKET(StartTime, 1h), Location
  ORDER BY time asc
```

The query might return a table with three columns.

StartTime	Temp	Location
09:00	24	LGA
09:00	20	BOS
10:00	26	LGA
10:00	22	BOS

The table format is *long* formatted time series, also called *tall*. It has repeated timestamps, and repeated values in Location. In this case, two time series in the set would be identified as `Temp {Location=LGA}` and `Temp {Location=BOS}`.

Individual time series from the set are extracted by using the following dimensions:

- The time typed column `StartTime` as the time index of the time series
- The numeric typed column `Temp` as the series name
- The name and values of the string typed `Location` column to build the labels, such as Location=LGA

Multiple dimensions

If the query is updated to select and group by more than one string column (for example, `GROUP BY BUCKET(StartTime, 1h), Location, Sensor`), an additional dimension is added.

StartTime	Temp	Location	Sensor
09:00	24	LGA	A
09:00	24.1	LGA	B
09:00	20	BOS	A
09:00	20.2	BOS	B
10:00	26	LGA	A
10:00	26.1	LGA	B
10:00	22	BOS	A
10:00	22.2	BOS	B

In this case, the labels that represent the dimensions have two keys based on the two string typed columns, `Location` and `Sensor`. The data result in four series:

- `Temp {Location=LGA,Sensor=A}`
- `Temp {Location=LGA,Sensor=B}`
- `Temp {Location=BOS,Sensor=A}`
- `Temp {Location=BOS,Sensor=B}`

> **Note**
>
> **Note:** Multiple dimensions are not supported in a way that maps to multiple alerts in Grafana. Instead, they are treated as multiple conditions to a single alert.

Multiple values

In the case of SQL-like data sources, more than one numeric column can be selected, with or without additional string columns to be used as dimensions; for example, `AVG(Temperature) AS AvgTemp, MAX(Temperature) AS MaxTemp`. This, if combined with multiple dimensions, can result in numerous series. Selecting multiple values is currently designed to be used only with visualization.

Introduction to histograms and heatmaps

A histogram is a graphical representation of the distribution of numerical data. It groups values into buckets (sometimes also called bins). Then it counts how many values fall into each bucket.

Instead of graphing the actual values, histograms graph the buckets. Each bar represents a bucket, and the bar height represents the frequency (such as count) of values that fell into the interval of that bucket.

Histograms look only at *value distributions* over a specific time range. The problem with histograms is that you cannot see any trends or changes in the distribution over time. This is where heatmaps become useful.

Heatmaps

A *heatmap* is like a histogram over time, where each time slice represents its own histogram. Instead of using bar height as a representation of frequency, it uses cells, coloring a cell proportional to the number of values in the bucket.

Pre-bucketed data

A number of data sources support histogram over time, including the following:

- Amazon OpenSearch Service (by using a histogram bucket aggregation)
- Prometheus (with the histogram metric type and the *Format as* option set to **Heatmap**)

Generally, you can use any data source that returns series with names representing bucket bound or returns series sorted by the bound in ascending order.

Raw data vs. aggregated data

If you use the heatmap with regular time series data (not pre-bucketed), it's important to remember that your data are often already aggregated by your time series backend. Most time

series queries don't return raw sample data. Instead, they include a group by time interval or maxDataPoints limit coupled with an aggregation function (usually average).

It depends on the time range of your query. The important point is to know that the histogram bucketing that Grafana performs might be done on already aggregated and averaged data. For more accurate heatmaps, it's better to do the bucketing during metric collection or to store the data in OpenSearch, or in the other data source that supports doing histogram bucketing on the raw data.

If you remove or lower the group by time (or raise maxDataPoints) in your query to return more data points, your heatmap is more accurate. But this can also put a heavy load on your CPU and memory. If the number of data points becomes unreasonably large, it might cause stalls and crashes.

Extend your workspace with plugins

Grafana plugins add the ability to connect to new data sources, or add visualization or other functionality to the workspace. Broadly, plugins have three types:

- **Panel plugins** – Panel plugins add new visualization types that are available for use in your dashboards. These define the rendering of the data in the frontend.
- **Data source plugins** – Data source plugins communicate with external sources of data, and return the data in a format that Grafana can use.
- **App plugins** – Applications, also known as app plugins. These include bundle data sources and panels, and can provide a cohesive experience within your Grafana workspace.

> **Note**
>
> When Amazon determines that a plugin is often failing or hasn't been maintained, it might remove the plugin from the list of available plugins in the console.

For Amazon Managed Grafana workspaces that support version 9 or newer, you can enable plugin management. This allows workspace admins to install or uninstall plugins from the *plugin catalog*.

Find plugins with the plugin catalog

Your Amazon Managed Grafana workspace includes a page that shows all of your installed plugins and a list of all plugins that are available to install in your workspace. This page is the *plugin catalog*. In addition to the plugins that are installed by default, you can install up to 50 more plugins.

The available plugins fall broadly into the following categories:

- **AWS Data Sources** – This is an application plugin, provided by Amazon Managed Grafana, to easily discover AWS resources in your account. This is installed by default. For more information, see Use the AWS Data Sources plugin to find AWS data.
- **Core plugins** – These plugins are provided by default in Grafana. They include popular data sources and panel visualizations. They are tagged as **Core** in the plugin catalog. These are installed by default and cannot be removed.
- **Enterprise plugins** – These plugins are available to Grafana workspaces that have an enterprise license. These are not installed by default. They are tagged as **Enterprise** in the plugin catalog. They can only be installed if you have a valid enterprise license. For details about how to upgrade a workspace to an Enterprise license, see Managing your access to Amazon Managed Grafana Enterprise plugins.
- **Community plugins** – These plugins are provided for Grafana workspaces from various sources, including Grafana Labs, AWS, and others. In Grafana workspaces that support version 9 or newer, these are not installed by default (earlier workspaces have some of these installed automatically). These are typically open source plugins. You can install or remove these plugins.

> **Note**
>
> Using community plugins is at your discretion. As part of the shared responsibility model between you and AWS, you are expected to understand what you are installing into your workspace for these third party plugins. You are also responsible for the plugins meeting your security needs.

Plugin support

Plugins come from a wide variety of sources, and support varies for them.

- **AWS Data Sources plugin** – This plugin is provided by and supported by AWS.

- **Enterprise plugins** – The Enterprise plugins are supported by both AWS and Grafana Labs—you can submit issues through either support team.
- **Core plugins** – The core plugins and other plugins provided by AWS or Grafana Labs are supported in Amazon Managed Grafana by AWS. You can submit an issue in GitHub for bug-fixes or enhancements, or create a ticket with AWS or Grafana Labs.
- **Community plugins** – Community plugins not created by AWS or Grafana Labs are typically supported through GitHub issues or other forums. Support information in those cases is included in the details for the plugin in the plugin catalog.

You can also submit issues for plugins through the GitHub forums for Amazon Managed Grafana or Grafana.

Plugin versions

Most plugins are updated on a regular cadence. The plugin catalog in a Amazon Managed Grafana workspace shows the most recent versions of a plugin, and you choose which version to install. When a plugin has an outdated version with a known security issue, the outdated version is removed from availability.

You can also update plugins that are already installed.

> **Note**
>
> Sometimes, a new version of a plugin is made available that fixes a security issue in an installed plugin. For severe issues, Amazon Managed Grafana might automatically update the plugin in your workspaces to the version with the fix.

Manage plugins with the plugin catalog

You manage the plugins for your Amazon Managed Grafana workspace from the plugin catalog. You can only install plugins that are listed in the plugin catalog within your workspace.

The following describes the prerequisites for using the plugin catalog, and how to find the plugin catalog.

Prerequisites

- You must have an Amazon Managed Grafana workspace that supports version 9, and have an account that can log into that workspace.
- The workspace must have plugin management enabled.
- Your user account must be an admin for your Amazon Managed Grafana workspace.
- To install and use Enterprise plugins, you must first upgrade to an Enterprise license.

To view the plugin catalog

1. Sign in to your Amazon Managed Grafana workspace.
2. From the left menu, choose **Administration**, then **Plugins**. This opens the plugin catalog.
3. By default, the plugin catalog lists the installed plugins. To view all available plugins, choose **All** under the **State** filter at the top of the catalog. Installed plugins include a tag that says **Installed**.

Install or remove a plugin

> **Note**
>
> You must meet the prerequisites from the preceding section, or you will not have permission to modify plugins.

To install or remove a Grafana plugin

1. Go to the plugin catalog.
2. By default, the plugin catalog lists the installed plugins only. To view all available plugins, choose **All** under the **State** filter at the top of the catalog. Installed plugins include a tag that says **Installed**.
3. Select the plugin to install or uninstall. For example, if you want to remove the *Datadog* data source, select the **Datadog** plugin.
4. On the plugin details page, choose the uninstall or install option.
5. After a plugin is installed, it can take up to a few minutes before the change is synced across all parts of the workspace. It is good to wait a few minutes before using the new plugin.

> **Note**
>
> You can have 50 plugins installed in a workspace (beyond the default Core plugins).

Update a plugin

To update an existing Grafana plugin

1. Sign in to your Amazon Managed Grafana workspace.
2. From the left menu, choose **Administration**, then **Plugins**. This opens the plugin catalog, listing only the installed plugins.
3. Select the plugin to update.
4. On the plugin details page, check to see if there is an update available. If so, choose the option to update the plugin and choose the version to update to.

> **Note**
>
> If you see a note that you do not have permission to modify the plugin, confirm that plugin management is enabled for your workspace. You must also be an admin for the Amazon Managed Grafana workspace.

Use the AWS Data Sources plugin to find AWS data

AWS provides an application plugin to make it easier to discover and use AWS resources as data sources in your Amazon Managed Grafana workspace. The *AWS Data Sources* plugin is installed by default in a new workspace.

The AWS Data Sources plugin requires permissions to access your resources for discovery. For more information, see Required permissions.

Open the AWS Data Sources plugin

To open the AWS Data Sources plugin

1. Sign into your Amazon Managed Grafana workspace.
2. From the **menu** in the top left, choose **Apps**, then **AWS Data Sources**.

The AWS Data Sources plugin interface appears listing AWS services that you can search for resources.

Discover resources

To discover resources from your AWS account

1. Open the AWS Data Sources plugin.
2. From the list of AWS services, select the one that you want to find resources to use as a data source. For example, select **Amazon Managed Service for Prometheus**. This will take you to the **Data sources** tab, with the **Service** selected for you.
3. Choose the AWS Region where you want to find resources. For example, select `US East (N. Virginia)`.

   > **Note**
   >
   > To find resources, the plugin must have the appropriate permissions to access that service in that region.

4. Some services can have multiple resources in a region. If there are multiple resources in the region, the AWS Data Sources plugin provides a list to choose from.

   From the list of resources (in this case, Amazon Managed Service for Prometheus), select the resource that you want to use as a data source. For example, selecting a Amazon Managed Service for Prometheus workspace will setup that resource as a data source The data source is ready to use in your dashboards or monitoring with Amazon Managed Grafana.
5. The resources in that service and Region that you have provisioned appear at the bottom of the page.

   (Optional) You can choose **Go to settings** to view and edit the settings for that data source.

> **Note**
>
> The AWS Data Sources plugin depends on the individual data source plugins being installed in your workspace. For example, if you want to use the AWS X-Ray functionality, you must have the X-Ray data source plugin installed from the plugin catalog.

Versions and updating the plugin

The AWS Data Sources plugin is regularly updated. The version installed with a new workspace is not typically the latest version. Newer versions can have more functionality than the one installed in your workspace. For example, a newer version might support additional AWS services as data sources.

To see the changes in each version of the AWS Data Sources plugin, you can view the Changelog.

To update to a newer version of the plugin, follow the standard instructions for Update a plugin.

> **Note**
>
> If you update to a newer version of the AWS Data Sources plugin, you need to provide additional permissions for new data sources that are not managed by Amazon Managed Grafana.

Required permissions

The AWS Data Sources plugin requires permission to access your AWS resources. The easiest way to do that is to allow Amazon Managed Grafana to manage the permissions for you. To learn how to set up service-managed permissions for data sources, see Manage permissions for data sources and notification channels. Amazon Managed Grafana can manage the permissions for the AWS resources that are included in the AWS Data Sources plugin by default.

If you update the AWS Data Sources plugin to a newer version than is included by default in your workspace, it could add support for AWS resources whose permissions are not managed by Amazon Managed Grafana automatically. In these cases, you must add the permissions yourself. For example, AWS IoT TwinMaker was added to a recent version of the plugin, (version 1.9.0), and permissions for these are not managed by Amazon Managed Grafana.

To learn more about the permissions for any specific data source, see the details for that data source provided in the Connect to data sources section. For example, the Connect to an AWS IoT TwinMaker data source section includes details about giving Amazon Managed Grafana permissions to access AWS IoT TwinMaker.

Connect to data sources

Amazon Managed Grafana supports many different *data sources*. Data sources are storage backends that you can query in Grafana to do things like building dashboards. Each data source has a specific query editor that is customized for the features and capabilities that the particular data source exposes.

The query language and capabilities of each data source are different. You can combine data from multiple data sources onto a single dashboard.

Every AWS account that uses Amazon Managed Grafana has access to create or configure many data sources. Some data sources require you to install the respective plugin for that data source. If you upgrade your workspace to Amazon Managed Grafana Enterprise plugins, you may also need to install the plugins for the Enterprise data sources. The following sections describe details of many of the data sources available, but the Grafana community sometimes adds new data sources that may be available in the plugin catalog within your workspace.

> **Note**
>
> To help you discover AWS resources in your account and setup data sources to query them, Amazon Managed Grafana provides the Use the AWS Data Sources plugin to find AWS data.

Special data sources

Amazon Managed Grafana includes three special data sources:

- **Grafana** (called *TestDB* in earlier versions of Grafana – Use this built-in data source to generate random walk data, or list files. This is useful for testing visualizations and running experiments.
- **Mixed** – Use this to query multiple data sources in the same panel. When you use this data source, you can specify a data source for every new query that you add. The first query uses the data source that you specified before selecting **Mixed**.

  You cannot change an existing query to use a mixed data source.
- **Dashboard** – Use this to use a result set from another panel in the same dashboard.

Important

Amazon Managed Grafana has a data source timeout limit, which might override any timeout limit configured on data sources. The lower of the two limits supersedes the other. To learn about the Amazon Managed Grafana data source timeout limit, see Amazon Managed Grafana service quotas in the *AWS General Reference*.

Topics

- How Amazon Managed Grafana works with AWS Organizations for AWS data source access
- Connect to built-in data sources
- Connect to Enterprise data sources

How Amazon Managed Grafana works with AWS Organizations for AWS data source access

With AWS Organizations, you can centrally manage data source configuration and permission settings for multiple AWS accounts. In an AWS account with an Amazon Managed Grafana workspace, you can specify other organizational units to make their AWS data sources available for viewing in the primary account.

For example, you can use one account in the organization as an Amazon Managed Grafana *management account*, and give this account access to data sources in other accounts in the organization. In the management account, list all the organizational units that have AWS data sources that you want to access with the management account. This automatically creates the roles and permissions policies that you need for setting up these data sources, which you can see in the Grafana console in the Amazon Managed Grafana workspace.

For more information about Organizations, see What is AWS Organizations.

Amazon Managed Grafana uses CloudFormation StackSets to automatically create the AWS Identity and Access Management (IAM) roles necessary for Amazon Managed Grafana to connect to data sources across your AWS organization. Before Amazon Managed Grafana can manage your IAM policies to access data sources across your organization, you must enable AWS CloudFormation StackSets in the management account of your organization. Amazon Managed Grafana automatically enables this the first time that it's needed.

Deployment scenarios for integration with AWS IAM Identity Center and Organizations

If you are using Amazon Managed Grafana with both AWS IAM Identity Center and Organizations, we recommend that you create an Amazon Managed Grafana workspace in your organization using one of the following three scenarios. For each scenario, you need to be signed in to an account with sufficient permissions. For more information, see Sample policies for Amazon Managed Grafana.

Standalone account

A standalone account is an AWS account that is not a member of an organization in Organizations. This is a likely scenario if you are trying out AWS for the first time.

In this scenario, Amazon Managed Grafana automatically enables AWS IAM Identity Center and Organizations when you are signed in to an account that has the **AWSGrafanaAccountAdministrator**, **AWSSSOMemberAccountAdministrator**, and **AWSSSODirectoryAdministrator** policies. For more information, see Create and manage Amazon Managed Grafana workspaces and users in a single standalone account using IAM Identity Center.

Member account of an existing organization in which IAM Identity Center is already configured

To create a workspace in a member account, you must be signed in to an account that has the **AWSGrafanaAccountAdministrator**, **AWSSSOMemberAccountAdministrator**, and **AWSSSODirectoryAdministrator** policies. For more information, see Grafana administrator in a member account using IAM Identity Center.

If you create a workspace in a member account and you want that workspace to access resources from other AWS accounts in your organization, you must use customer managed permissions in the workspace. For more information, see Customer-managed permissions.

To use service-managed permissions to allow a workspace to access resources from other AWS accounts in the organization, you would have to create the workspace in the management account of the organization. However, it is not a best practice to create Amazon Managed Grafana workspaces or other resources in the management account of an organization. For more information about Organizations best practices, see Best practices for the management account.

> **Note**
>
> If you enabled AWS IAM Identity Center in the management account before November 25, 2019, you must also enable IAM Identity Center-integrated applications in the

management account. Optionally, you can also enable IAM Identity Center-integrated applications in the member accounts after you do so in the management account. To enable these applications, choose **Enable access** in the IAM Identity Center **Settings** page in the IAM Identity Center-integrated applications section. For more information, see IAM Identity Center-integrated application enablement.

Member account of an existing organization in which IAM Identity Center isn't deployed yet

In this scenario, sign in as the organization administrator first, and enable IAM Identity Center in the organization. Then, create the Amazon Managed Grafana workspace in a member account in the organization.

If you are not an organization administrator, you must contact an administrator for Organizations and request that they enable IAM Identity Center. After IAM Identity Center is enabled, you can then create the workspace in a member account.

If you create a workspace in a member account and you want that workspace to access resources from other AWS accounts in your organization, you must use customer managed permissions in the workspace. For more information, see Customer-managed permissions.

To create a workspace in a member account, you must be signed in to an account that has the **AWSGrafanaAccountAdministrator**, **AWSSSOMemberAccountAdministrator**, and **AWSSSODirectoryAdministrator** policies. For more information, see Grafana administrator in a member account using IAM Identity Center.

Connect to built-in data sources

The following data sources are supported in every Amazon Managed Grafana workspace.

For workspaces that support version 9 and newer, some of these data sources might not be installed by default. The core data sources are available on all installations, but the data sources that are not part of the core set might need the correct Grafana plugin to be installed. You can install plugins for additional data sources that aren't listed here. For more information about managing plugins, see Extend your workspace with plugins.

Topics

- Connect to an Alertmanager data source
- Connect to an Amazon CloudWatch data source

- Connect to an Amazon OpenSearch Service data source
- Connect to an AWS IoT SiteWise data source
- Connect to an AWS IoT TwinMaker data source
- Connect to Amazon Managed Service for Prometheus and open-source Prometheus data sources
- Connect to an Amazon Managed Service for Prometheus data source
- Connect to an Amazon Timestream data source
- Connect to an Amazon Athena data source
- Connect to an Amazon Redshift data source
- Connect to an AWS X-Ray data source
- Connect to an Azure Monitor data source
- Connect to a Graphite data source
- Connect to a Google Cloud Monitoring data source
- Connect to an InfluxDB data source
- Connect to a Jaeger data source
- Connect to a Loki data source
- Connect to a Microsoft SQL Server data source
- Connect to a MySQL data source
- Connect to an OpenSearch data source
- Connect to an OpenTSDB data source
- Connect to a PostgreSQL data source
- Connect to a Tempo data source
- Configure a TestData data source for testing
- Connect to a Zipkin data source

Connect to an Alertmanager data source

Grafana includes built-in support for Prometheus Alertmanager. Once Grafana alerting is configured, you can use the Grafana alerting UI to manage silences, contact points as well as notification policies. A drop-down option in these pages allows you to switch between Grafana and any configured Alertmanager data sources.

Alertmanager implementations

Prometheus, Cortex, and Grafana Mimir implementations of Alertmanager are supported. You can specify implementation in the data source settings page. Prometheus contact points and notification policies are read-only in the Grafana alerting UI, as it does not support updating configuration via HTTP API.

Configuring an Alertmanager data source

You can configure an Alertmanager data source to use with Grafana alerting.

Prerequisites

To configure Alertmanager, you must have the following pre-requisite complete:

- A Prometheus instance, with ingested metrics, and at least one alert or recording rule configured. You will need the URL for your workspace.
- Permissions defined for Amazon Managed Grafana to have read access to your alerts, alert groups, silences, and contact points from your Alertmanager implementation.

To configure an Alertmanager data source

1. From your Grafana console, in the Grafana menu, choose the **Data source** page under **Configuration**.
2. Choose **Add data source**, and select **Alertmanager** from the list of data source types.
3. Provide the following information for your new data source.
   - For **Name**, provide a name of your choosing for your data source.
   - For **Implementation**, choose your Alertmanager implementation – either **Prometheus**, **Mimir**, or **Cortex**.
   - Under **HTTP**, for **URL**, provide the Alertmanager URL. For Prometheus, this is the workspace URL, with `alertmanager` appended. For example, `https://myprometheus/workspaces/ws-example-1234-5678-abcd-xyz00000001/alertmanager`.
   - Under **Auth**, configure the authentication details needed to access your Alertmanager implementation.
4. Choose **Save & test** to finish your data source setup.

   If your data source is set up correctly, you will see a message saying **Health check passed**.

Connect to an Amazon CloudWatch data source

With Amazon Managed Grafana, you can add Amazon CloudWatch as a data source by using the AWS data source configuration option in the Grafana workspace console. This feature simplifies adding CloudWatch as a data source by discovering your existing CloudWatch accounts and manages the configuration of the authentication credentials that are required to access CloudWatch. You can use this method to set up authentication and add CloudWatch as a data source. Alternatively, you can manually set up the data source and the necessary authentication credentials using the same method that you would on a self-managed Grafana server.

Topics

- Use AWS data source configuration to add CloudWatch as a data source
- Manually add CloudWatch as a data source
- Using the query editor
- Curated dashboards
- Templated queries
- Using ec2_instance_attribute examples
- Using JSON format template variables
- Pricing
- Service quotas
- Cross-account observability

Use AWS data source configuration to add CloudWatch as a data source

To use AWS data source configuration, first you use the Amazon Managed Grafana console to enable service-managed IAM roles that grant the workspace the IAM policies necessary to read the CloudWatch resources in your account or in your entire organizational unit. Then you use the Amazon Managed Grafana workspace console to add CloudWatch as a data source.

To use AWS data source configuration to add CloudWatch as a data source

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. On the navigation pane, choose the menu icon and then choose **All workspaces**.
3. Choose the name of the workspace.

4. If you didn't choose to use service-managed permissions for this workspace when you created it, then change from using customer-managed permissions to use service-managed permissions to ensure that the proper IAM roles and policies are enabled for using the AWS data source configuration option in the Grafana workspace console. To do so, choose the edit icon by **IAM role** and then choose **Service managed**, **Save changes**. For more information, see Amazon Managed Grafana permissions and policies for AWS data sources.
5. Choose the **Data sources** tab.
6. Select the check box for **Amazon CloudWatch** and then choose **Actions**, **Enable service-managed policy**.
7. Choose the **Data sources** tab again.
8. Choose **Configure in Grafana** in the **Amazon CloudWatch** row.
9. Sign in to the Grafana workspace console using IAM Identity Center, if necessary.
10. On the navigation bar in the Grafana workspace console, choose the AWS icon and then choose **AWS services**, **CloudWatch**.
11. Select the default Region that you want the CloudWatch data source to query from.
12. Select the accounts that you want, and then choose **Add data source**.

Manually add CloudWatch as a data source

To manually add the CloudWatch data source

1. In the Grafana console side menu, hover over the **Configuration** (gear) icon, and then choose **Data Sources**.
2. Choose **Add data source**.
3. Choose the **CloudWatch** data source. If necessary, you can start typing **`CloudWatch`** in the search box to help you find it.

CloudWatch settings

The following CloudWatch settings apply.

Name	Description
`Name`	The data source name. This is how you see the data source in panels and queries.

Name	Description
`Default`	Designates the data source to be pre-selected for new panels.
`Default Region`	Set the Region in the query editor. Can be changed on per-query basis.
`Namespaces of Custom Metrics`	Specifies the CloudWatch namespaces of custom metrics. Can include multiple namespaces, separated by commas.
`Auth Provider`	Specifies the provider to get credentials.
`Assume Role Arn`	Specifies the Amazon Resource Name (ARN) of the role to assume.
`External ID`	(Optional) Specifies the external ID. Use if you are assuming a role in another AWS account that has been created with an external ID.
`Timeout`	Configure the timeout specifically for CloudWatch Logs queries.
`X-Ray trace links`	To automatically add links in your logs when the log contains the `@xrayTraceId` field, link an X-Ray data source in the **X-Ray trace link** section of the data source configuration. You must already have an X-Ray data source configured.

Authentication

To enable authentication between Amazon Managed Grafana and CloudWatch, you can use the Amazon Managed Grafana console to quickly create the policies and permissions that are needed. Alternatively, you can manually set up authentication using some of the same methods that you would on a self-managed Grafana server.

To use Amazon Managed Grafana data source configuration to quickly set up the policies, follow the steps in Use AWS data source configuration to add CloudWatch as a data source.

To set up the permissions manually, use one of the methods in the following section.

AWS credentials

There are three different authentication methods available.

- **AWS SDK Default**— Uses the permissions defined in the role that is attached to your workspace. For more information, see Customer-managed permissions.
- **Access and secret key**— Corresponds to the AWS SDK for Go `StaticProvider`. Uses the given access key ID and secret key to authenticate. This method doesn't have any fallbacks, and will fail if the provided key pair doesn't work.

IAM roles

Currently, all access to CloudWatch is done server-side by the Grafana backend using the official AWS SDK. If you choose the *AWS SDK Default* authentication method, and your Grafana server is running on AWS, you can use IAM roles to handle authentication automatically.

For more information, see IAM roles.

IAM policies

Grafana needs permissions granted through IAM to be able to read CloudWatch metrics and EC2 tags, instances, and Regions. You can attach these permissions to IAM roles and use the built-in Grafana support for assuming roles.

The following code example shows a minimal policy.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowReadingMetricsFromCloudWatch",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarmsForMetric",
                "cloudwatch:DescribeAlarmHistory",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:GetMetricData",
```

```
                "cloudwatch:GetInsightRuleReport"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadingLogsFromCloudWatch",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:GetLogGroupFields",
                "logs:StartQuery",
                "logs:StopQuery",
                "logs:GetQueryResults",
                "logs:GetLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadingTagsInstancesRegionsFromEC2",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeTags",
                "ec2:DescribeInstances",
                "ec2:DescribeRegions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadingResourcesForTags",
            "Effect": "Allow",
            "Action": "tag:GetResources",
            "Resource": "*"
        },
        {
            "Sid": "AllowReadingAcrossAccounts",
            "Effect": "Allow",
            "Action": [
                "oam:ListSinks",
                "oam:ListAttachedLinks"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

Assuming a role

The `Assume Role ARN` field allows you to specify which IAM role to assume, if any. If you keep this blank, the provided credentials are used directly and the associated role or user should have the required permissions. If this field is not blank, the provided credentials are used to perform an `sts:AssumeRole` call.

Using the query editor

The CloudWatch data source in Amazon Managed Grafana provides a powerful query editor that allows you to retrieve and analyze metrics and logs from various AWS services that send data to CloudWatch. The query editor supports two distinct query modes: Metric Search and CloudWatch Logs.

The query editor mode for metrics uses the CloudWatch API to find metrics uploaded to CloudWatch. The mode for logs uses the CloudWatch Logs APIs to find log records. Each mode has its own specialized query editor. You select which API you want to query with by using the query mode switch at the top of the editor.

Topics

- Using the metric query editor
- Using the Amazon CloudWatch Logs query editor

Using the metric query editor

The metric query editor allows you to build two types of queries - **Metric Search** and **Metric Query**. The **Metric Query** option queries data using CloudWatch Metrics Insights.

Common query editor fields

There are three fields that are common to both **Metric Search** and **Metric Query** modes.

Common fields

Id

The `GetMetricData` API requires that all queries have a unique ID. Use this field to specify an ID of choice. The ID can include numbers, letters, and underscores, and must start with a

lowercase letter. If no ID is specified, Amazon Managed Grafana generates an ID using the following pattern: `query[refId of the current query row]`. For example, `queryA` represents the first query row in the panel editor.

Period

A period is the length of time associated with a specific CloudWatch statistic. Periods are defined in numbers of seconds. Valid values include 1, 5, 10, 30, or any multiple of 60. If you keep the period field blank or set to `auto`, then it calculates automatically based on the time range and the CloudWatch retention policy. The formula used is `time range in seconds / 2000`, and then it moves on to the next higher value in an array of predefined periods [60, 300, 900, 3600, 21600, 86400] after removing periods based on retention. To see which period Amazon Managed Grafana is using, choose **Show Query Preview** in the query editor.

Alias

The following alias patterns apply.

Alias pattern	Description	Example result
`{{region}}`	Returns the Region.	`us-east-1`
`{{period}}`	Returns the period.	`3000`
`{{metric}}`	Returns the metric.	`CPUUtilization`
`{{label}}`	Returns the label returned by the API operation (**Metric Search** only).	`i-01343`
`{{namespace}}`	Returns the namespace (**Metric Search** only).	`AWS/EC2`
`{{stat}}`	Returns the statistic (**Metric Search** only).	`Average`
`{{[dimension name]}}`	Returns the dimension name (**Metric Search** only).	`i-01343`

Using the metric search option

To create a valid query in **Metric Search**, you must specify the namespace, metric name and at least one statistic. If **Match Exact** is turned on, you must also specify all dimensions of the metric that you're querying. The metrics schema must match exactly. For more information, see CloudWatch search expression syntax .

If **Match Exact** is turned off, you can specify any number of dimensions by which you want to filter. Up to 100 metrics matching your filter criteria are returned.

Dynamic queries using dimension wildcard characters

You can monitor a dynamic list of metrics by using the asterisk (*) wildcard character for one or more dimension values.

This helps you monitor metrics for AWS resources, such as EC2 instances or containers. For example, when new instances get created as part of an auto scaling event, they will automatically appear in the graph without you having to track the new instance IDs. This capability is currently limited to retrieving up to 100 metrics. You can choose **Show Query Preview** to see the search expression that is automatically built to support wildcard characters.

By default, the search expression is defined in such a way that the queried metrics must match the defined dimension names exactly. This means that in the example only metrics with exactly one dimension with name `InstanceId` are returned.

To include metrics that have other dimensions defined, you can turn off **Match Exact**. Turning off **Match Exact** also creates a search expression even if you don't use wildcard characters. Grafana searches for any metric that matches at least the namespace, metric name, and all defined dimensions.

Multi-value template variables

When defining dimension values based on multi-valued template variables, a search expression is used to query for the matching metrics. This enables the use of multiple template variables in one query. You can also use template variables for queries that have the **Match Exact** option turned off.

Search expressions are currently limited to 1024 characters, so your query might fail if you have a long list of values. If you want to query all metrics that have any value for a certain dimension name, we recommend using the asterisk (*) wildcard character instead of the `All` option.

The use of multi-valued template variables is only supported for dimension values. Using multi-valued template variables for `Region`, `Namespace`, or `Metric Name` is not supported.

Metric math expressions

You can create new time series metrics by operating on top of CloudWatch metrics using mathematical functions. Arithmetic operators, unary subtraction, and other functions are supported and can be applied to CloudWatch metrics. For more information about CloudWatch metric math functions, see Using metric math.

As an example, to apply arithmetic operations on a metric, give an ID (a unique string) to the raw metric. You can then use this ID and apply arithmetic operations to it in the `Expression` field of the new metric.

If you use the `Expression` field to reference another query, such as `queryA * 2`, you cannot create an alert rule based on that query.

Period

A period is the length of time associated with a specific Amazon CloudWatch statistic. Periods are defined in numbers of seconds. Valid values include 1, 5, 10, 30, or any multiple of 60.

If you keep the period field blank or set to **auto**, then it calculates automatically based on the time range. The formula used is `time range in seconds / 2000`, and then it moves on to the next higher value in an array of predefined periods `[60, 300, 900, 3600, 21600, 86400]`. To see which period Amazon Managed Grafana is using, choose **Show Query Preview** in the query editor.

Deep linking from Grafana panels to the CloudWatch console

Choosing a time series in the panel shows a context menu with a link to **View in CloudWatch console**. Choosing that link opens a new tab that takes you to the CloudWatch console and displays all the metrics for that query. If you're not currently signed in to the CloudWatch console, the link forwards you to the sign-in page. The provided link is valid for any AWS account but only displays the correct metrics if you're signed in to the AWS account that corresponds to the selected data source in Grafana.

This feature is not available for metrics that are based on metric math expressions.

Using the metric query option to query CloudWatch Metrics Insights data

> **Note**
>
> Amazon CloudWatch Metrics Insights is in preview. CloudWatch Metrics Insights features are open to all AWS accounts. Features might be subject to change.

You can query CloudWatch Metrics Insights data by choosing the `metric query` mode in the **Metric query editor**.

CloudWatch Metrics Insights is a powerful high-performance SQL query engine that you can use to query your metrics at scale. It is a fast, flexible, SQL-based query engine that you can use to identify trends and patterns within all of your CloudWatch metrics in real time. It uses a dialect of SQL. For more information about the Metrics Insights query syntax, see Query syntax and keywords.

Query syntax and keywords

CloudWatch Metrics Insights uses a dialect of SQL. The following example shows the query syntax.

```
SELECT FUNCTION(metricName)
FROM namespace | [ SCHEMA(namespace[, labelKey [, ...] ]) ]
     [ WHERE labelKey OPERATOR labelValue [AND|OR|([...])*] [, ...] ]
[ GROUP BY labelKey [, ...]]
[ ORDER BY FUNCTION() [DESC | ASC] ]
[ LIMIT number]
```

Keywords are not case-sensitive, but the identifiers are case-sensitive. Identifiers include the names of metrics, namespaces, and dimensions.

The following table provides the query keywords and their descriptions.

Keywor	Description
FUNCTI	Required. Specifies the aggregate function to use, and also specifies the name of the metric to query. Valid values are AVG, COUNT, MAX, MIN, and SUM.
Metric e	Required. For example, CPUUtilization .
FROM	Required. Specifies the source of the metric. You can specify either the metric namespace that contains the metric to query, or a SCHEMA table function. Some namespace examples are AWS/EC2 and AWS/Lambda .
SCHEMA	(Optional) Filters the query results to show only the metrics that are an exact match, or the metrics that do not match.

Keywor	Description
WHERE	(Optional) Filters the results to show only the metrics that match your specified expression. For example, `WHERE InstanceType != 'c3.4xlarge'` .
GROUP BY	(Optional) Groups the query results into multiple time series. For example, `GROUP BY ServiceName` .
ORDER BY	(Optional) Specifies the order of time series to return. Options are ASC and DESC.
LIMIT	(Optional) Limits the number of time series to return.

The following are some examples:

- ```
  SELECT AVG(CPUUtilization) FROM "AWS/EC2"
  ```

  Matches all `CPUUtilization` metrics in the `AWS/EC2` namespace, ignoring their dimensions, and returns a single aggregated time series.

- ```
  SELECT AVG(CPUUtilization) FROM SCHEMA("AWS/EC2")
  ```

  Matches only the `CPUUtilization` metrics in the `AWS/EC2` namespace that do not have any dimensions defined.

- ```
  SELECT AVG(CPUUtilization) FROM SCHEMA("AWS/EC2", InstanceId)
  ```

  Matches only the `CPUUtilization`metrics that were reported to CloudWatch with exactly one dimension, `InstanceId`.

- ```
  SELECT SUM(RequestCount) FROM SCHEMA("AWS/ApplicationELB", LoadBalancer,
  AvailabilityZone)
  ```

  Mtches only the `RequestCount` metrics that were reported to CloudWatch from `AWS/ApplicationELB` with exactly two dimensions, `LoadBalancer`and `AvailabilityZone`.

Label values must be enclosed by single quotation marks.

Escape characters

In a query, label values must always be surrounded with single quotation marks. For example, `SELECT MAX(CPUUtilization) FROM "AWS/EC2" WHERE AutoScalingGroupName = 'my-production-fleet'`.

Metric namespaces, metric names, and label keys that contain characters other than letters, numbers, and underscores (_) must be surrounded by double quotation marks. For example, `SELECT MAX("My.Metric")`. If one of these contains a double quotation mark itself (such as `Bytes"Input"`), you must escape that double quotation mark with backslashes, as in `SELECT AVG("Bytes\"Input\"")`. If a metric namespace, metric name, or label key, contains a word that is a reserved keyword in Metrics Insights, these must also be enclosed in double quotation marks. For example, if you have a metric named `LIMIT`, you would use `SELECT AVG("LIMIT")`. It is also valid to enclose any namespace, metric name, or label in double quotation marks even if it does not include a reserved keyword.

Builder mode and code mode

You can create a query in `Builder` mode or `Code` mode.

To create a query in `Builder` mode

1. Browse and select a metric namespace, metric name, filter, group, and order options using information from the preceding table.
2. For each of these options, choose from the list of possible options.

To create a query in `Code` mode

1. Write your query in the code editor.
2. To run the query, choose **Run query** in the code editor.

To create a query in the `builder` mode:

- Browse and select a metric namespace, metric name, filter, group, and order options using information from the table above.
- For each of these options, choose from the list of possible options.

Grafana automatically constructs a SQL query based on your selections.

To create a query in the code mode:

- Write your query in the code editor.
- To run the query, choose the **Run query** on the code editor.

The code editor has a built-in autocomplete feature that gives suggestions for keywords, aggregations, namespaces, metrics, labels, and label values. The suggestions are shown when you enter a space, comma, or dollar sign. You can also use the keyboard combination `CTRL+Space`.

Code editor can autocomplete the query. However, use of template variables in the code editor might interfere with the autocompletion.

CloudWatch Metrics Insights examples

> **Note**
>
> CloudWatch Metrics Insights is in open preview. The preview is open to all AWS accounts and you do not need to request access. Features might be added or changed before announcing General Availability.

This section contains examples of useful CloudWatch Metrics Insights queries that you can copy and use directly or copy and modify in query editor. Some of these examples are already available in the console, and you can access them by choosing **Add query** in the **Metrics** view.

EC2 examples

View CPU utilization per instance metrics

```
SELECT AVG(CPUUtilization)
FROM "AWS/EC2"
GROUP BY InstanceId
```

View the average CPU utilization across the entire fleet

```
SELECT AVG(CPUUtilization)
```

```
FROM SCHEMA("AWS/EC2", InstanceId)
```

View the 10 instances with the highest average CPU utilization

```
SELECT MAX(CPUUtilization)
FROM "AWS/EC2"
GROUP BY InstanceId
LIMIT 10
```

View the 10 instances with the highest CPU utilization, ordered by the maximum, in descending order

```
SELECT AVG(CPUUtilization)
FROM "AWS/EC2"
GROUP BY InstanceId
ORDER BY MAX() DESC
LIMIT 10
```

In this case, the CloudWatch agent is collecting a CPUUtilization metric per application. This query filters the average of this metric for a specific application name.

```
SELECT AVG(CPUUtilization)
FROM "AWS/CWAgent"
WHERE ApplicationName = 'eCommerce'
SELECT AVG(ConcurrentExecutions)
FROM "AWS/Lambda"
```

View average execution time for the top 10 Lambda functions, ordered by the maximum, in descending order

```
SELECT AVG(Duration)
FROM "AWS/Lambda"
GROUP BY FunctionName
ORDER BY MAX() DESC
```

```
LIMIT 10
```

View the maximum, average, and minimum of Lambda execution times

```
SELECT MAX(Duration)
FROM "AWS/Lambda"
```

Application Load Balancer examples

View metrics that have the dimensions **LoadBalancer** and **AvailabilityZone**

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
```

View metrics with number of active concurrent TCP connections

```
SELECT AVG(ActiveConnectionCount)
FROM "AWS/ApplicationELB"
```

Amazon EBS examples

View top 10 average write bytes per volume in descending order

```
SELECT AVG(VolumeWriteBytes)
FROM "AWS/EBS"
GROUP BY VolumeId
ORDER BY MAX() DESC
LIMIT 10
```

View average Amazon EBS volume write time

```
SELECT AVG(VolumeTotalWriteTime)
FROM "AWS/EBS"
```

View average Amazon EBS volume idle time

```
SELECT AVG(VolumeIdleTime)
FROM "AWS/EBS"
View average burst balance per volume
SELECT AVG(BurstBalance)
FROM "AWS/EBS"
GROUP BY VolumeId
View average read bytes across Amazon EBS volumes
SELECT AVG(VolumeReadBytes)
FROM "AWS/EBS"
```

View average write bytes across Amazon EBS volumes

```
SELECT AVG(VolumeWriteBytes)
FROM "AWS/EBS"
```

Amazon Simple Storage Service examples

View average latency group by bucket name

```

SELECT AVG(TotalRequestLatency)
FROM "AWS/S3"
GROUP BY BucketName
```

View average number of objects per bucket across all Amazon S3 buckets

```

SELECT AVG(NumberOfObjects)
FROM "AWS/S3"
GROUP BY BucketName

```

Amazon Simple Notification Service examples

Amazon-simple-notificaation-service-examples

```

SELECT AVG(NumberOfMessagesPublished)
FROM "AWS/SNS"
```

View average number of messages failed for each topic name

```
SELECT AVG(NumberOfNotificationsFailed)
FROM "AWS/SNS"
GROUP BY TopicName
```

AWS API usage examples

View top 20 AWS APIs by the number of calls in your account

```
SELECT COUNT(CallCount)
FROM "AWS/Usage"
WHERE "Type" = 'API'
GROUP BY "Service", "Resource"
ORDER BY SUM() DESC
LIMIT 20
```

CloudWatch Metrics Insights limits

CloudWatch Metrics Insights currently has the following limits:

- You can query only the most recent three hours of data.
- A single query can process no more than 10,000 metrics. This means that if the `SELECT`, `FROM`, and `WHERE` clauses would match more than 10,000 metrics, only the first 10,000 of these metrics that are found will be processed by the query.
- A single query can return no more than 500 time series. This means that if the query is processing more than 500 metrics, not all metrics will be returned in the query results. If you use an `ORDER BY` clause, then all the metrics being processed will be sorted and the 500 that have the highest or lowest values according to your `ORDER BY` clause will be returned. If you do not include an `ORDER BY` clause, you can't control which 500 matching metrics are returned.
- Each `GetMetricData` operation can have only one query, but you can have multiple widgets in a dashboard that each include a query.

Using the Amazon CloudWatch Logs query editor

To query CloudWatch Logs, select the Region and up to 20 log groups that you want to query. Use the main input area to write your query. For more information, see CloudWatch Logs Insights query syntax.

You can also write queries returning time series data by using the `stats` command in CloudWatch Logs Insights. When making `stats` queries in Explore, you have to make sure you are in Metrics Explore mode.

To the right of the query input field is a CloudWatch Logs Insights link that opens the CloudWatch Logs Insights console with your query. You can continue exploration there if necessary.

Using template variables

As with several other data sources, the CloudWatch data source supports the use of template variables in queries. For more information, see Templates and variables.

Deep linking from Grafana panels to the CloudWatch Logs console

If you want to view your query in the CloudWatch Logs Insights console, choose the **CloudWatch Logs Insights** button next to the query editor. If you're not currently signed in to the CloudWatch console, the link forwards you to the sign-in page. The provided link is valid for any AWS account but only displays the correct metrics if you're signed in to the AWS account that corresponds to the selected data source in Grafana.

Alerting

Because CloudWatch Logs queries can return numeric data, for example, through the use of the `stats` command, alerts are supported. For more information, see Grafana alerting.

Curated dashboards

The updated CloudWatch data source ships with pre-configured dashboards for five of the most popular AWS services:

- Amazon EC2
- Amazon Elastic Block Store
- AWS Lambda
- Amazon CloudWatch Logs
- Amazon Relational Database Service

To import the pre-configured dashboards, go to the configuration page of your CloudWatch data source and choose the **Dashboards** tab. Choose **Import** for the dashboard you want to use. To customize the dashboard, we recommend saving the dashboard under a different name, because otherwise the dashboard will be overwritten when a new version of the dashboard is released.

Templated queries

Instead of hardcoding details such as servers, applications, and sensor names in your metric queries, you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. You can use these dropdown boxes to change the data being displayed in your dashboard.

See Templates for an introduction to the templating feature and the different types of template variables.

Query variable

The CloudWatch data source provides the following queries that you can specify in the **Query** field in the **Variable** edit view. You can use these to fill a variable's options list with things such as `region`, `namespaces`, `metric names`, and `dimension keys/values`.

In place of `region`, you can specify `default` to use the default Region configured in the data source for the query.

Name	Description
`regions()`	Returns a list of all AWS Regions.
`namespaces()`	Returns a list of namespaces that CloudWatch supports.
`metrics(namespace, [region])`	Returns a list of metrics in the namespace. (Specify the Region or use `"default"` for custom metrics.)
`dimension_keys(namespace)`	Returns a list of dimension keys in the namespace.
`dimension_values(region, namespace , metric, dimension_key, [filters])`	Returns a list of dimension values matching the specified `region`, `namespace` , `metric`, or `dimension_key` . ternatively, you can use dimension `filters` to get a more specific result.
`ebs_volume_ids(region, instance_id)`	Returns a list of volume IDs matching the specified `region`, `instance_id` .
`ec2_instance_attribute(region,`	Returns a list of attributes matching the specified `region`, `attribute_name` , `filters`.

Name	Description
`attribute_name, filters)`	
`resource_arns(region, resource_type, tags)`	Returns a list of ARNs matching the specified `region`, `resource_type` , and `tags`.
`statistics()`	Returns a list of all the standard statistics.

For details about the metrics that CloudWatch provides, see AWS services that publish CloudWatch metrics.

Examples of templated queries

The following table shows example dimension queries that return a list of resources for individual AWS services.

Query	Service
`dimension_values(us-east-1,AWS/ELB,RequestCount,LoadBalancerName)`	Elastic Load Balancing
`dimension_values(us-east-1,AWS/ElastiCache,CPUUtilization,CacheClusterId)`	Amazon ElastiCache
`dimension_values(us-east-1,AWS/Redshift,CPUUtilization,ClusterIdentifier)`	Amazon Redshift
`dimension_values(us-east-1,AWS/RDS,CPUUtilization,DBInstanceIdentifier)`	Amazon RDS
`dimension_values(us-east-1,AWS/S3,BucketSizeBytes,BucketName)`	Amazon Simple Storage Service

Query	Service
	(Amazon S3)
`dimension_values(us-east-1,CWAgent,disk_used_percent,device,{"InstanceId":"$instance_id"})`	CloudWatch Agent
`resource_arns(eu-west-1,elasticloadbalancing:loadbalancer,{"elasticbeanstalk:environment-name":["myApp-dev","myApp-prod"]})`	Elastic Load Balancing
`resource_arns(eu-west-1,ec2:instance,{"elasticbeanstalk:environment-name":["myApp-dev","myApp-prod"]})`	Amazon EC2

Using ec2_instance_attribute examples

JSON filters

The `ec2_instance_attribute` query takes `filters` in JSON format. You can specify pre-defined filters of `ec2:DescribeInstances`. Note that the actual filtering takes place in AWS, not in Grafana.

The following code example shows the filters syntax.

```
{ filter_name1: [ filter_value1 ], filter_name2: [ filter_value2 ] }
```

The following example shows the `ec2_instance_attribute()` query.

```
ec2_instance_attribute(us - east - 1, InstanceId, { 'tag:Environment':
 ['production'] });
```

Selecting attributes

Only one attribute per instance can be returned. Any flat attribute can be selected (that is, if the attribute has a single value and isn't an object or array). The following flat attributes are available.

- `AmiLaunchIndex`
- `Architecture`

- ClientToken
- EbsOptimized
- EnaSupport
- Hypervisor
- IamInstanceProfile
- ImageId
- InstanceId
- InstanceLifecycle
- InstanceType
- KernelId
- KeyName
- LaunchTime
- Platform
- PrivateDnsName
- PrivateIpAddress
- PublicDnsName
- PublicIpAddress
- RamdiskId
- RootDeviceName
- RootDeviceType
- SourceDestCheck
- SpotInstanceRequestId
- SriovNetSupport
- SubnetId
- VirtualizationType
- VpcId

Tags can be selected by prefixing the tag name with `Tags`.

The following example shows the `ec2_instance_attribute()` query.

```
ec2_instance_attribute(us - east - 1, Tags.Name, { 'tag:Team': ['sysops'] });
```

Using JSON format template variables

Some queries accept filters in JSON format and Grafana supports the conversion of template variables to JSON.

If `env = 'production', 'staging'`, the following query will return ARNs of EC2 instances for which the `Environment` tag is `production` or `staging`.

```
resource_arns(us-east-1, ec2:instance, {"Environment":${env:json}})
```

Pricing

The Amazon CloudWatch data source for Grafana uses the `ListMetrics` and `GetMetricData` CloudWatch API calls to list and retrieve metrics. Pricing for CloudWatch Logs is based on the amount of data ingested, archived, and analyzed via CloudWatch Logs Insights queries. For more information, see Amazon CloudWatch Pricing.

Every time you pick a dimension in the query editor, Grafana issues a `ListMetrics` request. Whenever you change the queries in the query editor, one new request to GetMetricData will be issued.

API requests to retrieve data samples use the `GetMetricData` operation. This operation provides better support for CloudWatch metric math. It also supports the automatic generation of search expressions when using wildcard characters or turning off the **Match Exact** option. The `GetMetricData` operation incurs charges. For more information, see Amazon CloudWatch Pricing.

Service quotas

AWS defines quotas, or limits, for resources, operations, and items in your AWS account. Depending on the number of queries in your dashboard and the number of users accessing the dashboard, you might reach the usage limits for various CloudWatch and CloudWatch Logs resources. Note that quotas are defined per account and per AWS Region. If you're using multiple Regions or you have set up more than one CloudWatch data source to query against multiple accounts, you must request a quota increase for each account and each Region in which you reach the limit.

For more information, see CloudWatch service quotas.

Cross-account observability

> **Warning**
>
> This feature requires your Grafana workspace to be version 9 or later.

The CloudWatch plugin enables you to monitor and troubleshoot applications across multiple regional accounts. Using cross-account observability, you can seamlessly search, visualize and analyze metrics and logs without worrying about account boundaries.

To enable cross-account observability, first enable it in CloudWatch, then add the proper IAM actions to the role/user running the plugin. If your Amazon Managed Grafana workspace is running within a VPC, then you must also have a NAT gateway to support internet access.

- To learn how to enable the feature, see CloudWatch cross-account observability in the *Amazon CloudWatch User Guide*.
- The following actions are the proper IAM actions to add for the role/user that is running the plugin.

```
{
"Sid":  "AllowReadingAcrossAccounts",
"Effect":  "Allow",
"Action": [
  "oam:ListSinks",
  "oam:ListAttachedLinks"
],
"Resource":  "*"
}
```

- Cross-account observability for the CloudWatch data source relies on Amazon CloudWatch Observability Access Manager. The Observability Access Manager does not support a VPC endpoint. If your Amazon Managed Grafana workspace is running within a VPC, then you must also have a NAT Gateway that allows the workspace to call APIs on the internet.

> **Note**
>
> You must also have IAM permissions to read the CloudWatch data in the account you are trying to access.

Connect to an Amazon OpenSearch Service data source

> **Note**
>
> In workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

With Amazon Managed Grafana, you can add Amazon OpenSearch Service as a data source by using the AWS data source configuration option in the Grafana workspace console. This data source supports OpenSearch Service domains, which run OpenSearch clusters as well as legacy Elasticsearch clusters.

The AWS data source configuration option simplifies adding OpenSearch Service as a data source by discovering your existing OpenSearch Service accounts, and manages the configuration of the authentication credentials that are required to access OpenSearch. You can use this method to set up authentication and add OpenSearch Service as a data source, or you can manually set up the data source and the necessary authentication credentials using the same method that you would on a self-managed Grafana server.

The OpenSearch Service data source supports piped processing language (PPL). For more information about PPL, see Querying Amazon OpenSearch Service data using Piped Processing Language.

You can use the OpenSearch Service data source to perform many types of simple or complex OpenSearch queries in order to visualize logs or metrics stored in OpenSearch. You can also annotate your graphs with log events stored in OpenSearch.

Topics

- Use AWS data source configuration to add OpenSearch Service as a data source
- Manually add Amazon OpenSearch Service as a data source

- OpenSearch Service settings
- Using the Amazon OpenSearch Service data source
- Amazon OpenSearch Service Serverless
- Traces support

Use AWS data source configuration to add OpenSearch Service as a data source

To use AWS data source configuration, first you use the Amazon Managed Grafana console to enable service-managed IAM roles that grant the workspace the IAM policies necessary to read the OpenSearch Service resources in your account or in your entire organizational units. Then you use the Amazon Managed Grafana workspace console to add OpenSearch Service as a data source.

To use AWS data source configuration to add OpenSearch Service as a data source

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the upper left corner of the page, choose the menu icon and then choose **All workspaces**.
3. Choose the name of the workspace.
4. If you didn't choose to use service-managed permissions for this workspace when you created it, then change from using customer-managed permissions to use service-managed permissions to ensure that the proper IAM roles and policies are enabled for using the AWS data source configuration option in the Grafana workspace console. To do so, choose the edit icon by **IAM role** and then choose **Service managed**, **Save changes**. For more information, see Amazon Managed Grafana permissions and policies for AWS data sources.
5. Choose the **Data sources** tab. Then select the check box for **Amazon OpenSearch Service**, and choose **Actions**, **Enable service-managed policy**.
6. Choose the **Data sources** tab again, and then choose **Configure in Grafana** in the **Amazon OpenSearch Service** row.
7. Sign into the Grafana workspace console using IAM Identity Center if necessary.
8. In the left navigation bar in the Grafana workspace console, choose the AWS icon and then choose **AWS services**, **Amazon OpenSearch Service**.
9. Select the Region that you want Amazon Managed Grafana to search to discover OpenSearch Service resources, and then select the accounts and the OpenSearch Service domains you want to add, configure the index settings, and then choose **Add data sources**.

Manually add Amazon OpenSearch Service as a data source

To manually add the Amazon OpenSearch Service data source

1. In the Grafana console side menu, choose the **AWS** icon, then choose **Data Sources**.
2. Choose the **Amazon OpenSearch Service** data source. If necessary, you can start typing **OpenSearch** in the search box to help you find it.
3. Choose the **Region** you want to search data from.
4. Choose **Add data source**.

> **Note**
>
> If you don't see the **Data Sources** link in your side menu, it means that your current user does not have the `Admin` role.

OpenSearch Service settings

Nam	Description
Name	The data source name. This is how you see the data source in panels and queries.
Defa	Default data source means that it will be pre-selected for new panels.
Url	The endpoint of your OpenSearch Service domain. The endpoint takes the following format: https://search-my-domain.us-east-1.es.amazonaws.com.
Acce	Server (default) = URL must be accessible from the Grafana backend/server. Browser = URL must be accessible from the browser.

Access mode controls how requests to the data source will be handled. Server should be the preferred way if nothing else is stated.

Server access mode (default)

All requests are made from the browser to Grafana backend or server, which forwards the requests to the data source, circumventing possible Cross-Origin Resource Sharing (CORS) requirements. If you select this access mode, the URL must be accessible from the Grafana backend or server.

Browser (direct) access

Amazon Managed Grafana does not support browser direct access.

Index settings

Here you can specify a default for the `time field` and specify the name of your OpenSearch index. You can use a time pattern for the index name or a wildcard character.

OpenSearch/Elasticsearch version

Specify your OpenSearch or legacy Elasticsearch version in the version dropdown menu. The version is important because there are differences in how queries are composed for each version. Currently, Grafana supports OpenSearch 1.0.x. Supported versions of Elasticsearch are `2.0+`, `5.0+`, `5.6+`, `6.0+`, and `7.0+`. The value `5.6+` means version 5.6 or higher, but lower than 6.0. The value `6.0+` means version 6.0 or higher, but lower than 7.0. Finally, `7.0+` means version 7.0 or higher, but lower than 8.0.

Min time interval

A lower limit for the auto group by time interval. Recommended to be set to write frequency; for example, `1m` if your data is written every minute. This option can also be overridden/configured in a dashboard panel under data source options. This value **must** be formatted as a number followed by a valid time identifier; for example, `1m` (1 minute) or `30s` (30 seconds). The following time identifiers are supported.

Identifier	Description
`y`	Year
`M`	Month
`w`	Week
`d`	Day
`h`	Hour
`m`	Minute
`s`	Second

Identifier	Description
ms	Millisecond

Logs

Two parameters, `Message field name` and `Level field name`, can optionally be configured from the data source settings page that determine which fields will be used for log messages and log levels when visualizing logs in Explore.

For example, if you use a default setup of Filebeat for shipping logs to OpenSearch Service, the following configuration should work.

- **Message field name:** message
- **Level field name:** fields.level

Data links

Data links create a link from a specified field that can be accessed in logs view in Explore.

Each data link configuration consists of the following:

- **Field** – Name of the field used by the data link.
- **URL/query** – If the link is external, then enter the full link URL. If the link is internal link, then this input serves as query for the target data source. In both cases, you can interpolate the value from the field with `${__value.raw }` macro.
- **Internal link** – Select this if the link is internal or external. If the link is internal, a data source selector allows you to select the target data source. Only tracing data sources are supported.

Using the Amazon OpenSearch Service data source

Metric query editor

The OpenSearch query editor allows you to select multiple metrics and group by multiple terms or filters. Use the plus and minus icons to the right to add/remove metrics or group by clauses. Some metrics and group by clauses have options. Choose the option text to expand the\ row to view and edit metric or group by options.

Using Piped Processing Language (PPL)

The Amazon OpenSearch Service data source supports Piped Processing Language (PPL), which enables simpler yet powerful querying and visualization capabilities for OpenSearch. PPL enables customers to explore and find data without having to compose lengthy OpenSearch Domain Specific Language (DSL) statements or write queries using JSON objects. With PPL, you can write queries as a set of commands delimited by pipes similar to UNIX pipes.

Take the following sample DSL query as an example:

```
GET opensearch_sample_data_logs/_search{"from":0,"size":0,"timeout":"1m","query":
{"bool":{"should":[{"term":{"response.keyword":{"value":"404","boost":1}}},{"term":
{"response.keyword":
{"value":"503","boost":1}}}],"adjust_pure_negative":true,"boost":1}},"sort":
[{"_doc":{"order":"asc"}}],"aggregations":{"composite_buckets":{"composite":
{"size":1000,"sources":[{"host":{"terms":
{"field":"host.keyword","missing_bucket":true,"order":"asc"}}},{"response":{"terms":
{"field":"response.keyword","missing_bucket":true,"order":"asc"}}}]},"aggregations":
{"request_count":{"value_count":{"field":"request.keyword"}},"sales_bucket_sort":
{"bucket_sort":{"sort":[{"request_count":{"order":"desc"}}],"size":10}}}}}}>
```

The preceding DSL query can be replaced with the following PPL command that is concise and human readable.

```
source = opensearch_sample_data_logs | where response='404' or response='503' | stats
 count(request) as request_count by host, response | sort -request_count
```

For more information about PPL, see Querying Amazon OpenSearch Service data using Piped Processing Language.

Series naming and alias patterns

You can control the name for time series using the `Alias` input field.

Pattern	Description
`{{term fieldname}}`	Replaced with value of a term Group By.
`{{metric}}`	Replaced with metric name (ex. Average, Min, Max).

Pattern	Description
`{{field}}`	Replaced with the metric field name.

Pipeline metrics

Some metric aggregations are called pipeline aggregations; for example, *Moving Average* and *Derivative*. OpenSearch pipeline metrics require another metric to be based on. Use the eye icon next to the metric to hide metrics from appearing in the graph. This is useful for metrics you only have in the query for use in a pipeline metric.

Templating

Instead of hardcoding things such as server, application, and sensor name in your metric queries you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. You can use these dropdown boxes to change the data being displayed in your dashboard.

For more information about templating and template variables, see Templates and variables.

Query variable

The OpenSearch Service data source supports two types of queries you can use in the *Query* field of *Query* variables. The query is written using a custom JSON string.

Query	Description
`{"find": "fields", "type": "keyword"}`	Returns a list of field names with the index type `keyword`.
`{"find": "terms", "field": "@hostname", "size": 1000}`	Returns a list of values for a field using term aggregation. Query will use current dashboard time range as time range for query.
`{"find": "terms", "field": "@hostname", "query": '<lucene query>'}`	Returns a list of values for a field using term aggregation and a specified Lucene query filter. Query will use current dashboard time range as time range for query.

There is a default size limit of 500 on terms queries. To set a custom limit, set the size property in your query. You can use other variables inside the query. The following code example shows the query definition for a variable named `$host`.

```
{"find": "terms", "field": "@hostname", "query": "@source:$source"}
```

In the previous example, we use another variable named `$source` inside the query definition. Whenever you change, using the dropdown list, the current value of the `$source` variable, it initiates an update of the `$host` variable. After the update, the `$host` variable contains only hostnames filtered by in this case the `@source` document property.

These queries by default return results in term order (which can then be sorted alphabetically or numerically as for any variable). To produce a list of terms sorted by doc count (a top-N values list), add an `orderBy` property of `doc_count`. This automatically selects a descending sort. Using `asc` with doc_count (a bottom-N list) can be done by setting `order: "asc"`, but it is discouraged because it increases the error on document counts. To keep terms in the doc count order, set the variable's **Sort** dropdown list to **Disabled**. Alternatively, you might alternatively still want to use **Alphabetical** to re-sort them.

```
{"find": "terms", "field": "@hostname", "orderBy": "doc_count"}
```

Using variables in queries

There are two syntaxes:

- `$<varname>` Example: @hostname:$hostname
- `[[varname]]` Example: @hostname:[[hostname]]

Why two ways? The first syntax is easier to read and write, but it does not allow you to use a variable in the middle of a word. When the *Multi-value* or *Include all value* options are enabled, Grafana converts the labels from plaintext to a Lucene-compatible condition.

In the previous example, we have a lucene query that filters documents based on the `@hostname` property using a variable named `$hostname`. It is also using a variable in the *Terms* group by field input box. This allows you to use a variable to quickly change how the data is grouped.

Annotations

Annotations allow you to overlay rich event information on top of graphs. You add annotation queries using the Dashboard menu or Annotations view. Grafana can query any OpenSearch index for annotation events. For more information, see Annotations.

Nar	Description
Que	You can keep the search query blank or specify a Lucene query.
Tim	The name of the time field; must be date field.
Tim End	Optional name of the time end field must be date field. If set, annotations will be marked as a region between time and time-end.
Tex	Event description field.
Tag	Optional field name to use for event tags (can be an array or a CSV string).

Querying logs

Querying and displaying log data from OpenSearch is available in Explore. To display your logs, select the OpenSearch Service data source, and then optionally enter a Lucene query. For more information, see Explore.

Log queries

After the result is returned, the log panel shows a list of log rows and a bar chart where the x-axis shows the time and the y-axis shows the frequency or count.

Filtering log messages

Optionally, enter a Lucene query into the query field to filter the log messages. For example, using a default Filebeat setup, you should be able to use `fields.level:error` to show only error log messages.

Amazon OpenSearch Service Serverless

> **Note**
>
> OpenSearch Service Serverless support is only available with Grafana workspaces that are running Grafana version 9.4 and later.

You can use the OpenSearch Service data source to access Amazon OpenSearch Service Serverless data with Amazon Managed Grafana. Access to the data is controlled by data access policies. The following example shows a policy that allows users to query a specific collection and index. Be sure to replace *collection_name*, *index_name*, and *principal_arn* with the correct values for your use case.

```
[
  {
    "Rules": [
      {
        "Resource": ["collection/{collection_name}"],
        "Permission": ["aoss:DescribeCollectionItems"],
        "ResourceType": "collection"
      },
      {
        "Resource": ["index/{collection_name}/{index_name}"],
        "Permission": ["aoss:DescribeIndex", "aoss:ReadDocument"],
        "ResourceType": "index"
      }
    ],
    "Principal": ["principal_arn"],
    "Description": "read-access"
  }
]
```

Traces support

The OpenSearch plugin has support for viewing a list of traces in table form, and a single trace in **Trace View**, which shows the timeline of trace spans.

> **Note**
>
> Querying OpenSearch traces is only available using Lucene queries.

Trace support is only available for Grafana workspaces that support version 9.4 or newer.

To create a query showing all traces, use the Lucene query type `Traces` with a blank query. If necessary, select the **Table** visualization type.

Selecting a trace ID in the table will open that trace in the trace view.

To create a query showing a single trace, use the query `traceid: {traceId}`, and, if necessary, select the **Traces** visualization type.

Connect to an AWS IoT SiteWise data source

Note

In workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

With Amazon Managed Grafana, you can add AWS IoT SiteWise as a data source by using the AWS data source configuration option in the Grafana workspace console. This feature simplifies adding AWS IoT SiteWise as a data source by discovering your existing AWS IoT SiteWise accounts and manages the configuration of the authentication credentials that are required to access AWS IoT SiteWise. You can use this method to set up authentication and add AWS IoT SiteWise as a data source, or you can manually set up the data source and the necessary authentication credentials using the same method that you would on a self-managed Grafana server.

Topics

- Use AWS data source configuration to add AWS IoT SiteWise as a data source
- Manually adding the AWS IoT SiteWise data source
- AWS IoT SiteWise settings
- Using the AWS IoT SiteWise data source

Use AWS data source configuration to add AWS IoT SiteWise as a data source

To use AWS data source configuration, first you use the Amazon Managed Grafana console to enable service-managed IAM roles that grant the workspace the IAM policies necessary to read the

AWS IoT SiteWise resources in your account or in your entire organizational units. Then you use the Amazon Managed Grafana workspace console to add AWS IoT SiteWise as a data source.

To use AWS data source configuration to add AWS IoT SiteWise as a data source

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the upper left corner of the page, choose the menu icon and then choose **All workspaces**.
3. Choose the name of the workspace.
4. If you didn't choose to use service-managed permissions for this workspace when you created it, then change from using customer-managed permissions to use service-managed permissions to ensure that the proper IAM roles and policies are enabled for using the AWS data source configuration option in the Grafana workspace console. To do so, choose the edit icon by **IAM role** and then choose **Service managed**, **Save changes**. For more information, see Amazon Managed Grafana permissions and policies for AWS data sources.
5. Choose the **Data sources** tab. Then select the check box for **AWS IoT SiteWise**, and choose **Actions**, **Enable service-managed policy**.
6. Choose the **Data sources** tab again, and then choose **Configure in Grafana** in the **AWS IoT SiteWise** row.
7. Sign into the Grafana workspace console using IAM Identity Center if necessary.
8. In the left navigation bar in the Grafana workspace console, choose the AWS icon and then choose **AWS services**, **IoT SiteWise**.
9. Select the default Region that you want the AWS IoT SiteWise data source to query from, select the accounts, and then choose **Add data source**.

Manually adding the AWS IoT SiteWise data source

To manually add the AWS IoT SiteWise data source

1. In the Grafana console side menu, pause on the **Configuration** (gear) icon, then choose **Data Sources**.
2. Choose **Add data source**.
3. Choose the **AWS IoT SiteWise** data source. If necessary, you can start typing **`SiteWise`** in the search box to help you find it.

AWS IoT SiteWise settings

Name	Description
Name	The data source name. This is how you see the data source in panels and queries.
Auth Provider	Specify the provider to get credentials.
Default Region	Used in query editor to set the region (can be changed on per query basis).
Credentials profile name	Specify the name of the profile to use (if you use `~/.aws/credentials` file); keep blank for default.
Assume Role Arn	Specify the ARN of the role to assume.
Endpoint (optional)	If you must specify an alternate service endpoint.

Using the AWS IoT SiteWise data source

For information about how to use the AWS IoT SiteWise data source, see AWS IoT SiteWise Datasource on Github.

Connect to an AWS IoT TwinMaker data source

> **Note**
>
> In workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

With Amazon Managed Grafana, you can add AWS IoT TwinMaker, a powerful industrial data analytics service, as an app and data source in your Grafana workspace. With AWS IoT TwinMaker, you can create end-user 3D digital twin applications to monitor industrial operations. The AWS IoT TwinMaker is a service that makes it faster for developers to create digital replicas of real-world

systems, helping more customers realize the potential of digital twins to optimize operations. The AWS IoT TwinMaker for Grafana provides custom panels, dashboard templates, and a data source to connect to your digital twin data.

Manually adding the AWS IoT TwinMaker data source

Prerequisites

Before you begin, ensure that you have access to **AWS IoT TwinMaker** from your AWS account.

To learn how to add permission to your workspace IAM role to access AWS IoT TwinMaker, see Adding the permission for AWS IoT TwinMaker to your workspace user role.

To add the AWS IoT TwinMaker data source:

1. Ensure that your user role is admin or editor.
2. In the Grafana console side menu, hover over the **Configuration** (gear) icon and then choose **Data Sources**.
3. Choose **Add data source**.
4. Choose the **AWS IoT TwinMaker** data source. If necessary, you can start typing **`TwinMaker`** in the search box to help you find it.
5. This opens the **Connection Details** page. Follow the steps in configuring the AWS IoT TwinMaker connection details settings.

Adding the permission for AWS IoT TwinMaker to your workspace user role

To add permissions for AWS IoT TwinMaker to your workspace user role, assume role permission between Amazon Managed Grafana workspace and TwinMaker dashboard roles.

1. Go to https://console.aws.amazon.com/iam/.
2. Manually create a dashboard role. For more information about creating a dashboard role, see To manually create a Grafana AWS IoT TwinMaker dashboard role.

AWS IoT TwinMaker connection details settings

Configure connection details settings

1. In the **Connection Details** menu, select the authentication provider (recommended: **Workspace IAM Role**).

2. Choose the **Default Region** you want to query.
3. In the **TwinMaker settings**, enter the AWS IoT TwinMaker workspace name.

Using the AWS IoT TwinMaker data source

For information about how to use the AWS IoT TwinMaker data source, see AWS IoT TwinMaker Datasource on GitHub.

To manually create a Grafana AWS IoT TwinMaker dashboard role

To manually create a Grafana AWS IoT TwinMaker dashboard role

1. Sign in to the IAM console at https://console.aws.amazon.com/iam/.
2. Locate your Amazon Managed Grafana workspace role in the summary. It appears as follows:

```
AmazonGrafanaServiceRole-random_ID
```

3. Add the following inline policy to the role:

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "arn:aws:iam::111122223333:role/TwinMakerDashboardRole"
    }
}
```

4. Add a new inline policy for each dashboard role. Alternatively, add a list of role Amazon Resource Names (ARNs) on the **Resource** line.
5. Find your dashboard role in the IAM console. It should have a `SceneViewer` policy and, optionally, a `VideoPlayer` policy.
6. Choose the **Trust relationship** tab.
7. Choose **Edit trust relationship**.
8. Enter the following policy, replacing *AMGWorkspaceRoleArn* with the Arn from your account:

JSON

```
 {
"Version":"2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "AWS": "AMGWorkspaceRoleARN"
    },
    "Action": "sts:AssumeRole"
  }
]
}
```

Example of an AWS IoT TwinMaker policy

The following is a minimal AWS IoT TwinMaker policy that you can attach to a dashboard role. You must replace the values for the AWS IoT TwinMaker workspace ARN and ID, as well as the Amazon S3 bucket ARN, based on your own resources.

Connect to Amazon Managed Service for Prometheus and open-source Prometheus data sources

In Amazon Managed Grafana, the Prometheus data source supports using both self-managed Prometheus servers and Amazon Managed Service for Prometheus workspaces as data sources. For more information about Amazon Managed Service for Prometheus, see What is Amazon Managed Service for Prometheus?

With Amazon Managed Grafana, you can add an Amazon Managed Service for Prometheus workspace as a data source by using the AWS data source configuration option in the Grafana workspace console. This feature simplifies adding Amazon Managed Service for Prometheus as a data source by discovering your existing Amazon Managed Service for Prometheus accounts and manages the configuration of the authentication credentials that are required to access Amazon Managed Service for Prometheus.

Note

You can view your Prometheus alerts in the unified Grafana alerting interface, by Configuring an Alertmanager data source.

Important

Starting in Amazon Managed Grafana version 12, SigV4 authentication support in the Core Prometheus plugin has been removed. All Amazon Managed Service for Prometheus data sources that were previously using the Core Prometheus plugin are automatically migrated to the Amazon Managed Service for Prometheus plugin. Any dashboards using these data sources are automatically updated to reflect this change. See Connect to an Amazon Managed Service for Prometheus data source.

Topics

- Use AWS data source configuration to add Amazon Managed Service for Prometheus as a data source
- Manually adding the Prometheus data source
- Using the Prometheus data source
- Visualize alerts from Amazon Managed Service for Prometheus
- Configure exemplars

Use AWS data source configuration to add Amazon Managed Service for Prometheus as a data source

Important

Starting in Amazon Managed Grafana version 12, SigV4 authentication support in the Core Prometheus plugin has been removed. All Amazon Managed Service for Prometheus data sources that were previously using the Core Prometheus plugin are automatically migrated to the Amazon Managed Service for Prometheus plugin. Any dashboards using these data sources are automatically updated to reflect this change. See Connect to an Amazon Managed Service for Prometheus data source.

To use AWS data source configuration, first you use the Amazon Managed Grafana console to enable service-managed IAM roles that grant the workspace the IAM policies necessary to read the Amazon Managed Service for Prometheus resources in your account or in your entire organizational units. Then you use the Amazon Managed Grafana workspace console to add Amazon Managed Service for Prometheus as a data source.

To use AWS data source configuration to add Amazon Managed Service for Prometheus as a data source

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the upper left corner of the page, choose the menu icon and then choose **All workspaces**.
3. Choose the name of the workspace.
4. If you didn't choose to use service-managed permissions for this workspace when you created it, then change from using customer-managed permissions to use service-managed permissions to ensure that the proper IAM roles and policies are enabled for using the AWS data source configuration option in the Grafana workspace console. To do so, choose the edit icon by **IAM role** and then choose **Service managed**, **Save changes**. For more information, see Amazon Managed Grafana permissions and policies for AWS data sources.
5. Choose the **Data sources** tab. Then select the check box for **Amazon Managed Service for Prometheus**, and choose **Actions**, **Enable service-managed policy**.
6. Choose the **Data sources** tab again, and then choose **Configure in Grafana** in the **Amazon Managed Service for Prometheus** row.
7. Sign into the Grafana workspace console using IAM Identity Center if necessary.
8. In the left navigation bar in the Grafana workspace console, choose the AWS icon and then choose **AWS services**, **Prometheus**.
9. Select the Region that you want Amazon Managed Grafana to search to discover Amazon Managed Service for Prometheus workspaces, and then select the accounts and Amazon Managed Service for Prometheus workspaces that you want to add, and then choose **Add data source**.

Manually adding the Prometheus data source

Important

Starting in Amazon Managed Grafana version 12, SigV4 authentication support in the Core Prometheus plugin has been removed. All Amazon Managed Service for Prometheus

data sources that were previously using the Core Prometheus plugin are automatically migrated to the Amazon Managed Service for Prometheus plugin. Any dashboards using these data sources are automatically updated to reflect this change. See Connect to an Amazon Managed Service for Prometheus data source.

To manually add the Prometheus data source

1. In the Grafana console side menu, pause on the **Administration** menu item (or the **Configuration** (gear) icon in Grafana v8), then choose **Data Sources**.
2. Choose **Add data source**.
3. Choose the **Prometheus** data source. If necessary, you can start typing **`Prometheus`** in the search box to help you find it.

Using the Prometheus data source

Important

Starting in Amazon Managed Grafana version 12, SigV4 authentication support in the Core Prometheus plugin has been removed. All Amazon Managed Service for Prometheus data sources that were previously using the Core Prometheus plugin are automatically migrated to the Amazon Managed Service for Prometheus plugin. Any dashboards using these data sources are automatically updated to reflect this change. See Connect to an Amazon Managed Service for Prometheus data source.

Prometheus settings

Name	Description
`Name`	The data source name. This is how you see the data source in panels and queries.
`Default`	Default data source means that it will be pre-selected for new panels.
`Url`	The URL of your Prometheus server; for example, `https://prometheus.example.org:9090` .

Name	Description
`Access`	Server (default) = URL must be accessible from the Grafana backend/server.
`Basic Auth`	Enable basic authentication to the Prometheus data source.
`User`	User name for basic authentication.
`Passwor`	Password for basic authentication.
`Scrape interva`	Set this to the typical scrape and evaluation interval configured in Prometheus. Defaults to 15s.
`Disable metrics lookup`	Checking this option will disable the metrics chooser and metric/label support in the query field's autocomplete. This helps if you have performance issues with bigger Prometheus instances.
`Custom Query Paramet s`	Add custom parameters to the Prometheus query URL. For example `timeout`, `partial_response` , dedup, or `max_source_resolution` . Multiple parameter s should be concatenated together with an "&".

Prometheus query editor

The following sections provide information and options for Prometheus query editor in the dashboard and in Explore.

Query editor in dashboards

Open a graph in edit mode by choosing the title and then choosing **Edit** (or by pressing **e** key while pausing on the panel).

N	Description
Q e n	For more information about Prometheus query expressions, see the Prometheus documenta tion.

N	Description
L f	Controls the name of the time series, using name or pattern. For example `{{hostname}}` is replaced with the label value for the label `hostname`.
M s	An additional lower limit for the `step` parameter of Prometheus range queries and for the `$__interval` and `$__rate_interval` variables. The limit is absolute and not modified by the *Resolution* setting.
R n	`1/1` sets both the `$__interval` variable and the `step` parameter of Prometheus range queries such that each pixel corresponds to one data point. For better performance, use lower resolutions. `1/2` only retrieves a data point for every other pixel, and `1/10` retrieves one data point per 10 pixels. Note that both *Min time interval* and *Min step* limit the final value of `$__interval` and `step`.
M l	Search for metric names in this input field.
F a	Switch between `Table`, `Time series`, or `Heatmap`. `Table` works only in the table panel. `Heatmap` is suitable for displaying metrics of the histogram type on a heatmap panel. It converts cumulative histograms to regular ones and sorts series by the bucket bound.
I	Perform an "instant" query, to return only the latest value that Prometheus has scraped for the requested time series. Instant queries return results much faster than normal range queries. Use them to look up label sets.
M t i	This value multiplied by the denominator from the *Resolution* setting sets a lower limit to both the `$__interval` variable and the `step` parameter of Prometheus range queries. Defaults to *Scrape interval* as set in the data source options.

Note

Amazon Managed Grafana modifies the request dates for queries to align them with the dynamically calculated step. This ensures consistent display of metrics data, but it can result in a small gap of data at the right edge of a graph.

Instant queries in dashboards

The Prometheus data source allows you to run instant queries, which query only the latest value. You can visualize the results in a table panel to see all available labels of a time series.

Instant query results are made up of only one data point per series. They can be shown in the graph panel with the help of series overrides. To show them in the graph as a latest value point, add a series override and select `Points > true`. To show a horizontal line across the whole graph, add a series override and select `Transform > constant` For more information about series overrides, see Series overrides.

Query editor in Explore

Na	Description
Qu ex n	For more information about Prometheus query expression, see the Prometheus documentation.
St	`Step` parameter of Prometheus range queries. Time units can be used here, for example: 5s, 1m, 3h, 1d, 1y. Default unit if no unit specified is `s` (seconds).
Qu ty	`Range`, `Instant`, or `Both`. When running **Range query**, the result of the query is displayed in graph and table. Instant query returns only the latest value that Prometheus has scraped for the requested time series and it is displayed in the table. When **Both** is selected, both instant query and range query is run. Result of range query is displayed in graph and the result of instant query is displayed in the table.

Metrics browser

The metrics browser allows you to quickly find metrics and select relevant labels to build basic queries. When you open the browser you will see all available metrics and labels. If supported by your Prometheus instance, each metric will show its HELP and TYPE as a tooltip.

When you select a metric, the browser narrows down the available labels to show only the ones applicable to the metric. You can then select one or more labels for which the available label values are shown in lists in the bottom section. Select one or more values for each label to tighten your query scope.

> **Note**
>
> If you do not remember a metric name to start with, you can also select a few labels first, to narrow down the list and then find relevant label values.

All lists in the metrics browser have a search field above them to quickly filter for metrics or labels that match a certain string. The values section only has one search field. Its filtering applies to all labels to help you find values across labels once they have been selected, for example, among your labels app, job, job_name only one might with the value you are looking for.

Once you are satisfied with your query, click “Use query” to run the query. The **Use as rate query** button adds a rate(...)[$__interval] around your query to help write queries for counter metrics. The “Validate selector” button will check with Prometheus how many time series are available for that selector.

Limitations

The metrics browser has a hard limit of 10,000 labels (keys) and 50,000 label values (including metric names). If your Prometheus instance returns more results, the browser will continue functioning. However, the result sets will be cut off above those maximum limits.

Templating

Instead of hardcoding things such as server, application and sensor name in your metric queries, you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. You can use these dropdown boxes to change the data being displayed in your dashboard.

For more information about templating and template variables, see Templates and variables.

Query variable

Variable of the type *Query* allows you to query Prometheus for a list of metrics, labels, or label values. The Prometheus data source plugin provides the following functions you can use in the **Query** input field.

Name	Description
`label_names()`	Returns a list of label names.

Name	Description
`label_values(label)`	Returns a list of label values for the `label` in every metric.
`label_values(metric, label)`	Returns a list of label values for the `label` in the specified metric.
`metrics(metric)`	Returns a list of metrics matching the specified `metric` regex.
`query_result(query)`	Returns a list of Prometheus query result for the `query`.

For information about what *metric names*, *label names* and *label values* are, see the Prometheus documentation.

Using interval and range variables

> **Note**
>
> Support for `$__range`, `$__range_s`, and `$__range_ms` are available only from Grafana v5.3.

You can use some global variables in query variables: `$__interval`, `$__interval_ms`, `$__range`, `$__range_s`, and `$__range_ms`. For more information, see Global variables. These can be convenient to use with the `query_result` function when you must filter variable queries because the `label_values` function doesn't support queries.

To get the correct instances when changing the time range on the dashboard, make sure to set the variable's `refresh` trigger to be `On Time Range Change`.

The following code example shows how to populate a variable with the busiest five request instances based on average QPS over the time range shown in the dashboard.

```
Query: query_result(topk(5, sum(rate(http_requests_total[$__range])) by (instance)))
Regex: /"([^"]+)"/
```

The following code example shows how to populate a variable with the instances having a certain state over the time range shown in the dashboard, using `$__range_s`.

```
Query: query_result(max_over_time(<metric>[${__range_s}s]) != <state>)
Regex:
```

Using `$__rate_interval` variable

The `$__rate_interval` variable is meant to be used in the rate function. It is defined as max(`$__interval` + *Scrape interval*, 4 * *Scrape interval*). *Scrape interval* is the Min step setting (AKA query_interval, a setting per PromQL query), if any is set, and otherwise the *Scrape interval* as set in the Prometheus data source (but ignoring any Min interval setting in the panel, because the latter is modified by the resolution setting).

Using variables in queries

There are two syntaxes:

- `$<varname>` Example: rate(http_requests_total{job=~"$job"}[5m])
- `[[varname]]` Example: rate(http_requests_total{job=~"[[job]]"}[5m])

Why two ways? The first syntax is easier to read and write but does not allow you to use a variable in the middle of a word. When the *Multi-value* or *Include all value* options are enabled, Grafana converts the labels from plaintext to a regex compatible string. Which means you have to use =~ instead of =.

Annotations

You can use annotations to overlay rich event information on top of graphs. You add annotation queries using the Dashboard menu or Annotations view. For more information, see Annotations.

Prometheus supports two ways to query annotations.

- A regular metric query
- A Prometheus query for pending and firing alerts. For more information, see Inspecting alerts during runtime).

The step option is useful to limit the number of events returned from your query.

Visualize alerts from Amazon Managed Service for Prometheus

> **Important**
>
> Starting in Amazon Managed Grafana version 12, SigV4 authentication support in the Core Prometheus plugin has been removed. All Amazon Managed Service for Prometheus data sources that were previously using the Core Prometheus plugin are automatically migrated to the Amazon Managed Service for Prometheus plugin. Any dashboards using these data sources are automatically updated to reflect this change. See Connect to an Amazon Managed Service for Prometheus data source.

You can visualize your Amazon Managed Service for Prometheus or Prometheus alerts in Amazon Managed Grafana by configuring an Alertmanager data source for Prometheus data sources that you are already connected to.

Prerequisites

To configure an Alertmanager for use with Amazon Managed Service for Prometheus, you must have the following pre-requisites complete:

- An Amazon Managed Service for Prometheus instance, with ingested metrics, and at least one alert or recording rule configured. You will need the URL for your workspace (from the details of your workspace in Amazon Managed Service for Prometheus you can see the **Endpoint** URL. The workspace URl is the Endpoint URL without the `api/v1/remote_write` at the end).
- An Amazon Managed Grafana workspace created with the Prometheus instance configured as a data source.
- Amazon Managed Grafana must have the following permissions for your Prometheus resources. You must add them to either the service-managed or customer-managed policies described in Amazon Managed Grafana permissions and policies for AWS data sources.
  - `aps:ListRules`
  - `aps:ListAlertManagerSilences`
  - `aps:ListAlertManagerAlerts`
  - `aps:GetAlertManagerStatus`
  - `aps:ListAlertManagerAlertGroups`
  - `aps:PutAlertManagerSilences`

- `aps:DeleteAlertManagerSilence`

To configure an Alertmanager data source for use with Amazon Managed Service for Prometheus

1. From your Grafana console, in the Grafana menu, choose the **Data source** page under **Configuration**.
2. Choose **Add data source**, and select **Alertmanager** from the list of data source types.
3. Provide the following information for your new data source.
   - For **Implementation**, choose **Prometheus**.
   - Under **HTTP**, for **URL**, provide the Prometheus workspace URL, with `alertmanager` appended. For example, `https://aps-workspaces.us-east1.amazonaws.com/workspaces/ws-example-1234-5678-abcd-xyz00000001/alertmanager`.
   - Under **Auth**, turn on **SigV4Auth**. This tells Grafana to use the AWS authentication for the requests.
   - Under **SigV4Auth Details**, for **Default Region**, provide the region of your Prometheus instance, for example `us-east-1`.
4. Choose **Save & test** to finish your data source setup.

   If your data source is set up correctly, you will see a message saying **Health check passed**.

To connect your new Alertmanager data source to the Prometheus data source

1. From your Grafana console, in the Grafana menu, choose the **Data source** page under **Configuration**.
2. Select your original Amazon Managed Service for Prometheus data source and turn on the **Manage alerts via Alerting UI** toggle switch.
3. Choose **Save & test** to finish configuring your data source.

Configure exemplars

Important

Starting in Amazon Managed Grafana version 12, SigV4 authentication support in the Core Prometheus plugin has been removed. All Amazon Managed Service for Prometheus data sources that were previously using the Core Prometheus plugin are automatically migrated to the Amazon Managed Service for Prometheus plugin. Any dashboards using these data sources are automatically updated to reflect this change. See Connect to an Amazon Managed Service for Prometheus data source.

Note

This feature requires Prometheus version 2.26 or later.
Exemplars are not supported in Amazon Managed Service for Prometheus.

You can show exemplars data alongside a metric both in Explore and Dashboards. Exemplars associate higher-cardinality metadata from a specific event with traditional time series data.

You can configure exemplars in the data source settings by adding links to your exemplars. You can use macros in your URL. For example, you could create a URL such as `https://example.com/${__value.raw}`.

Connect to an Amazon Managed Service for Prometheus data source

Note

The Amazon Managed Service for Prometheus data source is available starting in Amazon Managed Grafana version 12.

In Amazon Managed Grafana, you can use Amazon Managed Service for Prometheus workspaces as data sources. For more information about Amazon Managed Service for Prometheus, see What is Amazon Managed Service for Prometheus?

With Amazon Managed Grafana, you can add an Amazon Managed Service for Prometheus workspace as a data source by using the AWS data source configuration option in the Grafana

workspace console. This feature simplifies adding Amazon Managed Service for Prometheus as a data source by discovering your existing Amazon Managed Service for Prometheus accounts and manages the configuration of the authentication credentials that are required to access Amazon Managed Service for Prometheus.

> **Note**
>
> You can view your Prometheus alerts in the unified Grafana alerting interface, by Configuring an Alertmanager data source.

Topics

- Use AWS data source configuration to add Amazon Managed Service for Prometheus as a data source
- Using the Prometheus data source
- Visualize alerts from Amazon Managed Service for Prometheus

Use AWS data source configuration to add Amazon Managed Service for Prometheus as a data source

> **Note**
>
> The Amazon Managed Service for Prometheus data source is available starting in Amazon Managed Grafana version 12.

To use AWS data source configuration, first you use the Amazon Managed Grafana console to enable service-managed IAM roles that grant the workspace the IAM policies necessary to read the Amazon Managed Service for Prometheus resources in your account or in your entire organizational units. Then you use the Amazon Managed Grafana workspace console to add Amazon Managed Service for Prometheus as a data source.

To use AWS data source configuration to add Amazon Managed Service for Prometheus as a data source

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the upper left corner of the page, choose the menu icon and then choose **All workspaces**.

3. Choose the name of the workspace.
4. If you didn't choose to use service-managed permissions for this workspace when you created it, then change from using customer-managed permissions to use service-managed permissions to ensure that the proper IAM roles and policies are enabled for using the AWS data source configuration option in the Grafana workspace console. To do so, choose the edit icon by **IAM role** and then choose **Service managed**, **Save changes**. For more information, see Amazon Managed Grafana permissions and policies for AWS data sources.
5. Choose the **Data sources** tab. Then select the check box for **Amazon Managed Service for Prometheus**, and choose **Actions**, **Enable service-managed policy**.
6. Choose the **Data sources** tab again, and then choose **Configure in Grafana** in the **Amazon Managed Service for Prometheus** row.
7. Sign into the Grafana workspace console using IAM Identity Center if necessary.
8. In the left navigation bar in the Grafana workspace console, choose the **Connections** tab, and then choose the **Data sources** sub-tab.
9. Choose **Add new data source**, and then select **Amazon Managed Service for Prometheus**.
10. Configure the fields in the settings page.

Using the Prometheus data source

> **Note**
>
> The Amazon Managed Service for Prometheus data source is available starting in Amazon Managed Grafana version 12.

Prometheus settings

Name	Description
`Name`	The data source name. This is how you see the data source in panels and queries.
`Default`	Default data source means that it will be pre-selected for new panels.
`Url`	The URL of your Amazon Managed Service for Prometheus workspace; for example, `https://aps-workspaces.us-east-1.amazonaws.com/workspaces/ws-example1234/` .

Name	Description
Access	Server (default) = URL must be accessible from the Grafana backend/server.
Scrape interval	Set this to the typical scrape and evaluation interval configured in Prometheus. Defaults to 15s.
Disable metrics lookup	Checking this option will disable the metrics chooser and metric/label support in the query field's autocomplete. This helps if you have performance issues with bigger Prometheus instances.
Custom Query Parameters	Add custom parameters to the Prometheus query URL. For example `timeout`, `partial_response` , dedup, or `max_source_resolution` . Multiple parameters should be concatenated together with an "&".

Prometheus query editor

The following sections provide information and options for Prometheus query editor in the dashboard and in Explore.

Query editor in dashboards

Open a graph in edit mode by choosing the title and then choosing **Edit** (or by pressing **e** key while pausing on the panel).

N	Description
Q e n	For more information about Prometheus query expressions, see the Prometheus documentation.
L f	Controls the name of the time series, using name or pattern. For example `{{hostname}}` is replaced with the label value for the label `hostname`.
M s	An additional lower limit for the `step` parameter of Prometheus range queries and for the `$__interval` and `$__rate_interval` variables. The limit is absolute and not modified by the *Resolution* setting.

N	Description
R n	1/1 sets both the `$__interval` variable and the step parameter of Prometheus range queries such that each pixel corresponds to one data point. For better performance, use lower resolutions. 1/2 only retrieves a data point for every other pixel, and 1/10 retrieves one data point per 10 pixels. Note that both *Min time interval* and *Min step* limit the final value of `$__interval` and `step`.
M l	Search for metric names in this input field.
F a	Switch between `Table`, `Time series`, or `Heatmap`. `Table` works only in the table panel. `Heatmap` is suitable for displaying metrics of the histogram type on a heatmap panel. It converts cumulative histograms to regular ones and sorts series by the bucket bound.
I	Perform an "instant" query, to return only the latest value that Prometheus has scraped for the requested time series. Instant queries return results much faster than normal range queries. Use them to look up label sets.
M t i	This value multiplied by the denominator from the *Resolution* setting sets a lower limit to both the `$__interval` variable and the step parameter of Prometheus range queries. Defaults to *Scrape interval* as set in the data source options.

> **Note**
>
> Amazon Managed Grafana modifies the request dates for queries to align them with the dynamically calculated step. This ensures consistent display of metrics data, but it can result in a small gap of data at the right edge of a graph.

Instant queries in dashboards

The Prometheus data source allows you to run instant queries, which query only the latest value. You can visualize the results in a table panel to see all available labels of a time series.

Instant query results are made up of only one data point per series. They can be shown in the graph panel with the help of series overrides. To show them in the graph as a latest value point, add a series override and select `Points > true`. To show a horizontal line across the whole

graph, add a series override and select `Transform > constant` For more information about series overrides, see Series overrides.

Query editor in Explore

Na	Description
Qu ex n	For more information about Prometheus query expression, see the Prometheus documentation.
St	`Step` parameter of Prometheus range queries. Time units can be used here, for example: 5s, 1m, 3h, 1d, 1y. Default unit if no unit specified is `s` (seconds).
Qu ty	`Range`, `Instant`, or `Both`. When running **Range query**, the result of the query is displayed in graph and table. Instant query returns only the latest value that Prometheus has scraped for the requested time series and it is displayed in the table. When **Both** is selected, both instant query and range query is run. Result of range query is displayed in graph and the result of instant query is displayed in the table.

Metrics browser

The metrics browser allows you to quickly find metrics and select relevant labels to build basic queries. When you open the browser you will see all available metrics and labels. If supported by your Prometheus instance, each metric will show its HELP and TYPE as a tooltip.

When you select a metric, the browser narrows down the available labels to show only the ones applicable to the metric. You can then select one or more labels for which the available label values are shown in lists in the bottom section. Select one or more values for each label to tighten your query scope.

> **Note**
>
> If you do not remember a metric name to start with, you can also select a few labels first, to narrow down the list and then find relevant label values.

All lists in the metrics browser have a search field above them to quickly filter for metrics or labels that match a certain string. The values section only has one search field. Its filtering applies to all labels to help you find values across labels once they have been selected, for example, among your labels app, job, job_name only one might with the value you are looking for.

Once you are satisfied with your query, click "Use query" to run the query. The **Use as rate query** button adds a rate(...)[$__interval] around your query to help write queries for counter metrics. The "Validate selector" button will check with Prometheus how many time series are available for that selector.

Limitations

The metrics browser has a hard limit of 10,000 labels (keys) and 50,000 label values (including metric names). If your Prometheus instance returns more results, the browser will continue functioning. However, the result sets will be cut off above those maximum limits.

Templating

Instead of hardcoding things such as server, application and sensor name in your metric queries, you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. You can use these dropdown boxes to change the data being displayed in your dashboard.

For more information about templating and template variables, see Templates and variables.

Query variable

Variable of the type *Query* allows you to query Prometheus for a list of metrics, labels, or label values. The Prometheus data source plugin provides the following functions you can use in the **Query** input field.

Name	Description
`label_names()`	Returns a list of label names.
`label_values(label)`	Returns a list of label values for the `label` in every metric.
`label_values(metric, label)`	Returns a list of label values for the `label` in the specified metric.

Name	Description
`metrics(metric)`	Returns a list of metrics matching the specified `metric` regex.
`query_result(query)`	Returns a list of Prometheus query result for the `query`.

For information about what *metric names*, *label names* and *label values* are, see the Prometheus documentation.

Using interval and range variables

> **Note**
>
> Support for `$__range`, `$__range_s`, and `$__range_ms` are available only from Grafana v5.3.

You can use some global variables in query variables: `$__interval`, `$__interval_ms`, `$__range`, `$__range_s`, and `$__range_ms`. For more information, see Global variables. These can be convenient to use with the `query_result` function when you must filter variable queries because the `label_values` function doesn't support queries.

To get the correct instances when changing the time range on the dashboard, make sure to set the variable's `refresh` trigger to be `On Time Range Change`.

The following code example shows how to populate a variable with the busiest five request instances based on average QPS over the time range shown in the dashboard.

```
Query: query_result(topk(5, sum(rate(http_requests_total[$__range])) by (instance)))
Regex: /"([^"]+)"/
```

The following code example shows how to populate a variable with the instances having a certain state over the time range shown in the dashboard, using `$__range_s`.

```
Query: query_result(max_over_time(<metric>[${__range_s}s]) != <state>)
Regex:
```

Using `$__rate_interval` variable

The `$__rate_interval` variable is meant to be used in the rate function. It is defined as max(`$__interval` + *Scrape interval*, 4 * *Scrape interval*). *Scrape interval* is the Min step setting (AKA query_interval, a setting per PromQL query), if any is set, and otherwise the *Scrape interval* as set in the Prometheus data source (but ignoring any Min interval setting in the panel, because the latter is modified by the resolution setting).

Using variables in queries

There are two syntaxes:

- `$<varname>` Example: rate(http_requests_total{job=~"$job"}[5m])
- `[[varname]]` Example: rate(http_requests_total{job=~"[[job]]"}[5m])

Why two ways? The first syntax is easier to read and write but does not allow you to use a variable in the middle of a word. When the *Multi-value* or *Include all value* options are enabled, Grafana converts the labels from plaintext to a regex compatible string. Which means you have to use =~ instead of =.

Annotations

You can use annotations to overlay rich event information on top of graphs. You add annotation queries using the Dashboard menu or Annotations view. For more information, see Annotations.

Prometheus supports two ways to query annotations.

- A regular metric query
- A Prometheus query for pending and firing alerts. For more information, see Inspecting alerts during runtime).

The step option is useful to limit the number of events returned from your query.

Visualize alerts from Amazon Managed Service for Prometheus

> **Note**
>
> The Amazon Managed Service for Prometheus data source is available starting in Amazon Managed Grafana version 12.

You can visualize your Amazon Managed Service for Prometheus or Prometheus alerts in Amazon Managed Grafana by configuring an Alertmanager data source for Prometheus data sources that you are already connected to.

Prerequisites

To configure an Alertmanager for use with Amazon Managed Service for Prometheus, you must have the following pre-requisites complete:

- An Amazon Managed Service for Prometheus instance, with ingested metrics, and at least one alert or recording rule configured. You will need the URL for your workspace (from the details of your workspace in Amazon Managed Service for Prometheus you can see the **Endpoint** URL. The workspace URl is the Endpoint URL without the `api/v1/remote_write` at the end).
- An Amazon Managed Grafana workspace created with the Prometheus instance configured as a data source.
- Amazon Managed Grafana must have the following permissions for your Prometheus resources. You must add them to either the service-managed or customer-managed policies described in Amazon Managed Grafana permissions and policies for AWS data sources.
  - `aps:ListRules`
  - `aps:ListAlertManagerSilences`
  - `aps:ListAlertManagerAlerts`
  - `aps:GetAlertManagerStatus`
  - `aps:ListAlertManagerAlertGroups`
  - `aps:PutAlertManagerSilences`
  - `aps:DeleteAlertManagerSilence`

To configure an Alertmanager data source for use with Amazon Managed Service for Prometheus

1. From your Grafana console, in the Grafana menu, choose the **Data source** page under **Configuration**.
2. Choose **Add data source**, and select **Alertmanager** from the list of data source types.
3. Provide the following information for your new data source.
   - For **Implementation**, choose **Prometheus**.

- Under **HTTP**, for **URL**, provide the Prometheus workspace URL, with `alertmanager` appended. For example, `https://aps-workspaces.us-east1.amazonaws.com/workspaces/ws-example-1234-5678-abcd-xyz00000001/alertmanager`.
- Under **Auth**, turn on **SigV4Auth**. This tells Grafana to use the AWS authentication for the requests.
- Under **SigV4Auth Details**, for **Default Region**, provide the region of your Prometheus instance, for example `us-east-1`.

4. Choose **Save & test** to finish your data source setup.

   If your data source is set up correctly, you will see a message saying **Health check passed**.

To connect your new Alertmanager data source to the Prometheus data source

1. From your Grafana console, in the Grafana menu, choose the **Data source** page under **Configuration**.
2. Select your original Amazon Managed Service for Prometheus data source and turn on the **Manage alerts via Alerting UI** toggle switch.
3. Choose **Save & test** to finish configuring your data source.

Connect to an Amazon Timestream data source

> **Note**
>
> In workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

With Amazon Managed Grafana, you can add Amazon Timestream as a data source by using the AWS data source configuration option in the Grafana workspace console. This feature simplifies adding Timestream as a data source by discovering your existing Timestream accounts and manages the configuration of the authentication credentials that are required to access Timestream. You can use this method to set up authentication and add Timestream as a data source, or you can manually set up the data source and the necessary authentication credentials using the same method that you would on a self-managed Grafana server.

Use AWS data source configuration to add Timestream as a data source

To use AWS data source configuration, first you use the Amazon Managed Grafana console to enable service-managed IAM roles that grant the workspace the IAM policies necessary to read the Timestream resources in your account or in your entire organizational units. Then you use the Amazon Managed Grafana workspace console to add Timestream as a data source.

To use AWS data source configuration to add Timestream as a data source

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the upper left corner of the page, choose the menu icon and then choose **All workspaces**.
3. Choose the name of the workspace.
4. If you didn't choose to use service-managed permissions for this workspace when you created it, then change from using customer-managed permissions to use service-managed permissions to ensure that the proper IAM roles and policies are enabled for using the AWS data source configuration option in the Grafana workspace console. To do so, choose the edit icon by **IAM role** and then choose **Service managed**, **Save changes**. For more information, see Amazon Managed Grafana permissions and policies for AWS data sources.
5. Choose the **Data sources** tab. Then select the check box for **Amazon Timestream**, and choose **Actions**, **Enable service-managed policy**.
6. Choose the **Data sources** tab again, and then choose **Configure in Grafana** in the **Amazon Timestream** row.
7. Sign into the Grafana workspace console using IAM Identity Center if necessary.
8. In the left navigation bar in the Grafana workspace console, choose **Apps** then **AWS Data Sources** (in Grafana v8, choose the AWS icon from the left menu).
9. Choose the **AWS services** tab, then **Timestream**.
10. Select the default region that you want the **Timestream** data source to query from, select the accounts, and then choose **Add data source**.

Manually adding the Timestream data source

To manually add the Timestream data source

1. In the Grafana console side menu, pause on the **Configuration** (gear) icon, then choose **Data Sources**.
2. Choose **Add data source**.

3. Choose the **Amazon Timestream** data source. If necessary, you can start typing **`Timestream`** in the search box to help you find it.

Timestream settings

Name	Description
Name	The data source name. This is how you see the data source in panels and queries.
Auth Provider	Specify the provider to get credentials.
Default Region	Used in query editor to set region (can be changed on per query basis).
Credentials profile name	Specify the name of the profile to use (if you use `~/.aws/credentials` file), keep blank for default.
Assume Role Arn	Specify the ARN of the role to assume.
Endpoint (optional)	If you must specify an alternate service endpoint.

Authentication

This section covers the different types of authentication that you can use for the Amazon Timestream data source.

Example AWS credentials

You can't use the credentials file method of authentication in Amazon Managed Grafana.

Using the Timestream data source

Query editor

The query editor accepts Timestream syntax in addition to the macros listed previously and any dashboard template variables.

Press **Ctrl+Space** to open the IntelliSense suggestions.

Macros

To simplify syntax and to allow for dynamic parts, such as date range filters, the query can contain macros.

Macro example	Description
$__database	Will specify the selected database. This uses the default from the data source configuration, or the explicit value from the query editor.
$__table	Will specify the selected database. This uses the default from the datasource config, or the explicit value from the query editor.
$__measure	Will specify the selected measure. This uses the default from the datasource config, or the explicit value from the query editor.
$__timeFilter	Will be replaced by an expression that limits the time to the dashboard range
$__interval_ms	Will be replaced by a number that represents the amount of time a single pixel in the graph should cover.

Connect to an Amazon Athena data source

> **Note**
>
> In workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

> **Note**
>
> This guide assumes that you are familiar with the Amazon Athena service before you use the Athena data source.

With Amazon Managed Grafana, you can add Athena as a data source by using the AWS data source configuration option in the Grafana workspace console. This feature simplifies adding Athena as a data source by discovering your existing Athena accounts and manages the configuration of the authentication credentials that are required to access Athena. You can use this method to set up authentication and add Athena as a data source, or you can manually set up the data source and the necessary authentication credentials using the same method that you would on a self-managed Grafana server.

There are prerequisites for Athena to be accessible by Amazon Managed Grafana. For prerequisites associated with using the Athena data source, see Prerequisites.

Prerequisites

To use the managed policies for Amazon Managed Grafana for Athena, complete the following tasks before you configure the Athena data source:

- Tag your Athena work groups with `GrafanaDataSource: true`.
- Create an S3 bucket with a name that starts with `grafana-athena-query-results-`. This policy provides permissions for writing query results into an S3 bucket with that naming convention.

The Amazon S3 permissions for accessing the underlying data source of an Athena query are not included in this managed policy. You must add the necessary permissions for the Amazon S3 buckets manually, on a case-by-case basis. For more information, see Identity-based policy examples in Amazon Managed Grafana in this guide.

Use AWS data source configuration to add Amazon Athena as a data source

Prerequisites

- The AWS CLI is installed and configured in your environment.
- You have access to Athena from your account.

To use AWS data source configuration, first go to the Amazon Managed Grafana console to enable service-managed IAM roles that grant the workspace the IAM policies necessary to read the Athena resources in your account or in your entire organizational units. Then you use the Amazon Managed Grafana workspace console to add Athena as a data source.

To use AWS data source configuration to add Athena as a data source

1. Ensure that your user role is admin or editor.
2. Select the workspace that you want to work on from the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
3. If you didn't choose to use service-managed permissions for this workspace when you created it, then change from using customer-managed permissions to use service-managed permissions to ensure that the proper IAM roles and policies are enabled for using the AWS data source configuration option in the Grafana workspace console. To do so, choose the edit icon by **IAM role** and then choose **Service managed**, **Save changes**. For more information, see Amazon Managed Grafana permissions and policies for AWS data sources.
4. Choose the **Data sources** tab. Then select the check box for **Amazon Athena**, and choose **Actions**, **Enable service-managed policy**.
5. Choose the **Data sources** tab again, and then choose **Configure in Grafana** in the **Amazon Athena** row.
6. Sign into the Grafana workspace console using IAM Identity Center if necessary. The user should have Athena access policy attached to the user/role in order to have access to Athena data source. See AWS managed policy: AmazonGrafanaAthenaAccess for more info.
7. In the left navigation bar in the Grafana workspace console, choose the lower AWS icon (there are two) and then choose **Athena** from **Data sources** menu.
8. Select the default Region that you want the Athena data source to query from, and then select the accounts that you want, and then choose **Add data source**.
9. Follow the steps to configure **Athena Details** in **Athena Details** settings

Athena Details settings

Configure Athena Detailssettings

1. In **Connection Details** menu, select the authentication provider (recommended: **Workspace IAM Role**).

2. Select your targeted Athena data source where you have your Athena account with. If you do not select any data source, there is a default data source in the drop-down.

   To create a new Athena account, follow the instructions at Getting started with Athena

3. Select your targeted Athena database in the data source selected above.
4. Select the Workgroup. **Primary** is by default.
5. If your workgroup doesn't have an output location configured already, specify an S3 bucket and folder to use for query results. For example, `s3://grafana-athena-plugin-test-data/query-result-output/`
6. Select **Save & Test**.

Manually adding the Athena data source

Prerequisites

- The AWS CLI is installed and configured in your environment.
- You have access to **Amazon Athena** from your account.

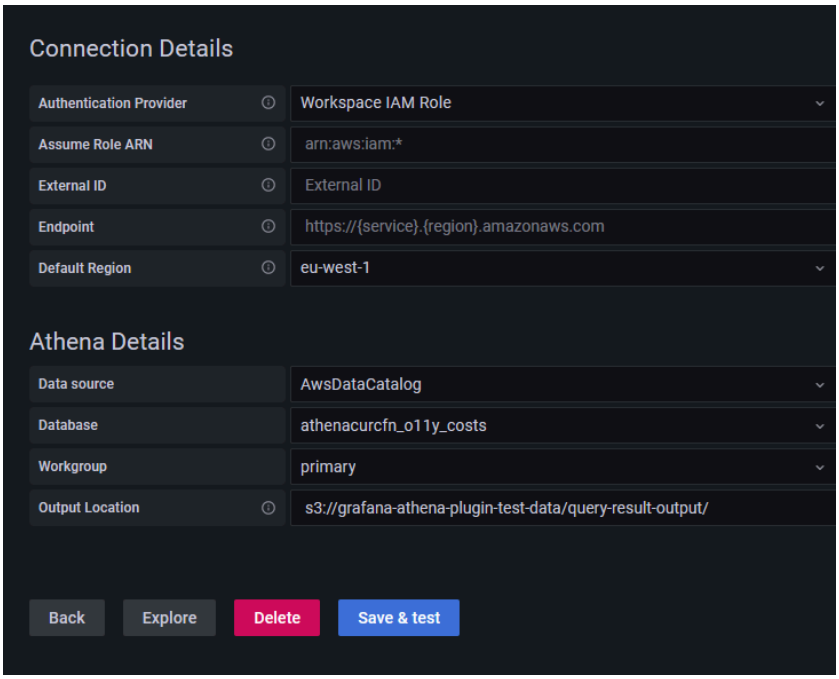
To manually add the Athena data source:

1. In the Grafana console side menu, pause on the **Configuration** (gear) icon, then choose **Data Sources**.
2. Choose **Add data source**.
3. Choose the **AWS Athena** data source. If necessary, you can start typing **`Athena`** in the search box to help you find it.
4. In **Connection Details** menu, configure the authentication provider (recommended: **Workspace IAM Role**)
5. Select your targeted Athena data source, database, and workgroup.

   To create a new Athena account, follow the instructions at Getting started with Athena.

6. If your workgroup doesn't have an output location configured already, specify an S3 bucket and folder to use for query results. For example, `s3://grafana-athena-plugin-test-data/query-result-output/` .
7. Select **Save & Test**.

The following is an example of the **Athena Details** settings.



Using Athena data source

IAM policies

Grafana needs permissions granted via IAM to be able to read Athena metrics. You can attach these permissions to IAM roles and utilize Grafana's built-in support for assuming roles. Note that you will need to Configure the required policy for your role before adding the data source to Grafana. You will need an admin or an editor role for adding a data source. The built-in Amazon Grafana Athena access policy is defined in the AWS managed policy: AmazonGrafanaAthenaAccess section.

Query Athena data

Athena data source provides a standard SQL query editor. Amazon Managed Grafana includes some macros to help with writing more complex timeseries queries.

Macros

Macro	Description	Example	Output Example
`$__dateFilter(column)`	`$__dateFilter` creates a conditional filter that selects the data (using `column`)	`$__date(my_date)`	`my_date BETWEEN date '2017-07-18' AND date '2017-07-18'`

Macro	Description	Example	Output Example
	based on the date range of the panel.		
`$__parseTime(column,format)`	`$__parseTime` casts a **varchar** as a **timestamp** with the given format.	`$__parseTime(eventtime, 'yyyy-MM-dd''T''HH:mm:ss''Z')`	`parse_datetime(time,'yyyy-MM-dd''T''HH:mm:ss''Z')`
`$__timeFilter(column,format)`	`$__timeFilter` creates a conditional that filters the data (using `column`) based on the time range of the panel. The second argument is used to optionally parse the column from a varchar to a timestamp with a specific format.	`$__timeFilter(time, 'yyyy-MM-dd HH:mm:ss')`	`TIMESTAMP time BETWEEN TIMESTAMP '2017-07-18T11:15:52Z' AND TIMESTAMP '2017-07-18T11:15:52Z'`
`$__timeFrom()`	`$__timeFrom` outputs the current starting time of the range of the panel with quotes.	`$__timeFrom()`	`TIMESTAMP '2017-07-18 11:15:52'`
`$__timeTo()`	`$__timeTo` outputs the current ending time of the range of the panel with quotes.	`$__timeTo()`	`TIMESTAMP '2017-07-18 11:15:52'`

Macro	Description	Example	Output Example
`$__timeGroup(column, '1m', format)`	`$__timeGroup` groups timestamps so that there is only 1 point for every period on the graph. The third argument is used to optionally parse the column from a varchar to a timestamp with a specific format.	`$__timeGroup(time, '5m','yyyy-MM-dd''T''HH:mm:ss.SSSSSS''Z')`	`FROM_UNIXTIME(FLOOR(TO_UNIXTIME(parse_datetime(time,'yyyy-MM-dd''T''HH:mm:ss.SSSSSS''Z'))/300)*300)`
`$__table`	`$__table` returns the table selected in the **Table** selector.	`$__table`	`my_table`
`$__column`	`$__column` returns the column selected in the **Column** selector (it requires a table).	`$__column`	`col1`

Visualization

Most queries in Athena will be best represented by a table visualization. A query displays return data in a table. If it can be queried, then it can be put displayed as a table.

This example returns results for a table visualization:

```
SELECT {column_1}, {column_2} FROM {table};
```

Timeseries/ Graph visualizations

For timeseries and graph visualizations, you must:

- select a column with a `date` or a `datetime` type. The `date` column must be in ascending order (using `ORDER BY column ASC`).
- also select a numeric column.

Inspecting the query

Amazon Managed Grafana supports macros that Athena does not, which means a query might not work when copied and pasted directly into Athena. To view the full interpolated query, which works directly in Athena, click the **Query Inspector** button. The full query is displayed under the **Query** tab.

Templates and variables

For more information about adding a Athena query variable, see Adding a query variable. Use your Athena data source as your data source for the available queries.

Any value queried from an Athena table can be used as a variable. Avoid selecting too many values, as this can cause performance issues.

After creating a variable, you can use it in your Athena queries by using Variable syntax. For more information about variables, see Templates and variables.

Annotations

Annotations allow you to overlay rich event information on top of graphs. You can add annotations by selecting the panel or by adding annotation queries using the **Dashboard** menu **Annotations** view.

An example query to automatically add annotations:

```
SELECT
  time as time,
  environment as tags,
  humidity as text
FROM
  tableName
WHERE
  $__dateFilter(time) and humidity > 95
```

The following table represents the descriptions of the columns that can be used to render annotations:

Name	Description
`Time`	The name of the date/time field. Could be a column with a native SQL date/time data type or epoch value.
`Timeend`	Optional name of the end date/time field. Could be a column with a native SQL date/time data type or epoch value. (Grafana v6.6+)
`Text`	Event description field.
`Tags`	Optional field name to use for event tags as a comma separated string.

Async query data support

Athena queries in Amazon Managed Grafana are handled in an asynchronous manner to avoid timeouts. Asynchronous queries use separate requests to start the query, then check on its progress, and finally to fetch the results. This avoids timeouts for queries that run for a long time.

Query result reuse

You can reuse the results of previous queries to improve query performance. To enable query reuse, enable is in the **Query result reuse** section of the query editor. This must be done for each query you want to reuse queries.

> **Note**
>
> This feature requires that your Athena instance be on engine version 3. For more information, see Changing Athena engine versions in the *Amazon Athena User Guide*.

Connect to an Amazon Redshift data source

> **Note**
>
> In workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

> **Note**
>
> This guide assumes that users are familiar with the Amazon Redshift service before using the Amazon Redshift data source.

With Amazon Managed Grafana, you can add Amazon Redshift as a data source by using the AWS data source configuration option in the Grafana workspace console. This feature simplifies adding Amazon Redshift as a data source by discovering your existing Amazon Redshift accounts and manages the configuration of the authentication credentials that are required to access Amazon Redshift. You can use this method to set up authentication and add Amazon Redshift as a data source, or you can manually set up the data source and the necessary authentication credentials using the same method that you would on a self-managed Grafana server.

There are prerequisites for Amazon Redshift to be accessible by Amazon Managed Grafana. For prerequisites associated with using the Amazon Redshift data source, see Prerequisites.

Prerequisites

To use the AWS managed policies for Amazon Managed Grafana, complete the following tasks before you configure the Amazon Redshift data source:

- Tag your Amazon Redshift cluster with `GrafanaDataSource: true`. Otherwise, it won't be accessible.
- Create the database credentials in one of the following mutually exclusive ways:
  - If you want to use the default mechanism (the temporary credentials options) to authenticate against the Redshift database, you must create a database user named `redshift_data_api_user`.
  - If you want to use the credentials from Secrets Manager, you must tag the secret with `RedshiftQueryOwner: true`. For more information, see Identity-based policy examples in Amazon Managed Grafanain this guide.

Use AWS data source configuration to add Amazon Redshift as a data source

To use AWS data source configuration to add Amazon Redshift as a data source

1. Ensure that your user role is admin or editor.

2. Select the workspace that you want to work on from the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
3. If you didn't choose to use service-managed permissions for this workspace when you created it, then change from using customer-managed permissions to use service-managed permissions to ensure that the proper IAM roles and policies are enabled for using the AWS data source configuration option in the Grafana workspace console. To do so, choose the edit icon by **IAM role** and then choose **Service managed**, **Save changes**. For more information, see Amazon Managed Grafana permissions and policies for AWS data sources.
4. Choose the **Data sources** tab. Then select the check box for **Amazon Redshift**, and choose **Actions**, **Enable service-managed policy**.
5. Choose the **Data sources** tab again, and then choose **Configure in Grafana** in the **Amazon Redshift** row.
6. Sign into the Grafana workspace console using IAM Identity Center if necessary.
7. In the left navigation bar in the Grafana workspace console, choose the lower AWS icon (there are two) and then choose **Redshift**.
8. Select the default region that you want the Amazon Redshift data source to query from, and then select the accounts that you want, and then choose **Add data source**.
9. Follow the steps to configure **Connection Details** in **Connection details** settings.

Manually adding the Amazon Redshift data source

Prerequisites

- You have access to **Amazon Redshift** from your account.

To add the Amazon Redshift data source:

1. Attach the AmazonRedshiftAccessPolicy to your workspace user role.
2. Ensure your user role is admin or editor.
3. Select the workspace you want to work on from the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
4. In the Grafana console side menu, pause on the **Configuration** (gear) icon, then choose **Data Sources**.
5. Choose **Add data source**.

6. Choose the **AWS Redshift** data source. If necessary, you can start typing **`Redshift`** in the search box to help you find it.
7. This opens the **Connection Details** page. Follow the steps in configuring the **Connection details** settings.

Configuring Amazon Redshift

After adding your Amazon Redshift data source to your workspace, configure Amazon Redshift settings as the following:

Prerequisites

- You have access to **Amazon Redshift** from your account.

Connection details settings

Configure Connection details settings

1. In the **Connection Details** menu, select the authentication provider (recommended: **Workspace IAM Role**).
2. Choose the **Default Region** you want to query.

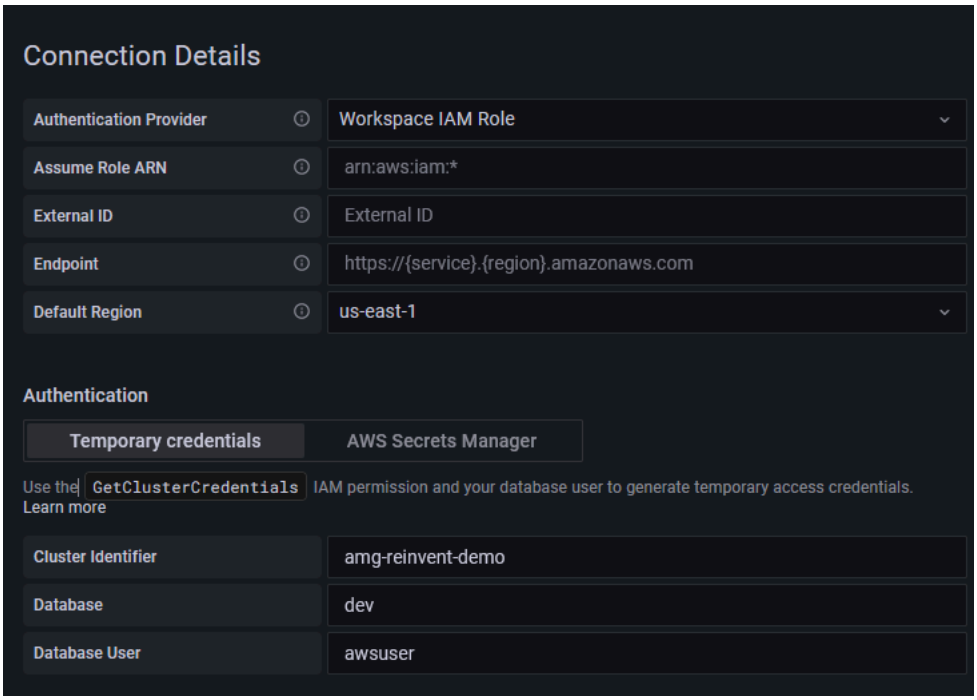
Authentication settings

Configure Authentication settings

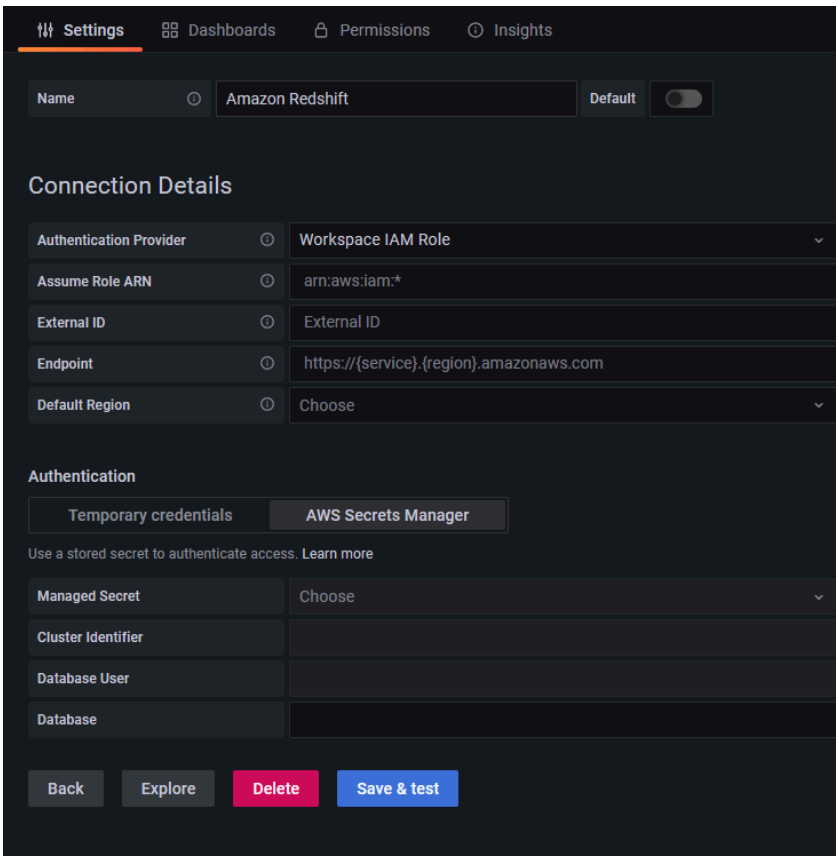
1. In the **Authentication** menu, choose either the **Temporary Credentials** or **AWS Secrets Manager** tab as your access credentials provider. For details on Temporary Credentials and AWS Secrets Manager, refer to AWS managed policy: AmazonGrafanaRedshiftAccess
2. If you choose **Temporary credentials** which is by default, follow the steps below. If you choose **AWS Secrets Manager**, enter your **AWS Secrets Manager** credentials in the input fields.
3. Choose the **Cluster Identifier** of the cluster you created in Amazon Redshift.

   For more information about the Redshift cluster, see Redshift connections.
4. Choose your targeted Redshift database.
5. Select the database user you created for the above cluster.
6. Choose **Save & Test**.

The following is an example of the **Temporary Credentials** settings.



The following is an example of the **AWS Secrets Manager** menu.



Using the Amazon Redshift data source

IAM policies

Grafana needs permissions granted using IAM to be able to read Redshift metrics. You can attach these permissions to IAM roles and utilize Grafana's built-in support for assuming roles. The built-in Amazon Grafana Redshift access policy is defined in the AWS managed policy: AmazonGrafanaRedshiftAccess section.

Query Amazon Redshift data

Amazon Redshift data source provides a standard SQL query editor. Amazon Managed Grafana includes some macros to help with writing more complex timeseries queries.

Macros

Macro	Description	Output example
`$__timeEpoch(column)`	`$__timeEpoch` will be replaced by an expression to convert to a UNIX timestamp and rename the column to time	`UNIX_TIMESTAMP(dateColumn) as "time"`
`$__timeFilter(column)`	`$__timeFilter` creates a conditional that filters the data (using `column`) based on the time range of the panel	`time BETWEEN '2017-07-18T11:15:52Z' AND '2017-07-18T11:15:52Z'`
`$__timeFrom()`	`$__timeFrom` outputs the current starting time of the range of the panel with quotes	`'2017-07-18T11:15:52Z'`
`$__timeTo()`	`$__timeTo` outputs the current ending time of the range of the panel with quotes	`'2017-07-18T11:15:52Z'`
`$__timeGroup(column, '1m')`	`$__timeGroup` groups timestamps so that there	`floor(extract(epoch from time)/60)*60 AS "time"`

Macro	Description	Output example
	is only 1 point for every period on the graph	
`$__schema`	`$__schema` uses the selected schema	`public`
`$__table`	`$__table` outputs a table from the given `$__schema` (it uses the public schema by default)	`sales`
`$__column`	`$__column` outputs a column from the current `$__table`	`date`
`$__unixEpochFilter(column)`	`$__unixEpochFilter` be replaced by a time range filter using the specified column name with times represented as Unix timestamp	`column >= 1624406400 AND column <= 1624410000`
`$__unixEpochGroup(column)`	`$__unixEpochGroup` is the same as `$__timeGroup` but for times stored as Unix timestamp	`floor(time/60)*60 AS "time"`

Visualization

Most queries in Redshift are best represented by a table visualization. Any query will display data in a table. If it can be queried, then it can be put in a table.

This example returns results for a table visualization:

```
SELECT {column_1}, {column_2} FROM {table};
```

Time series and graph visualizations

For time series and graph visualizations, there are a few requirements:

- A column with a `date` or a `datetime` type must be selected.

- The `date` column must be in ascending order (using `ORDER BY column ASC`).
- You must select a numeric column.

To make a more reasonable graph, be sure to use the `$__timeFilter` and `$__timeGroup` macros.

Example timeseries query:

```
SELECT
  avg(execution_time) AS average_execution_time,
  $__timeGroup(start_time, 'hour'),
  query_type
FROM
  account_usage.query_history
WHERE
  $__timeFilter(start_time)
group by
  query_type,start_time
order by
  start_time,query_type ASC;
```

Fill mode

Grafana also autocompletes frames without a value with some default. To configure this value, change the **Fill Value** in the query editor.

Inspecting the query

Because Grafana supports macros that Redshift does not, the fully rendered query, which can be copied and pasted directly into Redshift, is visible in the Query Inspector. To view the full interpolated query, choose the **Query Inspector** menu, and the full query is visible on the **Query** tab.

Templates and variables

For more information about how to add a new Redshift query varialble, see Adding a query variable. Use your Redshift data source as your data source for the available queries.

Any value queried from a Amazon Redshift table can be used as a variable. Be sure to avoid selecting too many values, as this can cause performance issues.

After creating a variable, you can use it in your Redshift queries by using Variable syntax. For more information about variables, see Templates and variables.

Annotations

Annotations allows you to overlay rich event information on top of graphs. You can add annotations by selecting the panel or by adding annotation queries using the **Annotations** view, opened from the **Dashboard** menu.

Example query to automatically add annotations:

```
SELECT
  time as time,
  environment as tags,
  humidity as text
FROM
  $__table
WHERE
  $__timeFilter(time) and humidity > 95
```

The following table represents the values of the columns taken into account to render annotations:

Name	Description
`Time`	The name of the date or time field. Could be a column with a native SQL date or time data type or epoch value.
`Timeend`	Optional name of the end date or time field. Could be a column with a native SQL dateor time data type or epoch value.
`Text`	Event description field.
`Tags`	Optional field name to use for event tags as a comma separated string.

Connect to an AWS X-Ray data source

> **Note**
>
> In workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

Add AWS X-Ray as a data source, and then build dashboards or use Explore with X-Ray to look at traces, analytics, or insights.

With Amazon Managed Grafana, you can add X-Ray as a data source by using the AWS data source configuration option in the Grafana workspace console. This feature simplifies adding X-Ray as a data source by discovering your existing X-Ray accounts and manages the configuration of the authentication credentials that are required to access X-Ray. You can use this method to set up authentication and add X-Ray as a data source, or you can manually set up the data source and the necessary authentication credentials using the same method that you would on a self-managed Grafana server.

Topics

- Use AWS data source configuration to add X-Ray as a data source
- Manually adding the X-Ray data source
- X-Ray settings
- Using the X-Ray data source

Use AWS data source configuration to add X-Ray as a data source

To use AWS data source configuration, first you use the Amazon Managed Grafana console to enable service-mananged IAM roles that grant the workspace the IAM policies necessary to read the X-Ray resources in your account or in your entire organizational units. Then you use the Amazon Managed Grafana workspace console to add X-Ray as a data source.

To use AWS data source configuration to add X-Ray as a data source

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the upper left corner of the page, choose the menu icon and then choose **All workspaces**.
3. Choose the name of the workspace.

4. If you didn't choose to use service-managed permissions for this workspace when you created it, then change from using customer-managed permissions to use service-managed permissions to ensure that the proper IAM roles and policies are enabled for using the AWS data source configuration option in the Grafana workspace console. To do so, choose the edit icon by **IAM role** and then choose **Service managed**, **Save changes**. For more information, see Amazon Managed Grafana permissions and policies for AWS data sources.
5. Choose the **Data sources** tab. Then select the check box for **AWS X-Ray**, and choose **Actions**, **Enable service-managed policy**.
6. Choose the **Data sources** tab again, and then choose **Configure in Grafana** in the **AWS X-Ray** row.
7. Sign into the Grafana workspace console using IAM Identity Center if necessary.
8. In the left navigation bar in the Grafana workspace console, choose the AWS icon and then choose**AWS services**, **X-Ray**.
9. Select the default Region that you want the X-Ray data source to query from, select the accounts, and then choose **Add data source**.

Manually adding the X-Ray data source

To manually add the X-Ray data source

1. In the Grafana console side menu, pause on the **Configuration** (gear) icon, then choose **Data Sources**.
2. Choose **Add data source**.
3. Choose the **X-Ray** data source. If necessary, you can start typing **`X-Ray`** in the search box to help you find it.

X-Ray settings

Name	Description
Name	The data source name. This is how you see the data source in panels and queries.
Default	Default data source means that it will be pre-selected for new panels.

Name	Description
Default Region	Used in query editor to set region (can be changed on per query basis).
Auth Provider	Specify the provider to get credentials.
Credentials profile name	Specify the name of the profile to use (if you use `~/.aws/credentials` file), keep blank for default.
Assume Role Arn	Specify the ARN of the role to assume.
External ID	If you are assuming a role in another account, that has been created with an external ID, specify the external ID here.

Authentication

This section covers the different types of authentication that you can use for X-Ray data source.

IAM roles

Currently, all access to X-Ray is done server side by the Grafana workspace backend using the official AWS SDK. If your Grafana server is running on AWS, you can use IAM roles and authentication will be handled automatically.

For more information, see IAM roles.

IAM policies

Grafana needs permissions granted via IAM to be able to read X-Ray data and EC2 tags/instances/regions. You can attach these permissions to IAM roles and use the built-in Grafana support for assuming roles.

The following code example shows a minimal policy.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:BatchGetTraces",
                "xray:GetTraceSummaries",
                "xray:GetTraceGraph",
                "xray:GetGroups",
                "xray:GetTimeSeriesServiceStatistics",
                "xray:GetInsightSummaries",
                "xray:GetInsight",
                "ec2:DescribeRegions"
            ],
            "Resource": "*"
        }
    ]
}
```

Example AWS credentials

You can't use the credentials file method in Amazon Managed Grafana.

Using the X-Ray data source

Query editor

The most important field in the editor is the query type. There are four query types:

- Trace List (Traces in AWS)
- Trace Statistics
- Trace Analytics (Analytics in AWS)
- Insights

Trace List

The Trace List type allows you to search for traces, which are shown in a table. Choosing the trace ID in the first column opens the trace on the right side. Notice the query field in the editor. You can write queries, filter expressions, or you can insert a single trace ID that will be shown in a trace view. For more details about filter expressions, see the AWS X-Ray documentation.

> **Note**
>
> The trace list will show only the first 1000 traces.

Trace Statistics

In Trace Statistics, you can see a graph and a table showing information about error, fault, throttle, success, and total count. You can use the columns field in the query editor to see only specified columns.

Trace Analytics

In Trace Analytics, you can visualize the following tables.

- Root Cause
  - Response Time
    - Root Cause Service (Last service in path)
    - Path (multiple paths)
  - Error
    - Root Cause Service (Last service in path)
    - Path
    - Error Message
  - Fault
    - Root Cause Service (Last service in path)
    - Path
    - Error Message
- End-user Impact
- URL
- HTTP Status Code

Insights

In Insights, you can see the summary table for Insights. Choosing the InsightId will take you to AWS Management console.

Alerting

Because X-Ray queries can return numeric data, alerts are supported. For more information, see Grafana alerting.

Connect to an Azure Monitor data source

The Azure Monitor data source supports multiple services in the Azure cloud:

- **Azure Monitor service** is the platform service that provides a single source for monitoring Azure resources. For more information, see Querying the Azure Monitor service.
- **Application Insights server** is an extensible Application Performance Management (APM) service for web developers on multiple platforms and can be used to monitor your live web application - it will automatically detect performance anomalies. For more information, see Querying the Application Insights Analytics service.
- **Azure Log Analytics** (or Azure Logs) gives you access to log data collected by Azure Monitor. For more information, see Querying the Azure Log Analytics service.
- Use the **Application Insights Analytics service** to query Application Insights data using the same query language used for Azure Log Analytics. For more information, see Querying the Application Insights Analytics service.

Adding the data source

The data source can access metrics from four different services. You can configure access to the services that you use. It is also possible to use the same credentials for multiple services if that is how you have set it up in Azure Entra ID.

- Register a Microsoft Entra app and create a service principal

1. Accessed from the Grafana main menu, newly installed data sources can be added immediately within the Data Sources section. Next, choose the **Add data source** button in the upper right. The Azure Monitor data source will be available for selection in the Cloud section in the list of data sources.
2. In the name field, Grafana will automatically fill in a name for the data source: `Azure Monitor` or something such as `Azure Monitor - 3`. If you are configuring multiple data sources, change the name to something more informative.

3. If you are using Azure Monitor, you need four pieces of information from the Azure portal (for detailed instructions, see the link provided earlier):
   - **Tenant Id** (Azure Entra ID, Properties, Directory ID)
   - **Client Id** (Azure Entra ID, App Registrations, Choose your app, Application ID)
   - **Client Secret** (Azure Entra ID, App Registrations, Choose your app, Keys)
   - **Default Subscription Id** (Subscriptions, Choose subscription, Overview, Subscription ID)
4. Paste these four items into the fields in the Azure Monitor API Details section.
   - The Subscription Id can be changed per query. Save the data source and refresh the page to see the list of subscriptions available for the specified Client Id.
5. If you are also using the Azure Log Analytics service, you must specify these two configuration values or reuse the Client Id and Secret from the previous step.
   - Client Id (Azure Entra ID, App Registrations, Choose your app, Application ID)
   - Client Secret (Azure Entra ID, App Registrations, Choose your app, Keys, Create a key, Use client secret)
6. If you are using Application Insights, you need two pieces of information from the Azure Portal (for detailed instructions, see the link provided earlier):
   - Application ID
   - API Key
7. Paste these two items into the appropriate fields in the Application Insights API Details section.
8. Test that the configuration details are correct by choosing the **Save & Test** button.

Alternatively on step 4, if you are creating a new Azure Entra ID App, use the Azure CLI:

```
az ad sp create-for-rbac -n "http://localhost:3000"
```

Choosing a service

In the query editor for a panel, after you choose your Azure Monitor data source, the first step is to select a service. There are four options:

- `Azure Monitor`

- `Application Insights`
- `Azure Log Analytics`
- `Insights Analytics`

The query editor changes depending on which option you select. Azure Monitor is the default.

Querying the Azure Monitor service

The Azure Monitor service provides metrics for all the Azure services that you have running. It helps you understand how your applications on Azure are performing, andit proactively finds issues affecting your applications.

If your Azure Monitor credentials give you access to multiple subscriptions, choose the appropriate subscription first.

Examples of metrics that you can get from the service are:

- `Microsoft.Compute/virtualMachines - Percentage CPU`
- `Microsoft.Network/networkInterfaces - Bytes sent`
- `Microsoft.Storage/storageAccounts - Used Capacity`

The query editor allows you to query multiple dimensions for metrics that support them. Metrics that support multiple dimensions are those listed in the Azure Monitor supported Metrics List that have one or more values listed in the **Dimension** column for the metric.

Formatting legend keys with aliases for Azure Monitor

The default legend formatting for the Azure Monitor API is:

```
metricName{dimensionName=dimensionValue,dimensionTwoName=DimensionTwoValue}
```

These can be long, but you can change this formatting by using aliases. In the **Legend Format** field, you can combine the following aliases any way that you want.

Azure Monitor examples:

- `Blob Type: {{ blobtype }}`
- `{{ resourcegroup }} - {{ resourcename }}`

Alias patterns for Azure Monitor

- `{{ resourcegroup }}` = replaced with the value of the Resource Group
- `{{ namespace }}` = replaced with the value of the Namespace (for example, Microsoft.Compute/virtualMachines)
- `{{ resourcename }}` = replaced with the value of the Resource Name
- `{{ metric }}` = replaced with metric name (for example, Percentage CPU)
- `{{ dimensionname }}` = *Legacy as of 7.1+ (for backwards compatibility)* replaced with the first dimension's key/label (as sorted by the key/label) (for example, blobtype)
- `{{ dimensionvalue }}` = *Legacy as of 7.1+ (for backwards compatibility)* replaced with first dimension's value (as sorted by the key/label) (for example, BlockBlob)
- `{{ arbitraryDim }}` = *Available in 7.1+* replaced with the value of the corresponding dimension. (for example, `{{ blobtype }}` becomes BlockBlob)

Creating template variables for Azure Monitor

Instead of hardcoding things such as server, application and sensor name in your metric queries you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. You can use these dropdown boxes to change the data being displayed in your dashboard.

Note that the Azure Monitor service does not support multiple values yet. To visualize multiple time series (for example, metrics for server1 and server2), add multiple queries so that you can view them on the same graph or in the same table.

The Azure Monitor data source plugin provides the following queries that you can specify in the **Query** field in the Variable edit view. You can use them to fill a variable's options list.

Name	Description
`Subscriptions()`	Returns a list of subscriptions.
`ResourceGroups()`	Returns a list of resource groups.
`ResourceGroups(12345678-aaaa-bbbb-cccc-123456789aaa)`	Returns a list of resource groups for a specified subscription.

Name	Description
`Namespaces(aResourceGroup)`	Returns a list of namespaces for the specified resource group.
`Namespaces(12345678-aaaa-bbbb-cccc-123456789aaa, aResourceGroup)`	Returns a list of namespaces for the specified resource group and subscription.
`ResourceNames(aResourceGroup, aNamespace)`	Returns a list of resource names.
`ResourceNames(12345678-aaaa-bbbb-cccc-123456789aaa, aResourceGroup, aNamespace)`	Returns a list of resource names for a specified subscription.
`MetricNamespace(aResourceGroup, aNamespace, aResourceName)`	Returns a list of metric namespaces.
`MetricNamespace(12345678-aaaa-bbbb-cccc-123456789aaa, aResourceGroup, aNamespace, aResourceName)`	Returns a list of metric namespaces for a specified subscription.
`MetricNames(aResourceGroup, aNamespace, aResourceName)`	Returns a list of metric names.
`MetricNames(12345678-aaaa-bbbb-cccc-123456789aaa, aResourceGroup, aNamespace, aResourceName)`	Returns a list of metric names for a specified subscription.

Examples:

- Resource Groups query: `ResourceGroups()`
- Passing in metric name variable: `Namespaces(cosmo)`
- Chaining template variables: `ResourceNames($rg, $ns)`
- Do not quote parameters: `MetricNames(hg, Microsoft.Network/publicIPAddresses, grafanaIP)`

For more information about templating and template variables, see Templates.

List of supported Azure Monitor metrics

Not all metrics returned by the Azure Monitor API have values. To make building a query easier, the Grafana data source has a list of supported Azure Monitor metrics, and it ignores metrics that will never have values. This list is updated regularly as new services and metrics are added to the Azure cloud.

Azure Monitor alerting

Grafana alerting is supported for the Azure Monitor service. This is not Azure Alerts support. For more information about Grafana alerting, see Grafana alerting.

Querying the Application Insights service

Formatting legend keys with aliases for Application Insights

The default legend formatting is:

```
metricName{dimensionName=dimensionValue,dimensionTwoName=DimensionTwoValue}
```

In the Legend Format field, the following aliases can be combined any way you want.

Application Insights examples:

- `city: {{ client/city }}`
- `{{ metric }} [Location: {{ client/countryOrRegion }}, {{ client/city }}]`

Alias patterns for Application Insights

- `{{ groupbyvalue }}` = *Legacy as of Grafana 7.1+ (for backwards compatibility)* replaced with the first dimension's key/label (as sorted by the key/label)
- `{{ groupbyname }}` = *Legacy as of Grafana 7.1+ (for backwards compatibility)* replaced with first dimension's value (as sorted by the key/label) (for example, BlockBlob)
- `{{ metric }}` = replaced with metric name (for example, requests/count)
- `{{ arbitraryDim }}` = *Available in 7.1+* replaced with the value of the corresponding dimension. (for example, `{{ client/city }}` becomes Chicago)

Filter expressions for Application Insights

The filter field takes an OData filter expression.

Examples:

- `client/city eq 'Boydton'`
- `client/city ne 'Boydton'`
- `client/city ne 'Boydton' and client/city ne 'Dublin'`
- `client/city eq 'Boydton' or client/city eq 'Dublin'`

Templating with variables for Application Insights

Use the one of the following queries in the **Query** field in the Variable edit view.

For more information about templating and template variables, see Templates.

Name	Description
`AppInsightsMetricNames()`	Returns a list of metric names.
`AppInsightsGroupBys(aMetricName)`	Returns a list of `group by` clauses for the specified metric name.

Examples:

- Metric Names query: `AppInsightsMetricNames()`
- Passing in metric name variable: `AppInsightsGroupBys(requests/count)`
- Chaining template variables: `AppInsightsGroupBys($metricnames)`

Application Insights alerting

Grafana alerting is supported for Application Insights. This is not Azure Alerts support. For more information about Grafana alerting, see Grafana alerting.

Querying the Azure Log Analytics service

Queries are written in the new Azure Log Analytics (or KustoDB) Query Language. A Log Analytics query can be formatted as time series data or as table data.

If your credentials give you access to multiple subscriptions, then choose the appropriate subscription before entering queries.

Time series queries

Time series queries are for the graph panel and other panels such as the SingleStat panel. Each query must contain at least a datetime column and a numeric value column. The result must be sorted in ascending order by the datetime column.

The following code example shows a query that returns the aggregated count grouped by hour.

```
Perf
| where $__timeFilter(TimeGenerated)
| summarize count() by bin(TimeGenerated, 1h)
| order by TimeGenerated asc
```

A query can also have one or more non-numeric/non-datetime columns, and those columns are considered dimensions and become labels in the response. For example, a query that returns the aggregated count grouped by hour, Computer, and the CounterName.

```
Perf
| where $__timeFilter(TimeGenerated)
| summarize count() by bin(TimeGenerated, 1h), Computer, CounterName
| order by TimeGenerated asc
```

You can also select additional number value columns (with, or without multiple dimensions). For example, getting a count and average value by hour, Computer, CounterName, and InstanceName:

```
Perf
| where $__timeFilter(TimeGenerated)
| summarize Samples=count(), ["Avg Value"]=avg(CounterValue)
    by bin(TimeGenerated, $__interval), Computer, CounterName, InstanceName
```

```
| order by TimeGenerated asc
```

> **Note**
>
> **Tip**: In the previous query, the Kusto syntax and `Samples=count()["Avg Value"]=...` are used to rename those columns — the second syntax allowing for the space. This changes the name of the metric that Grafana uses. As a result, things such as series legends and table columns will match what you specify. In this example, `Samples` is displayed instead of `_count`.

Table queries

Table queries are mainly used in the table panel, and they show a list of columns and rows. This example query returns rows with the six specified columns.

```
AzureActivity
| where $__timeFilter()
| project TimeGenerated, ResourceGroup, Category, OperationName, ActivityStatus, Caller
| order by TimeGenerated desc
```

Formatting the display name for Log Analytics

The default display name format is:

`metricName{dimensionName=dimensionValue,dimensionTwoName=DimensionTwoValue}`

This can be customized by using the display name field option.

Azure Log Analytics macros

To make writing queries easier, Grafana provides several macros that you can use in the where clause of a query:

- `$__timeFilter()` – Expands to `TimeGenerated ≥ datetime(2018-06-05T18:09:58.907Z)` and `TimeGenerated ≤ datetime(2018-06-05T20:09:58.907Z)` where the from and to datetimes are from the Grafana time picker.

- `$__timeFilter(datetimeColumn)` – Expands to `datetimeColumn ≥ datetime(2018-06-05T18:09:58.907Z) and datetimeColumn ≤ datetime(2018-06-05T20:09:58.907Z)` where the from and to datetimes are from the Grafana time picker.
- `$__timeFrom()` – Returns the From datetime from the Grafana picker. Example: `datetime(2018-06-05T18:09:58.907Z)`.
- `$__timeTo()` – Returns the From datetime from the Grafana picker. Example: `datetime(2018-06-05T20:09:58.907Z)`.
- `$__escapeMulti($myVar)` – is to be used with multi-value template variables that contain illegal characters. If `$myVar` has the following two values as a string `'\\grafana-vm\Network(eth0)\Total','\\hello!'`, then it expands to: `@'\\grafana-vm \Network(eth0)\Total', @'\\hello!'`. If using single value variables there is no need for this macro, escape the variable inline instead: `@'\$myVar'`.
- `$__contains(colName, $myVar)` – is to be used with multi-value template variables. If `$myVar` has the value `'value1','value2'`, it expands to: `colName in ('value1','value2')`.

  If using the **All** option, check the **Include All Option** check box and in the **Custom all value** field, enter the following value: **`all`**. If `$myVar` has the value `all`, the macro will instead expand to `1 == 1`. For template variables with numerous options, this increases the query performance by not building a large "where..in" clause.

Azure Log Analytics built-in variables

There are also some Grafana variables that can be used in Azure Log Analytics queries:

- `$__interval` - Grafana calculates the minimum time grain that can be used to group by time in queries. It returns a time grain such as `5m` or `1h` that can be used in the bin function; for example, `summarize count() by bin(TimeGenerated, $__interval)`. For more information about interval variables, see Adding an interval variable.

Templating with variables for Azure Log Analytics

Any Log Analytics query that returns a list of values can be used in the **Query** field in the Variable edit view. There is also one Grafana function for Log Analytics that returns a list of workspaces.

For information about templates and template variables, see Templates and variables.

Name	Description
`workspaces()`	Returns a list of workspaces for the default subscription.
`workspaces(12345678-aaaa-bbbb-cccc-123456789aaa)`	Returns a list of workspaces for the specified subscription (the parameter can be quoted or unquoted).

The following table shows example variable queries.

Query	Description
`subscriptions()`	Returns a list of Azure subscriptions.
`workspaces()`	Returns a list of workspaces for default subscription.
`workspaces("12345678-aaaa-bbbb-cccc-123456789aaa")`	Returns a list of workspaces for a specified subscription.
`workspaces("$subscription")`	With template variable for the subscription parameter.
`workspace("myWorkspace").Heartbeat | distinct Computer`	Returns a list of virtual machines.
`workspace("$workspace").Heartbeat | distinct Computer`	Returns a list of virtual machines with template variable.
`workspace("$workspace").Perf | distinct ObjectName`	Returns a list of objects from the Perf table.
`workspace("$workspace").Perf \| where ObjectName == "$object" | distinct CounterName`	Returns a list of metric names from the Perf table.

The following code xample shows a time series query using variables.

```
Perf
| where ObjectName == "$object" and CounterName == "$metric"
| where TimeGenerated >= $__timeFrom() and TimeGenerated <= $__timeTo()
| where  $__contains(Computer, $computer)
| summarize avg(CounterValue) by bin(TimeGenerated, $__interval), Computer
| order by TimeGenerated asc
```

Deep linking from Grafana panels to the Log Analytics query editor in Azure Portal

Choose a time series in the panel to see a context menu with a link to **View in Azure Portal**. Choosing that link opens the Azure Log Analytics query editor in the Azure Portal and runs the query from the Grafana panel there.

If you're not currently logged in to the Azure Portal, then the link opens the login page. The provided link is valid for any account, but it only displays the query if your account has access to the Azure Log Analytics workspace specified in the query.

Azure Log Analytics alerting

Grafana alerting is supported for Application Insights. This is not Azure Alerts support. For more information about alerting in Grafana workspaces, see Grafana alerting.

Querying the Application Insights Analytics service

If you change the service type to **Insights Analytics**, then a similar editor to the Log Analytics service is available. This service also uses the Kusto language, so the instructions for querying data are identical to Querying the Azure Log Analytics service, except that you query Application Insights Analytics data instead.

Connect to a Graphite data source

Grafana has an advanced Graphite query editor that lets you quickly navigate the metric space, add functions, change function parameters and much more. The editor can handle all types of graphite queries. It can even handle complex nested queries through the use of query references.

Graphite settings

To access Graphite settings, pause on the **Configuration** (gear) icon, then choose **Data Sources**, and then choose the Graphite data source.

Name	Description
Name	The data source name. This is how you see the data source in panels and queries.
Default	Default data source means that it will be pre-selected for new panels.
URL	The HTTP protocol, IP, and port of your graphite-web or graphite-api install.
Access	Server (default) = URL must be accessible from the Grafana backend/server.
Auth	
Basic Auth	Enable basic authentication to the data source.
User	User name for basic authentic ation.
Password	Password for basic authentication.
Custom HTTP Headers	Choose **Add header** to add a custom HTTP header.
Header	Enter the custom header name.
Value	Enter the custom header value.
Graphite details	
Version	Select your version of Graphite.
Type	Select your type of Graphite.

Access mode controls how requests to the data source will be handled. Server should be the preferred way if nothing else is stated.

Server access mode (default)

All requests are made from the browser to Amazon Managed Grafana, which forwards the requests to the data source, circumventing possible Cross-Origin Resource Sharing (CORS) requirements. If you select this access mode, the URL must be accessible from Amazon Managed Grafana.

Browser access mode

Amazon Managed Grafana does not support browser direct access for the Graphite data source.

Graphite query editor

Grafana includes a Graphite-specific query editor to help you build your queries.

To see the raw text of the query that is sent to Graphite, choose the **Toggle text edit mode** (pencil) icon.

Choosing metrics to query

Choose **Select metric** to navigate the metric space. After you start, you can continue using the pointer or keyboard arrow keys. You can select a wildcard character and still continue.

Functions

To add a function, choose the plus icon next to **Function**. You can search for the function or select it from the menu. After a function is selected, it will be added and your focus will be in the text box of the first parameter. To edit or change a parameter, choose it and it will turn into a text box. - To delete a function, choose the function name followed by the x icon.

Some functions, such as `aliasByNode`, support an optional second argument. To add an argument, pause on the first argument, and then choose the + symbol that appears. To remove the second optional parameter, choose it and keep it blank. The editor will remove it.

Sort labels

If you want consistent ordering, use `sortByName`. This can be annoying when you have the same labels on multiple graphs, and they are both sorted differently and using different colors. To fix this, use `sortByName()`.

Nested queries

You can reference queries by the row *letter* that they're on (similar to Microsoft Excel). If you add a second query to a graph, you can reference the first query by typing in #A. This provides a convenient way to build compounded queries.

Avoiding many queries by using wildcard characters

Occasionally, you might want to see multiple time series plotted on the same graph. For example, you might want to see how the CPU is being used on a machine. You might initially create the graph by adding a query for each time series, such as `cpu.percent.user.g`, `cpu.percent.system.g`, and so on. This results in *n* queries made to the data source, which is inefficient.

To be more efficient one can use wildcard characters in your search, returning all the time series in one query. For example, `cpu.percent.*.g`.

Modifying the metric name in tables or charts

Use `alias` functions to change metric names on Grafana tables or graphs; for example, `aliasByNode()` or `aliasSub()`.

Point consolidation

All Graphite metrics are consolidated so that Graphite doesn't return more data points than there are pixels in the graph. By default, this consolidation is done using `avg` function. You can control how Graphite consolidates metrics by adding the Graphite consolidateBy function.

> **Note**
>
> This means that legend summary values (max, min, total) cannot all be correct at the same time. They are calculated client-side by Grafana. And depending on your consolidation function, only one or two can be correct at the same time.

Combining time series

To combine time series, choose **Combine** in the **Functions** list.

Data exploration and tags

In Graphite, everything is a tag.

When exploring data, previously selected tags are used to filter the remaining result set. To select data, you use the `seriesByTag` function, which takes tag expressions (=, !=, =~, !=~) to filter time series.

The Grafana query builder does this for you automatically when you select a tag.

> **Note**
>
> **Tip:** The regular expression search can be slow on high-cardinality tags, so try to use other tags to reduce the scope first. Starting off with a particular name or namespace helps reduce the results.

Template variables

Instead of hardcoding things such as server, application, and sensor name in your metric queries, you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. You can use these dropdown boxes to change the data being displayed in your dashboard.

For more information about templating and template variables, see Templates and variables.

To create a variable using tag values, use the Grafana functions `tags` and `tag_values`.

Query	Description
`tags()`	Returns all tags.
`tags(server=~backend\*)`	Returns only tags that occur in series matching the filter expressio n.
`tag_values(server)`	Return tag values for the specified tag.

Query	Description
`tag_values(server, server=~backend\*)`	Returns filtered tag values that occur for the specified tag in series matching those expressions.
`tag_values(server, server=~backend\*, app=~${apps:regex})`	Multiple filter expressions and expressions can contain other variables.

For more details, see Graphite docs on the autocomplete API for tags.

Query variable

The query you specify in the query field should be a metric find type of query. For example, a query such as `prod.servers.*` will fill the variable with all possible values that exist in the wildcard position.

You can also create nested variables that use other variables in their definition. For example `apps.$app.servers.*` uses the variable `$app` in its query definition.

Using `__searchFilter` to filter query variable results

Using `__searchFilter` in the query field will filter the query result based on what you enter in the dropdown select box. When you enter nothing, the default value for `__searchFilter` is `*` and `` when used as part of a regular expression.

The following example shows how to use `__searchFilter` as part of the query field to enable searching for `server` while the user enters text in the dropdown select box.

Query

```
apps.$app.servers.$__searchFilter
```

TagValues

```
tag_values(server, server=~${__searchFilter:regex})
```

Variable usage

You can use a variable in a metric node path or as a parameter to a function.

There are two syntaxes:

- `$<varname>` Example: apps.frontend.$server.requests.count
- `${varname}` Example: apps.frontend.${server}.requests.count

Why two ways? The first syntax is easier to read and write but does not allow you to use a variable in the middle of a word. Use the second syntax in expressions such as `my.server${serverNumber}.count`.

Variable usage in tag queries

Multi-value variables in tag queries use the advanced formatting syntax introduced in Grafana 5.0 for variables: `{var:regex}`. Non-tag queries will use the default glob formatting for multi-value variables.

The following code example shows a tag expression with regex formatting and using the Equal Tilde operator, =~.

```
server=~${servers:regex}
```

For more information, see Advanced variable format options.

Annotations

Annotations enable you to overlay rich event information on top of graphs. You add annotation queries via the Dashboard menu / Annotations view. For more information, see Annotations.

Graphite supports two ways to query annotations:

- A regular metric query. For this, you use the **Graphite query** text box.
- A Graphite events query. For this, you use the `Graphite event tags` text box, and specify a tag or wildcard character (keeping it empty should also work).

Connect to a Google Cloud Monitoring data source

> **Note**
>
> In earlier versions of Grafana, this data source was named Google Stackdriver.

Add the Google Cloud Monitoring data source to be able to build dashboards for your Google Cloud Monitoring metrics.

Adding the data source

1. Open the side menu by choosing the Grafana icon in the top header.
2. In the side menu, under the **Dashboards** link, you should find the **Data Sources** link.
3. Choose the **+ Add data source** button in the top header.
4. Select **Google Cloud Monitoring** from the **Type** dropdown list.
5. Upload or paste in the Service Account Key file. See later in this document for steps to create a Service Account Key file.

> **Note**
>
> If you don't see the **Data Sources** link in your side menu, your current user does not have the `Admin` role.

Name	Description
`Name`	The data source name. This is how you refer to the data source in panels and queries.

Name	Description
`Default`	Default data source means that it will be pre-selected for new panels.
`Service Account Key`	Service account key file for a GCP Project. See the instructions later in this document about how to create it.

Authentication

There are two ways to authenticate the Google Cloud Monitoring plugin

- Upload a Google JWT file
- Automatically retrieve credentials from Google metadata server

The latter option is only available when running Grafana on GCE virtual machine.

Using a Google service account key file

To authenticate with the Google Cloud Monitoring API, you must create a Google Cloud Platform (GCP) Service Account for the Project that you want to show data for. A Grafana data source integrates with one GCP Project. To visualize data from multiple GCP Projects, you must create one data source per GCP Project.

Enabling APIs

The following APIs must be enabled first:

- Monitoring API
- Cloud Resource Manager API

Choose the links listed, and then choose the **Enable** button.

Creating a GCP service account for a Project

1. Navigate to the APIs and Services Credentials page.
2. Choose the **Create credentials** dropdown/button and choose the **Service account key** option.

{{< docs-imagebox img="/img/docs/v71/cloudmonitoring_create_service_account_button.png" class="docs-image–no-shadow" caption="Create service account button" >}}

3. On the **Create service account key** page, choose key type `JSON`. Then, in the **Service Account** dropdown list, choose the **New service account** option.

   {{< docs-imagebox img="/img/docs/v71/cloudmonitoring_create_service_account_key.png" class="docs-image–no-shadow" caption="Create service account key" >}}

4. Some new fields will appear. Fill in a name for the service account in the **Service account name** field and then choose the **Monitoring Viewer** role from the **Role** dropdown list.

   {{< docs-imagebox img="/img/docs/v71/cloudmonitoring_service_account_choose_role.png" class="docs-image–no-shadow" caption="Choose role" >}}

5. Choose the **Create** button. A JSON key file will be created and downloaded to your computer. Store this file in a secure place as it allows access to your Google Cloud Monitoring data.
6. Upload it to Grafana on the data source **Configuration** page. You can either upload the file or paste in the contents of the file.

   {{< docs-imagebox img="/img/docs/v71/cloudmonitoring_grafana_upload_key.png" class="docs-image–no-shadow" caption="Upload service key file to Grafana" >}}

7. The file contents will be encrypted and saved in the Grafana database. Don’t forget to save after uploading the file!

   {{< docs-imagebox img="/img/docs/v71/cloudmonitoring_grafana_key_uploaded.png" class="docs-image–no-shadow" caption="Service key file is uploaded to Grafana" >}}

Using the query editor

The Google Cloud Monitoring query editor allows you to build two types of queries - **Metric** and **Service Level Objective (SLO)**. Both types return time series data.

Metric queries

The metric query editor allows you to select metrics, group/aggregate by labels and by time, and use filters to specify which time series you want in the results.

To create a metric query, follow these steps:

1. Choose the option **Metrics** in the **Query Type** dropdown list.
2. Choose a project from the **Project** dropdown list.
3. Choose a Google Cloud Platform service from the **Service** dropdown list.
4. Choose a metric from the **Metric** dropdown list.
5. To add or remove filters or group by clauses, use the plus and minus icons in the filter and group by sections. This step is optional.

Google Cloud Monitoring metrics can be of different kinds (GAUGE, DELTA, CUMULATIVE) and these kinds have support for different aggregation options (reducers and aligners). The Grafana query editor shows the list of available aggregation methods for a selected metric and sets a default reducer and aligner when you select the metric. Units for the Y-axis are also automatically selected by the query editor.

Filters

To add a filter, choose the plus icon, choose a field to filter by, and enter a filter value. For example, enter `instance_name = grafana-1`. You can remove the filter by choosing the filter name and selecting `--remove filter--`.

Simple wildcard characters

When the operator is set to or ,=!= it is possible to add wildcard characters to the filter value field. For example, `us-*` captures all values that start with "us-", and `*central-a` captures all values that end with "central-a". `*-central-*` captures all values that have the substring of `central-`. Simple wildcard characters are less expensive than regular expressions.

Regular expressions

When the operator is set to or ,=~!=~ it is possible to add regular expressions to the filter value field. For example, `us-central[1-3]-[af]` matches all values that start with "us-central", followed by a number in the range of 1 to 3, a dash and then either an "a" or an "f". Leading and trailing slashes are not needed when creating regular expressions.

Aggregation

The aggregation field lets you combine time series based on common statistics. For more information about aggregation, refer to aggregation options.

The `Aligner` field allows you to align multiple time series after the same group by time interval. For more information about aligner, refer to alignment metric selector.

Alignment Period and grouping by time

The `Alignment Period` groups a metric by time if an aggregation is chosen. The default is to use the GCP Google Cloud Monitoring default groupings (which allows you to compare graphs in Grafana with graphs in the Google Cloud Monitoring UI). The option is called `cloud monitoring auto` and the defaults are:

- 1m for time ranges < 23 hours
- 5m for time ranges >= 23 hours and < 6 days
- 1h for time ranges >= 6 days

The other automatic option is `grafana auto`. This will automatically set the group by time depending on the time range chosen and the width of the graph panel. For more information, see Adding an interval variable.

It is also possible to choose fixed time intervals to group by, such as `1h` or `1d`.

Group By

Group by resource or metric labels to reduce the number of time series and to aggregate the results by a group by. For example, group by instance_name to see an aggregated metric for a compute instance.

Metadata labels

Resource metadata labels contain information to uniquely identify a resource in Google Cloud. Metadata labels are only returned in the time series response if they're part of the **Group By** segment in the time series request. There's no API for retrieving metadata labels, so it's not possible to populate the group by dropdown list with the metadata labels that are available for the selected service and metric. However, the **Group By** field dropdown list comes with a pre-defined list of common system labels.

User labels cannot be pre-defined, but it's possible to enter them manually in the **Group By** field. If a metadata label, user label, or system label is included in the **Group By** segment, you can create filters based on it and expand its value in the **Alias** field.

Alias patterns

The Alias By field allows you to control the format of the legend keys. The default is to show the metric name and labels. This can be long and hard to read. Using the following patterns in the alias field, you can format the legend key the way you want it.

Metric Type patterns

Alias pattern	Description	Example result
`{{metric.type}}`	Returns the full Metric Type.	`compute.googleapis.com/instance/cpu/utilization`
`{{metric.name}}`	Returns the metric name part.	`instance/cpu/utilization`
`{{metric.service}}`	Returns the service part.	`compute`

Label patterns

In the Group By dropdown list, you can see a list of metric and resource labels for a metric. These can be included in the legend key using alias patterns.

Alias pattern format	Description	Alias pattern example	Example result
`{{metric.label.xxx}}`	Returns the metric label value.	`{{metric.label.instance_name}}`	`grafana-1-prod`
`{{resource.label.xxx}}`	Returns the resource label value.	`{{resource.label.zone}}`	`us-east1-b`
`{{metadata.system_labels.xxx}}`	Returns the metadata system label value.	`{{metadata.system_labels.name}}`	`grafana`

Alias pattern format	Description	Alias pattern example	Example result
`{{metadata.user_labels.xxx}}`	Returns the metadata user label value.	`{{metadata.user_labels.tag}}`	`production`

Example Alias By: `{{metric.type}} - {{metric.label.instance_name}}`

Example Result: `compute.googleapis.com/instance/cpu/usage_time - server1-prod`

It is also possible to resolve the name of the Monitored Resource Type.

Alias pattern format	Description	Example result
`{{resource.type}}`	Returns the name of the monitored resource type.	`gce_instance`

Example Alias By: `{{resource.type}} - {{metric.type}}`

Example Result: `gce_instance - compute.googleapis.com/instance/cpu/usage_time`

SLO queries

> **Note**
>
> SLO queries are available only in Grafana v7.0+

The SLO query builder in the Google Cloud Monitoring data source allows you to display SLO data in time series format. To get an understanding of the basic concepts in service monitoring, refer to the Google Cloud Monitoring official documentation.

Creating an SLO query

To create an SLO query, follow these steps:

1. Choose the option **Service Level Objectives (SLO)** in the **Query Type** dropdown list.
2. Choose a project from the **Project** dropdown list.
3. Choose an SLO service from the **Service** dropdown list.
4. Choose an SLO from the **SLO** dropdown list.
5. Choose a time series selector from the **Selector** dropdown list.

The friendly names for the time series selectors are shown in Grafana. The following table shows the mapping from the friendly name to the system name that is used in the Service Monitoring documentation.

Selector dropdown list value	Corresponding time series selector used
SLI Value	select_slo_health
SLO Compliance	select_slo_compliance
SLO Error Budget Remaining	select_slo_budget_fraction

Alias patterns for SLO queries

You can use the Alias By field to control the format of the legend keys for SLO queries.

Alias pattern	Description	Example result
`{{project}}`	Returns the GCP project name.	`myProject`
`{{service}}`	Returns the service name.	`myService`
`{{slo}}`	Returns the SLO.	`latency-slo`
`{{selector}}`	Returns the selector.	`select_slo_health`

Alignment Period and grouping by time for SLO queries

SLO queries use the same alignment period functionality as metric queries. For more information, see Metric queries.

Templating

Instead of hardcoding things such as server, application and sensor name in your metric queries you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. You can use these dropdown boxes to change the data being displayed in your dashboard.

For more information about templating and template variables, see Templates and variables.

Query variable

Variable of the type *Query* allows you to query Google Cloud Monitoring for various types of data. The Google Cloud Monitoring data source plugin provides the following `Query Types`.

Name	Description
`Metric Types`	Returns a list of metric type names that are available for the specified service.
`Labels Keys`	Returns a list of keys for `metric label` and `resource label` in the specified metric.
`Labels Values`	Returns a list of values for the label in the specified metric.
`Resource Types`	Returns a list of resource types for the specified metric.
`Aggregations`	Returns a list of aggregations (cross series reducers) for the specified metric.
`Aligners`	Returns a list of aligners (per series aligners) for the specified metric.
`Alignment periods`	Returns a list of all alignment periods that are available in Google Cloud Monitoring query editor in Grafana.
`Selectors`	Returns a list of selectors that can be used in SLO (Service Level Objective s) queries.
`SLO Services`	Returns a list of Service Monitoring services that can be used in SLO queries.

Name	Description
`Service Level Objectives (SLO)`	Returns a list of SLO's for the specified SLO service.

Using variables in queries

There are two syntaxes:

- `$<varname>` Example: `metric.label.$metric_label`
- `[[varname]]` Example: `metric.label.[[metric_label]]`

Why two ways? The first syntax is easier to read and write but does not allow you to use a variable in the middle of a word. When the *Multi-value* or *Include all value* options are enabled, Grafana converts the labels from plaintext to a regex compatible string, which means you have to use =~ instead of =.

Annotations

You can use annotations to overlay rich event information on top of graphs. You add annotation queries via the Dashboard menu / Annotations view. Annotation rendering is expensive so it is important to limit the number of rows returned. There is no support for showing Google Cloud Monitoring annotations and events yet but it works well with custom metrics in Google Cloud Monitoring.

For more information about annotations, see Annotations.

With the query editor for annotations, you can select a metric and filters. The **Title** and **Text** fields support templating and can use data returned from the query. For example, the Title field could have the following text:

```
{{metric.type}} has value: {{metric.value}}
```

Example Result: `monitoring.googleapis.com/uptime_check/http_status has this value: 502`

Patterns for the annotation query editor

Alias pattern format	Description	Alias pattern example	Example result
`{{metric.value}}`	Value of the metric/point.	`{{metric.value}}`	`555`
`{{metric.type}}`	Returns the full Metric Type.	`{{metric.type}}`	`compute.googleapis.com/instance/cpu/utilization`
`{{metric.name}}`	Returns the metric name part.	`{{metric.name}}`	`instance/cpu/utilization`
`{{metric.service}}`	Returns the service part.	`{{metric.service}}`	`compute`
`{{metric.label.xxx}}`	Returns the metric label value.	`{{metric.label.instance_name}}`	`grafana-1-prod`
`{{resource.label.xxx}}`	Returns the resource label value.	`{{resource.label.zone}}`	`us-east1-b`

Deep linking from Grafana panels to the Metrics Explorer in Google Cloud Console

> **Note**
>
> This feature is available only for Metric queries.

Choose a time series in the panel to see a context menu with a link to View in Metrics Explorer in Google Cloud Console. Choosing that link opens the Metrics Explorer in the Google Cloud Console and runs the query from the Grafana panel there. The link navigates the user first to the Google Account Chooser. After successfully selecting an account, the user is redirected to the Metrics

Explorer. The provided link is valid for any account, but it only displays the query if your account has access to the GCP project specified in the query.

Connect to an InfluxDB data source

Grafana ships with a feature-rich data source plugin for InfluxDB. The plugin includes a custom query editor and supports annotations and query templates.

Adding the data source

1. Open the side menu by choosing the Grafana icon in the top header.
2. In the side menu under the link,**Dashboards** you should find a link named **Data Sources**.
3. Choose the **+ Add data source** button in the top header.
4. Select **InfluxDB** from the **Type** dropdown list.
5. Select **InfluxQL** or **Flux** from the **Query Language** list.

> **Note**
>
> If you don't see the **Data Sources** link in your side menu, it means that your current user does not have the `Admin` role.

Connect to a Jaeger data source

The Jaeger data source provides open-source, end-to-end distributed tracing.

Adding the data source

To access Jaeger settings, choose the **Configuration** (gear) icon, then choose **Data Sources**, and then choose **Jaeger**.

Name	Description
`Name`	The data source name. This is how you see the data source in panels, queries, and Explore.
`Default`	Default data source means that it will be pre-selected for new panels.

Name	Description
URL	The URL of the Jaeger instance; e.g., `http://localhost:16686` .
Access	Server (default) = URL must be accessible from the Grafana backend/server.
Basic Auth	Enable basic authentication to the Jaeger data source.
User	User name for basic authentication.
Passwo	Password for basic authentication.

Query traces

You can query and display traces from Jaeger via Explore. For more information, see Explore.

The Jaeger query editor allows you to query by trace ID directly or selecting a trace from trace selector. To query by trace ID, insert the ID into the text input.

Use the trace selector to pick particular trace from all traces logged in the time range you have selected in Explore. The trace selector has three levels of nesting: 1. The service you are interested in. 1. Particular operation is part of the selected service. 1. Specific trace in which the selected operation occurred, represented by the root operation name and trace duration.

Linking to the trace ID from logs

You can link to Jaeger trace from logs in Loki by configuring a derived field with internal link. For more information, see Derived fields.

Connect to a Loki data source

The Loki data source provides access to Loki, Grafana's log aggregation system.

Adding the data source

1. Open the Grafana workspace and make sure you are logged in.
2. In the side menu under the **Configuration** link you should find a **Data Sources** link.
3. choose the **Add data source** button at the top.
4. Select **Loki** from the list of data sources.

> **Note**
>
> If you don't see the **Data Sources** link in your side menu, it means that your current user does not have the `Admin` role.

Name	Description
`Name`	The data source name. This is how you see the data source in panels, queries, and Explore.
`Defaul`	Default data source means that it will be pre-selected for new panels.
`URL`	The URL of the Loki instance; e.g., `http://localhost:3100` . This could be the URL for an Amazon EC2 host, or an Application Load Balancer in front of an Amazon EKS cluster, or any other URL for a Loki instance.
`Maximu lines`	Upper limit for number of log lines returned by Loki (default is 1000). Decrease if your browser is sluggish when displaying logs in Explore.

Derived fields

You can use the *derived fields* configuration to do the following:

- Add fields parsed from the log message.
- Add a link that uses the value of the field.

You can use this functionality to link to your tracing backend directly from your logs, or link to a user profile page if a userId is present in the log line. These links appear in the log details. For more information, see Labels and detected fields.

Each derived field consists of the following:

- **Name** – Shown in the log details as a label.
- **Regex** – A Regex pattern that runs on the log message and captures part of it as the value of the new field. Can only contain a single capture group.

- **URL/query** – If the link is external, then enter the full link URL. If the link is internal link, then this input serves as query for the target data source. In both cases, you can interpolate the value from the field with `${__value.raw }` macro.
- **Internal link** – Select if the link is internal or external. In case of internal link, a data source selector allows you to select the target data source. Only tracing data sources are supported.

You can use a debug section to see what your fields extract and how the URL is interpolated. choose **Show example log message** to show the text area where you can enter a log message.

The new field with the link shown in log details.

Querying logs

Querying and displaying log data from Loki is available via Explore and with the logs panel in visualizations. Select the Loki data source, and then enter a LogQL query to display your logs. For more information about LogQL, see LogQL.

Log queries

A log query consists of two parts: **log stream selector**, and a **search expression**. For performance reasons, you must start by choosing a log label for a log stream.

The Logs Explorer (the **Log labels** button) next to the query field shows a list of labels of available log streams. An alternative way to write a query is to use the query field's automatic completion. You start by typing a left curly brace `{` and the autocomplete menu will suggest a list of labels. Press the **Enter** key to run the query.

After the result is returned, the log panel shows a list of log rows and a bar chart where the x-axis shows the time and the y-axis shows the frequency/count.

Log Stream Selector

For the label part of the query expression, wrap it in curly braces `{}` and then use the key value syntax for selecting labels. Multiple label expressions are separated by a comma:

```
{app="mysql",name="mysql-backup"}
```

The following label matching operators are currently supported:

- = exactly equal.
- != not equal.

- =~ regex-match.
- !~ do not regex-match.

Examples:

- `{name=~"mysql.+"}`
- `{name!~"mysql.+"}`

Another way to add a label selector is in the table section. choose **Filter** beside a label to add the label to the query expression. This even works for multiple queries and will add the label selector to each query.

Search expressions

After writing the Log Stream Selector, you can filter the results further by writing a search expression. The search expression can be just text or a regex expression.

Example queries:

- `{job="mysql"} |= "error"`
- `{name="kafka"} |~ "tsdb-ops.*io:2003"`
- `{instance=~"kafka-[23]",name="kafka"} != "kafka.server:type=ReplicaManager"`

Filter operators can be chained and will sequentially filter down the expression. The resulting log lines will satisfy every filter.

Example

```
{job="mysql"} |= "error" != "timeout"
```

The following filter types are currently supported:

- |= line contains string.
- != line doesn't contain string.
- |~ line matches regular expression.
- !~ line does not match regular expression.

Note

For more information about LogQL, Loki's query language, see Loki LogQL.

Log context

When using a search expression as detailed above, you now have the ability to retrieve the context surrounding your filtered results. By choosing the `Show Context` link on the filtered rows, you'll be able to investigate the log messages that came before and after the log message you're interested in.

Templating

Instead of hardcoding things such as server, application and sensor name in your metric queries, you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. You can use these dropdown boxes to change the data being displayed in your dashboard.

For more information about templating and template variables, see Templates and variables.

Annotations

You can use any non-metric Loki query as a source for annotations. Log content will be used as annotation text and your log stream labels as tags, so there is no need for additional mapping.

Connect to a Microsoft SQL Server data source

Use the Microsoft SQL Server (MSSQL) data source to query and visualize data from any Microsoft SQL Server 2005 or newer, including Microsoft Azure SQL Database.

Important

Grafana version 8.0 changes the underlying data structure for data frames for the Microsoft SQL Server, Postgres, and MySQL. As a result, a time series query result is returned in a wide format. For more information, see Wide format in the Grafana data frames documentation.

To make your visualizations work as they did before, you might have to do some manual migrations. One solution is documented on Github at Postgres/MySQL/MSSQL: Breaking change in v8.0 related to time series queries and ordering of data column.

Adding the data source

1. Open the side menu by choosing the Grafana icon in the top header.
2. In the side menu under the link,**Configuration** you should find a **Data Sources** link.
3. Choose the **+ Add data source** button in the top header.
4. Select **Microsoft SQL Server** from the **Type** dropdown list.

Data source options

Name	Description
`Name`	The data source name. This is how you see the data source in panels and queries.
`Default`	Default data source means that it will be pre-selected for new panels.
`Host`	The IP address/hostname and optional port of your MSSQL instance. If port is omitted, default 1433 will be used.
`Database`	Name of your MSSQL database.
`User`	Database user's login/username.
`Password`	Database user's password.
`Encrypt`	This option determines whether or to which extent a secure SSL TCP/IP connection will be negotiated with the server, default `false` (Grafana v5.4+).
`Max open`	The maximum number of open connections to the database, default `unlimited` (Grafana v5.4+).
`Max idle`	The maximum number of connections in the idle connection pool, default 2 (Grafana v5.4+).

Name	Description
`Max lifetime`	The maximum amount of time in seconds a connection can be reused, default `14400`/4 hours.

Min time interval

A lower limit for the `$_interval` `$_interval_ms` variables. Recommended to be set to write frequency, for example `1m` if your data is written every minute. This option can also be overridden/configured in a dashboard panel under data source options. This value **must** be formatted as a number followed by a valid time identifier; for example, `1m` (1 minute) or `30s` (30 seconds). The following time identifiers are supported.

Identifier	Description
`y`	Year
`M`	Month
`w`	Week
`d`	Day
`h`	Hour
`m`	Minute
`s`	Second
`ms`	Millisecond

Database user permissions

> **Important**
>
> The database user that you specify when you add the data source should only be granted SELECT permissions on the specified database and tables you want to query. Grafana

does not validate that the query is safe. The query could include any SQL statement. For example, statements such as `DELETE FROM user;` and `DROP TABLE user;` would be run. To protect against this, we highly recommend that you create a specific MSSQL user with restricted permissions.

The following example code shows creating a specific MSSQL user with restricted permissions.

```
CREATE USER grafanareader WITH PASSWORD 'password'
GRANT SELECT ON dbo.YourTable3 TO grafanareader
```

Make sure that the user does not get any unwanted permissions from the public role.

Known issues

If you're using an older version of Microsoft SQL Server such as 2008 and 2008R2, you might need to disable encryption to be able to connect. If possible, we recommend you to use the latest service pack available for optimal compatibility.

Query editor

You will find the MSSQL query editor in the metrics tab in the graph, Singlestat, or table panel's edit mode. You enter edit mode by choosing the panel title and then choosing Edit. The editor allows you to define a SQL query to select data to be visualized.

1. Select *Format as* `Time series` (for use in Graph or Singlestat panel's among others) or `Table` (for use in Table panel among others).
2. This is the actual editor where you write your SQL queries.
3. Show help section for MSSQL below the query editor.
4. Show the SQL query that was run. Will be available first after a successful query has been run.
5. Add an additional query where an additional query editor will be displayed.

Macros

To simplify syntax and to allow for dynamic parts, such as date range filters, the query can contain macros.

Macro example	Description
`$__time(dateColumn)`	Will be replaced by an expression to rename the column to *time*. For example, *dateColumn as time* .
`$__timeEpoch(dateColumn)`	Will be replaced by an expression to convert a DATETIME column type to Unix timestamp and rename it to *time*. For example, *DATEDIFF(second, "1970-01-01", dateColumn) AS time*.
`$__timeFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name. For example, *dateColumn BETWEEN "2017-04-21T05:01:17Z"*

Macro example	Description
	AND "2017-04-21T05:06:17Z".
`$__timeFrom()`	Will be replaced by the start of the currently active time selection. For example, *"2017-04-21T05:01:17Z"*.
`$__timeTo()`	Will be replaced by the end of the currently active time selection. For example, *"2017-04-21T05:06:17Z"*.

Macro example	Description
`$__timeGroup(dateColumn,'5m'[, fillvalue])`	Will be replaced by an expression usable in GROUP BY clause. Providing a *fillValue* of *NULL* or *floating value* will automatically fill empty series in time range with that value. For example, *CAST(ROUND(DATEDIF F(second, "1970-01-01", time_column)/300.0, 0) as bigint)*300*.
`$__timeGroup(dateColumn,'5m', 0)`	Same as preceding but with a fill parameter so missing points in that series will be added by grafana and 0 will be used as value.

Macro example	Description
`$__timeGroup(dateColumn,'5m', NULL)`	Same as above but NULL will be used as value for missing points.
`$__timeGroup(dateColumn,'5m', previous)`	Same as above but the previous value in that series will be used as fill value if no value has been seen yet NULL will be used (only available in Grafana 5.3+).

The query editor has a **Generated SQL** link that shows up after a query has been run, while in panel edit mode. Choose it and it will expand and show the raw interpolated SQL string that was run.

Table queries

If the query option is set to ,**Format asTable** then you can basically do any type of SQL query. The table panel will automatically show the results of whatever columns and rows your query returns.

The following example code shows a database table.

```
CREATE TABLE [event] (
  time_sec bigint,
  description nvarchar(100),
  tags nvarchar(100),
)
```

```
CREATE TABLE [mssql_types] (
  c_bit bit, c_tinyint tinyint, c_smallint smallint, c_int int, c_bigint bigint,
 c_money money, c_smallmoney smallmoney, c_numeric numeric(10,5),
  c_real real, c_decimal decimal(10,2), c_float float,
  c_char char(10), c_varchar varchar(10), c_text text,
  c_nchar nchar(12), c_nvarchar nvarchar(12), c_ntext ntext,
  c_datetime datetime,  c_datetime2 datetime2, c_smalldatetime smalldatetime, c_date
 date, c_time time, c_datetimeoffset datetimeoffset
)

INSERT INTO [mssql_types]
SELECT
  1, 5, 20020, 980300, 1420070400, '$20000.15', '£2.15', 12345.12,
  1.11, 2.22, 3.33,
  'char10', 'varchar10', 'text',
  N'#nchar12#', N'#nvarchar12#', N'#text#',
  GETDATE(), CAST(GETDATE() AS DATETIME2), CAST(GETDATE() AS SMALLDATETIME),
 CAST(GETDATE() AS DATE), CAST(GETDATE() AS TIME), SWITCHOFFSET(CAST(GETDATE() AS
 DATETIMEOFFSET), '-07:00')
```

The following example code shows a query.

```
SELECT * FROM [mssql_types]
```

You can control the name of the Table panel columns by using regular AS SQL column selection syntax, as shown in the following example code.

```
SELECT
  c_bit as [column1], c_tinyint as [column2]
FROM
  [mssql_types]
```

The resulting table panel:

Time series queries

If you set **Format as** to **Time series**, for use in Graph panel for example, the query must have a column named `time` that returns either a SQL datetime or any numeric data type representing Unix epoch in seconds. You can return a column named `metric` that is used as metric name for the

value column. Any column except `time` and `metric` is treated as a value column. If you omit the `metric` column, the name of the value column will be the metric name. You can select multiple value columns, each will have its name as metric. If you return multiple value columns and a column named `metric` then this column is used as prefix for the series name.

Result sets of time series queries must be sorted by time.

The following example code shows a database table.

```
CREATE TABLE [event] (
  time_sec bigint,
  description nvarchar(100),
  tags nvarchar(100),
)
```

```
CREATE TABLE metric_values (
  time datetime,
  measurement nvarchar(100),
  valueOne int,
  valueTwo int,
)

INSERT metric_values (time, measurement, valueOne, valueTwo) VALUES('2018-03-15
 12:30:00', 'Metric A', 62, 6)
INSERT metric_values (time, measurement, valueOne, valueTwo) VALUES('2018-03-15
 12:30:00', 'Metric B', 49, 11)
...
INSERT metric_values (time, measurement, valueOne, valueTwo) VALUES('2018-03-15
 13:55:00', 'Metric A', 14, 25)
INSERT metric_values (time, measurement, valueOne, valueTwo) VALUES('2018-03-15
 13:55:00', 'Metric B', 48, 10)
```

The following example code shows one `value` and one `metric` column.

```
SELECT
  time,
  valueOne,
  measurement as metric
FROM
```

```
  metric_values
WHERE
  $__timeFilter(time)
ORDER BY 1
```

When the preceding query is used in a graph panel, it will produce two series named `Metric A` and `Metric B` with the values `valueOne` and `valueTwo` plotted over `time`.

The following example code shows multiple `value` columns.

```
SELECT
  time,
  valueOne,
  valueTwo
FROM
  metric_values
WHERE
  $__timeFilter(time)
ORDER BY 1
```

When the preceding query is used in a graph panel, it will produce two series named `Metric A` and `Metric B` with the values `valueOne` and `valueTwo` plotted over `time`.

The following example code shows using the $__timeGroup macro.

```
SELECT
  $__timeGroup(time, '3m') as time,
  measurement as metric,
  avg(valueOne)
FROM
  metric_values
WHERE
  $__timeFilter(time)
GROUP BY
  $__timeGroup(time, '3m'),
  measurement
ORDER BY 1
```

When the previous query is used in a graph panel, it will produce two series named `Metric A` and `Metric B` with the values `valueOne` and `valueTwo` plotted over `time`. Any two series lacking a

value in a three-minute window will render a line between those two lines. You'll notice that the graph to the right never goes down to zero.

The following example code shows using the $__timeGroup macro with fill parameter set to zero.

```
SELECT
  $__timeGroup(time, '3m', 0) as time,
  measurement as metric,
  sum(valueTwo)
FROM
  metric_values
WHERE
  $__timeFilter(time)
GROUP BY
  $__timeGroup(time, '3m'),
  measurement
ORDER BY 1
```

When this query is used in a graph panel, the result is two series named `Metric A` and `Metric B` with a sum of `valueTwo` plotted over `time`. Any series lacking a value in a 3 minute window will have a value of zero which you'll see rendered in the graph to the right.

Templating

Instead of hardcoding things such as server, application and sensor name in your metric queries you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. You can use these dropdown boxes to change the data being displayed in your dashboard.

For more information about templating and template variables, see Templates and variables.

Query variable

If you add a template variable of the type `Query`, you can write a MSSQL query that can return things such as measurement names, key names or key values that are shown as a dropdown select box.

For example, you can have a variable that contains all values for the `hostname` column in a table if you specify a query such as this in the templating variable *Query* setting.

```
SELECT hostname FROM host
```

A query can return multiple columns and Grafana will automatically create a list from them. For example, the following query will return a list with values from `hostname` and `hostname2`.

```
SELECT [host].[hostname], [other_host].[hostname2] FROM host JOIN other_host ON [host].
[city] = [other_host].[city]
```

Another option is a query that can create a key/value variable. The query should return two columns that are named `__text` and `__value`. The `__text` column value should be unique (if it is not unique then the first value is used). The options in the dropdown list will have a text and value that allow you to have a friendly name as text and an id as the value. An example query with `hostname` as the text and `id` as the value:

```
SELECT hostname __text, id __value FROM host
```

You can also create nested variables. For example, if you had another variable named `region`. Then you could have the hosts variable only show hosts from the current selected region with a query such as this (if `region` is a multi-value variable, then use the `IN` comparison operator rather than = to match against multiple values).

```
SELECT hostname FROM host WHERE region IN ($region)
```

Using variables in queries

> **Note**
>
> Template variable values are only quoted when the template variable is a `multi-value`.

If the variable is a multi-value variable then use the `IN` comparison operator rather than = to match against multiple values.

There are two syntaxes:

$<varname> Example with a template variable named `hostname`:

```
SELECT
  atimestamp time,
  aint value
FROM table
WHERE $__timeFilter(atimestamp) and hostname in($hostname)
ORDER BY atimestamp
```

`[[varname]]` Example with a template variable named `hostname`:

```
SELECT
  atimestamp as time,
  aint as value
FROM table
WHERE $__timeFilter(atimestamp) and hostname in([[hostname]])
ORDER BY atimestamp
```

Turning off quoting for multi-value variables

Grafana automatically creates a quoted, comma-separated string for multi-value variables. For example, if `server01` and `server02` are selected then it will be formatted as: `'server01', 'server02'`. To turn off quoting, use the csv formatting option for variables.

`${servers:csv}`

For more information about variable formatting options, see Templates and variables.

Annotations

You can use annotations to overlay rich event information on top of graphs. You add annotation queries via the Dashboard menu / Annotations view. For more information, see Annotations.

Columns:

Name	Description
`time`	The name of the date/time field. Could be a column with a native SQL date/time data type or epoch value.

Name	Description
`timeend`	Optional name of the end date/time field. Could be a column with a native SQL date/time data type or epoch value.
`text`	Event description field.
`tags`	Optional field name to use for event tags as a comma separated string.

The following example code shows database tables.

```
CREATE TABLE [events] (
  time_sec bigint,
  description nvarchar(100),
  tags nvarchar(100),
)
```

We also use the database table defined in Time series queries.

The following example code shows a query using a time column with epoch values.

```
SELECT
  time_sec as time,
  description as [text],
  tags
FROM
  [events]
WHERE
  $__unixEpochFilter(time_sec)
ORDER BY 1
```

The following example code shows a region query using time and timeend columns with epoch values.

```
SELECT
  time_sec as time,
  time_end_sec as timeend,
  description as [text],
  tags
FROM
  [events]
WHERE
  $__unixEpochFilter(time_sec)
ORDER BY 1
```

The following example code shows a query using a time column of native SQL date/time data type.

```
SELECT
  time,
  measurement as text,
  convert(varchar, valueOne) + ',' + convert(varchar, valueTwo) as tags
FROM
  metric_values
WHERE
  $__timeFilter(time_column)
ORDER BY 1
```

Stored procedure support

Stored procedures have been verified to work. However, there might be edge cases where it won't work as you would expect. Stored procedures should be supported in table, time series, and annotation queries as long as you use the same naming of columns and return data in the same format as described previously in the respective sections.

Macro functions will not work inside a stored procedure.

Examples

For the following examples, the database table is defined in Time series queries. Let's say that you want to visualize four series in a graph panel, such as all combinations of columns `valueOne`, `valueTwo` and `measurement`. The graph panel to the right visualizes what we want to achieve. To solve this, you must use two queries:

The following example code shows the first query.

```
SELECT
  $__timeGroup(time, '5m') as time,
  measurement + ' - value one' as metric,
  avg(valueOne) as valueOne
FROM
  metric_values
WHERE
  $__timeFilter(time)
GROUP BY
  $__timeGroup(time, '5m'),
  measurement
ORDER BY 1
```

The following example code shows the second query.

```
SELECT
  $__timeGroup(time, '5m') as time,
  measurement + ' - value two' as metric,
  avg(valueTwo) as valueTwo
FROM
  metric_values
GROUP BY
  $__timeGroup(time, '5m'),
  measurement
ORDER BY 1
```

Stored procedure using time in epoch format

You can define a stored procedure that will return all data that you need to render four series in a graph panel such as above. In this case, the stored procedure accepts two parameters, `@from` and `@to`, of `int` data types, which should be a time range (from-to) in epoch format which will be used to filter the data to return from the stored procedure.

This mimics the `$__timeGroup(time, '5m')` in the select and group by expressions, and that's why numerous lengthy expressions are needed. These could be extracted to MSSQL functions, if wanted.

```
CREATE PROCEDURE sp_test_epoch(
```

```
  @from int,
  @to   int
)   AS
BEGIN
  SELECT
    cast(cast(DATEDIFF(second, {d '1970-01-01'}, DATEADD(second,
 DATEDIFF(second,GETDATE(),GETUTCDATE()), time))/600 as int)*600 as int) as time,
    measurement + ' - value one' as metric,
    avg(valueOne) as value
  FROM
    metric_values
  WHERE
    time >= DATEADD(s, @from, '1970-01-01') AND time <= DATEADD(s, @to, '1970-01-01')
  GROUP BY
    cast(cast(DATEDIFF(second, {d '1970-01-01'}, DATEADD(second,
 DATEDIFF(second,GETDATE(),GETUTCDATE()), time))/600 as int)*600 as int),
    measurement
  UNION ALL
  SELECT
    cast(cast(DATEDIFF(second, {d '1970-01-01'}, DATEADD(second,
 DATEDIFF(second,GETDATE(),GETUTCDATE()), time))/600 as int)*600 as int) as time,
    measurement + ' - value two' as metric,
    avg(valueTwo) as value
  FROM
    metric_values
  WHERE
    time >= DATEADD(s, @from, '1970-01-01') AND time <= DATEADD(s, @to, '1970-01-01')
  GROUP BY
    cast(cast(DATEDIFF(second, {d '1970-01-01'}, DATEADD(second,
 DATEDIFF(second,GETDATE(),GETUTCDATE()), time))/600 as int)*600 as int),
    measurement
  ORDER BY 1
END
```

Then you can use the following query for your graph panel.

```

DECLARE
  @from int = $__unixEpochFrom(),
  @to int = $__unixEpochTo()

EXEC dbo.sp_test_epoch @from, @to
```

Stored procedure using time in datetime format

You can define a stored procedure that will return all data that you need to render four series in a graph panel such as above. In this case, the stored procedure accepts two parameters, `@from` and `@to`, of `datetime` data types, which should be a time range (from-to) that will be used to filter the data to return from the stored procedure.

This mimics the `$__timeGroup(time, '5m')` in the select and group by expressions, and that's why numerous lengthy expressions are needed. These could be extracted to MSSQL functions, if wanted.

```
CREATE PROCEDURE sp_test_datetime(
  @from datetime,
  @to   datetime
)   AS
BEGIN
  SELECT
    cast(cast(DATEDIFF(second, {d '1970-01-01'}, time)/600 as int)*600 as int) as time,
    measurement + ' - value one' as metric,
    avg(valueOne) as value
  FROM
    metric_values
  WHERE
    time >= @from AND time <= @to
  GROUP BY
    cast(cast(DATEDIFF(second, {d '1970-01-01'}, time)/600 as int)*600 as int),
    measurement
  UNION ALL
  SELECT
    cast(cast(DATEDIFF(second, {d '1970-01-01'}, time)/600 as int)*600 as int) as time,
    measurement + ' - value two' as metric,
    avg(valueTwo) as value
  FROM
    metric_values
  WHERE
    time >= @from AND time <= @to
  GROUP BY
    cast(cast(DATEDIFF(second, {d '1970-01-01'}, time)/600 as int)*600 as int),
    measurement
  ORDER BY 1
END
```

Then you can use the following query for your graph panel.

```
DECLARE
  @from datetime = $__timeFrom(),
  @to datetime = $__timeTo()

EXEC dbo.sp_test_datetime @from, @to
```

Alerting

Time series queries should work in alerting conditions. Table formatted queries are not yet supported in alert rule conditions.

Connect to a MySQL data source

Add the MySQL data source to be able to query and visualize data from a MySQL compatible database.

> **Important**
>
> Grafana version 8.0 changes the underlying data structure for data frames for the MySQL, Postgres, and Microsoft SQL Server data sources. As a result, a time series query result is returned in a wide format. For more information, see Wide format in the Grafana data frames documentation.
> To make your visualizations work as they did before, you might have to do some manual migrations. One solution is documented on Github at Postgres/MySQL/MSSQL: Breaking change in v8.0 related to time series queries and ordering of data column.

Adding the data source

1. Open the side menu by choosing the Grafana icon in the top header.
2. In the side menu under the **Dashboards** link, you should find a link named **Data Sources**.
3. Choose the **+ Add data source** button in the top header.
4. Select **MySQL** from the **Type** dropdown list.

Data source options

Name	Description
`Name`	The data source name. This is how you see the data source in panels and queries.
`Default`	Default data source means that it will be pre-selected for new panels.
`Host`	The IP address/hostname and optional port of your MySQL instance.
`Database`	Name of your MySQL database.
`User`	Database user's login/username.
`Password`	Database user's password.
`Max open`	The maximum number of open connectio ns to the database, default `unlimited` (Grafana v5.4+).
`Max idle`	The maximum number of connections in the idle connection pool, default 2 (Grafana v5.4+).
`Max lifetime`	The maximum amount of time in seconds a connection can be reused, default `14400`/4 hours. This should always be lower than configured wait_timeout in MySQL (Grafana v5.4+).

Min time interval

A lower limit for the `$_interval` `$_interval_ms` variables. Recommended to be set to write frequency, for example `1m` if your data is written every minute. This option can also be overridden/configured in a dashboard panel under data source options. This value **must** be formatted as a

number followed by a valid time identifier; for example, `1m` (1 minute) or `30s` (30 seconds). The following time identifiers are supported.

Identifier	Description
`y`	Year
`M`	Month
`w`	Week
`d`	Day
`h`	Hour
`m`	Minute
`s`	Second
`ms`	Millisecond

Database user permissions

> **Important**
>
> The database user that you specify when you add the data source should be granted only SELECT permissions on the specified database and tables that you want to query. Grafana does not validate that the query is safe. The query could include any SQL statement. For example, statements such as `USE otherdb;` and `DROP TABLE user;` would be run. To protect against this, we strongly recommend that you create a specific MySQL user with restricted permissions.

The following code example shows creating a specific MySQL user with restricted permissions.

```
CREATE USER 'grafanaReader' IDENTIFIED BY 'password';
GRANT SELECT ON mydatabase.mytable TO 'grafanaReader';
```

To grant access to more databases and tables, you can use wildcard characters (*) in place of database or table if you want.

Query editor

You find the MySQL query editor in the metrics tab in a panel's edit mode. You enter edit mode by choosing the panel title, then **Edit**.

The query editor has a **Generated SQL** link that shows up after a query has been run, while in panel edit mode. Choose it, and it will expand and show the raw interpolated SQL string that was run.

Select table, time column, and metric column (FROM)

When you enter edit mode for the first time or add a new query, Grafana will try to prefill the query builder with the first table that has a timestamp column and a numeric column.

In the FROM field, Grafana will suggest tables that are in the configured database. To select a table or view in another database that your database user has access to, you can manually enter a fully qualified name (database.table) such as `otherDb.metrics`.

The Time column field refers to the name of the column holding your time values. Selecting a value for the Metric column field is optional. If a value is selected, the Metric column field will be used as the series name.

The metric column suggestions will only contain columns with a text data type (text, tinytext, mediumtext, longtext, varchar, char). If you want to use a column with a different data type as metric column, you can enter the column name with a cast: `CAST(numericColumn as CHAR)`. You can also enter arbitrary SQL expressions in the metric column field that evaluate to a text data type such as `CONCAT(column1, " ", CAST(numericColumn as CHAR))`.

Columns and aggregation functions (SELECT)

In the `SELECT` row, you can specify what columns and functions you want to use. In the column field, you can write arbitrary expressions instead of a column name such as `column1 * column2 / column3`.

If you use aggregate functions, you must group your result set. The editor will automatically add a `GROUP BY time`if you add an aggregate function.

You can add further value columns by choosing the plus button and selecting `Column` from the menu. Multiple value columns will be plotted as separate series in the graph panel.

Filtering data (WHERE)

To add a filter, choose the plus icon to the right of the `WHERE` condition. You can remove filters by choosing on the filter and selecting `Remove`. A filter for the current selected time range is automatically added to new queries.

Group By

To group by time or any other columns, choose the plus icon at the end of the GROUP BY row. The suggestion dropdown list will only show text columns of your currently selected table but you can manually enter any column. You can remove the group by choosing on the item and then selecting `Remove`.

If you add any grouping, all selected columns must have an aggregate function applied. The query builder will automatically add aggregate functions to all columns without aggregate functions when you add groupings.

Gap filling

Grafana can fill in missing values when you group by time. The time function accepts two arguments. The first argument is the time window that you want to group by, and the second argument is the value you want Grafana to fill missing items with.

Text editor mode (raw)

You can switch to the raw query editor mode by choosing the hamburger icon and selecting **Switch editor mode** or by choosing **Edit SQL** below the query.

> **Note**
>
> If you use the raw query editor, be sure that your query at minimum has `ORDER BY time` and a filter on the returned time range.

Macros

To simplify syntax and to allow for dynamic parts, such as date range filters, the query can contain macros.

Macro example	Description
`$__time(dateColumn)`	Will be replaced by an expression to convert to a UNIX timestamp and rename the column to `time_sec`; for example, *UNIX_TIMESTAMP(dateColumn) as time_sec*.
`$__timeEpoch(dateColumn)`	Will be replaced by an expression to convert to a UNIX timestamp and rename the column to `time_sec`; for example, *UNIX_TIMESTAMP(dateColumn) as time_sec*.
`$__timeFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name. For example, *dateColum*

Macro example	Description
	n BETWEEN FROM_UNIX TIME(1494 410783) AND FROM_UNIX TIME(1494 410983).
`$__timeFrom()`	Will be replaced by the start of the currently active time selection. For example, *FROM_UNIX TIME(1494 410783)*.
`$__timeTo()`	Will be replaced by the end of the currently active time selection. For example, *FROM_UNIX TIME(1494 410983)*.

Macro example	Description
`$__timeGroup(dateColumn,'5m')`	Will be replaced by an expression usable in GROUP BY clause. For example, *cast(cast (UNIX_TIMESTAMP(dateColumn) /(300) as signed)*300 as signed),*
`$__timeGroup(dateColumn,'5m', 0)`	Same as the previous row, but with a fill parameter so missing points in that series will be added by grafana and 0 will be used as value.
`$__timeGroup(dateColumn,'5m', NULL)`	Same as above but NULL will be used as value for missing points.

Macro example	Description
`$__timeGroup(dateColumn,'5m', previous)`	Same as above but the previous value in that series will be used as fill value if no value has been seen yet NULL will be used (only available in Grafana 5.3+).
`$__timeGroupAlias(dateColumn,'5m')`	Will be replaced identical to `$__timeGroup` but with an added column alias (available only in Grafana 5.3+).

Macro example	Description
`$__unixEpochFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name with times represented as Unix timestamp. For example, `dateColumn > 1494410783 AND dateColumn < 1494497183`.
`$__unixEpochFrom()`	Will be replaced by the start of the currently active time selection as Unix timestamp. For example, `1494410783`.
`$__unixEpochTo()`	Will be replaced by the end of the currently active time selection as Unix timestamp. For example, `1494497183`.

Macro example	Description
`$__unixEpochNanoFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name with times represented as nanosecond timestamp. For example, `dateColumn > 1494410783152415214 AND dateColumn < 1494497182142514872`.
`$__unixEpochNanoFrom()`	Will be replaced by the start of the currently active time selection as nanosecond timestamp. For example, `1494410783152415214`.

Macro example	Description
`$__unixEpochNanoTo()`	Will be replaced by the end of the currently active time selection as nanosecond timestamp. For example, 1494497182514251487 2 .
`$__unixEpochGroup(dateColumn,"5m", [fillmode])`	Same as `$__timeGroup` but for times stored as Unix timestamp (available only in Grafana 5.3+).
`$__unixEpochGroupAlias(dateColumn,"5m", [fillmode])`	Same as above but also adds a column alias (available only in Grafana 5.3+).

The query editor has a **Generated SQL** link that shows up after a query has run, while in panel edit mode. Choose it, and it will expand and show the raw interpolated SQL string that was run.

Table queries

If the **Format as** query option is set to **Table**, you can basically do any type of SQL query. The table panel will automatically show the results of whatever columns and rows your query returns.

The following code shows an example query.

```
SELECT
  title as 'Title',
  user.login as 'Created By' ,
  dashboard.created as 'Created On'
 FROM dashboard
INNER JOIN user on user.id = dashboard.created_by
WHERE $__timeFilter(dashboard.created)
```

You can control the name of the Table panel columns by using regular `as` SQL column selection syntax.

Time series queries

If you set **Format as** to **Time series**, for use in a graph panel for example, the query must return a column named `time` that returns either a SQL datetime or any numeric data type representing Unix epoch. Any column except `time` and `metric` is treated as a value column. You can return a column named `metric` that is used as metric name for the value column. If you return multiple value columns and a column named `metric`, this column is used as prefix for the series name (available only in Grafana 5.3+).

Result sets of time series queries must be sorted by time.

The following code example shows the `metric` column.

```
SELECT
  $__timeGroup(time_date_time,'5m'),
  min(value_double),
  'min' as metric
FROM test_data
WHERE $__timeFilter(time_date_time)
GROUP BY time
ORDER BY time
```

The following code example shows using the fill parameter in the $__timeGroup macro to convert null values to be zero instead.

```
SELECT
  $__timeGroup(createdAt,'5m',0),
  sum(value_double) as value,
  measurement
FROM test_data
WHERE
  $__timeFilter(createdAt)
GROUP BY time, measurement
ORDER BY time
```

The following code example shows multiple columns.

```
SELECT
  $__timeGroup(time_date_time,'5m'),
  min(value_double) as min_value,
  max(value_double) as max_value
FROM test_data
WHERE $__timeFilter(time_date_time)
GROUP BY time
ORDER BY time
```

There is no support for a dynamic group by time based on time range and panel width.

Templating

Instead of hardcoding things such as server, application and sensor name in your metric queries you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. You can use these dropdown boxes to change the data being displayed in your dashboard.

For more information about templating and template variables, see Templates.

Query variable

If you add a template variable of the type `Query`, you can write a MySQL query that can return things such as measurement names, key names, or key values that are shown as a dropdown select box.

For example, you can have a variable that contains all values for the `hostname` column in a table if you specify a query such as this in the templating variable *Query* setting.

```
SELECT hostname FROM my_host
```

A query can return multiple columns and Grafana will automatically create a list from them. For example, the following query will return a list with values from `hostname` and `hostname2`.

```
SELECT my_host.hostname, my_other_host.hostname2 FROM my_host JOIN my_other_host ON
 my_host.city = my_other_host.city
```

To use time range dependent macros such as `$__timeFilter(column)` in your query, the refresh mode of the template variable must be set to *On Time Range Change*.

```
SELECT event_name FROM event_log WHERE $__timeFilter(time_column)
```

Another option is a query that can create a key/value variable. The query should return two columns that are named `__text` and `__value`. The `__text` column value should be unique (if it is not unique, the first value is used). The options in the dropdown list will have a text and value so that you can have a friendly name as text and an ID as the value.

The following code example shows a query with `hostname` as the text and `id` as the value.

```
SELECT hostname AS __text, id AS __value FROM my_host
```

You can also create nested variables. For example, if you had another variable named `region`. Then you could have the hosts variable show only hosts from the current selected region with a query such as this (if `region` is a multi-value variable then use the `IN` comparison operator rather than = to match against multiple values).

```
SELECT hostname FROM my_host  WHERE region IN($region)
```

Using `__searchFilter` to filter results in Query Variable

Using `__searchFilter` in the query field will filter the query result based on what the user types in the dropdown select box. When nothing has been entered by the user, the default value for `__searchFilter` is `%`.

> **Note**
>
> Important that you surround the `__searchFilter` expression with quotes as Grafana does not do this for you.

The following example shows how to use `__searchFilter` as part of the query field to enable searching for `hostname` while the user types in the dropdown select box.

```
SELECT hostname FROM my_host  WHERE hostname LIKE '$__searchFilter'
```

Using variables in queries

From Grafana 4.3.0 to 4.6.0, template variables are always quoted automatically so if it is a string value do not wrap them in quotes in where clauses.

From Grafana 4.7.0, template variable values are only quoted when the template variable is a `multi-value`.

If the variable is a multi-value variable, use the `IN` comparison operator rather than = to match against multiple values.

There are two syntaxes:

$<varname> Example with a template variable named `hostname`:

```
SELECT
  UNIX_TIMESTAMP(atimestamp) as time,
  aint as value,
  avarchar as metric
FROM my_table
WHERE $__timeFilter(atimestamp) and hostname in($hostname)
ORDER BY atimestamp ASC
```

`[[varname]]` Example with a template variable named `hostname`:

```
SELECT
```

```
  UNIX_TIMESTAMP(atimestamp) as time,
  aint as value,
  avarchar as metric
FROM my_table
WHERE $__timeFilter(atimestamp) and hostname in([[hostname]])
ORDER BY atimestamp ASC
```

Turning off quoting for multi-value variables

Grafana automatically creates a quoted, comma-separated string for multi-value variables. For example: if `server01` and `server02` are selected then it will be formatted as: `'server01', 'server02'`. To turn off quoting, use the csv formatting option for variables.

```
${servers:csv}
```

For more information about variable formatting options, see Advanced variable format options.

Annotations

You can use annotations to overlay rich event information on top of graphs. You add annotation queries via the Dashboard menu / Annotations view. For more information, see .

The following example code shows a query using a time column with epoch values.

```
SELECT
  epoch_time as time,
  metric1 as text,
  CONCAT(tag1, ',', tag2) as tags
FROM
  public.test_data
WHERE
  $__unixEpochFilter(epoch_time)
```

The following example code shows a region query using time and timeend columns with epoch values.

> **Note**
>
> Available only in Grafana v6.6+.

```
SELECT
  epoch_time as time,
  epoch_timeend as timeend,
  metric1 as text,
  CONCAT(tag1, ',', tag2) as tags
FROM
  public.test_data
WHERE
  $__unixEpochFilter(epoch_time)
```

The following example code shows a query using a time column of native SQL date/time data type.

```
SELECT
  native_date_time as time,
  metric1 as text,
  CONCAT(tag1, ',', tag2) as tags
FROM
  public.test_data
WHERE
  $__timeFilter(native_date_time)
```

Name	Description
`time`	The name of the date/time field. Could be a column with a native SQL date/time data type or epoch value.
`timeend`	Optional name of the end date/time field. Could be a column with a native SQL date/time data type or epoch value.
`text`	Event description field.
`tags`	Optional field name to use for event tags as a comma-separated string.

Alerting

Time series queries should work in alerting conditions. Table formatted queries are not yet supported in alert rule conditions.

Connect to an OpenSearch data source

> **Note**
>
> In workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

With Amazon Managed Grafana, you can add open-source OpenSearch (or legacy Elasticsearch) as a data source. You can perform many types of simple or complex OpenSearch queries to visualize logs or metrics stored in OpenSearch. You can also annotate your graphs with log events stored in OpenSearch.

Add OpenSearch as a data source

> **Note**
>
> To be able to add the OpenSearch data source, you need to add your Grafana IAM account to the ALL_ACCESS and SECURITY_MANAGER roles.

1. Open the side menu by choosing the Grafana icon in the top header.
2. In the side menu under the **Dashboards** link, you should find the named **Data Sources** link.
3. Choose the **+ Add data source** button in the top header.
4. Select **OpenSearch** from the **Type** dropdown list.

> **Note**
>
> If you're not seeing the **Data Sources** link in your side menu, it means that your current user does not have the `Admin` role.

Nam	Description
Name	The data source name. This is how you see the data source in panels and queries.
Defa	Default data source means that it will be pre-selected for new panels.
Url	The HTTP protocol, IP, and port of your OpenSearch server.
Acce	Server (default) = URL must be accessible from the Grafana backend/server. Browser = URL must be accessible from the browser.

Access mode controls how requests to the data source will be handled. Server should be the preferred way if nothing else is stated.

Server access mode (default)

All requests are made from the browser to Grafana backend or server, which forwards the requests to the data source, circumventing possible Cross-Origin Resource Sharing (CORS) requirements. If you select this access mode, the URL must be accessible from the Grafana backend or server.

Browser (direct) access

Amazon Managed Grafana does not support browser direct access for the OpenSearch data source.

Index settings

Here you can specify a default for the `time field` and specify the name of your OpenSearch index. You can use a time pattern for the index name or a wildcard character.

OpenSearch/Elasticsearch version

Specify your OpenSearch or legacy Elasticsearch version in the version dropdown menu. The version is important because there are differences in how queries are composed for each version. Currently, Grafana supports OpenSearch 1.0.x. Supported versions of Elasticsearch are `2.0+`, `5.0+`, `5.6+`, `6.0+`, and `7.0+`. The value `5.6+` means version 5.6 or higher, but lower than 6.0. The value `6.0+` means version 6.0 or higher, but lower than 7.0. Finally, `7.0+` means version 7.0 or higher, but lower than 8.0.

Min time interval

A lower limit for the auto group by time interval. Recommended to be set to write frequency; for example, `1m` if your data is written every minute. This option can also be overridden/configured in a dashboard panel under data source options. This value **must** be formatted as a number followed by a valid time identifier; for example, `1m` (1 minute) or `30s` (30 seconds). The following time identifiers are supported.

Identifier	Description
`y`	Year
`M`	Month
`w`	Week
`d`	Day
`h`	Hour
`m`	Minute
`s`	Second
`ms`	Millisecond

Logs

Two parameters, `Message field name` and `Level field name`, can optionally be configured from the data source settings page that determine which fields will be used for log messages and log levels when visualizing logs in Explore.

For example, if you use a default setup of Filebeat for shipping logs to OpenSearch, the following configuration should work.

- **Message field name:** message
- **Level field name:** fields.level

Data links

Data links create a link from a specified field that can be accessed in logs view in Explore.

Each data link configuration consists of the following:

- **Field** – Name of the field used by the data link.
- **URL/query** – If the link is external, then enter the full link URL. If the link is internal link, then this input serves as query for the target data source. In both cases, you can interpolate the value from the field with `${__value.raw }` macro.
- **Internal link** – Select this if the link is internal or external. If the link is internal, a data source selector allows you to select the target data source. Only tracing data sources are supported.

Using the OpenSearch data source

Metric query editor

The OpenSearch query editor allows you to select multiple metrics and group by multiple terms or filters. Use the plus and minus icons to the right to add/remove metrics or group by clauses. Some metrics and group by clauses have options. Choose the option text to expand the row to view and edit metric or group by options.

Series naming and alias patterns

You can control the name for time series via the `Alias` input field.

Pattern	Description
`{{term fieldname}}`	Replaced with value of a term Group By.
`{{metric}}`	Replaced with metric name (ex. Average, Min, Max).
`{{field}}`	Replaced with the metric field name.

Pipeline metrics

Some metric aggregations are called pipeline aggregations; for example, *Moving Average* and *Derivative*. OpenSearch pipeline metrics require another metric to be based on. Use the eye icon

next to the metric to hide metrics from appearing in the graph. This is useful for metrics you only have in the query for use in a pipeline metric.

Templating

Instead of hardcoding things such as server, application, and sensor name in your metric queries you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. You can use these dropdown boxes to change the data being displayed in your dashboard.

For more information about templating and template variables, see Templates and variables.

Query variable

The OpenSearch data source supports two types of queries you can use in the *Query* field of *Query* variables. The query is written using a custom JSON string.

Query	Description
`{"find": "fields", "type": "keyword"}`	Returns a list of field names with the index type `keyword`.
`{"find": "terms", "field": "@hostname", "size": 1000}`	Returns a list of values for a field using term aggregation. Query will use current dashboard time range as time range for query.
`{"find": "terms", "field": "@hostname", "query": '<lucene query>'}`	Returns a list of values for a field using term aggregation and a specified Lucene query filter. Query will use current dashboard time range as time range for query.

There is a default size limit of 500 on terms queries. To set a custom limit, set the size property in your query. You can use other variables inside the query. The following code example shows the query definition for a variable named `$host`.

```
{"find": "terms", "field": "@hostname", "query": "@source:$source"}
```

In the previous example, we use another variable named `$source` inside the query definition. Whenever you change, via the dropdown list, the current value of the `$source` variable, it initiates an update of the `$host` variable. After the update, the `$host` variable contains only hostnames filtered by in this case the `@source` document property.

These queries by default return results in term order (which can then be sorted alphabetically or numerically as for any variable). To produce a list of terms sorted by doc count (a top-N values list), add an `orderBy` property of `doc_count`. This automatically selects a descending sort. Using `asc` with doc_count (a bottom-N list) can be done by setting `order: "asc"`, but it is discouraged because it increases the error on document counts. To keep terms in the doc count order, set the variable's **Sort** dropdown list to **Disabled**. Alternatively, you might alternatively still want to use **Alphabetical** to re-sort them.

```
{"find": "terms", "field": "@hostname", "orderBy": "doc_count"}
```

Using variables in queries

There are two syntaxes:

- `$<varname>` Example: @hostname:$hostname
- `[[varname]]` Example: @hostname:[[hostname]]

Why two ways? The first syntax is easier to read and write, but it does not allow you to use a variable in the middle of a word. When the *Multi-value* or *Include all value* options are enabled, Grafana converts the labels from plaintext to a Lucene-compatible condition.

In the previous example, we have a lucene query that filters documents based on the `@hostname` property using a variable named `$hostname`. It is also using a variable in the *Terms* group by field input box. This allows you to use a variable to quickly change how the data is grouped.

Annotations

Annotations allow you to overlay rich event information on top of graphs. You add annotation queries via the Dashboard menu or Annotations view. Grafana can query any OpenSearch index for annotation events. For more information, see Annotations.

Nar	Description
Que	You can keep the search query blank or specify a Lucene query.
Tim	The name of the time field; must be date field.
Tim End	Optional name of the time end field must be date field. If set, annotations will be marked as a region between time and time-end.
Tex	Event description field.
Tag	Optional field name to use for event tags (can be an array or a CSV string).

Querying logs

Querying and displaying log data from OpenSearch is available in Explore. To display your logs, select the OpenSearch data source, and then optionally enter a Lucene query. For more information, see Explore.

Log queries

After the result is returned, the log panel shows a list of log rows and a bar chart where the x-axis shows the time and the y-axis shows the frequency or count.

Filtering log messages

Optionally, enter a Lucene query into the query field to filter the log messages. For example, using a default Filebeat setup, you should be able to use `fields.level:error` to show only error log messages.

Connect to an OpenTSDB data source

Amazon Managed Grafana ships with advanced support for OpenTSDB.

Adding the data source

1. Open the side menu by choosing the Grafana icon in the top header.
2. In the side menu under the **Dashboards** link, you should find a **Data Sources** link.
3. Choose the **+ Add data source** button in the top header.
4. Select **OpenTSDB** from the **Type** dropdown list.

> **Note**
>
> If you don't see the **Data Sources** link in your side menu, it means that your current user does not have the `Admin` role.

Name	Description
`Name`	The data source name. This is how you see the data source in panels and queries.
`Defau`	Default data source means that it will be pre-selected for new panels.
`Url`	The HTTP protocol, ip and port of your opentsdb server (default port is usually 4242).
`Acces`	Server (default) = URL must be accessible from the Grafana backend/server.
`Versi`	Version = opentsdb version, either <=2.1 or 2.2.
`Resol n`	Metrics from opentsdb can have data points with either second or millisecond resolutio n.

Query editor

Open a graph in edit mode by choose the title. Query editor will differ if the data source has version <=2.1 or = 2.2. In the former version, only tags can be used to query OpenTSDB. But in the latter version, filters as well as tags can be used to query opentsdb. Fill Policy is also introduced in OpenTSDB 2.2.

> **Note**
>
> While using the OpenTSDB 2.2 data source, make sure you use either Filters or Tags as they are mutually exclusive. If used together, might give you weird results.

Using autocomplete suggestions

As soon as you start typing metric names, tag names and tag values , you should see highlighted auto complete suggestions for them. The autocomplete only works if the OpenTSDB suggest API is enabled.

Templating queries

Instead of hardcoding things such as server, application and sensor name in your metric queries you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. You can use these dropdown boxes to change the data being displayed in your dashboard.

For more information about templating and template variables, see Templates and variables.

Query variable

The OpenTSDB data source supports template variable queries. This means you can create template variables that fetch the values from OpenTSDB. For example, metric names, tag names, or tag values.

When using OpenTSDB with a template variable of `query` type you can use following syntax for lookup.

Query	Description
`metrics(prefix)`	Returns metric names with specific prefix (can be empty).
`tag_names(cpu)`	Returns tag names (i.e., keys) for a specific cpu metric.
`tag_values(cpu, hostname)`	Returns tag values for metric cpu and tag key hostname.
`suggest_tagk(prefix)`	Returns tag names (i.e., keys) for all metrics with specific prefix (can be empty).
`suggest_tagv(prefix)`	Returns tag values for all metrics with specific prefix (can be empty).

If you do not see template variables being populated in `Preview of values` section, you must enable `tsd.core.meta.enable_realtime_ts` in the OpenTSDB server settings. Also, to populate metadata of the existing time series data in OpenTSDB, you must run `tsdb uid metasync` on the OpenTSDB server.

Nested templating

One template variable can be used to filter tag values for another template variable. First parameter is the metric name, second parameter is the tag key for which you need to find tag values, and after that all other dependent template variables. Some examples are mentioned below to make nested template queries work successfully.

Query	Description
`tag_values(cpu, hostname, env=$env)`	Returns tag values for cpu metric, selected env tag value, and tag key hostname.
`tag_values(cpu, hostname, env=$env, region=$region)`	Returns tag values for cpu metric, selected env tag value, selected region tag value, and tag key hostname.

For more information about OpenTSDB metric queries, see OpenTSDB documentation

Connect to a PostgreSQL data source

You can use the PostgreSQL data source to query and visualize data from your Amazon Aurora PostgreSQL databases.

Important

Grafana version 8 changes the underlying data structure for data frames for the Postgres, MySQL, and Microsoft SQL Server data sources. As a result, a time series query result is returned in a wide format. For more information, see Wide format in the Grafana data frames documentation. To make your visualizations work as they did before version 8, you might have to do some manual migrations. One solution is documented on Github at Postgres/MySQL/MSSQL: Breaking change in v8.0 related to time series queries and ordering of data column.

In *Grafana version 9*, the PostgreSQL data source sets up the root certificate for connecting to your database differently than in previous versions. If you update your workspace from version 8 to 9, you might need to change how you connect. See Troubleshooting issues with updated workspaces for more information.

Adding the data source

1. Open the side menu by choosing the Grafana icon in the top header.
2. In the side menu under the **Configuration** icon, you should find a **Data Sources** link.
3. Choose the **+ Add data source** button in the top header.
4. Select **PostgreSQL** from the **Type** dropdown list.

Data source options

Name	Description
`Name`	The data source name. This is how you see the data source in panels and queries.
`Default`	Default data source means that it will be pre-selected for new panels.
`Host`	The IP address/hostname and optional port of your PostgreSQL instance. *Do not* include the database name. The connection string for connecting to Postgres will not be correct and will cause errors.
`Database`	Name of your PostgreSQL database.
`User`	Database user's login/username.
`Password`	Database user's password
`SSL Mode`	This option determines whether or with what priority a secure SSL TCP/IP connection will be negotiated with the server.
`Max open`	The maximum number of open connectio ns to the database, default `unlimited` (Grafana v5.4+).

Name	Description
`Max idle`	The maximum number of connections in the idle connection pool, default 2 (Grafana v5.4+).
`Max lifetime`	The maximum amount of time in seconds a connection can be reused, default `14400`/4 hours (Grafana v5.4+).
`Version`	This option determines which functions are available in the query builder (only available in Grafana 5.3+).
`TimescaleDB`	TimescaleDB is a time-series database built as a PostgreSQL extension. If enabled, Grafana will use `time_bucket` in the `$__timeGroup` macro and display TimescaleDB specific aggregate functions in the query builder (only available in Grafana 5.3+).

Min time interval

A lower limit for the `$_interval` `$_interval_ms` variables. Recommended to be set to write frequency, for example `1m` if your data is written every minute. This option can also be overridden/configured in a dashboard panel under data source options. This value **must** be formatted as a number followed by a valid time identifier; for example, `1m` (1 minute) or `30s` (30 seconds). The following time identifiers are supported.

Identifier	Description
`y`	Year
`M`	Month
`w`	Week

Identifier	Description
d	Day
h	Hour
m	Minute
s	Second
ms	Millisecond

Database user permissions

> **Important**
>
> The database user that you specify when you add the data source should only be granted SELECT permissions on the specified database and tables you want to query. Grafana does not validate that the query is safe. The query could include any SQL statement. For example, statements such as `DELETE FROM user;` and `DROP TABLE user;` would be run. To protect against this, we highly recommend that you create a specific PostgreSQL user with restricted permissions.

The following example code shows creating a specifice PostgreSQL user with restricted permissions.

```
CREATE USER grafanareader WITH PASSWORD 'password';
GRANT USAGE ON SCHEMA schema TO grafanareader;
GRANT SELECT ON schema.table TO grafanareader;
```

Make sure that the user does not get any unwanted permissions from the public role.

Query editor

You find the PostgreSQL query editor in the metrics tab in Graph or Singlestat panel's edit mode. You enter edit mode by choosing the panel title, then edit.

The query editor has a **Generated SQL** link that shows up after a query has been run, while in panel edit mode. Choose it, and it will expand and show the raw interpolated SQL string that was run.

Select table, time column, and metric column (FROM)

When you enter edit mode for the first time or add a new query, Grafana will try to prefill the query builder with the first table that has a timestamp column and a numeric column.

In the FROM field, Grafana will suggest tables that are in the `search_path` of the database user. To select a table or view not in your ,`search_path` you can manually enter a fully qualified name (schema.table) such as `public.metrics`.

The Time column field refers to the name of the column holding your time values. Selecting a value for the Metric column field is optional. If a value is selected, the Metric column field will be used as the series name.

The metric column suggestions will only contain columns with a text data type (char,varchar,text). To use a column with a different data type as metric column, you can enter the column name with a cast: `ip::text`. You can also enter arbitrary SQL expressions in the metric column field that evaluate to a text data type such as `hostname || ' ' || container_name`.

Columns, window, and aggregation functions (SELECT)

In the `SELECT` row, you can specify what columns and functions you want to use. In the column field, you can write arbitrary expressions instead of a column name such as `column1 * column2 / column3`.

The available functions in the query editor depend on the PostgreSQL version you selected when configuring the data source. If you use aggregate functions, you must group your result set. If you add an aggregate function, the editor will automatically add a `GROUP BY time`.

The editor tries to simplify and unify this part of the query.

You can add further value columns by choosing the plus button and selecting **Column** from the menu. Multiple value columns will be plotted as separate series in the graph panel.

Filtering data (WHERE)

To add a filter, choose the plus icon to the right of the `WHERE` condition. You can remove filters by choosing the filter and selecting **Remove**. A filter for the current selected time range is automatically added to new queries.

Group By

To group by time or any other columns choose the plus icon at the end of the GROUP BY row. The suggestion dropdown list will only show text columns of your currently selected table but you can manually enter any column. You can remove the group by choosing the item and then selecting **Remove**.

If you add any grouping, all selected columns must have an aggregate function applied. The query builder will automatically add aggregate functions to all columns without aggregate functions when you add groupings.

Gap filling

Amazon Managed Grafana can fill in missing values when you group by time. The time function accepts two arguments. The first argument is the time window that you want to group by, and the second argument is the value you want Grafana to fill missing items with.

Text editor mode (RAW)

You can switch to the raw query editor mode by choosing the hamburger icon and selecting **Switch editor mode** or by choosing **Edit SQL** below the query.

> **Note**
>
> If you use the raw query editor, be sure that your query at minimum has `ORDER BY time` and a filter on the returned time range.

Macros

Macros can be used within a query to simplify syntax and allow for dynamic parts.

Macro example	Description
`$__time(dateColumn)`	Will be replaced by an expression to convert to a UNIX timestamp and rename

Macro example	Description
	the column to `time_sec`. For example, *UNIX_TIMESTAMP(dateColumn) as time_sec*.
`$__timeEpoch(dateColumn)`	Will be replaced by an expression to convert to a UNIX timestamp and rename the column to `time_sec`. For example, *UNIX_TIMESTAMP(dateColumn) as time_sec*.

Macro example	Description
`$__timeFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name. For example, *dateColumn BETWEEN FROM_UNIXTIME(1494410783) AND FROM_UNIXTIME(1494410983)*.
`$__timeFrom()`	Will be replaced by the start of the currently active time selection. For example, *FROM_UNIXTIME(1494410783)*.
`$__timeTo()`	Will be replaced by the end of the currently active time selection. For example, *FROM_UNIXTIME(1494410983)*.

Macro example	Description
`$__timeGroup(dateColumn,'5m')`	Will be replaced by an expression usable in GROUP BY clause. For example, *cast(cast (UNIX_TIMESTAMP(dateColumn) /(300) as signed)*300 as signed),*
`$__timeGroup(dateColumn,'5m', 0)`	Same as the preivious row, but with a fill parameter so missing points in that series will be added by grafana and 0 will be used as value.
`$__timeGroup(dateColumn,'5m', NULL)`	Same as above but NULL will be used as value for missing points.

Macro example	Description
`$__timeGroup(dateColumn,'5m', previous)`	Same as above but the previous value in that series will be used as fill value if no value has been seen yet NULL will be used (only available in Grafana 5.3+).
`$__timeGroupAlias(dateColumn,'5m')`	Will be replaced identical to $__timeGroup but with an added column alias
`$__unixEpochFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name with times represented as Unix timestamp . For example, *dateColumn > 1494410783 AND dateColumn < 1494497183*

Macro example	Description
`$__unixEpochFrom()  `	| Will be replaced by the start of the currently active time selection as Unix timestamp . For example, *1494410783*
`$__unixEpochTo()`	Will be replaced by the end of the currently active time selection as Unix timestamp . For example, *1494497183*
`$__unixEpochNanoFilter(dateColumn)`	Will be replaced by a time range filter using the specified column name with times represented as nanosecon d timestamp . For example, *dateColumn > 149441078 3152415214 AND dateColum n < 149449718 3142514872*

Macro example	Description
`$__unixEpochNanoFrom()`	Will be replaced by the start of the currently active time selection as nanosecond timestamp. For example, *1494410783152415214*
`$__unixEpochNanoTo()`	Will be replaced by the end of the currently active time selection as nanosecond timestamp. For example, *1494497183142514872*
`$__unixEpochGroup(dateColumn,"5m", [fillmode])`	Same as $__timeGroup but for times stored as Unix timestamp.

Table queries

If the query option is set to **Format as Table**, you can basically do any type of SQL query. The table panel will automatically show the results of whatever columns and rows your query returns.

You can control the name of the Table panel columns by using regular `as` SQL column selection syntax.

Time series queries

If you set **Format as** to `Time series`, for use in a graph panel for example, the query must return a column named `time` that returns either a SQL datetime or any numeric data type representing Unix epoch. Any column except `time` and `metric` is treated as a value column. You can return a column named `metric` that is used as metric name for the value column. If you return multiple value columns and a column named `metric`, this column is used as prefix for the series name.

Result sets of time series queries must be sorted by time.

The following example code shows a `metric` column.

```
SELECT
  $__timeGroup("time_date_time",'5m'),
  min("value_double"),
  'min' as metric
FROM test_data
WHERE $__timeFilter("time_date_time")
GROUP BY time
ORDER BY time
```

The folowing code example shows using the fill parameter in the $__timeGroup macro to convert null values to be zero instead.

```
SELECT
  $__timeGroup("createdAt",'5m',0),
  sum(value) as value,
  measurement
FROM test_data
WHERE
  $__timeFilter("createdAt")
GROUP BY time, measurement
ORDER BY time
```

The following example code shows multiple columns.

```
SELECT
  $__timeGroup("time_date_time",'5m'),
```

```
  min("value_double") as "min_value",
  max("value_double") as "max_value"
FROM test_data
WHERE $__timeFilter("time_date_time")
GROUP BY time
ORDER BY time
```

Templating

Instead of hardcoding things such as server, application and sensor name in your metric queries you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. You can use these dropdown boxes to change the data being displayed in your dashboard.

For more information about templating and template variables, see Templates.

Query variable

If you add a template variable of the type `Query`, you can write a PostgreSQL query that can return things such as measurement names, key names, or key values that are shown as a dropdown select box.

For example, you can have a variable that contains all values for the `hostname` column in a table if you specify a query such as this in the templating variable *Query* setting.

```
SELECT hostname FROM host
```

A query can return multiple columns and Grafana will automatically create a list from them. For example, the following query will return a list with values from `hostname` and `hostname2`.

```
SELECT host.hostname, other_host.hostname2 FROM host JOIN other_host ON host.city =
 other_host.city
```

To use time range dependent macros such as `$__timeFilter(column)` in your query, the refresh mode of the template variable must be set to *On Time Range Change*.

```
SELECT event_name FROM event_log WHERE $__timeFilter(time_column)
```

Another option is a query that can create a key/value variable. The query should return two columns that are named `__text` and `__value`. The `__text` column value should be unique (if it is not unique, the first value is used). The options in the dropdown list will have a text and value that allows you to have a friendly name as text and an id as the value. An example query with `hostname` as the text and `id` as the value:

```
SELECT hostname AS __text, id AS __value FROM host
```

You can also create nested variables. Using a variable named `region`, you could have the hosts variable show only hosts from the current selected region. The following code example shows a query such as this (if `region` is a multi-value variable, use the `IN` comparison operator rather than = to match against multiple values).

```
SELECT hostname FROM host  WHERE region IN($region)
```

Using `__searchFilter` to filter results in Query Variable

Using `__searchFilter` in the query field will filter the query result based on what the user types in the dropdown select box. When nothing has been entered by the user, the default value for `__searchFilter` is `%`.

> **Note**
>
> Important that you surround the `__searchFilter` expression with quotes as Grafana does not do this for you.

The following example shows how to use `__searchFilter` as part of the query field to enable searching for `hostname` while the user types in the dropdown select box.

```
SELECT hostname FROM my_host  WHERE hostname LIKE '$__searchFilter'
```

Using variables in queries

Template variable values are only quoted when the template variable is a `multi-value`.

If the variable is a multi-value variable, use the `IN` comparison operator rather than = to match against multiple values.

There are two syntaxes:

`$<varname>` Example with a template variable named `hostname`:

```
SELECT
  atimestamp as time,
  aint as value
FROM table
WHERE $__timeFilter(atimestamp) and hostname in($hostname)
ORDER BY atimestamp ASC
```

`[[varname]]` Example with a template variable named `hostname`:

```
SELECT
  atimestamp as time,
  aint as value
FROM table
WHERE $__timeFilter(atimestamp) and hostname in([[hostname]])
ORDER BY atimestamp ASC
```

Turning off quoting for multi-value variables

Amazon Managed Grafana automatically creates a quoted, comma-separated string for multi-value variables. For example: if `server01` and `server02` are selected then it will be formatted as: `'server01', 'server02'`. To turn off quoting, use the csv formatting option for variables.

`${servers:csv}`

For more information about variable formatting options, see Templates and variables.

Annotations

Use annotations to overlay rich event information on top of graphs. You add annotation queries via the Dashboard menu / Annotations view. For more information, see Annotations.

The following example code shows a query using a time column with epoch values.

```
SELECT
  epoch_time as time,
  metric1 as text,
  concat_ws(', ', metric1::text, metric2::text) as tags
FROM
  public.test_data
WHERE
  $__unixEpochFilter(epoch_time)
```

The following example code shows a region query using time and timeend columns with epoch values.

> **Note**
>
> This is available only in Grafana v6.6+.

```
SELECT
  epoch_time as time,
  epoch_time_end as timeend,
  metric1 as text,
  concat_ws(', ', metric1::text, metric2::text) as tags
FROM
  public.test_data
WHERE
  $__unixEpochFilter(epoch_time)
```

The following example code shows a query using a time column of native SQL date/time data type.

```
SELECT
  native_date_time as time,
  metric1 as text,
  concat_ws(', ', metric1::text, metric2::text) as tags
FROM
  public.test_data
WHERE
  $__timeFilter(native_date_time)
```

Name	Description
`time`	The name of the date/time field. Could be a column with a native SQL date/time data type or epoch value.
`timeend`	Optional name of the end date/time field. Could be a column with a native SQL date/time data type or epoch value (Grafana v6.6+).
`text`	Event description field.
`tags`	Optional field name to use for event tags as a comma-separated string.

Alerting

Time series queries should work in alerting conditions. Table formatted queries are not yet supported in alert rule conditions.

Connect to a Tempo data source

Tempo is a high-volume, minimal dependency trace storage, OSS tracing solution from Grafana Labs.

Adding the data source

To access Tempo settings, choose the **Configuration** (gear) icon, then choose **Data Sources**, and then choose **Tempo**.

Name	Description
`Name`	The name using which you will refer to the data source in panels, queries, and Explore.
`Default`	Default data source means that it will be pre-selected for new panels.
`URL`	The URL of the Tempo instance; e.g., `http://tempo` .

Name	Description
`Basic Auth`	Enable basic authentication to the Tempo data source.
`User`	User name for basic authentication.
`Passwo`	Password for basic authentication.

Trace to logs

This is a configuration for the trace to logs feature. The target data source currently must be Loki. For more information, see Tracing integration.

- **Data source** – Target data source.
- **Tags** – The tags that will be used in the Loki query. The default is `'cluster', 'hostname', 'namespace', pod'`
- **Span start time shift** – Shift in the start time for the Loki query based on the span start time. In order to extend to the past, you need to use a negative value. Time units can be used here, for example, 5s, 1m, 3h. The default is 0.
- **Span end time shift** – Shift in the end time for the Loki query based on the span end time. Time units can be used here, for example, 5s, 1m, 3h. The default is 0.

Query traces

You can query and display traces from Tempo via Explore. You can search for traces if you set up the trace to logs setting in the data source configuration page. To find traces to visualize, use the Loki query editor. To get search results, you must have derived fields configured, which point to this data source.

To query a particular trace, select the TraceID query type, and then put the ID into the Trace ID field.

Linking to the trace ID from logs

You can link to Tempo trace from logs in Loki or Elastic by configuring an internal link. For more information, see Derived fields.

Configure a TestData data source for testing

Grafana ships with a TestData data source, which creates simulated time series data for any panel. You can use it to build your own fake and random time series data and render it in any panel, which helps you verify dashboard functionality and safely and easily share the data.

Configure the data source

To access the data source configuration for TestData

1. Choose the **Configuration** (gear) icon.
2. Choose **Data Sources**.
3. choose **TestData**.

The data source doesn't provide any settings beyond the most basic options common to all data sources:

Name	Description
`Name`	The name of the data source in panels, queries, and Explore.
`Default`	Whether this datasource will be pre-selected for new panels.

Create mock data

Added the TestData data source, your Grafana instance's users can use it as a data source in any metric panel, and it will provide mock data that you can use, based on the TestData scenario you choose.

Choose a scenario

Instead of providing a query editor, the TestData data source helps you select a **Scenario** that generates simulated data for panels.

You can assign an **Alias** to each scenario, and many have their own options that appear when selected.

Available scenarios:

- **Annotations**
- **Conditional Error**
- **CSV Content**
- **CSV File**
- **CSV Metric Values**
- **Datapoints Outside Range**
- **Exponential heatmap bucket data**
- **Grafana API**
- **Grafana Live**
- **Linear heatmap bucket data**
- **Load Apache Arrow Data**
- **Logs**
- **No Data Points**
- **Node Graph**
- **Predictable CSV Wave**
- **Predictable Pulse**
- **Random Walk**
- **Random Walk (with error)**
- **Random Walk Table**
- **Raw Frames**
- **Simulation**
- **Slow Query**
- **Streaming Client**
- **Table Static**
- **USA generated data**

Import a pre-configured dashboard

TestData also provides an example dashboard.

To import the example dashboard

1. Navigate to the data source's configuration page.
2. Select the **Dashboards** tab.
3. Select **Import** for the **Simple Streaming Example** dashboard.

To customize an imported dashboard:

To customize the imported dashboard, we recommend that you save it under a different name. If you don't, upgrading Grafana can overwrite the customized dashboard with the new version.

Use test data to report issues

If you report an issue to GrafanaLabs on GitHub involving the use or rendering of time series data, we strongly recommend that you use this data source to replicate the issue. That makes it much easier for the developers to replicate and solve your issue.

Connect to a Zipkin data source

Zipkin is an open source, distributed tracing system. Add the Zipkin data source to be able to query your traces in Explore in Amazon Managed Grafana

Adding the data source

To access Zipkin settings, choose the **Configuration** (gear) icon, then choose **Data Sources**, and then choose **Zipkin**.

Name	Description
`Name`	The data source name. This is how you see the data source in panels, queries, and Explore.
`Default`	Default data source means that it will be pre-selected for new panels.
`URL`	The URL of the Zipkin instance; e.g., `http://localhost:9411` .
`Access`	Server (default) = URL needs to be accessible from the Grafana backend/server.
`Basic Auth`	Enable basic authentication to the Zipkin data source.

Name	Description
`User`	User name for basic authentication.
`Passwo`	Password for basic authentication.

Query traces

Querying and displaying traces from Zipkin is available via Explore.

The Zipkin query editor allows you to query by trace ID directly or selecting a trace from trace selector. To query by trace ID, insert the ID into the text input.

Use the trace selector to pick particular trace from all traces logged in the time range you have selected in Explore. The trace selector has three levels of nesting: 1. The service you are interested in. 1. Particular operation is part of the selected service 1. Specific trace in which the selected operation occurred, represented by the root operation name and trace duration.

Data mapping in the trace UI

Zipkin annotations are shown in the trace view as logs with annotation value shown under annotation key.

Linking to the trace ID from logs

You can link to Zipkin trace from logs in Loki by configuring a derived field with internal link. For more information, see Derived fields.

Connect to Enterprise data sources

The following data sources are supported in workspaces that have been upgraded to Amazon Managed Grafana Enterprise plugins. For more information, see Manage access to Enterprise plugins.

Enterprise plugins are regularly updated. This includes both updates to the existing plugins, and sometimes new data sources. This following documentation may not include all available data sources. For a list of the current Enterprise plugins supported by Amazon Managed Grafana Enterprise plugins, see Grafana Enterprise plugins in the *Grafana documentation*.

For workspaces that support version 9 and newer, Enterprise data sources are no longer installed by default. You must install the correct data source plugin. You can install plugins for all Enterprise

data sources, including any that aren't listed here. You can also choose to update the version of a plugin that you already have installed. For more information about managing plugins, see Extend your workspace with plugins.

Topics

- Connect to an AppDynamics data source
- Connect to a Databricks data source
- Connect to a Datadog data source
- Connect to a Dynatrace data source
- Connect to a GitLab data source
- Connect to a Honeycomb data source
- Connect to a Jira data source
- Connect to a MongoDB data source
- Connect to a New Relic data source
- Connect to an Oracle Database data source
- Connect to a Salesforce data source
- Connect to an SAP HANA data source
- Connect to a ServiceNow data source
- Connect to a Snowflake data source
- Connect to a Splunk data source
- Connect to a Splunk Infrastructure Monitoring data source
- Connect to a Wavefront data source (VMware Tanzu Observability by Wavefront)

Connect to an AppDynamics data source

The AppDynamics data source for Amazon Managed Grafana enables you to query metrics from AppDynamics using its Metrics API and visualize them in Grafana dashboards.

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.

Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

Note on the Data source configuration

Use Server (proxy) access (to avoid CORS and users looking up your password) and basic authentication. Remember that the username should be "user@account", (that is, your.name@customer1 or my_user@saas_account_name).

Configure the password using the following steps:

1. Navigate to https://accounts.appdynamics.com/subscriptions
2. Choose the link in the **Name** column on the row for your subscription.
3. Navigate to the **License details** by choosing the tab at top of the page.
4. The Access Key field has a **Show** button. Choose the **Show** button to show the Access Key.
5. Copy the Access Key into the Password field in the Basic Auth Details on config page in Grafana.

Set up a user and role for Amazon Managed Grafana using the following steps.

1. In AppDynamics, navigate to Settings, Administration.
2. Select the **Roles** tab, and choose the "+" button to create a new role; for example, `grafana_readonly`.
3. In the **Account** tab of the Create Role section, add the permission `View Business Flow`.
4. In the **Applications** tab, check the **View** box to allow Grafana to view application data.
5. In the **Databases** tab, check the **View** box to allow Grafana to view database data.
6. In the **Analytics** tab, check the **Can view data from all Applications** box to allow Grafana to view application analytics data.
7. In the **Users** tab of the Administration page, create a new user; for example, `grafana`. Assign the new user (or a Group to which the user belongs) to the role you just created; for example, `grafana_readonly`.

Templating

The supported template queries for now are:

1. `Applications` (All Applications)
2. `AppName.BusinessTransactions` (All BTs for the Application Name)
3. `AppName.Tiers` (All Tiers for the Application Name)
4. `AppName.Nodes` (All Nodes for the Application Name)
5. `AppName.TierName.BusinessTransactions` (All BTs for a specific Tier)
6. `AppName.TierName.Nodes` (All Nodes for a specific Tier)
7. `AppName.Path.<Any Metric Path>` (Any metric Path can be specified)

Legend keys

The default for the legend key can be quite long but this formatting can be customized.

The legend key can be prefixed with the application name by choosing the `App on legend` option. For example: `MyApp - Overall Application Performance|Average Response Time (ms)`.

If the query is for a singlestat or other panel where you cannot see the legend key, then choose the Show Metadata option to see what the legend key (also called an alias) for the query is.

The Legend dropdown list has three options: `Full Path`, `Segments` and `Custom`.

Legend option – full path

The legend key is the full metric path; for example, `Overall Application Performance|Average Response Time (ms)`.

Legend option – segments

The metric name is made up of segments. You can choose which segments to show.

For example, with a metric name:

`Errors|mywebsite|Error|Errors per Minute`

entering the following `2,4` in the Segments field returns `mywebsite|Errors per minute`.

The indexing starts with 1 so 1 returns `Errors`.

Legend option – custom

Create a custom legend by combining text with the following aliasing patterns to be able to mix in metric metadata.

- `{{app}}` returns the Application name
- `{{1}}` returns a segment from the metric path.

  For example, the metric: `Overall Application Performance|Average Response Time (ms)` has two segments. `{{1}}` returns the first segment, `{{2}}` returns the second segment.

Examples of legend key patterns and the legend keys that are generated:

- `custom legend key` => `custom legend key`
- `App: {{app}} MetricPart2: {{2}}` => `App: myApp MetricPart2: Average Response Time (ms)`

Connect to a Databricks data source

The Databricks data source enables you to query and visualize Databricks data within Amazon Managed Grafana. It includes a SQL editor to format and color code your queries.

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.
> Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

Adding a Databricks data source

Follow these steps to add a Databricks data source in the Grafana console.

To add a Databricks data source

1. Open the side menu by choosing the Grafana icon in the top header.
2. In the side menu, under the **Dashboards** link, select **Data Sources**.

> **Note**
>
> If you don't see the **Data Sources** link, you do not have the `Admin` role for Grafana.

3. Choose the **+ Add data source** button in the top header.
4. Select **Databricks** from the **Type** dropdown list.

> **Note**
>
> If you don't see the Databricks option, and need it, you must upgrade to Grafana Enterprise.

5. Choose the options to connect to and edit your data.

Notes when using the Databricks data source

Time series

Time series visualizations are selectable when you add a `datetime` field to your query. This field will be used as the timestamp for the series. If the field does not include a specific time zone, Grafana will assume that the time is UTC.

Multi-line time series

To create a multi-line time series visualization, the query must include at least three fields in the following order.

1. A `datetime` field with an alias of `time`.
2. A value to `GROUP BY`.
3. One or more metric values to visualize.

The following is an example of a query that will return multi-line time series options.

```
SELECT log_time AS time, machine_group, avg(disk_free) AS avg_disk_free
FROM mgbench.logs1
GROUP BY machine_group, log_time
ORDER BY log_time
```

Connect to a Datadog data source

The Datadog data source enables you to visualize metrics from the Datadog monitoring service in Amazon Managed Grafana.

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.
> Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

Usage

Caching

For large dashboards, that make lots of queries it is possible to be rate limited by the Datadog API (reach the maximum number of API calls per hour that the Datadog API allows). The caching feature caches unique queries for 60 seconds. This interval can be changed to be longer or shorter on the config page.

Query editor

It's easy - select aggregation and metric. If you want to filter result, select one or more tags.

The Datadog data source supports all of the advanced functions that the Datadog query editor supports. Select it from the dropdown list and arrange by choosing a function name.

Alias by field usage possibilities:

- Enter the alias into the "Alias by" field.
- Use scoped variables:
  - `$__metric` = replaced with metric name
  - `$__display_name` = replaced with metric name
  - `$__expression` = replaced with full metric expression
  - `$__aggr` = replaced with metric aggregation function (for example, avg, max, min, sum)

- $__scope = replaced with metric scope (for example, region, site, env, host)

- Use regular expressions:
  - Enter your regular expression into "Alias RegExp" field in /you regexp here/flags format.
  - If "Alias by" field is empty, RegExp results will be joined using. Example with metric expression = avg:system.load.5{*} : "Alias by" field input: """Alias RegExp" field input: avg:(.+)\.(\d) Result: system.load, 5
  - Use $<group_number> variables in "Alias by" field. Example with metric expression = avg:system.load.5{*} : "Alias by" field input: $1: $2 seconds "Alias RegExp" field input: avg:(.+)\.(\d) Result: system.load: 5 seconds
  - Use $0 to get the whole expression. Example with metric expression = datadog.dogstatsd.packet.count{*} : "Alias by" field input: Expression: $0 "Alias RegExp" field input: DOGstatsd\.(.*)\.(.*){*}/i Result: Expression: datadog.dogstatsd.packet.count{*}

  Note: you'll get an error using nonexistent group number.

Metric arithmetic

To use metric arithmetic set *Query type* to *Arithmetic*. Link to the metric that you want by using # sign. For example, #A * 2 will double the result of query A. Arithmetic between two metrics works in the same way - add queries which results you want to use for the calculation and then link to these metrics in the third query, such as #A / #B.

Annotations

An annotation is an event that is overlaid on top of graphs - an example of an event is a deployment or an outage. With this data source, you can fetch events from Datadog and overlay them on graphs in Amazon Managed Grafana. Annotations events can be filtered by source, tag or priority.

Templating

There are a few options for getting values of template variable - metrics and tags. To fetch the list of available metrics specify * in the *Query* field.

To return all tags use the value: tag or scope.

To return tags for a specified tag group then use one of the following default category values:

- `host`
- `device`
- `env`
- `region`
- `site`
- `status`
- `version`

For custom tag groups, then just enter the tag group name. For example, if your custom tag group name is `subscription_name`, enter that in the *Query* field.

Filter results by using the *Regex* field. Multi-value variables are supported when using tags - multiple selected tag values will be converted into a comma separated list of tags.

Ad-hoc filters

There is a new special type of template variable in Grafana called *Ad-hoc filters*. This variable will apply to *all* the Datadog queries in a dashboard. This allows using it like a quick filter. An ad-hoc variable for Datadog fetches all the key-value pairs from tags, for example, `region:east`, `region:west`, and uses them as query tags. To create this variable, select the *Ad-hoc filters* type and choose your Datadog data source. You can set any name for this variable.

Connect to a Dynatrace data source

Data source for https://www.dynatrace.com/. To use this data source, you must have a Dynatrace account.

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.
> Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

Known limitations

Template variables can't be multi-select. Only single selection is supported.

Only v2 metric APIs are supported.

Features

Core features

- Template Variables
  - Metric Names
  - Single selection only (**no multi-select**)
  - Ad-Hoc Filters
- Annotations
  - Not currently supported
- Aliasing
  - Metric Names
  - Aggregation
  - Display Name
  - Host
  - Description
- Alerting
  - Full alerting support

Dynatrace specific features

Supports both built-in and custom metrics using the Dynatrace metrics v2 API. For more information, see the Dynatrace documentation: Metrics API v2 and Metric ingestion.

Depending on the metric, the API might support additional transformation options.

Dynatrace permissions

You will need the following permissions in Dynatrace - Read metrics using API V2 (metrics.read) permission - Read entities using API V2 (entities.read) permission

Get an API key from Dynatrace

To set up an API token, see Dynatrace API - Tokens and authentication

Set the `metrics.read` and `entities.read` permissions for your API token.

Configuration

1. Choose **Settings/Data Sources** within the logical Grafana server UI and choose **Add data source**.
2. On the **Add data source** page, filter for **Dynatrace**, and select the Dynatrace plugin.
3. Configuring a Dynatrace data source requires the following parameters:

   - `Name` - The name you want to apply to the Dynatrace data source (default: Dynatrace).
   - `Dynatrace API Type` - The type of Dynatrace instance that you're connecting to. This is either `SaaS` or `Managed Cluster`.
   - `Dynatrace API Token` - This is the API token you generated in the previous step..

   The next two settings depend on whether you are Dynatrace SaaS or managed

   - In a SaaS example of `yfc55578.live.dynatrace.com`, your **Environment ID** would be `yfc55578`.
   - In the Managed example of `yd8888.managed-sprint.dynalabs.io/e/abc99984-3af2-55tt-72kl-0672983gc45`, your **Environment ID** would be `abc99984-3af2-55tt-72kl-0672983gc45` and your **Domain** would be `yd8888.managed-sprint.dynalabs.io`
4. After all of the configuration values have been set, choose **Save & Test** to validate the configuration and save your changes.

Query the data source

Use the query editor to query Dynatrace metrics and problems. The query type can be `metric` or `problem`.

Metric query type

- `Metric`— Select the metric that you want to see. To get the metric list from Dynatrace again, choose **Refresh** button.

- `Aggregations`— Select the aggregation you want to use for a specific metric. Choose the aggregations value to change the aggregation type or choose **+** to add another aggregation.
- `Transformations`— You can select transformations in the query editor. Afterwards, enter a number of parameters into the selected transformation. Currently, only the merge transformation is supported. For more information about the merge transforms, see Merge transformation.
- `Filters`— The Dynatrace data source dynamically queries the appropriate filters for each metric. To add a filter, choose the **+** symbol next to the **Filters** label on the Dynatrace query editor, select which field to filter on, select the operator to use, and then select a value to filter by. The Dynatrace data source allows you to create Filter Groups that you can join together to create complex logical comparisons. For most use cases, Filter Groups are not required. When creating filters with Tags, regardless of the conjunction selected, Dynatrace will always use AND. Dynatrace does not support OR filters with Tags.
- `Alias`— There are two different types of aliases you will encounter while using the Dynatrace data source. The first is a static alias. An alias of this type is available on every query that you build, and the name of the alias starts with a lowercase letter. The second is a dynamic alias, which changes based on the metric that you are using in your query, and the name of the alias starts with an uppercase letter. The Dynatrace plugin supports several different aliases: `Metric Names`, `Aggregation`, `Display Name`, `Host`, and `Description`.

Name	Value
`$name`	builtin:apps.other.keyUserActions.reportedErrorCount.os
`$aggregation`	auto,value
`$displayName`	Reported error count (by key user action, OS) [mobile, custom]

Problems query type

- `Problem Query Type`— Select a problem query type. Currently, only the feed problem query type is supported. For information about the feed problem query type, see Merge transformation
- `Status Filter`— Filter the result problems by the status.

- `Impact Filter`— Filter the result problems by the impact level.
- `Severity Filter`— Filter the result problems by the severity level.
- `Expand Details`— Include related events to the response, if set..

Using template variables

To add a new Dynatrace query variable, see add a new template variable. Use your Dynatrace data source as your data source for the following available queries:

- `Query type`— Select a query type. The query type associates some data with some key or descriptor.

Query type	Description
`Metric names`	Returns a list of all metric names
`Filter keys`	Returns a list of all the possible dimensions (e.g. Hostname) that can be used to filter
`Filter values for key`	Returns a list of all filtered values by a key name or a key name template variable
`Problem status options`	Returns a list of all problem statuses
`Problem impact options`	Returns a list of all problem impacted areas
`Problem severity options`	Returns a list of all problem severity types

- `Regex`— (Optional) Filter out any of the returned values from your query with a regular expression.

> **Note**
>
> `Multi-value` and `Include All option` are currently not supported by the Dynatrace data source.

After creating a variable, you can find it in the **Metric** drop-down menu.

Import a dashboard for Dynatrace

To import a dashboard, see Importing a dashboard. Imported dashboards can be found in **Configuration** > **Data Sources** > select your Dynatrace data source > select the **Dashboards** tab to see available pre-made dashboards.

Connect to a GitLab data source

The GitLab data source allows you to keep track of detailed GitLab statistics, such as top contributors, commits per day, or deployments per day. You can also use template variables, such as projects, to set up filters for your dashboards. You can combine data from the GitLab API with data from other sources.

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.
> Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

Known limitations

Alerting is not supported yet on this plugin because transformations are not supported in alert queries and transformations is the only way to obtain meaningful aggregate metrics from GitLab API raw data.

Adding the data source

1. Open the Grafana console in the Amazon Managed Grafana workspace and make sure you are logged in.
2. In the side menu under **Configuration** (the gear icon), choose **Data Sources**.
3. Choose **Add data source**.

> **Note**
>
> If you don't see the **Data Sources** link in your side menu, it means that your current user does not have the `Admin` role.

4. Select **GitLab** from the list of data sources.
5. Enter the following information:
   - For **Name**, enter a name for this GitLab data source.
   - For **URL**, enter the root URL for your GitLab instance, such as **`https://gitlab.com/api/v4`**.
   - For **Access token**, enter your GitLab personal access token.

Query the GitLab data source

From the GitLab Query Editor you can select different resource types, such as commits, issues, or releases.

Filter and view projects

1. From the dropdown menu, choose **Projects**.
2. (Optional) Filter by the projects that you own.
3. Use the dropdown and select **Yes** or **No** to filter the results.

> **Note**
>
> Fetching all the projects **Owned = No** can take a long time.

Filter and view commits

1. From the dropdown menu, choose **Commits**.
2. Use the input field to add the project ID.
3. (Optional) To filter by branch/tag use the input field to add a branch/tag reference.

Filter and view issues

1. From the dropdown menu, choose **Issues**.
2. Use the input field to add the project ID.
3. (Optional) To filter by title/description, use the input field to search issues based on their **title** and **description**.

View releases

1. From the dropdown menu, choose **Deployments**.
2. Use the input field to add the project ID.
3. (Optional) To filter by environment/status, use the input fields. The **status** attribute can be one of the following values: `created`, `running`, `success`, `failed`, or `canceled`.

View labels

1. From the dropdown menu, choose **Labels**.
2. Use the input field to add the project ID.

Templates and variables

To add a new GitLab query variable, see Adding a query variable. Use your GitLab data source as the data source. Choose a resource type: **Releases**, **Projects**, or **Labels**.

To get a dynamic list of projects, labels, and so on to choose from, create a Query type variable. Query type variables use the GitLab Query Editor to query and return Projects, Labels, and so on. The following example creates a Project variable to parameterize your queries

Create a Project variable to parameterize your queries

1. Add a variable of type **Query** named **`project`**.
2. Select your GitLab data source and refresh **On Dashboard Load**.
3. Select the **Projects** resource type, **Yes** for **Owned**, **name** for **display field** and **id** for **value field** .
4. Choose **Update** to add the variable to the dashboard.
5. Add a new panel to the dashboard and use **`$project`** as the project ID.

   Now, when choosing from the dropdown, you get the results that belong to that project.

Using transformations from Grafana to answer common questions

Now that you can perform basic GitLab queries to find commits, issues, etc, you can use Transformations to visualize, aggregate, group, and join datasets, along with many other types of

transformations to transform simple results into answers for complex questions. Below are a few common questions and how to use transformations to answer them.

How many commits/issues/deployments per day in my project?

1. Add a query. Select **Commits** for the resource type and add the project ID.
2. Add a new **Group by** transformation: for **Group by**, select **created_at_date** and then calculate **(Count)=id**
3. Choose the **Graph** visualization.

What is the average time to close issues in my project?

1. Add a query. Select **Issues** for the resource type and add the project ID.
2. Add a new **Add field from calculation** transformation: for **Mode**, select **Binary Operation**, for **Operation**, select **closed_at = created_at** and for **Alias** choose **resolution_time**.
3. Add a new **Add field from calculation** transformation: for **Mode**, select **Binary Operation**, for **Operation**, select **resolution_time / 86400000** and for **Alias** choose **resolution_time**.

   For **Replace all fields**, choose **True**.
4. Choose the **Stat** visualization.

   - Show = Calculate
   - Calculation = Mean
   - Fields = **resolution_time**

Connect to a Honeycomb data source

The Honeycomb data source allows you to query and visualize Honeycomb metrics and link to Honeycomb traces from within Amazon Managed Grafana.

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.

Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

Known limitations

- This data source does not support ad-hoc queries.
- Because of API limitations, the variable editor can only return the first 1000 unique values for a selected column.
- Because of API limitations, the data source can query only the last 7 days of data.

Adding the data source

1. Open the Grafana console in the Amazon Managed Grafana workspace and make sure you are logged in.
2. In the side menu under **Configuration** (the gear icon), choose **Data Sources**.
3. Choose **Add data source**.
4. Select **Honeycomb** from the list of data sources.

Note

If you don't see the **Data Sources** link in your side menu, it means that your current user does not have the `Admin` role.

Honeycomb settings

Name	Description
`Name`	The data source name. This is how you see the data source in panels, queries, and Explore.
`Honeyc API key`	The API key that you obtained from Honeycomb.

Name	Description
URL	The URL of the Honeycomb API. For example, `https://api.honeycomb.io` .
Team	The Honeycomb team associated with the API key.

Query the Honeycomb data source

To query metrics, enter values into the editor fields:

- Select a dataset.
- The default query is a COUNT over the selected dataset.
- To refine the query, select values for any of the remaining fields, such as **Visualization**, **Visualization**, **Where**, **Constraint**, **Group by**, **Order by**, or **Limit**.

Templates and variables

To add a new Honeycomb query variable, see Adding a query variable.

YOu can create variables containing Datasets, Columns, or Column Values.

- If no dataset is selected, the variable will contain datasets.
- If only a dataset is selected, the variable will contain column names.
- If both a dataset and a column are selected, the variable will contain column values. Column values can be further constrained using the **Where** fields in the editor. .

View query in Honeycomb UI

To see the query you have created in the Honeycomb UI from the dashboard panel, choose any point in the graph and choose **Open in Honeycomb**.

To see the query you have created in the Honeycomb UI from the Query Editor, choose **Open in Honeycomb**.

Import a dashboard for Honeycomb

To import a dashboard, see Importing a dashboard.

To find your imported dashboards, choose **Configuration**, **Data sources**.

To see the avaialble pre-made dashboards, choose the Honeycomb data source and choose the **Dashboards** tab.

Connect to a Jira data source

Get the whole picture of your development process by combining issue data from Jira with application performance data from other sources.

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.
> Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

- Create annotations based on issue creation or resolution, to see the relationship between issues and metrics.
- Track detailed Jira stats, such as mean time to resolution and issue throughput.

In order to use the Jira data source, you need an Atlassian account with access to a Jira project.

Known limitations

Custom field types from Jira addons might not be supported.

Adding the data source

1. Open the Grafana console in the Amazon Managed Grafana workspace and make sure you are logged in.
2. In the side menu under **Configuration** (the gear icon), choose **Data Sources**.
3. Choose **Add data source**.

> **Note**
>
> If you don't see the **Data Sources** link in your side menu, it means that your current user does not have the `Admin` role.

4. Select **Jira** from the list of data sources.
5. Enter the following information:

   - For **Name**, enter a name for this Jira data source.
   - For **URL**, enter the root URL for your Atlassian instance, such as **`https://bletchleypark.atlassian.net`**.
   - For **User**, enter an email address for the user/service account.
   - For **API token**, enter an API token generated for the user.

Query the Jira data source

From the Jira Query Editor you can select fields and query issues.

The Jira data source queries Jira for issues, which can represent bugs, user stories, support tickets, or other tasks in Jira

Filter and view issues

1. Choose **Fields** choose the dropdown and use type-ahead to select from any of the fields in your Jira instance, including custom fields. Some fields to try:

   - **Summary**— The name of the issue
   - **Epic Name**— The epis that an issue belongs to
   - **Story Point Estimate**— The number of story points that the team has estimated for an issue
2. Filter or sort the issues. To do so, enter any valid JQL expression to filter or sort the issues based on any of their fields such as **Project**, **Assignee**, or **Sprint** with the Atlassian query language JQL.

From here, you can display your data in a table or use Grafana transformations to manipulate that issue data, run calculations, or turn the data into a time series graph. For more information, see Applying a transformation.

Time series query

To show time series data, choose a **Date** field along with a numeric field, then switch to graph visualization. For example: **Sprint Start Date**, **Story point estimate**.

The preceding example, on its own, is not very useful. The numeric field can be (and will most likely be) calculated from Transformations. Using the **Group By** Transformation would allow grouping by **Sprint Start Date** and summarizing the **Story point estimate** allowing a visualization of Story Points over time per Sprint. For more information about transformations, see Applying a transformation.

Templates and variables

To add a new Jira query variable, see Adding a query variable. Use your Jira data source as the data source.

You can define variables on your dashboards and reference them in JQL expressions. For example, you can create a project status dashboard and choose between projects, or an epic status dashboard and choose different epics, or a task status dashboard and choose different assignees.

To get a dynamic list of projects, epics, assignees, and so on to choose from, create a Query type variable. Query type variables use JQL to query issues and return Projects, Epics, Assignees, or anything related to issues. The following is an example:

Create an Assignee variable to get the status of issues by Assignee

1. Add a variable of type **Query** named **`assignee`**.
2. Select **Field: Assignee**.
3.)Optional) Add a JQL filter **project = 'your project'**.
4. Choose **Run** to see a list of Assignees.
5. Choose **Update** to add the variable to the dashboard.
6. Add a new panel to the dashboard and edit the JQL to filter using your new variable **assignee = $assignee**.

   Now, when choosing from the dropdown, you see only the issues assigned to that user.

Multi-value variables allow selecting multiple options and can be used as part of the IN clause. For example, **assignee IN ($assignee)**.

Using transformations from Grafana to answer common questions

Macros are variables that reference the Dashboard time window so you can filter issues only within the range of the Dashboard window. There are 2 macros:

- **$__timeFrom**
- **$__timeTo.**

The following example JQL query filters issues created within the dashboard time window: `createdDate >= $__timeFrom AND createdDate <= $__timeTo`

Get the most out of the data source

Using Grafana's transformations and other built-in features can help you meaningly view your Jira data.

Using transformations to augment JQL

While there are many Transformations in Grafana to choose from, the following provide a powerful augmentation to give JQL some of the features/power of SQL.

Group By This transformation provides a key feature that is not part of the standard Jira JQL syntax: Grouping. Using the **Group By** transformation, you can group by Sprints or other Issue fields, and aggregate by group to get metrics like velocity and story point estimates vs actual completed in a Sprint.

Outer Join Similar to SQL joins, you can join 2 or more queries together by common fields. This provides a way to combine datasets from queries and use other transformations to calculate values from multiple queries/datasets.

Add Field from Calculation Similar to SQL expressions, this transformation allows adding new fields to your dataset based on calculations of other fields. The fields used in the calculation can be from a single query or from queries you've joined together. You can also chain together calculations and perform calculations from calculated fields.

Using transformations from Grafana to answer common questions

You can use Transformations to visualize, aggregate, group, and join datasets, along with many other types of transformations to transform simple results into answers for complex questions.

How do I show Velocity per Sprint?

1. Select Fields: **Sprint Name**, **Story point estimate**.
2. Add a JQL filter: `project = "Your Project" AND type != epic AND status = done order by created ASC`

3. Add a **Group By** transformation:
   - Sprint Name | Group By
   - Story Point Estimate | Calculate | Total
4. Choose the **Bar Gauage** visualization.

How do I show what was Completed vs Estimated in a Sprint?

1. Add a query. First, select Fields: **Sprint Name**, **Sprint Start Date,**, **Story point estimate**.

   Then add a JQL filter: `project = 'Your Project' AND type != epic`
2. Add a second query. First, select Fields: **Sprint Name**, **Sprint Start Date,**, **Story point estimate**.

   Then add a JQL filter: `project = 'Your Project' AND type != epic AND status = done`
3. Add a **Group By** transformation:
   - Sprint Name | Group By
   - Sprint Start Date | Group By
   - Story Point Estimate | Calculate | Total
4. Choose the **Graph** visualization.

What is the Average time to complete issues in my project?

1. Add a query. First, select Fields: **Created**, **Status Category Changed**.

   Then add a JQL filter: `project = 'Your Project' AND type != epic AND status = done`
2. Add a transformation: **Add field from calculation**
   - Mode = Reduce Row
   - Calculation = Difference
3. Add a transformation: **Add field from calculation**
   - Mode = Binary Operation
   - Operation = Difference / 86000000

- Alias = Days

4. Add a transformation: **Organize fields**
   - Hide Different field
5. Add a transformation: **Filter data by values**
   - Filter Type = Include
   - conditions = Match any
     - Field = Days | Match = Is Greater | Value = 1
6. Add a transformation: **Reduce**
   - Mode = Series to Rows
   - Calculations = mean
7. Choose the **Stat** visualization.

Connect to a MongoDB data source

The MongoDB Data source enables you to visualize data from MongoDB in Amazon Managed Grafana.

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.
> Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

Usage

Query editor

The query editor supports the same syntax as the MongoDB Shell, with some limitations: * You can only run one command/query. * Only read commands are supported: **find** and **aggregate** * *Most* Object constructors are not supported (with the exception of **ISODate**, which is supported)

The editor expands upon the MongoDB Shell syntax in the following ways:

- **Database selection** – You can supply the name of the database in place of the normal "db":

> **Note**
>
> You can still use "db". It will refer to the default database in your connection string.

```
sample_mflix.movies.find()
```

- **Aggregate sorting** – Normally sorting happens with a step within the aggregate pipeline, however the MongoDB Atlas free tier doesn't allow sorting. We have expanded the syntax to allow it for those using the free tier.

> **Note**
>
> MongoDB doesn't perform the sort with this syntax. The sort happens after the results are queried from the collection.

```
sample_mflix.movies.aggregate({}).sort({"time": 1})
```

- With a blank editor, **Ctrl + Space** will show a selection of all the available databases.
- Entering a dot after the database will show a selection of all the available collections for that database.
- Entering a dot after the collection will show the available query methods.
- Entering a dot after the query method will show additional functions: sort/limit.

Running the query

Press **Cmd + S** to run the query

Time series

When visualizing time series data, the plugin needs to know which field to use as the time. Simply project the field with a name alias of "time". The field data type must be a date.

You can coerce non-date data types to date. Doing so will allow using non-date fields as the time series time. The following example shows how to convert the int field "year" to a date that is projected as "time" using the MongoDB $dateFromParts pipeline operator.

```
sample_mflix.movies.aggregate([
{"$match": { "year": {"$gt" : 2000} }},
{"$group": { "_id": "$year", "count": { "$sum": 1 }}},
{"$project": { "_id": 0, "count": 1, "time": { "$dateFromParts": {"year": "$_id",
 "month": 2}}}}
]
).sort({"time": 1})
```

Diagnostics

Diagnostic Commands

The following diagnostic commands are currently supported: "stats", "serverStatus", "replSetGetStatus", "getLog", "connPoolStats", "connectionStatus", "buildInfo", "dbStats", "hostInfo", "lockInfo"

Examples:

```
admin.connectionStatus()  // run the connectionStatus command
admin.connectionStatus({"authInfo.authenticatedUserRoles": 1})  // run and only return
 the "authInfo.authenticatedUserRoles" field
admin.connPoolStats({arg: "pool"})  // run the connPoolStats command and pass 1
 argument
admin.serverStatus({args: {repl: 0, metrics:0}})  // run the serverStatus command and
 pass multiple args
```

Macros

You can reference the dashboard time range in your queries.

- `$__timeFrom` — a macro that references the dashboard start time
- `$__timeTo` — a macro that references the dashboard end time

```
          $__timeTo -  ``` sample_mflix.movies.find({released: {$gt:
          "$__timeFrom"}}).sort({year: 1})
```

Template variables

MongoDB supports the idea of "Compound Variables", which enable you to use one variable as multiple variables to perform complex multi-key filters.

To create a Compound Variable, use the naming convention of breaking the variables up by using underscores (must start with underscore): `_var1_var2` When querying, the response must be in the format: `val1-val2`

Example: I want to filter results on both movie name and year.

1. Create a variable of type Query: `_movie_year`
2. Set the variable query to a query that will return an array of items with one movie-year property, as shown in the following example.

   ```
   // Example sample_mflix.movies.aggregate([
            {"$match": {year: {"$gt": 2011}}},
            {"$project": {_id: 0, movie_year: {"$concat":
            ["$title", " - ", {"$toString":"$year"}]}}}
            ])
   ```

   ```
    // [{"movie-year": "Ted - 2016"},
            {"movie-year": "The Terminator -
            1985"}]
   ```
3. Now in your query, you can reference "Movie" and "Year" as separate template variables using the syntax "$_variable".

Using ad-hoc filters

In addition to the standard "ad-hoc filter" type variable of any name, a second helper variable must be created. It should be a "constant" type with the name `mongodb_adhoc_query` and a value compatible with the query editor. The query result will be used to populate the selectable filters. You can choose to hide this variable from view as it serves no further purpose.

```
sample_mflix.movies.aggregate([
{"$group": { "_id": "$year"}},
{"$project": { "year": "$_id","_id":
0 }} ] )
```

Connect to a New Relic data source

This section covers New Relic APM and Insights for Grafana.

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.
> Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

Features

- Template variables
  - Metric names
  - Metric values
- Annotations
- Aliasing
  - Metric names
  - Metric values
- Ad-hoc filters
  - Not currently supported
- Alerting

Configuration

Add the data source, filling out the fields for your admin API key, personal API key and account ID.

Usage

Service types

- **Metrics**; for querying New Relic APM via New Relic's REST API.
- **Insights**; for querying New Relic Insights via NRQL.

Aliases

You can combine plaintext with the following variables to produce custom output.

Variable	Description	Example value
`$__nr_metric`	Metric name	CPU/User time
`$__nr_metric_value`	Metric values	average_value

For example:

```
<para>
  Server: $__nr_server Metric: $__nr_metric
</para>
<programlisting>
```

Templates and variables

1. Create a template variable for your dashboard. For more information, see Templates and variables.
2. Select the "Query" type.
3. Select the "New Relic" data source.
4. Formulate a query using relative REST API endpoints (excluding file extensions).

List of available applications:

```
<para>
  applications
```

```
    </para>
    <programlisting>
```

List of available metrics for an application:

```
    <para>
      applications/{application_id}/metrics
    </para>
    <programlisting>
```

NRQL macros

To improve the writing experience when creating New Relic Query Language (NRQL) queries, the editor supports predefined macros:

- `$__timeFilter` (or `[[timeFilter]]`) will interpolate to `SINCE <from> UNTIL <to>` based on your dashboard's time range.

Example:

```
    <para>
      SELECT average(value) FROM $event_template_variable
      $__timeFilter TIMESERIES
    </para>
    <programlisting>
```

For further hints on how to use macros and template variables, refer to the editor's help section.

Alert events

Select your New Relic data source and set additional filters. Without any filters set, all events will be returned.

If you want to filter events by *Entity ID*, use template variables because you will be able to select the entity name instead of ID. For example, to filter events for a particular application, create a variable _$app_ which retrieves a list of apps and uses it as an *Entity ID* filter.

Deployment events

Application ID is a required field.

Connect to an Oracle Database data source

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.
> Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

Adding the data source

Select **Data sources** on the left panel of Grafana.

Select Add Datasource:

Enter **`oracle`** to find the data source.

Enter Oracle server details.

Enter a hostname (or IP address) along with the port number, and the username and password to connect.

With the tnsnames option toggle, any valid entry found in your tnsnames.ora configuration file can be used, along with basic authentication.

Similar to the previous example, but using Kerberos for authentication. See the kerberos specific setup guide for details on how to configure the OS or docker container to use kerberos.

Optionally change the time zone used to connect to the Oracle server and to be used by timezone aware macros. The default setting is UTC.

Save and Test the data source, you should see a green message with "Database Connection OK"

Usage

Macros

To simplify syntax and to allow for dynamic parts, such as date range filters, the query can contain macros. The column name must be contained within double-quotes ('').

Macro example	Description
$__time(dateColumn)* \| Will be replaced by an expression to rename the column to `time`. For example, `dateColumn as time` *$__timeEpoch(dateColumn)	Will be replaced by an expression to rename the column to `time` and converting the value to unix timestamp (in milliseconds).
$__timeFilter(dateColumn)* \| Will be replaced by a time range filter using the specified column name. For example, `dateColumn BETWEEN TO_DATE('19700101','yyyymmdd') + (1/24/60/60/1000) * 1500376552001 AND TO_DATE('19700101','yyyymmdd') + (1/24/60/60/1000) * 1500376552002` *$__timeFrom()	Will be replaced by the start of the currently active time selection converted to `DATE` data type. For example, `TO_DATE('19700101','yyyymmdd') + (1/24/60/60/1000) * 1500376552001 .`
$__timeTo()* \| Will be replaced by the end of the currently active time selection converted to `DATE` data type. *$__timeGroup(dateColumn,"5m")	Will be replaced by an expression usable in GROUP BY clause.
$__timeGroup(dateColumn,"5m"[, fillvalue])	Will be replaced by an expression usable in GROUP BY clause. Providing a fillValue of NULL or floating value will automatically fill empty series in time range with that value. For example, *$__timeGroup(dateColumn,'5m', 0)*.
$__timeGroup(dateColumn,"5m", previous)	Same as above but the previous value in that series will be used as fill value if no value has been seen yet NULL will be used.
$__unixEpochFilter(dateColumn) | Will be replaced by a time range filter using the specified column name with times represented as unix timestamp (in milliseconds). For example, `dateColumn >= 1500376552001`	Will be replaced by the start of the currently active time selection as unix timestamp. For example, `1500376552001 .`

Macro example	Description
AND dateColumn <= 1500376552002` * $__unixEpochFrom()*	
$__unixEpochTo()	Will be replaced by the end of the currently active time selection as unix timestamp. For example, `1500376552002` .

The plugin also supports notation using braces `{}`. Use this notation when queries are needed inside parameters.

> **Note**
>
> Use one notation type per query. If the query needs braces, all macros in the query must use braces.

```
$__timeGroup{"dateColumn",'5m'}
$__timeGroup{SYS_DATE_UTC("SDATE"),'5m'}
$__timeGroup{FROM_TZ(CAST("SDATE" as timestamp), 'UTC'), '1h'}
```

The query editor has a **Generated SQL** link that shows up after a query has run, while in panel edit mode. When you choose the link, it expands and shows the raw interpolated SQL string that was run.

Table queries

If the **Format as** query option is set to **Table** then you can basically do any type of SQL query. The table panel will automatically show the results of whatever columns & rows your query returns. You can control the name of the Table panel columns by using regular `as` SQL column selection syntax.

Time series queries

If you set **Format as** to **Time series**, for use in Graph panel for example, the query must return a column named `time` that returns either a SQL datetime or any numeric data type representing unix epoch in seconds. Grafana interprets DATE and TIMESTAMP columns without explicit time

zone as UTC. Any column except `time` and `metric` is treated as a value column. You can return a column named `metric` that is used as metric name for the value column.

The following code example shows the `metric` column.

```
SELECT
  $__timeGroup("time_date_time", '5m') AS time,
  MIN("value_double"),
  'MIN' as metric
FROM test_data
WHERE $__timeFilter("time_date_time")
GROUP BY $__timeGroup("time_date_time", '5m')
ORDER BY time
```

More queries – using oracle-fake-data-gen

```
SELECT
  $__timeGroup("createdAt", '5m') AS time,
  MIN("value"),
  'MIN' as metric
FROM "grafana_metric"
WHERE $__timeFilter("createdAt")
GROUP BY $__timeGroup("createdAt", '5m')
ORDER BY time
```

The following code example shows a Fake Data time series.

```
SELECT
  "createdAt",
  "value"
FROM "grafana_metric"
WHERE $__timeFilter("createdAt")
ORDER BY "createdAt" ASC
```

```
SELECT
  "createdAt" as time,
  "value" as value
```

```
FROM "grafana_metric"
WHERE $__timeFilter("createdAt")
ORDER BY time ASC
```

The following example shows a useful table result.

```
select tc.table_name Table_name
,tc.column_id Column_id
,lower(tc.column_name) Column_name
,lower(tc.data_type) Data_type
,nvl(tc.data_precision,tc.data_length) Length
,lower(tc.data_scale) Data_scale
,tc.nullable nullable
FROM all_tab_columns tc
,all_tables t
WHERE tc.table_name = t.table_name
```

Templating

Instead of hardcoding things such as server, application and sensor name in you metric queries you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. These dropdown boxes makes it easy to change the data being displayed in your dashboard.

Query variable

If you add a template variable of the type `Query`, you can write a Oracle query that can return things such as measurement names, key names or key values that are shown as a dropdown select box.

For example, you can have a variable that contains all values for the `hostname` column in a table if you specify a query like this in the templating variable *Query* setting.

```
SELECT "hostname" FROM host
```

A query can return multiple columns and Grafana will automatically create a list from them. For example, the following query will return a list with values from `hostname` and `hostname2`.

```
SELECT "host.hostname", "other_host.hostname2" FROM host JOIN other_host ON host.city =
 other_host.city
```

To use time range dependent macros such as `$__timeFilter("time_column")` in your query the refresh mode of the template variable needs to be set to *On Time Range Change*.

```
SELECT "event_name" FROM event_log WHERE $__timeFilter("time_column")
```

Another option is a query that can create a key/value variable. The query should return two columns that are named `__text` and `__value`. The `__text` column value should be unique (if it is not unique then the first value is used). The options in the dropdown list will have a text and value that allows you to have a friendly name as text and an id as the value. The following example code shows a query with `hostname` as the text and `id` as the value.

```
SELECT "hostname" AS __text, "id" AS __value FROM host
```

You can also create nested variables. For example, if you had another variable named `region`. Then you could have the hosts variable only show hosts from the current selected region with a query like this (if `region` is a multi-value variable then use the `IN` comparison operator rather than = to match against multiple values).

```
SELECT "hostname" FROM host WHERE region IN('$region')
```

Using variables in queries

Template variable values are only quoted when the template variable is a `multi-value`.

If the variable is a multi-value variable then use the `IN` comparison operator rather than = to match against multiple values.

There are two syntaxes:

$<varname> Example with a template variable named `hostname`:

```
SELECT
  "atimestamp" as time,
```

```
  "aint" as value
FROM table
WHERE $__timeFilter("atimestamp") AND "hostname" IN('$hostname')
ORDER BY "atimestamp" ASC
```

`[[varname]]` Example with a template variable named `hostname`:

```
SELECT
  "atimestamp" as time,
  "aint" as value
FROM table
WHERE $__timeFilter("atimestamp") AND "hostname" IN('[[hostname]]')
ORDER BY atimestamp ASC
```

Connect to a Salesforce data source

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.
> Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

The Salesforce data source allows you to visualize data from Salesforce within Amazon Managed Grafana.

To use this data source, you must have a Salesforce account and a Salesforce Connected App.

Known limitations

- Ad-hoc filters are not supported yet.
- Only SOQL queries, and data that is accessible via SOQL are currently supported. SOSL and SAQL query formats are not yet supported.

Required settings

The following settings are required.

Note

The plugin currently uses the OAuth 2.0 Username-Password Flow. The required callback URL in the Connected App is not used. Thus, you can set it to any valid URL.

Name	Description
Enable OAuth settin	You must check this to enable OAuth.
Callba URL	Not used in this plugin, so you can specify any valid URL.
Select OAuth Scopes (minim requir nts)	Access and Manage your data (api).
Requir Secret for Refres Token Flow	You can either enable or disable this.

Adding the data source

1. Open the Grafana console in the Amazon Managed Grafana workspace and make sure you are logged in.
2. In the side menu under **Configuration** (the gear icon), choose **Data Sources**.
3. Choose **Add data source**.

> **Note**
>
> If you don't see the **Data Sources** link in your side menu, it means that your current user does not have the `Admin` role.

4. Select **Salesforce** from the list of data sources.
5. Enter the following information:

   - For **User Name**, enter the username for the Salesforce account that you want to use to connect and query Salesforce.
   - For **Password**, enter the password for that user.
   - For **Security Token**, enter the security token for that user.
   - For **Consumer Key**, enter A Consumer key to connect to Salesforce. You can obtain this from your Salesforce Connected App.
   - For **Consumer Secret**, enter A Consumer secrete to connect to Salesforce. You can obtain this from your Salesforce Connected App.
   - For **Use Sandbox**, select this if you want to use a Salesforce sandbox.

Query the Salesforce data source

The query editor supports the modes Query Builder and SOQL Editor. SOQL stands for Salesforce Object Query Language.

Query Builder (SOQL Builder)

Query Builder is a user friendly interface for building SOQL queries. If you are not familiar with writing SOQL queries, you can use this mode to build the SOQL to query Salesforce objects. The **FROM** field in the query builder refers to the entity or entities in Salesforce. You need to select the **FROM** field before any other operation in the query builder. After you choose the **FROM** field, you need to choose the builder mode. SOQL Builder currently supports the following modes.

- `List`— List the items with their fields from the selected table/salesforce. Use this mode to get results such as, "Show me list of opportunities created in this fiscal quarter along with their name, value, and stage."

- `Aggregate`— Aggregate the items in an entity. Use this mode to get results such as,"Count the opportunities created in last month." or "What is the total value of the opportunities grouped by their stage name?"
- `Trend`— Display the aggregated results over time. Use this mode to get results such as, "Count the number of opportunities by CreatedDate." or "What is the total sum of value grouped by opportunities' closing dates."

After you choose the `Entity/FROM` and the **mode** in the query editor, build your query using the following options.

Fields	Appplicable to	Descriptions
SELECT	ALL	Select the list of fields that you want to see. For the aggregate or trend view, also select how you want to aggregate the values.
WHERE	ALL	(Optional) Specify the filter conditions. The results are filtered based on the conditions that you select.
ORDER BY	LIST, AGGREGATE	(Optional) Select the field name and the sort order that you want for the results.
LIMIT	LIST, AGGREGATE	(Optional) Limit the number fo results returned. The default is 100.
GROUP BY	AGGREGATE	(Optional) Select the field if you want to split the aggregated value by any specific field.
TIME FIELD	TREND	Specify the date field by which you want to group your results. Results are filtered based on Grafana's time picker range.

s you configure the preceding fields in the query editor, you will also see a preview of generated SOQL below the query editor. If you are blocked with any limitations in the query builder, you can safely switch to SOQL Editor, where you can customize the generated SOQL query.

SOQL editor

The raw SOQL editor provides the option to query Salesforce objects via raw a SOQL query. The SOQL editor provides autocomplete suggestions, such as available entities per tables and corresponding fields. Use Ctrl+Space after SELECT or WHERE to see the available entities per tables. You can see the available fields, if you enter a dot after the entity name.

Shortcuts

Use CTRL + SPACE to show code completion, which shows available contextual options.

CMD + S runs the query.

Query as time series

Make a time series query by aliasing a date field to time, and a metric field to metric, then grouping by the metric and date. The following is an example:

```
SELECT sum(Amount) amount, CloseDate time, Type metric from Opportunity
group by Type, CloseDate
```

Macros

To filter by the dashboard time range, you can use Macros in your SOQL queries:

- `$__timeFrom`— Will be replaced by the start of the currently active time selection converted to the `time` data type.
- `$__timeTo`— Will be replaced by the end of the currently active time selection converted to the `time` data type.
- `$__quarterStart`— The start of the fiscal quarter (derived from SalesForce Fiscal Year Settings).
- `$__quarterEnd`— The end of the fiscal quarter (derived from SalesForce Fiscal Year Settings).

```
SELECT UserId, LoginTime from LoginHistory where LoginTime > $__timeFrom
```

Templates and variables

To add a new Salesforce query variable, see Adding a query variable. Use your Salesforce data source as your data source. You can use any SOQL query here.

If you want to use name/value pairs, for example a user id and user name, return two fields from your SOQL query. The first field will be used as the ID. Do this when you want to filter by key (ID, etc) in your query editor SOQL.

Use the variable in your SOQL queries by using Variable syntax. For more information, see Variable syntax.

Connect to an SAP HANA data source

SAP HANAis a high-performance, in-memory database that speeds up data-driven, real-time decisions and \ actions. It is developed and marketed by SAP. The SAP HANA data source plugin helps you to connect your SAP HANA instance with Grafana.

With the SAP HANA Grafana Enterprise plugin, you can visualize your SAP HANA data alongside all of your other data sources in Grafana as well as log and metric data in context. This plugin includes a built-in query editor, supports annotations, and it allows you to set alerting thresholds, control access, set permissions, and more.

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.
> Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

Features

- **Query editor**— The plugin comes with an built-in SQL query editor with syntax highlighting that allows you to visualize time series or table data and auto completes basic Grafana macros.
- **Data source permissions**— Control who can view or query SAP HANA data in Grafana.
- **Annotations**— Overlay SAP HANA events or data on any Grafana graph to correlate events with other graph data.
- **Alerting**— Set alerts-based metrics stores in SAP HANA.
- **Variables for queries**— Create template variables in Grafana, which are based on SAP HANA data, and include variables in SAP HANA queries to make dashboards interactive.

Adding the data source

1. Open the Grafana console in the Amazon Managed Grafana workspace and make sure you are logged in.
2. In the side menu under **Configuration** (the gear icon), choose **Data Sources**.
3. Choose **Add data source**.

> **Note**
>
> If you don't see the **Data Sources** link in your side menu, it means that your current user does not have the `Admin` role.

4. Select **SAP HANA** from the list of data sources.
5. In the Config editor, enter the following information:

   - For **Server address**, Provide the address of the SAP HANA instance. Example : `xxxxxxx-xxxx-xxxx-xxxx-xxxxxxx.hana.trial-us10.hanacloud.ondemand.com`.
   - For **Server port**, provide the port of the SAP HANA instance.
   - For **Username**, enter the username to use to connect to the SAP HANA instance.
   - For **Password**, enter the password for this user.
   - (Optional) Enable **Skip TLS verify** if you want to skip TLS verification.
   - (Optional) Enable **TLS Client Auth** if you need to provide a client cert and key.
   - (Optional) Enable **With CA cert** if you want to enable verifying self-signed TLS Certs.
   - (Optional) For **Default schema**, enter a default schema to be used. If you omit this, you will need to specify the schema in every query.

Access and permissions

To connect Grafana to SAP HANA, use dedicated credentials. Only provide required permissions to the user. First, create a restricted user with username and password. The following query is an example to create a restricted user. This query also disables the force password change.

```
CREATE RESTRICTED USER <USER> PASSWORD <PASSWORD> NO FORCE_FIRST_PASSWORD_CHANGE;
```

Next, allow the the user to connect the system through clients such as Grafana with the following:

```
ALTER USER <USER> ENABLE CLIENT CONNECT;
```

Finally, give the user access to the necessary views, tables, and schemas.

```
ALTER USER <USER> GRANT ROLE PUBLIC;
GRANT SELECT ON SCHEMA <SCHEMA> TO <USER>;
```

User level permissions

Limit access to SAP HANA by clicking on the Permissions tab in the data source configuration page to enable data source permissions. On the permission page, Admins can enable permissions and restrict query permissions to specific Users and Teams.

Query editor

The SAP HANA Grafana plugin comes with an SQL query editor where you can enter any HANA queries. If your query return timeseries data, you can format it as timeseries for visualizing them in a graph panel. The query editor provides auto completion for supported Grafana macros and syntax highlighting of your SQL query.

Annotations

You can use SAP HANA queries as the sources of Grafana annotations. Your annotation query should return at least one time column and one text column. For more information about annotations, see Annotations.

To create annotations from SAP HANA

1. Choose the **Dashboard settings** gear icon.
2. From the left menu, choose **Annotations**, **New**.
3. From the **Data source** drop-down menu, select your SAP HANA data source instance.
4. In the **Query** field, enter a SAP HANA query that returns at least one time field and one text field.
5. In the **Format as** drop-down menu, select **Time Series**.
6. For each annotation, configure the **From** fields.

Templates and variables

To add a new SAP HANA query variable, see Adding a query variable. Use your SAP HANA data source as your data source.

The following example query returns the distinct list of `username` from the `users` table.

```
select distinct("username") from "users"
```

> **Note**
>
> Be sure to only select one column in your variable query. If your query returns two columns, the first column will used as display value and the second column will be used as the actual value of the variable. If your query returns more than two columns, they will be rejected.

Templates and variables

You can use any Grafana variable in your query. The following examples shows how to use the single / multi variable in your query.

```
-- For example, following query
select * from "users" where "city" = ${city}
-- will be translated into
select * from "users" where "city" = 'london'
--- where you can see ${city} variable translated into actual value in the variable
```

Similar to text, variables also works for numeric fields. In the below example, `${age}` is a text box variable where it accepts numbers and then compares against the numeric field in the table.

```
select * from "users" where "age" > ${age}
--- wil be translated into
select * from "users" where "age" > '36'
```

If your variable returns multiple values, then you can use it in SAP HANA query's `in` condition like below. Note the brackets surrounding the variable to make the `where in` condition valid in SAP HANA.

```
select * from "users" where "city" in (${cities})
```

```
--- will be translated into
select * from "users" where "city" in ('london','perth','delhi')
--- where you can see ${cities} turned into a list of grafana variables selected.
--- You can also write the same query using shorthand notation as shown below
select * from "users" where "city" in ($cities)
```

Macros

- `$__timeFilter(<time_column>)`— Applies Grafana's time range to the specified column when used in the raw query. Applicable to date/timestamp/long time columns.
- `$__timeFilter(<time_column>, <format>)`— Same as above. But gives the ability to specify the format of the time_column stored in the database.
- `$__timeFilter(<time_column>, "epoch", <format>)`— Same as above but can be used when your time column is in epoch. format can be one of 's','ms' and 'ns'.
- `$__fromTimeFilter(<time_column>)`— Same as above but can be used when your time column is in epoch. format can be one of 's','ms' and 'ns'.
- `$__fromTimeFilter(<time_column>, <comparison_predicate>)`— Same as above but able to specify comparison_predicate.
- `$__fromTimeFilter(<time_column>, <format>)`— Same as above but able to specify format of the time column.
- `$__fromTimeFilter(<time_column>, <format>, <comparison_predicate>)`— Same as above but able to specify comparison_predicate.
- `$__toTimeFilter(<time_column>)`— Returns time condition based on grafana's to time over a time field.
- `$__toTimeFilter(<time_column>, <comparison_predicate>)`— Same as above but able to specify comparison_predicate.
- `$__toTimeFilter(<time_column>, <format>)`— Same as above but able to specify format of the time column.
- `$__toTimeFilter(<time_column>, <comparison_predicate>)`— Same as above but able to specify comparison_predicate.
- `$__timeGroup(<time_column>, <interval>)`— Expands the time column into interval groups. Applicable to date/timestamp/long time columns..

$__timeFilter(<time_column>) macro

The following example explains the `$__timeFilter(<time_column>)` macro:

```
- In the following example, the query
select ts, temperature from weather where $__timeFilter(ts)
--- will be translated into
select ts, temperature from weather where ts > '2021-02-24T12:52:48Z' AND ts <
 '2021-03-24T12:52:48Z'
--- where you can see the grafana dashboard's time range is applied to the column ts in
 the query.
```

$__timeFilter(<time_column>, <format>) macro

In some cases, time columns in the database are stored in custom formats. The following example explains the `$__timeFilter(<time_column>, <format>)` macro, which helps to filter custom timestamps based on grafana's time picker:

```
SELECT TO_TIMESTAMP("TS",'YYYYMMDDHH24MISS') AS METRIC_TIME , "VALUE" FROM "SCH"."TBL"
 WHERE $__timeFilter("TS","YYYYMMDDHH24MISS") -- TS is in 20210421162012 format
SELECT TO_TIMESTAMP("TS",'YYYY-MON-DD') AS METRIC_TIME , "VALUE" FROM "SCH"."TBL" WHERE
 $__timeFilter("TS","YYYY-MON-DD") -- TS is in 2021-JAN-15 format
```

In the macro, the format can be one of the valid HANA formats matchting your timestamp column. For example, `YYYYMMDDHH24MISS` is a valid format when your data is stored in `20210421162012` format.

$__timeFilter(<time_column>, "epoch" <format>) macro

In some cases, timestamps are stored as epoch timestamps in your DB. The following example explains the `$__timeFilter(<time_column>, "epoch" <format>)` macro which helps to filter epoch timestamps based on grafana's time picker. In the macro, format can be one of ms,s or ns. If not specified, s will be treated as default format.

```
SELECT ADD_SECONDS('1970-01-01', "TIMESTAMP") AS "METRIC_TIME", "VALUE" FROM
 "SCH"."TBL" WHERE $__timeFilter("TIMESTAMP","epoch") -- Example : TIMESTAMP field
 stored in epoch_second format 1257894000
SELECT ADD_SECONDS('1970-01-01', "TIMESTAMP") AS "METRIC_TIME", "VALUE" FROM
 "SCH"."TBL" WHERE $__timeFilter("TIMESTAMP","epoch","s") -- Example : TIMESTAMP field
 stored in epoch_second format 1257894000
SELECT ADD_SECONDS('1970-01-01', "TIMESTAMP"/1000) AS "METRIC_TIME", "VALUE" FROM
 "SCH"."TBL" WHERE $__timeFilter("TIMESTAMP","epoch","ms") -- Example : TIMESTAMP field
 stored in epoch_ms format 1257894000000
```

```
SELECT ADD_SECONDS('1970-01-01', "TIMESTAMP"/1000000000) AS "METRIC_TIME", "VALUE" FROM
 "SCH"."TBL" WHERE $__timeFilter("TIMESTAMP","epoch","ns") -- Example : TIMESTAMP field
 stored in epoch_nanoseconds format 1257894000000000000
```

Instead of using third argument to the $__timeFilter, you can use one of epoch_s, epoch_ms or epoch_ns as your second argument..

```
SELECT ADD_SECONDS('1970-01-01', "TIMESTAMP"/1000) AS "METRIC_TIME", "VALUE" FROM
 "SCH"."TBL" WHERE $__timeFilter("TIMESTAMP","epoch","ms")
-- is same as
SELECT ADD_SECONDS('1970-01-01', "TIMESTAMP"/1000) AS "METRIC_TIME", "VALUE" FROM
 "SCH"."TBL" WHERE $__timeFilter("TIMESTAMP","epoch_ms")
```

$__fromTimeFilter() and $__toTimeFilter() macros

The `$__fromTimeFilter()` macro expands to a condition over a time field based on Grafana time picker's from time.

This accepts three parameters. First parameter is time field name. You can pass comparison_predicate or format of the time column as second argument. If you want to pass both, then format is second parameter and use comparison_predicate as your third parameter.

<format> If the format is not specified, plugin wil assume that the time column is of timestamp/date type. If your time column is stored in any other format than timestamp/date, then pass the format as second argument. <format> can be one of epoch_s, epoch_ms,epoch_ns or any other custom format like YYYY-MM-DD.

<comparison_predicate> optional parameter. If not passed, plugin will use > as comparison predicate. <comparison_predicate> can be one of =, !=, <>, <, <=, >, >=

`$__toTimeFilter()` works the same as $__fromTimeFilter(). Instead of using Grafana's from time, it will use to time. Also the default comparison predicate will be <.

$__timeGroup(<time_column>, <interval>)

For example, the macro $__timeGroup(timecol,1h) is expanded to SERIES_ROUND("timecol", 'INTERVAL 1 HOUR') in the query.

The following example explains the `$__timeGroup(<time_column>, <interval>) macro.`

```
SELECT $__timeGroup(timestamp,1h),  "user", sum("value") as "value"
```

```
FROM "salesdata"
WHERE $__timeFilter("timestamp")
GROUP BY $__timeGroup(timestamp,1h), "user"
ORDER BY $__timeGroup(timestamp,1h) ASC
```

This is translated into the following query where `$__timeGroup(timestamp,1h)` is expanded into `SERIES_ROUND("timestamp", 'INTERVAL 1 HOUR')`.

```
SELECT SERIES_ROUND("timestamp", 'INTERVAL 1 HOUR') as "timestamp",  "user",
 sum("value") as "value"
FROM "salesdata"
WHERE "timestamp" > '2020-01-01T00:00:00Z' AND "timestamp" < '2020-01-01T23:00:00Z'
GROUP BY SERIES_ROUND("timestamp", 'INTERVAL 1 HOUR'), "user"
ORDER BY "timestamp" ASC
```

Note

When using group by with $__timeGroup macro, make sure that your select, sort by fields follows the same name as your group by field. Otherwise, HANA might not recognize the query.

If you don't want to hard code the interval in $__timeGroup() function, then you can leave that to Grafana by specifying $__interval as your interval. Grafana will calculate that interval from dashboard time range. Example query:

```
SELECT $__timeGroup(timestamp, $__interval), sum("value") as "value"
FROM "salesdata"
WHERE $__timeFilter("timestamp")
GROUP BY $__timeGroup(timestamp, $__interval)
ORDER BY $__timeGroup(timestamp, $__interval) ASC
```

That query is translated into the followin query based on the dashboard time range.

```
SELECT SERIES_ROUND("timestamp", 'INTERVAL 1 MINUTE'), sum("value") as "value"
FROM "salesdata"
WHERE "timestamp" > '2019-12-31T23:09:14Z' AND "timestamp" < '2020-01-01T23:17:54Z'
GROUP BY SERIES_ROUND("timestamp", 'INTERVAL 1 MINUTE')
ORDER BY SERIES_ROUND("timestamp", 'INTERVAL 1 MINUTE') ASC
```

Alerting

To set up a SAP HANA alert in Grafana

1. Create a graph panel in your dashboard.
2. Create a SAP HANA query in time series format.
3. Choose the **Alert** tab and specify the alerting criteria.
4. Choose **Test Rule** to test the alert query.
5. Specify the alert recipients, message, and error handling.
6. Save the dashboard.

Alerting on non-timeseries data

To alert on non-timeseries data, use the `TO_TIMESTAMP('${__to:date}')` macro to make non-timeseries metrics into timeseries. This will convert your metric into single point time series query. The format of the query is given below

```
SELECT TO_TIMESTAMP('${__to:date}'),  <METRIC> FROM <TABLE# WHERE <YOUR CONDITIONS>
```

In the following example, a table has four fields called username, age, city and role. This table doesn't have any time field. We want to notify when the number of users with dev role is less than three.

```
SELECT  TO_TIMESTAMP('${__to:date}'), count(*) as "count" FROM (
   SELECT 'John' AS "username", 32 AS "age", 'Chennai' as "city", 'dev' as "role" FROM
 dummy
   UNION ALL SELECT 'Jacob' AS "username", 32 AS "age", 'London' as "city",
 'accountant' as "role" FROM dummy
   UNION ALL SELECT 'Ali' AS "username", 42 AS "age", 'Delhi' as "city", 'admin' as
 "role" FROM dummy
   UNION ALL SELECT 'Raja' AS "username", 12 AS "age", 'New York' as "city", 'ceo' as
 "role" FROM dummy
   UNION ALL SELECT 'Sara' AS "username", 35 AS "age", 'Cape Town' as "city", 'dev' as
 "role" FROM dummy
   UNION ALL SELECT 'Ricky' AS "username", 25 AS "age", 'London' as "city",
 'accountant' as "role" FROM dummy
   UNION ALL SELECT 'Angelina' AS "username", 31 AS "age", 'London' as "city", 'cxo' as
 "role" FROM dummy
) WHERE "role" = 'dev'
```

Connect to a ServiceNow data source

This is the ServiceNow data source that is used to connect to ServiceNow instances.

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.
> Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

Features

- Queries
  - Stat API Queries
  - Table API Queries
    - Incidents, Changes, and any other table
- Alerts
- Annotations (beta feature)
- Template Variables

Configuration

Select data sources on the left panel of Grafana.

Select Add Datasource:

Enter **`servicenow`** to find the data source plugin:

Enter ServiceNow URL:

Choose **Save & Test**. You should see a green message with "ServiceNow Connection OK".

Example dashboards

Pre-made dashboards are included with the plugin and can be imported through the data source configuration page, under the dashboards tab.

Usage

There are two ways to return data in the query editor.

- TableAPI
- AggregateAPI

Users can currently choose between querying pre-defined tables, such as the following:

- Changes
- Incidents

Or, as of `v1.4.0`, an API-driven list of tables and fields using the **Other (Custom Table)** option. This option will allow you to query data that is in any table available to the user used to set up the ServiceNow data source.

The **Custom Table** option should support all of the same features as the pre-defined table lists.

TableAPI queries

The TableAPI returns data suitable for displaying in a table panel. It allows for an ordered selection of fields to display plus filtering options. The query editor also provides a field to limit the number of rows returned by a query.

Example table panel showing results from the previous query.

Show

The *Show* row provides a selector for a field to be displayed. Multiple fields can be also be specified. The fields will be returned in the exact order specified.

Display Values

The *Display Values* flag will cause the query to return human-friendly values, or display vaules, instead of numeric values.

For example, a severity of 1 without this flag would only display 1. If the flag is enabled, the value displayed will be `1 - High`.

According to the ServiceNow API documentation, this can have a negative performance impact.

> **Note**
>
> […] specifying the display value can cause performance issues since it is not reading directly from the database and could include referencing other fields and records.

Filters (general)

The *Filters* row provides the ability to narrow down the displayed rows based on multiple field and value criteria.

All filters are combined with an *AND* or an *OR* operation.

The following fields are available when not using a custom table (this list will expand in the future).

```
Active
Asset
Group
Assigned To
Escalation
Issue Number
Description
Priority
State
Type
Change Risk
Change State
Start Date
End Date
On Hold
```

When selecting a custom table, fields are automatically populated from the Service Now API.

Date filters

Time Field	Operators	Value
Opened At	At or Before Today Not Today Before At or Before After At or After	timestamp javascript:gs.daysAgo(30)

Time Field	Operators	Value
Activity Due		
Closed At		
Due Date		
Expecte Start		
Reopen Time		
Resolve At		
Work End		
Work Start		
Ignore Time		

For additional date values, see: https://developer.servicenow.com/app.do#!/api_doc?v=newyork&id=r_SGSYS-dateGenerate_S_S

Operators (general, string-based)

- Starts With
- Ends With
- Like

- Not Like
- Equals
- Not Equals
- Is Empty

Operators (time-based)

- Today
- Not Today
- Before
- At or Before
- After
- At or After

Values

Value selection depends on the type of filter selected.

- Boolean filters have True/False options
- Text filters will allow typing any value
- Escalation, Priority has a fixed set of numerical values

Sort By

The *Sort By* row provides the ability to narrow down the displayed rows based on multiple field and value criteria.

All filters are combined with an *AND* operation. Support for additional operators will be added.

Limit

A row limit can be specified to prevent returning too much data. The default value is 25.

Time Field

The `Time Field` is what turns your queried data into a time series. Your data being handled as a time series means that values in your selected "time field" that do not fall within your dashboard / panel's time range will not be displayed.

The default time field used is "Opened At", but can be changed to any available field that holds a time value.

A special value "Ignore Time" is provided to allow results "up until now" and also to enable the filters to control what data is displayed.

AggregateAPI queries (Stats)

The AggregateAPI will always return metrics, with the following aggregations: avg, min, max, sum. Filtering is also available to narrow queries.

Show

The *Show* row provides a selector for a metric to be displayed. Multiple metrics can be also be specified.

Filters (general)

Aggregate *Filters* provide the ability to narrow down the displayed metrics based on field and value criteria, similar to the table option.

All filters are combined with an *AND* operation. Support for additional operators will be added.

Stat filter options are the same as the TableAPI.

Aggregation

There are four types of metric aggregations, plus a "count":

- Average
- Minimum
- Maximum
- Sum
- Count - this returns the "number" of metrics returned by a query

Group By

This selector provides the ability to split metrics into lesser aggregates. Grouping by "priority" would return the metrics with a "tag" of priority and the unique values separated.

Templating

Instead of hardcoding names in your queries, you can use variables in their place. Variables are shown as dropdown select boxes at the top of the dashboard. You can use these dropdown boxes to change the data being displayed on your dashboard.

See the example in the **Query Variable** section on how to add a query variable and reference that with a Template value.

Query variable

If you add a template variable of the type `Query`, you can write a query that can return items such as category names, key names, or key values that are shown as a dropdown select box.

For example, you can have a variable that contains all values for `categories` by specifying a query such as this in the templating variable *Query* setting.

When choosing the **Query** setting, a **Filter** section is displayed, allowing you to choose a **Type** and **Field**. Currently, **Type** is limited to Incidents and Changes. When selecting a type, you are provided with a list of fields applicable to that Type. Once a **Type** and **Field** are selected, a preview of values will be displayed at the bottom showing the options available for that Type/Field. Those values will be displayed in a dropdown list on the Dashboard, which you can use along with Templating to filter data on your Dashboard Panels.

For example, if you add a Variable named *category* then select Type = Incidents and Field = Category, you will see a list of options for Category. If you then add a Filter to a panel, and select Category Equals ${category}, the panel data will show only data for that Category that is selected from the Dashboard dropdown list.

Import the **Incidents By Category** dashboard to see an example.

Using variables in queries

There are two syntaxes:

$<varname> Example with a template variable named `hostname`:

`[[varname]]` Example with a template variable named `hostname`:

Alerting

Standard Grafana alerting is supported. Any queries defined in a graph panel can be used to generate alerts.

The following is an example query and an alert. This query will return a graph of all open critical high priority incidents:

This alert will be initiated when there are more than five open critical high priority incidents:

Testing the alert rule will display output from the alert rule, and selecting the state history will show the alert transitioning from ok to pending to alerting.

The graph view will show a vertical line and the heart icon at the top will turn orange while the alert is pending.

Once the criteria for alerting has been met, the rule transitions to red.

In the graph view, the red vertical line will appear and the heart icon at the top will turn red.

Writing incidents for alerts

Beta feature

- Configure a Notification Channel for your ServiceNow data source.

This will configure a Grafana Notification Channel which uses your configured user to create incidents on the ServiceNow instance for this data source.

This action requires that the ServiceNow data source user has permissions for writing incidents.

Annotations

Grafana Annotations are a **beta feature** as of `v1.4.0` of this data source. Annotations give you the ability to overlay events on graphs.

The Annotations query supports the same options as the standard query editor with a few minor differences:

- Only one "Show" column is selectable. This is likely going to be fixed in a future improvement.
- The time field is required.

FAQ

What if we don’t have the ITSM Roles Plugin?

Administrator access is required to perform the following actions

Option 1: Grant Grafana user admin permissions to allow access to all tables.

Option 2: Create a role and apply ACLs to all tables that must be accessed by Grafana.

Administrator access is required to perform the following actions.

1. The logged in administrator needs to elevate access to security_admin.
    a. In the top right navigation pane, choose the profile icon. The profile icon has dropdown caret indicator.
    b. From the dropdown list, choose **Elevate Roles**.
    c. From the modal that is shown, select the **security_admin** check box.
    d. Choose OK.
2. Create a new role with whatever naming convention you want.
    a. Navigate to the roles section in the left hand navigation System Security => Users and Groups => Roles
    b. Choose **New** at the top.
    c. Enter a name for the role and a relevant description.
    d. Choose **Submit**.
3. Create a new user or modify an existing user with the needed roles.
    a. The role you create in Step 2
    b. personalize_dictionary
    c. personalize_choices
    d. cmdb_read (this will grant read access to all cmdb tables)
4. Create Table ACLs for the required tables and fields.
    - Create an ACL for the sys_db_object table.
        i. In the second search header column **Name**, enter **`sys_db_object`**, and press **Enter**.
        ii. The filtered result should show **Table**. Choose **Table** to navigate into the record.

iii. On the tab section, choose **Controls**.

iv. On the lower portion of the page, make sure that **Access Controls** is the selected tab.

v. Choose **New** to create a new ACL.

vi. Change the **Operation** selection to read.

vii. In the **Requires Role** section in the lower part of the screen, choose (double-click) **Insert New Row**, and search for the role that you created.

viii. After you select the role you created, choose the green check mark.

ix. Choose **Submit** in the lower part of the screen to create the ACL, and then choose **Continue** when the modal appears.

5. Create ACLs for specific sys_db_object fields. The following steps must be repeated for each of the following fields: Name, Label, Display Name, and Extends table.

   a. While still on the table record view for sys_db_object, select the **Columns** tab in the tab group closest to the top of the screen.

   b. Locate the field name and select it.

   c. In the lower tab section, choose **New** on the **Access Controls** tab.

   d. Change the operation to read

   e. Choose (double-click) the insert a row text in the bottom "Requires role" table.

   f. Search for the role that you created, and choose the green check mark.

   g. Choose **Submit**.

   h. Make sure that you've repeated these steps for all required fields: Name, Label, Display Name, and Extends table.

6. Repeat the steps from 4.1 on Change, Incident, and any other non-CMDB tables that you want to query from Grafana. Do not repeat the steps from 4.2; that step is only required for sys_db_object.

Connect to a Snowflake data source

With the Snowflake Enterprise data source, you can visualize your Snowflake data alongside all of your other data sources in Grafana as well as log and metric data in context. This data source includes a powerful type-ahead query editor, supports complex annotations, set alerting thresholds, control access and permissions and more.

Note

This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.
Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

Overview

What is Snowflake?

Snowflake offers a cloud-based data storage and analytics service, generally termed "data warehouse-as-a-service" that offers a solution for data warehousing, data lakes, data engineering, data science, data application development, and data sharing. Over the last few years, Snowflake has gained massive popularity because of its ability to affordably store and analyze data using cloud-based hardware and software; recently culminating in the largest software IPO ever. Today, many companies use Snowflake as their primary database to store application and business data such as transaction counts, active user sessions, and even time series and metric data.

Making the most of Snowflake and Amazon Managed Grafana

Visualize Snowflake data without moving it: Grafana's unique architecture queries data directly where it lives rather than moving it and paying for redundant storage and ingestion.

Compose panels from varied sources: With pre-built and custom dashboards, bring data together from many different data sources into a single pane of glass.

Transform and compute at the user level: Users can transform data and run various computations on data they see, requiring less data preparation.

Combine, compute, and visualize within panels: Create mixed-data source panels that display related data from Snowflake and other sources.

Features

Query editor: The query editor is a Smart SQL autocompletion editor that allows you to visualize time series or table data, handles SQL syntax errors, and autocompletes basic SQL keywords.

Data source permissions: Control who can view or query Snowflake data in Grafana

Annotations: Overlay Snowflake events on any Grafana graph, to correlate events with other graph data

Alerting: Set alerts based metrics stores in Snowflake

Variables for queries: Create template variables in Grafana based on Snowflake data, and include variables in Snowflake queries to make interactive dashboards.

Multi-metric queries: Write a single query that returns multiple metrics, each in its own column

Get started with the Snowflake plugin

Here are five quick steps to get started with the Snowflake plugin in Grafana:

Step 1: Set up the Snowflake Data Source

To configure the data source, choose **Configuration**, **Data Sources**, **Add data source**, Snowflake.

Add your authentication details, and the data source is ready to query!

The following configuration fields are available.

Name	Description
Account	Account for Snowflake.
Username	Username for the service account.
Password	Password for the service account.
Schema (optional)	Sets a default schema for queries.
Warehouse (optional)	Sets a default warehouse for queries.
Database (optional)	Sets a default database for queries.
Role (optional)	Assumes a role for queries.

Step 2: Write queries for your Snowflake data

Create a panel in a dashboard, and select a Snowflake Data Source to start using the query editor.

- Date / time can appear anywhere in the query as long as it is included.
- A numerical column must be included. This can be an aggregation or an int/float column.
- Optionally, you can include string columns to create separate data series, if your time series data is formatted for different metrics.

Layout of a Snowflake query

```
select
  <time_column>,
  <any_numerical_column>
  <other_column_1>,
  <other_column_2>,
  <...>
from
  <any_table>
where
  $__timeFilter(<time_column>) // predefined where clause for time range
  and $<custom_variable> = 1 // custom variables start with dollar sign
```

SQL query format for time series group by interval

```
select
  $__timeGroup(created_ts, '1h'), // group time by interval of 1h
  <time_column>,
  <any_numerical_column>,
  <metric_column>
from
  <any_table>
where
  $__timeFilter(<time_column>) // predefined where clause for time range
  and $<custom_variable> = 1 // custom variables start with dollar sign
group by <time_column>
```

SQL query format for tables

```
select
  <time_column>, // optional if result format option is table
```

```
  <any_column_1>
  <any_column_2>
  <any_column_3>
from
  <any_table>
where
  $__timeFilter(time_column) // macro for time range, optional if format as option is
 table
  and $<custom_variable> = 1 // custom variables start with dollar sign
```

Step 3: Create and use Template Variables

Using template variables

You can include template variables in queries, as shown in the following example.

```
 select
   <column>
 from
   <table>
 WHERE column >= '$variable'
```

The following example shows using multi-value variables in a query.

```
select
  <column>
from
  <table>
WHERE <column> regexp '${variable:regex}'
```

Using the Snowflake data source to create variables

In the dashboard settings, choose **Variables**, and choose **New**.

Using the "Query" variable type, select the Snowflake data source as the "Data source".

> **Important**
>
> Be sure to select only one column in your variable query.

Example:

```
SELECT DISTINCT query_type from account_usage.query_history;
```

will give you these variables:

```
All DESCRIBE USE UNKNOWN GRANT SELECT CREATE DROP SHOW
```

Step 4: Set up an alert

You can set alerts on specific Snowflake metrics or on queries you've created.

Choose the alert tab button within the query editor, and choose **Create Alert**.

Step 5. Create an annotation

Annotations allow you to overlay events on a graph.

To create an annotation, in the dashboard settings, choose **Annotations**, and **New**, and select Snowflake as the data source.

Because annotations are events, they require at least one time column and one column to describe the event.

The following example code shows a query to annotate all failed logins to Snowflake.

```
SELECT
  EVENT_TIMESTAMP as time,
  EVENT_TYPE,
  CLIENT_IP
FROM ACCOUNT_USAGE.LOGIN_HISTORY
WHERE $__timeFilter(time) AND IS_SUCCESS!='YES'
ORDER BY time ASC;
```

And

- time: `TIME`

- title: `EVENT_TYPE`
- text: `CLIENT_IP`

This will overlay annotations of all failed logins to Snowflake on your dashboard panels.

Additional functionality

Using the Display Name field

This plugin uses the Display Name field in the Field tab of the Options panel to shorten or alter a legend key based on its name, labels, or values. Other data sources use custom `alias` functionality to modify legend keys, but the Display Name function is a more consistent way to do so.

Data source permissions

Limit access to Snowflake by choosing the **Permissions** tab in the data source configuration page to enable data source permissions. On the permission page, Admins can enable permissions and restrict query permissions to specific Users and Teams.

Understand your Snowflake billing and usage data

Within the Snowflake data source, you can import a billing and usage dashboard that shows you useful billing and usage information.

Add the dashboard in the Snowflake Data Source configuration page:

This dashboard uses the ACCOUNT_USAGE database, and requires the querier to have the ACCOUNTADMIN role. To do this securely, create a new Grafana data source that has a user with the ACCOUNTADMIN role. Then select that data source in the variables.

Connect to a Splunk data source

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.
> Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

Configuration

Data source configuration

When configuring the Data Source, ensure that the URL field utilizes `https` and points to the your configured Splunk port. The default Splunk API point is 8089, not 8000 (this is default web UI port). Enable *Basic Auth* and specify Splunk username and password.

Browser (direct) access mode and CORS

Amazon Managed Grafana does not support browser direct access for the Splunk data source.

Advanced options

Stream mode

Enable stream mode if you want to get search results as they become available. This is experimental feature, don't enable it until you really need it.

Poll result

Run search and then periodically check for result. Under the hood this option runs `search/jobs` API call with `exec_mode` set to `normal`. In this case API request returns job SID, and then Grafana checks job status time to time, in order to get job result. This option can be helpful for slow queries. By default this option is disabled and Grafana sets `exec_mode` to `oneshot` which allows returning search result in the same API call. See more about `search/jobs` API endpoint in Splunk docs.

Search polling interval

This option allow to adjust how often Amazon Managed Grafana will poll splunk for search results. Time for next poll choosing randomly from [min, max) interval. If you run a lot of heavy searches, it makes sense to increase these values. Tips: increase *Min* if search jobs execution takes a long time, and *Max* if you run a lot of parallel searches (a lot of splunk metrics on Grafana dashboard). Default is [500, 3000) milliseconds interval.

Automatic cancellation

If specified, the job automatically cancels after this many seconds of inactivity (0 means never auto-cancel). Default is 30.

Status buckets

The most status buckets to generate. 0 indicates to not generate timeline information. Default is 300.

Fields search mode

When you use visual query editor, data source attempts to get list of available fields for selected source type.

- quick - use first available result from preview
- full - wait for job finish and get full result.

Default earliest time

Some searches can't use dashboard time range (such as template variable queries). This option helps to prevent search for all time, which can slow down Splunk. The syntax is an integer and a time unit `[+|-]<time_integer><time_unit>`. For example `-1w`. Time unit can be `s, m, h, d, w, mon, q, y`.

Variables search mode

Search mode for template variable queries. Possible values:

- fast - Field discovery off for event searches. No event or field data for stats searches.
- smart - Field discovery on for event searches. No event or field data for stats searches.
- verbose - All event & field data.

Usage

Query editor

Editor modes

Query editor support two modes: raw and visual. To switch between these modes choose hamburger icon at the right side of editor and select *Toggle Editor Mode*.

Raw mode

Use `timechart` command for time series data, as shown in the following code example.

```
index=os sourcetype=cpu | timechart span=1m avg(pctSystem) as system, avg(pctUser) as
 user, avg(pctIowait) as iowait
index=os sourcetype=ps | timechart span=1m limit=5 useother=false avg(cpu_load_percent)
 by process_name
```

Queries support template variables, as shown in the following example.

```
sourcetype=cpu | timechart span=1m avg($cpu)
```

Keep in mind that Grafana is time series–oriented application and your search should return time series data (timestamp and value) or single value. You can read about timechart command and find more search examples in official Splunk Search Reference

Splunk Metrics and `mstats`

Splunk 7.x provides `mstats` command for analyzing metrics. To get charts working properly with `mstats`, it should be combined with `timeseries` command and `prestats=t` option must be set.

```
Deprecated syntax:
| mstats prestats=t avg(_value) AS Value WHERE index="collectd"
 metric_name="disk.disk_ops.read" OR metric_name="disk.disk_ops.write" by metric_name
 span=1m
| timechart avg(_value) span=1m by metric_name

Actual:
| mstats prestats=t avg(disk.disk_ops.read) avg(disk.disk_ops.write) WHERE
 index="collectd" by metric_name span=1m
| timechart avg(disk.disk_ops.read) avg(disk.disk_ops.write) span=1m
```

Read more about `mstats` command in Splunk Search Reference.

Format as

There are two supported result format modes - *Time series* (default) and *Table*. Table mode suitable for using with Table panel when you want to display aggregated data. That works with raw events (returns all selected fields) and `stats` search function, which returns table-like data. Examples:

```
index="os" sourcetype="vmstat" | fields host, memUsedMB
index="os" sourcetype="ps" | stats avg(PercentProcessorTime) as "CPU time",
 latest(process_name) as "Process", avg(UsedBytes) as "Memory" by PID
```

The result is similar to *Statistics* tab in Splunk UI.

Read more about `stats` function usage in Splunk Search Reference.

Visual mode

This mode provides step-by-step search creating. Note that this mode creates `timechart` splunk search. Just select index, source type, and metrics, and set split by fields if you want.

Metric

You can add multiple metrics to search by choosing *plus* button at the right side of metric row. Metric editor contains list of frequently used aggregations, but you can specify here any other function. Just choose agg segment (`avg` by default) and type what you need. Select interested field from the dropdown list (or enter it), and set alias if you want.

Split by and Where

If you set Split by field and use *Time series* mode, Where editor will be available. Choose *plus* and select operator, aggregation and value, for example *Where avg in top 10*. Note, this *Where* clause is a part of *Split by*. See more at timechart docs.

Options

To change default timechart options, choose **Options** at the last row.

See more about these options in timechart docs.

Rendered splunk search

Choose the target letter at the left to collapse the editor and show the rendered splunk search.

Annotations

Use annotations if you want to show Splunk alerts or events on graph. Annotation can be either predefined Splunk alert or regular splunk search.

Splunk alert

Specify an alert name, or keep the field blank to get all fired alerts. Template variables are supported.

Splunk search

Use splunk search to get needed events, as shown in the following example.

```
index=os sourcetype=iostat | where total_ops > 400
index=os sourcetype=iostat | where total_ops > $io_threshold
```

Template variables are supported.

The **Event field as text** option is suitable if you want to use field value as annotation text. The following example shows error message text from logs.

```
Event field as text: _raw
Regex: WirelessRadioManagerd\[\d*\]: (.*)
```

Regex allows to extract a part of message.

Template variables

Template variables feature supports Splunk queries which return list of values, for example with `stats` command.

```
index=os sourcetype="iostat" | stats values(Device)
```

This query returns list of `Device` field values from `iostat` source. Then you can use these device names for time series queries or annotations.

There are two possible types of variable queries can be used in Grafana. The first is a simple query (as presented earlier), which returns a list of values. The second type is a query that can create a key/value variable. The query should return two columns that are named `_text` and `_value`. The `_text` column value should be unique (if it is not unique then the first value is used). The options in the dropdown list will have a text and value so that you can have a friendly name as text and an ID as the value.

For example, this search returns table with columns `Name` (Docker container name) and `Id` (container id).

```
source=docker_inspect | stats count latest(Name) as Name by Id | table Name, Id
```

To use container name as a visible value for variable and id as it's real value, query should be modified, as in the following example.

```
source=docker_inspect | stats count latest(Name) as Name by Id | table Name, Id |
 rename Name as "_text", Id as "_value"
```

Multi-value variables

It's possible to use multi-value variables in queries. An interpolated search will be depending on variable usage context. There are a number of that contexts which plugin supports. Assume there's a variable `$container` with selected values `foo` and `bar`:

- Basic filter for `search` command

  ```
  source=docker_stats $container
  =>
  source=docker_stats (foo OR bar)
  ```

- Field-value filter

  ```
  source=docker_stats container_name=$container
  =>
  source=docker_stats (container_name=foo OR container_name=bar)
  ```

- Field-value filter with the `IN` operator and `in( )` function

  ```
  source=docker_stats container_name IN ($container)
  =>
  source=docker_stats container_name IN (foo, bar)

  source=docker_stats | where container_name in($container)
  =>
  source=docker_stats | where container_name in(foo, bar)
  ```

Multi-value variables and quotes

If variable wrapped in quotes (both double or single), its values also will be quoted, as in the following example.

```
source=docker_stats container_name="$container"
=>
source=docker_stats (container_name="foo" OR container_name="bar")

source=docker_stats container_name='$container'
=>
source=docker_stats (container_name='foo' OR container_name='bar')
```

Connect to a Splunk Infrastructure Monitoring data source

Provides support for Splunk Infrastructure Monitoring (formerly SignalFx).

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.
> Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

Adding the data source

1. Open the Grafana console in the Amazon Managed Grafana workspace and make sure you are logged in.
2. In the side menu under **Configuration** (the gear icon), choose **Data Sources**.
3. Choose **Add data source**.

> **Note**
>
> If you don't see the **Data Sources** link in your side menu, it means that your current user does not have the `Admin` role.

4. Select **Splunk Infrastructure Monitoring** from the list of data sources.
5. Enter the following information:

   - For **Access Token**, enter the token that is generated by your SignalFx account. For more information, see Authentication Tokens.
   - **Realm** A self-contained deployment that hosts your organization. You can find your realm name on your profile page when signed in to the SignalFx user interface.

Using the query editor

The query editor accepts a SignalFlow program/query.

For labels, a Signalflow label `publish(label = 'foo')` is applied as metadata to the results: **"label":"foo"**

For query type template variables, there is no **Query** field. Instead, you select one of the following query types:

- Dimensions
- Metrics
- Tags

Ad-hoc filters are supported, allowing global filters using dimensions.

Grafana annotations are supported. When you create annotations, use SignalFlow Alerts or Events queries.

Example of getting alerts for a detector:

```
alerts(detector_name='Deployment').publish();
```

Example of getting custom Events by type:

```
events(eventType='simulated').publish();
```

Connect to a Wavefront data source (VMware Tanzu Observability by Wavefront)

The Wavefront (VMware Tanzu Observability by Wavefront) data source enables Amazon Managed Grafana users to query and visualize the data they're collecting directly from Wavefront and easily

visualize it alongside any other metric, log, tracing, or other data source. This flexible, single-pane view makes it easier to track system health and debug issues.

> **Note**
>
> This data source is for Grafana Enterprise only. For more information, see Manage access to Enterprise plugins.
> Additionally, in workspaces that support version 9 or newer, this data source might require you to install the appropriate plugin. For more information, see Extend your workspace with plugins.

What is Wavefront?

Wavefront is a cloud monitoring and analytics tool developed by VMware. Wavefront is a cloud-hosted service where you send your time-series (metric) data – from CollectD, StatsD, JMX, Ruby's logger, AWS, or other tools. With Wavefront, users can perform mathematical operations on those series, render charts to see anomalies, track KPIs, and create alerts.

Maximizing your tech stack with Wavefront and Grafana

While on the surface, Grafana and Wavefront sound similar, many organizations use both Wavefront and Grafana as critical parts of their observability workflows.

Visualize without Moving Data Sources: Grafana's unique architecture queries data directly where it lives rather than moving it and paying for redundant storage and ingestion.

Compose Panels from Varied Sources With pre-built and custom dashboards, bring data together from many different data sources into a single pane of glass.

Transform and Compute at the User Level: Users can transform data and run various computations on data they see, requiring less data preparation.

Combine, Compute, and Visualize within Panels: Create mixed-data source panels that display related data from Waveferont and other sources, such as Prometheus and InfluxDB.

Documentation

Features

- Timeseries Visualizations

- Table Visualizations
- Heatmap Visualizations
- Single Stat Visualizations
- Guided Query Editor
- Raw WQL Query Editor
- Annotations for event data
- Template Variables
- Ad-Hoc Filters
- Alerting

Configuration

Configuring the Wavefront data source is relatively straightforward. There are only two fields required to complete the configuration: `API URL` and `Token`.

- `API URL` will be the URL you use to access your wavefront environment. Example: `https://myenvironment.wavefront.com`.
- `Token` must be generated from a user account or service account.

  1. To create a user account based token, log into your Wavefront environment, choose the cog on the top right corner of the page, choose your username (for example, `me@grafana.com`), select the **API Access** tab at the top of the user page, then copy an existing key or choose **generate**.
  2. To create a service account based token, log into your Wavefront environment, choose the cog on the top right corner of the page, choose account management. On the left navigation, select **Accounts, Groups, & Roles**, choose the **Service Accounts** tab at the top, and then choose **Create New Account**. Enter a name for the service account. This can be anything you want. Copy the token that is provided under the **Tokens** section.
  3. The last step is to make sure that the **Accounts, Groups, & Roles** check box selected under **Permissions**.

After you have the token, add that to the `Token` configuration field and you should be set!

The finalized configuration page should look similar to this:

Usage

Using the query editor

The Wavefront query editor has two modes: **Query Builder** and **Raw Query**. To toggle between them, use the selector in the top right of the query form:

In **Query Builder** mode, you will presented with four choices to make:

1. What metric do you want to query?
2. What aggregation do you want to perform on that metric?
3. How do you want to filter the results from that metric query?
4. Do you want to apply any additional functions to the result?

The metric selector is a categorized hierarchy. Select a category, then choose again to drill into the subcategories. Repeat this process until you have reached the metric that you want.

After selecting a metric, the available filters and filter values will be automatically populated for you.

In **Raw Query** mode, you will see a single field labeled **Query**. This allows you to run any WQL query that you want.

Using filters

The Wavefront plugin will dynamically query the appropriate filters for each metric.

To add a filter, choose the **+** next to the **Filters** label on the Wavefront query editor, select which field you want to filter on, and select a value to filter by.

Using functions

Functions provide an additional way to aggregate, manipulate, and perform calculations on the metric response data. To view the available functions, choose the dropdown list by the function label on the **Query Builder**. Based on the function you select, you will be able to perform further actions such as setting a group by field or applying thresholds. Users are able to chain multiple functions together to perform advanced calculations or data manipulations.

Adding a query template variable

1. To create a new Wavefront template variable for a dashboard, choose the settings cog on the top right portion of the dashboard.
2. In the panel at the left, choose **Variables**.
3. At the top right of the Variables page, choose **New**.
4. Enter a **Name** and a **Label** for the template variable you want to create. **Name** is the value you will use inside of queries to reference the template variable. **Label** is a human friendly name to display for the template variable on the dashboard select panel.
5. Select the type **Query** for the type field (it should be selected by default).
6. Under the **Query Options** heading, select **Wavefront** in the **Data source** dropdown list.
7. See Template Variable Query Structure for details on what should be entered into the **Query** field.
8. If you want to filter out any of the returned values from your query, enter a regular expression in the **Regex** input field.
9. Apply any sorting preferences you might have by choosing a sort type in the **Sort** dropdown list.
10. After verifying the configuration, choose **Add** to add the template variable, then choose **Save dashboard** on the left hand navigation panel to save your changes.

Template variable query structure

metric lists: metrics: ts(…)

source lists: sources: ts(…)

source tag lists: sourceTags: ts(…)

matching source tag lists: matchingSourceTags: ts(…)

tag name lists: tagNames: ts(…)

tag value lists: tagValues(<tag>): ts(…)

Notes

- The s at the end of each query type is optional
- Support for all lowercase. You can use tagnames or tagNames, but not TAGNAMES.

- Using spaces around the : is optional

WARNING

`Multi-value` and `Include All option` are currently not supported by the Wavefront plugin.

Using template variables

After completing the steps to add a new template variable, you're now ready to use the template variable within your dashboard panels to create dynamic visualizations.

1. Add a new dashboard panel using the panel+ icon in the top right corner of your dashboard.
2. Select the aggregate you want to use for your query.
3. Choose the + icon beside **Filters** label and select the key type that matches your template variable. `host=` for a host filter, for example.
4. Enter the name of the template variable you created in the **Value** input field of the filter.
5. Save the dashboard.

You should now be able to cycle through different values of your template variable and have your panel dynamically update!

Using Ad-Hoc filters

To use ad-hoc filters, we must create two template variables. The first one is a helper variable that will be used to select a metric so that add-hoc filters can be populated for that metric name. The other will be the actual ad-hoc filter variable.

Important

The helper variable that is required has to be named `metriclink`. This can be an custom variable with the list of metrics that you want to use or a query based variable using the Template Variable Query Structure. If you want to populate the ad-hoc filter fields with only the values from a single metric, you can hide the `metriclink` template variable.

After creating the `metriclink` variable, you can now add the ad-hoc filter by following the same steps detailed in Adding a Query Template Variable. The difference being that you will select **Ad Hoc Filters** as the **Type** and no inputs are required for a query.

Adding annotations

1. To create a new Wavefront annotation for a dashboard, choose the settings cog on the top right portion of the dashboard.
2. In the panel at the left, choose **Annotations**.
3. At the top right of the Annotations page, choose **New**.
4. Enter a name for the annotation (this will be used as the name of the toggle on the dashboard).
5. Select the **Data source** of Wavefront.
6. By default, annotations have a limit of 100 alert events that will be returned. To change that, set the **Limit** field to the value that you want.
7. Choose **Add**.

Using annotations

When annotations are toggled on, you should now see the alert events and issues that correlate with a given time period.

If you pause on the bottom of an annotated section of a visualization, a pop-up window will be displayed that shows the alert name and provides a direct link to the alert in Wavefront.

Using the Display Name field

This data source uses the Display Name field in the Field tab of the Options panel to shorten or alter a legend key based on its name, labels, or values. Other datasources use custom `alias` functionality to modify legend keys, but the Display Name function is a more consistent way to do so.

References

- WQL (Wavefront Query Language)

Working in Grafana version 12

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

When you create your Grafana workspace, you have the option of which version of Grafana to use. The following topics describe using a Grafana workspace that uses version 12 of Grafana.

Topics

- Dashboards in Grafana version 12
- Panels and visualizations in Grafana version 12
- Explore in Grafana version 12
- Drilldown apps in Grafana version 12
- Correlations in Grafana version 12
- Alerts in Grafana version 12

Dashboards in Grafana version 12

This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A dashboard is a set of one or more panels organized and arranged into one or more rows. Grafana ships with a variety of panels making it easy to construct the right queries, and customize the visualization so that you can create the perfect dashboard for your need. Each panel can interact with data from any configured Connect to data sources.

The dashboard rendering engine is built on the Scenes framework, providing improved performance, better template variable support, and more flexible layouts.

Dashboard snapshots are static. Queries and expressions cannot be re-executed from snapshots. As a result, if you update any variables in your query or expression, it will not change your dashboard data.

Topics

- Using dashboards
- Building dashboards
- Managing dashboards
- Managing playlists
- Sharing dashboards and panels
- Variables
- Assessing dashboard usage
- Troubleshoot dashboards
- Searching Dashboards in Grafana version 12

Using dashboards

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This topic provides an overview of dashboard features and shortcuts, and describes how to use dashboard search.

Features

You can use dashboards to customize the presentation of your data. The following image shows the dashboard interface in the Amazon Managed Grafana workspace.



Feature	Description
1. Home	Select the Grafana home icon to be redirecte d to the home page configured in the Grafana instance.
2. Title	When you select the dashboard title, you can search for dashboards contained in the current folder.
3. Sharing a dashboard	Use this option to share the current dashboard by link or snapshot. You can also export the dashboard definition from the share modal.
4. Adding a new panel	Use this option to add a panel, dashboard row, or library panel to the current dashboard.
5. Save dashboard	Choose the Save icon to save changes to your dashboard.
6. Dashboard insights	Choose to view analytics about your dashboard, including information about users,

Feature	Description
	activity, and query counts. For more informati on, see Assessing dashboard usage.
7. Dashboard settings	Use this option to change the dashboard name, folder, or tags and manage variables and annotation queries. For more informati on about dashboard settings, see Modifying dashboard settings.
8. Time picker dropdown	Use to select relative time range options and set custom absolute time ranges. You can change the **Timezone** and **fiscal year** settings from the time range controls by clicking the **Change time settings** button. Time settings are saved on a per-dashboard basis.
9. Zoom out time range	Use to zoom out the time range. For more information about how to use time range controls, see Setting dashboard time range.
10. Refresh dashboard	Select to immediately trigger queries and refresh dashboard data.
11. Refresh dashboard time interval	Select a dashboard auto refresh time interval.
12. View mode	Select to display the dashboard on a large screen such as a TV or a kiosk. View mode hides irrelevant information such as navigatio n menus.

Feature	Description
13. Dashboard panel	The primary building block of a dashboard is the panel. To add a new panel, dashboard row, or library panel, select **Add panel**. • Library panels can be shared among many dashboards. • To move a panel, drag the panel header to another location. • To resize a panel, select and drag the lower right corner of the panel.
14. Graph legend	Change series colors, y-axis, and series visibilit y directly from the legend.
15. Dashboard row	A dashboard row is a logical divider within a dashboard that groups panels together. • Rows can be collapsed or expanded to hide parts of the dashboard. • Panels inside a collapsed row do not issue queries. • Use repeating rows to create rows dynamically based on a template variable. For more information about repeating rows, see Creating dashboards.

Keyboard shortcuts

Grafana has a number of keyboard shortcuts available. To display all keyboard shortcuts available to you, press **?** or **h** on your keyboard.

- `Ctrl+S` saves the current dashboard.
- `f` opens the dashboard finder/search.
- `d+k` toggles kiosk mode (hides the menu).

- d+e expands all rows.
- d+s opens dashboard settings.
- Ctrl+K opens the command palette.
- Esc exits panel when in fullscreen view or edit mode. Also returns you to the dashboard from the dashboard settings.

Focused panel

To use shortcuts targeting a specific panel, hover over a panel with your pointer.

- e toggles panel edit view
- v toggles panel fullscreen view
- ps opens panel share feature
- pd duplicates panel
- pr removes panel
- pl toggles panel legend

Setting dashboard time range

Grafana provides several ways to manage the time ranges of the data being visualized, for dashboard, panels and also for alerting.

This section describes supported time units and relative ranges, the common time controls, dashboard-wide time settings, and panel-specific time settings.

Time units and relative ranges

Grafana supports the following time units: `s (seconds)`, `m (minutes)`, `h (hours)`, `d (days)`, `w (weeks)`, `M (months)`, `Q (quarters)`, and `y (years)`.

The minus operator enables you to step back in time, relative to the current date and time, or now. If you want to display the full period of the unit (day, week, or month), append `/<time unit>` to the end. To view fiscal periods, use `fQ (fiscal quarter)` and `fy (fiscal year)` time units.

The plus operator enables you to step forward in time, relative to now. For example, you can use this feature to look at predicted data in the future.

The following table provides example relative ranges.

Example relative range	From	To
Last 5 minutes	`now-5m`	now
The day so far	`now/d`	now
This week	`now/w`	`now/w`
This week so far	`now/w`	now
This month	`now/M`	`now/M`
This month so far	`now/M`	now
Previous Month	`now-1M/M`	`now-1M/M`
This year so far	`now/Y`	now
This Year	`now/Y`	`now/Y`
Previous fiscal year	`now-1y/fy`	`now-1y/fy`

Note

Grafana Alerting does not support the following syntaxes:

- `now+n` for future timestamps.
- `now-1n/n` for *start of n until end of n*, because this is an absolute timestamp.

Common time range controls

The dashboard and panel time controls have a common user interface. The following describes common time range controls.

- Current time range, also called the *time picker*, shows the time range currently displayed in the dashboard or panel you are viewing. Hover your cursor over the field to see the exact time stamps in the range and their source (such as the local browser time). Click the *current time range* to change it. You can change the current time using a *relative time range*, such as the

last 15 minutes, or an absolute time range, such as `2020-05-14 00:00:00` to `2020-05-15 23:59:59`.

- The **relative time range** can be selected from the **Relative time ranges** list. You can filter the list using the input field at the top. Some examples of time ranges include *Last 30 minutes*, *Last 12 hours*, *Last 7 days*, *Last 2 years*, *Yesterday*, *Day before yesterday*, *This day last week*, *Today so far*, *This week so far*, and *This month so far*.
- **Absolute time range** can be set in two ways: Typing exact time values or relative time values into the **From** and **To** fields and clicking **Apply time range**, or clicking a date or date range from the calendar displayed when you click the **From** or **To** field. To apply your selections, click **Apply time range**. You can also choose from a list of recently used absolute time ranges.
- **Semi-relative time range** can be selected in the absolute time range settings. For example, to show activity since a specific date, you can choose an absolute time for the start time, and a relative time (such as now) for the end time.

  Using a semi-relative time range, as time progresses, your dashboard will automatically and progressively zoom out to show more history and fewer details. At the same rate, as high data resolution decreases, historical trends over the entire time period will become more clear.

  > **Note**
  >
  > Alerting does not support semi-relative time ranges.

- **Zoom out** by selecting the zoom out icon (or by using Cmd+Z or Ctrl+Z as a keyboard shortcut). This increases the view, showing a larger time range in the dashboard or panel visualization.
- **Zoom in** by selecting a time range you want to view on the graph in the visualization.

  > **Note**
  >
  > Zooming in is only applicable to graph visualizations.

Refresh dashboards

Click the **Refresh dashboard** icon to immediately run every query on the dashboard and refresh the visualizations. Grafana cancels any pending requests when you trigger a refresh.

By default, Grafana does not automatically refresh the dashboard. Queries run on their own schedule according to the panel settings. However, if you want to regularly refresh the dashboard, then click the down arrow next to the **Refresh dashboard** icon and then select a refresh interval.

Control the time range using a URL

You can control the time range of a dashboard by providing the following query parameters in the dashboard URL.

- `from` defines the lower limit of the time range, specified in ms epoch, or relative time.
- `to` defines the upper limit of the time range, specified in ms epoch, or relative time.
- `time` and `time.window` defines a time range from `time-time.window/2` to `time+time.window/2`. Both parameters should be specified in ms. For example ? `time=1500000000000&time.window=10000` results in 10s time range from 1499999995000 to 1500000005000.

Building dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

After you create a Grafana workspace and sign in, you can create dashboards and modify settings to suit your needs. A dashboard is made up of panels with visualizations. Each panel has a query associated with it, to pull data from one of your Connect to data sources.

You can create more interactive and dynamic dashboards by adding and using variables. Instead of hard-coding the server, application, or other names in your metric queries, you can use variables in their place.

Topics

- Creating dashboards

- Importing dashboards
- Exporting dashboards
- Modifying dashboard settings
- Dashboard URL variables
- Managing library panels
- Managing dashboard version history
- Managing dashboard links
- Annotate visualizations
- Dashboard JSON model
- Best practices for dashboards

Creating dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Creating a dashboard

Dashboards and panels allow you to show your data in visual form using Grafana. Each panel needs at least one query to display a visualization. Before you get started, complete the following prerequisites.

- Ensure that you have the proper permissions. For more information about permissions, see Users, teams, and permissions.
- Identify the dashboard to which you want to add the panel.
- Understand the query language of the target data source.
- Ensure that data source for which you are writing a query has been added. For more information, see Connect to data sources.

To create a dashboard:

1. Sign into Grafana, and select **Dashboards** from the left menu.
2. Select **New**, then **New dashboard**.
3. On the empty dashboard, select **+ Add visualization**. This opens the new visualization dialog box.
4. Select a data source. You can choose an existing data source, one of Grafana's built in data sources for testing, or choose **Configure a new data source** to set up a new one (only users with Admin permissions can configure new data sources).

   The **Edit panel** view opens, with your data source selected. You can change the data source for the panel later, using the **Query** tab of the panel editor, if needed.
5. Write or construct a query in the query language of your data source. Choose the refresh dashboard icon to perform a query on the data source, seeing the results as you go.
6. In the **Visualization** list, select a visualization type. Grafana displays a preview of your query results with the visualization applied. For more information, see Visualizations options.
7. Under **Panel options**, you can enter a title and description for your panel.
8. Most visualizations need some adjustment before they display the exact information that you need. You can adjust panel settings in the following ways.
   - Configure value mappings
   - Visualization-specific options
   - Override field values
   - Configure thresholds
   - Configure standard options
9. When you've finished configuring your panel, choose **Save** to save the dashboard.

   Alternatively, select **Apply** to see changes without leaving the panel editor.
10. Add a note to describe the visualization (or describe your changes) and then click **Save** in the upper-right corner of the page.

    > **Note**
    >
    > Notes are helpful if you need to revert the dashboard to a previous version.

11. Choose **Save**.

12. Optionally, you can add more panels to the dashboard by choosing **Add** in the dashboard header, and selecting **Visualization** from the drop-down.

Copying an existing dashboard

You can quickly copy an existing dashboard, to jumpstart creating a new one.

To copy an existing dashboard

1. Select **Dashboards** from the left menu.
2. Choose the dashboard you want to copy, to open it.
3. Select **Settings** (gear icon) in the top right of the dashboard.
4. Select **Save as**in the top right corner of the dashboard.
5. (Optional) Specify the name, folder, description, and whether or not to copy the original dashboard tags for the copied dashboard.
6. Select **Save**.

Configuring repeating rows

You can configure Grafana to dynamically add panels or rows to a dashboard based on the value of a variable. Variables dynamically change your queries across all rows in a dashboard. For more information about repeating panels, see Configure repeating panels.

You can also repeat rows if you have variables set with `Multi-value` or `Include all values` selected.

Before you get started, ensure that the query includes a multi-value variable, then you should complete the following steps.

To configure repeating rows

1. Select **Dashboards** from the left menu, then choose the dashboard you want to modify.
2. At the top of the dashboard, select **Add**, and then select **Row** from the drop down.

   If the dashboard is empty, you can alternately select the **+ Add row** button in the middle of the dashboard.
3. Hover over the row title and select the **Settings** (gear) icon that appears.

4. On the **Row Options** dialog box, add a title and select the variable for which you want to add repeating rows.

> **Note**
>
> To provide context to dashboard users, add the variable to the row title.

5. Select **Update**.

Repeating rows and the Dashboard special data source

If a row includes panels using the special Dashboard data source—the data source that uses a result set from another panel in the smae dashboard—then corresponding panels in repeated rows will reference the panel in the original row, not the ones in the repeated rows.

For example, in a dashboard:

- `Row 1` includes `Panel 1A` and `Panel 1B`.
- `Panel 1B` uses the results from `Panel 1A` by using the `Dashboard` data source.
- Repeating `Row 2` includes `Panel 2A` and `Panel 2B`.
- `Panel 2B` references `Panel 1A`, not `Panel 2A`.

To move a panel

1. Open the dashboard.
2. Select the panel title and drag the panel to the new location. You can place a panel on a dashboard in any location.

To resize a panel

1. Open the dashboard.
2. To adjust the size of the panel, drag the lower-right corner of the panel. You can size a dashboard panel to suits your needs.

Importing dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can import preconfigured dashboards into your Amazon Managed Grafana workspace.

To import a dashboard

1. Sign into your Amazon Managed Grafana workspace.
2. Select **Dashboards** from the left menu.
3. Select **New** and choose **Import** in the drop down menu.
4. Next you need to choose the dashboard JSON definition to import. You have three choices for how to import JSON:
   - Upload a file containing dashboard JSON.
   - Directly copy JSON text into the text area.
   - Paste a Grafana Labs dashboard URL or ID into the field. For more information on grafana.com dashboard URLs, see the next section.
   - (Optional) Change any dashboard details that you wish to change.
   - Select a data source, if required.
   - Choose **Import**.
   - Save the dashboard.

Finding dashboards on grafana.com

The Dashboards page on grafana.com provides you with dashboards for common server applications. Browse a library of official and community-built dashboards and import them to quickly get up and running.

> **Note**
>
> To import dashboards from grafana.com, your Amazon Managed Grafana workspace must have access to the internet.

Exporting dashboards

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can use the Grafana UI or the HTTP API to export dashboards.

The dashboard export action creates a Grafana JSON file that contains everything you need, including layout, variables, styles, data sources, queries, and so on, so that you can later import the dashboard.

Making a dashboard portable

If you want to export a dashboard for others to use, you can add template variables for things like a metric prefix (use a constant variable) and server name.

A template variable of the type `Constant` will automatically be hidden in the dashboard, and will also be added as a required input when the dashboard is imported.

To export a dashboard

1. Open the dashboard that you want to export.
2. Select the share icon.
3. Choose **Export**.
4. Choose **Save to file.**

Note

Grafana downloads a JSON file to your local machine.

Modifying dashboard settings

This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The dashboard settings page enables you to:

- Edit general dashboard properties, including time settings.
- Add annotation queries.
- Add dashboard variables.
- Add links.
- View the dashboard JSON model

To access the dashboard setting page

1. Open a dashboard in edit mode.
2. Click **Dashboard settings** (gear icon) located at the top of the page.

Modifying dashboard time settings

Adjust dashboard time settings when you want to change the dashboard timezone, the local browser time, and specify auto-refresh time intervals.

To modify dashboard time settings

1. On the **Dashboard** settings page, select **General**.

2. Navigate to the **Time Options** section.
3. Specify time settings according to the following descriptions.
4.
   - **Timezone** – Specify the local time zone of the service or system that you are monitoring. This can be helpful when monitoring a system or service that operates across several time zones.
     - **Default** – Grafana uses the default selected time zone for the user profile, team, or organization. If no time zone is specified for the user profile, a team the user is a member of, or the organization, then Grafana uses the local browser time.
     - **Local browser time** – The time zone configured for the viewing user browser is used. This is usually the same time zone as set on the computer.
     - Use standard ISO 8601 time zones, including UTC.
   - **Auto-refresh** – Customize the options displayed for relative time and the auto-refresh options. Entries are comma separated and accept any valid time unit.
   - **Now delay** – Override the now time by entering a time delay. Use this option to accommodate known delays in data aggregation to avoid null values.
   - **Hide time picker** – Selecting this option if you do not want the dashboard to display the time picker.

> **Note**
>
> To have time controls, your data must include a time column. See the documentation for your specific data source for more information about including a time column.

Adding an annotation query

An annotation query is a query that queries for events. These events can be visualized in graphs across the dashboard as vertical lines along with a small icon you can hover over to see the event information.

To add an annotation query

1. On the **Dashboard settings** page, select **Annotations**.
2. Select **Add annotation query**.
3. Enter a name and select a data source.

4. Complete the rest of the form to build a query and annotation.

The query editor UI changes based on the data source that you select. see the Data source documentation for details on how to construct a query. Or, for data source plugins that you install from the Find plugins with the plugin catalog, you can use the documentation on the Grafana Labs website.

Adding a variable

Variables enable you to create more interactive and dynamic dashboards. Instead of hard-coding things like server, application, and sensor names in your metric queries, you can use variables in their place. Variables are displayed as dropdown lists at the top of the dashboard. These dropdowns make it easy to change the data being displayed in your dashboard.

For more information about variables, see Variables.

To add a variable

1. On the **Dashboard settings** page, click **Variable** in the left side section menu and then the **Add variable** button.
2. In the **General** section, add the name of the variable. This is the name that you will later use in queries.
3. Select a variable **Type**.

> **Note**
>
> The variable type that you select impacts which fields that you populate on the page.

4. Define the variable and click **Update**.

Adding a link

Dashboard links enable you to place links to other dashboards and websites directly below the dashboard header. Links provide for easy navigation to other, related dashboards and content.

To add a link

1. On the **Dashboard settings** page, choose **Links** from the left side section menu and then the **Add link** button.

2. Enter a title and in the **Type** field, select **Dashboard** or **Link**.
3. To add a dashboard link, add an optional tag, select any of the dashboard link Options, and click **Apply**.

> **Note**
>
> Using tags creates a dynamic dropdown of dashboards that all have a specific tag.

4. To add a web link, add a URL and tooltip text that appears when the user hovers over the link, select an icon that appears next to the link, and select any of the dashboard link options.

View dashboard JSON model

A dashboard in Grafana is represented by a JSON object, which stores the metadata of a dashboard. Dashboard metadata includes dashboard properties, metadata from panels, template variables, panel queries, and so on. The JSON metadata defines the dashboard.

To view a dashboard JSON model, on the **Dashboard settings** page, click **JSON**.

For more information about the JSON fields, see JSON fields.

Dashboard URL variables

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana can apply variable values passed as query parameters in dashboard URLs. For more information, see Manage dashboard links and Templates and variables.

Passing variables as query parameters

Grafana interprets query string parameters prefixed with `var-` as variables in the given dashboard.

For example, in this URL:

```
https://${your-domain}/path/to/your/dashboard?var-example=value
```

The query parameter `var-example=value` represents the dashboard variable example with a value of `value`.

Passing multiple values for a variable

To pass multiple values, repeat the variable parameter once for each value.

```
https://${your-domain}/path/to/your/dashboard?var-example=value1&var-example=value2
```

Grafana interprets `var-example=value1&var-example=value2` as the dashboard variable example with two values: `value1` and `value2`.

Adding variables to dashboard links

Grafana can add variables to dashboard links when you generate them from a dashboard's settings. For more information and steps to add variables, see Manage dashboard links.

Passing ad hoc filters

Ad hoc filters apply key or value filters to all metric queries that use a specified data source. For more information, see Ad hoc filters.

To pass an ad hoc filter as a query parameter, use the variable syntax to pass the ad hoc filter variable, and also provide the key, the operator as the value, and the value as a pipe-separated list.

For example, in this URL:

`https://${your-domain}/path/to/your/dashboard?var-adhoc=example_key|=|example_value`

The query parameter `var-adhoc=key|=|value` applies the ad hoc filter configured as the adhoc dashboard variable using the `example_key` key, the = operator, and the `example_value` value.

> **Note**
>
> When sharing URLs with ad hoc filters, remember to encode the URL. In the above example, replace the pipes (`|`) with `%7C` and the equality operator (=) with `%3D`.

Controlling time range using the URL

To set a dashboard's time range, use the `from`, `to`, `time`, and `time.window` query parameters. Because these are not variables, they do not require the `var-` prefix.

Managing library panels

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A library panel is a reusable panel that you can use in any dashboard. When you change a library panel, the change propagates to all instances of where the panel is used. Library panels streamline reuse of panels across multiple dashboards.

You can save a library panel in a folder alongside saved dashboards.

Creating a library panel

When you create a library panel, the panel on the source dashboard is converted to a library panel as well. You need to save the original dashboard after a panel is converted.

To create a library panel

1. Open the panel you want to convert to a library panel in edit mode.
2. In the panel display options, click the down arrow option to start changes to the visualization.
3. Select **Library panels**, and then **+ Create library panel**. This opens the create dialog.
4. In **Library panel name**, enter the name you want for the panel.
5. In **Save in folder**, select the folder to save the library panel.
6. Select **Create library panel** to save your changes to the library.
7. Save the dashboard.

After a library panel is created, you can modify the panel using any dashboard on which it appears. After you save the changes, all instances of the library panel reflect these modifications.

You can also create a library panel directly from the edit menu of any panel, by selecting **More...** then **Create library panel**.

Adding a library panel to a dashboard

Add a Grafana library panel to a dashboard when you want to provide visualizations to other dashboard users.

To add a library panel to a dashboard

1. Select **Dashboards** on the left menu.
2. Select **New**, and then choose **New dashboard** from the drop down.
3. On the empty dashboard, select **+ Import library panel**. You will see a list of your library panels.
4. Filter the list or search to find the panel you want to add.
5. Click a panel to add it to the dashboard.

Unlinking a library panel

Unlink a library panel when you want to make a change to the panel and not affect other instances of the library panel.

To unlink a library panel

1. Select **Dashboards** on the left menu.
2. Select **Library panels**.
3. Select a library panel that is being used in different dashboards.
4. Select the panel that you want to unlink.
5. Select the panel title (or hover the pointer anywhere over the panel), to display the actions menu on the top right corner of the panel.
6. Select **Edit**. The panel will open in edit mode.
7. Select **Unlink** on the top right corner of the page.
8. Choose **Yes, unlink**.

Viewing a list of library panels

You can view a list of available library panels and search for a library panel.

To view a list of library panels

1. Select **Dashboards** on the left menu.
2. Select **Library panels**. You can see a list of previously defined library panels.
3. Search for a specific library panel if you know its name. You can also filter the panels by folder or type.

Deleting a library panel

Delete a library panel when you no longer need it.

To delete a library panel

1. Select **Dashboards** on the left menu.
2. Select **Library panels**.
3. Select the delete icon next to the library panel name for the panel you wish to delete.

Managing dashboard version history

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Whenever you save a version of your dashboard, a copy of that version is saved so that previous versions of your dashboard are not lost. A list of these versions is available by entering the dashboard settings and then selecting **Versions** in the left side menu.

Note

The most recent 20 versions of a dashboard are saved.

The dashboard version history feature lets you compare and restore to previously saved dashboard versions.

Comparing two dashboard versions

To compare two dashboard versions, select the two versions from the list that you wish to compare. Click **Compare versions** to view the diff between the two versions. This brings up the version diff view. By default, you'll see a textual summary of the changes.

If you want to view the diff of the raw JSON that represents your dashboard, you can do that by clicking the **View JSON Diff** button at the bottom.

Restoring to a previously saved dashboard version

If you need to restore to a previously saved dashboard version, you can either select the **Restore** button on the right of a row in the dashboard version list, or select the **Restore to version *<x>*** button in the diff view. Selecting either of these will prompt you to confirm the restoration.

After restoring to a previous version, a new version will be created containing the same exact data as the previous version, only with a different version number. This is indicated in the **Notes column** for the row in the new dashboard version. This ensures your previous dashboard versions are not affected by the change.

Managing dashboard links

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can use links to navigate between commonly used dashboards or to connect others to your visualizations. Links let you create shortcuts to other dashboards, panels, and even external websites.

Grafana supports dashboard links, panel links, and data links. Dashboard links are displayed at the top of the dashboard. Panel links are accessible by clicking an icon on the top left corner of the panel.

Choosing which link to use

Start by figuring out how you're currently navigating between dashboards. If you're often jumping between a set of dashboards and struggling to find the same context in each, links can help optimize your workflow.

The next step is to figure out which link type is right for your workflow. Even though all the link types in Grafana are used to create shortcuts to other dashboards or external websites, they work in different contexts.

- If the link relates to most if not all of the panels in the dashboard, use dashboard links.
- If you want to drill down into specific panels, use panel links.
- If you want to link to an external site, you can use either a dashboard link or a panel link.
- If you want to drill down into a specific series, or even a single measurement, use data links.

Controlling the time range using a URL

To control the time range of a panel or dashboard, you can provide query parameters in the dashboard URL:

- `from` defines lower limit of the time range, specified in ms epoch.
- `to` defines upper limit of the time range, specified in ms epoch.
- `time` and `time.window` defines a time range from `time-time.window/2` to `time+time.window/2`. Both params should be specified in ms. For example ? `time=1500000000000&time.window=10000` will result in 10s time range from 1499999995000 to 1500000005000.

Dashboard links

When you create a dashboard link, you can include the time range and current template variables to directly jump to the same context in another dashboard. This way, you don't have to worry whether the person you send the link to is looking at the right data. For other types of links, see Data link variables.

Dashboard links can also be used as shortcuts to external systems, such as submitting a GitHub issue with the current dashboard name.

After adding a dashboard link, it will show up in the upper-right corner of your dashboard.

Adding links to dashboards

Add links to other dashboards at the top of your current dashboard.

To add a link to a dashboard

1. While viewing the dashboard you want to link, click the gear at the top of the screen to open **Dashboard settings**.
2. Select **Links** and then **Add Dashboard Link** or **New**.
3. In **Type**, select **dashboards**.
4. Select link options from the following.

   - **With tags** – Enter tags to limit the linked dashboards to only the ones with the tags you enter. Otherwise, Grafana includes links to all other dashboards.
   - **As dropdown** – If you are linking to many dashboards, By default, Grafana displays them all side-by-side across the top of your dashboard. Selecting this option and adding an optional title, will display the links in a dropdown.
   - **Time range** – Select this option to include the dashboard time range in the link. When the user clicks the link, the linked dashboard will open with the indicated time range already set.
   - **Variable values** – Select this option to include template variables currently used as query parameters in the link. When the user clicks the link, any matching templates in the linked dashboard are set to the values from the link. For more information, see Dashboard URL variables.
   - **Open in new tab** – Select this option if you want the dashboard link to open in a new tab or window.
5. Click **Add**.

Adding a URL link to a dashboard

Add a link to a URL at the top of your current dashboard. You can link to any available URL, including dashboards, panels, or external sites. You can even control the time range to ensure the user is zoomed in on the right data in Grafana.

To add a URL link to a dashboard

1. While viewing the dashboard you want to link, select the gear at the top of the screen to open **Dashboard settings**.

2. Select **Links** and then **Add Dashboard Link** or **New**.
3. In **Type**, select **Link**.
4. Select link options from the following.

   - **URL** – Enter the URL you want to link to. Depending on the target, you might want to include field values.
   - **Title** – Enter the title you want the link to display.
   - **Tooltip** – Enter the tooltip you want the link to display.
   - **Icon** – Choose the icon that you want displayed with the link.
   - **Time range** – Select this option to include the dashboard time range in the link. When the user clicks the link, the linked dashboard will open with the indicated time range set.
     - `from` – Defines lower limit of the time range, specified in ms epoch.
     - `to` – Defines upper limit of the time range, specified in ms epoch.
     - `time` and `time.window` – Define a time range from `time-time.window/2` to `time+time.window/2`. Both params should be specified in ms. For example ? `time=1500000000000&time.window=10000` will result in 10s time range from 1499999995000 to 1500000005000.
   - **Variable values** – Select this option to include template variables currently used as query parameters in the link. When the user clicks the link, any matching templates in the linked dashboard are set to the values from the link.

     The variable format is as follows:

     ```
     https://${you-domain}/path/to/your/dashboard?var-${template-varable1}=value1&var-{template-variable2}=value2
     ```

   - **Open in a new tab** – Select this option if you want the dashboard link to open in a new tab or window
5. Select **Add**.

Updating a dashboard link

To change or update an existing dashboard link, follow this procedure.

To update a dashboard link

1. In **Dashboard settings,** on the **Links** tab, select the existing link that you want to edit.
2. Change the settings and then choose **Update**.

Duplicating a dashboard link

To duplicate an existing dashboard link, select the duplicate icon next to the existing link that you want to duplicate.

Deleting a dashboard link

To delete an existing dashboard link, select the trash icon next to the link that you want to delete.

Panel links

Each panel can have its own set of links that are shown in the upper left corner of the panel. You can link to any available URL, including dashboards, panels, or external sites. You can even control the time range to ensure the user is zoomed in on the right data in Grafana.

To see available panel links, select the icon to the right of the panel title.

- **Adding a panel link**: Each panel can have its own set of links that are shown in the upper left corner of the panel. You can link to any available URL, including dashboards, panels, or external sites. You can even control the time range to ensure the user is zoomed in on the right data in Grafana. Click the icon on the top left corner of a panel to see available panel links.

  1. Hover your cursor over the panel to which you want to add a link.
  2. Select the menu, and choose **Edit**, or you can use the keyboard shortcut, `e`.
  3. Expand the **Panel options** section, and scroll down to **Panel links**.
  4. Select **Add link**.
  5. Enter a **Title**. This is a human-readable label for the link that will be displayed in the UI.
  6. Enter the **URL** you want to link to. Press `Ctrl+Space` (or `Cmd+Space`) and select the URL field to see available variables. By adding template variables to your panel link, the link sends the user to the right context, with the relevant variables already set.

     You can also use time variables.

     - `from` defines the lower limit of the time range, specified in ms epoch.

  - `to` defines the upper limit of the time range, specified in ms epoch.
  - `time` and `time.window` defines a time range from `time-time.window/2` to `time+time.window/2`. Both parameters should be specified in ms. For example `?time=1500000000000&time.window=10000` results in 10s time range from 1499999995000 to 1500000005000.

- **Updating a panel link**

  1. Select a panel (or place the cursor over the panel) to display an actions menu at the top right of the panel.
  2. From the menu, select the **Edit**.

     You can also use the keyboard shortcut, e.
  3. Expand the **Panel options** section, and scroll down to **Panel links**.
  4. Find the link that you want to change, and select the **Edit** (pencil) icon next to it.
  5. Make any necessary changes.
  6. Select **Save** to save changes and close the window.
  7. Save changes to your dashboard by selecting **Save** in the upper right.

- **Deleting a panel link**

  1. Select a panel (or place the cursor over the panel) to display an actions menu at the top right of the panel.
  2. From the menu, select the **Edit**.

     You can also use the keyboard shortcut, e.
  3. Expand the **Panel options** section, and scroll down to **Panel links**.
  4. Find the link that you want to delete, and select the **X** icon next to it.
  5. Select **Save** in the upper right to save your changes to the dashboard.

Annotate visualizations

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Annotations provide a way to mark points on a visualization with rich events. They are visualized as vertical lines and icons on all graph panels. When you hover over an annotation, you can get event description and event tags. The text field can include links to other systems with more detail.

Annotations support Cron syntax for time regions, providing more precise control over time-based annotations.

You can annotate visualizations in three ways:

- Directly in the panel, using the built-in annotations query.
- Using the Grafana HTTP API.
- Configuring annotation queries in the dashboard settings.

In the first two cases, you're creating new annotations, while in the last you're querying existing annotations from data sources. The built-in annotation query also supports this.

This section explains the first and third options; for information about using the Grafana HTTP API, refer to Annotations API.

Annotations are supported for the following visualization types:

- Time series
- State timeline
- Candlestick

Create annotations in panels

Grafana comes with the ability to add annotation events directly from a panel using the built-in annotations query that exists on all dashboards. Annotations that you create this way are stored in Grafana.

To add annotations directly in the panel:

- The dashboard must already be saved.

- The built-in query must be enabled.

To add an annotation

1. In the dashboard select the panel to which you're adding the annotation. A context menu will appear.
2. In the context menu, select **Add annotation**.
3. (Optional) Add an annotation description and tags.
4. Select **Save**.

Alternatively, to add an annotation, press `Ctrl` (or `Cmd`) while selecting the panel, and the **Add annotation** popover will appear.

Region annotations

You can also create annotations that cover a region, or period of time in a visualization.

To add a region annotation

1. In the dashboard press `Ctrl` (or `Cmd`) while selecting an area of the panel.
2. Add an annotation description and tags (optional).
3. Click **Save**.

To edit an annotation

1. In the dashboard, hover over an annotation indicator on a panel.
2. Select the **Edit** (pencil) icon in the annotation tooltip.
3. Modify the description and/or tags.
4. Select **Save**.

To delete an annotation

1. In the dashboard, hover over an annotation indicator on a panel.
2. Select the **Delete** (trash) icon in the annotation tooltip.

Fetch annotations through dashboard settings

In the dashboard settings, under **Annotations**, you can add new queries to fetch annotations using any data source, including the built-in data annotation data source. Annotation queries return events that can be visualized as event markers in graphs across the dashboard.

To add a new annotation query

1. Select the **Settings** (gear) icon in the dashboard header to open the settings menu.
2. Select **Annotations**.
3. Click **Add annotation query**.
4. Enter a name for the annotation query.

   This name is given to the toggle (checkbox) that will allow you to enable showing annotation events from this query.
5. Select the data source for the annotations.

   You can also choose **Open advanced data source picker** to see more options, including adding a new data source (available for Admins only).
6. If you don't want to use the annotation query right away, clear the **Enabled** checkbox.
7. If you don't want the annotation query toggle to be displayed in the dashboard, select the **Hidden** checkbox.
8. Select a color for the event markers.
9. In the **Show in** drop-down, choose one of the following options:

   - **All panels** – The annotations are displayed on all panels that support annotations.
   - **Selected panels** – The annotations are displayed on all the panels you select.
   - **All panels except** – The annotations are displayed on all panels except the ones you select.
10. Configure the query.

    The annotation query options are different for each data source. For information about annotations in a specific data source, see Connect to data sources.

Built-in query

After you add an annotation, they will still be visible. This is due to the built-in annotation query that exists on all dashboards. This annotation query will fetch all annotation events that originate

from the current dashboard, which are stored in Grafana, and show them on the panel where they were created. This includes alert state history annotations.

By default, the built-in annotation query uses the `Grafana` special data source, and manual annotations are only supported using this data source. You can use another data source in the built-in annotation query, but you'll only be able to create automated annotations using the query editor for that data source.

To add annotations directly to the dashboard, this query must be enabled.

To confirm the built-in query is enabled

1. Select the dashboard **settings** (gear) icon in the dashboard header to open the dashboard settings menu.
2. Select **Annotations**.
3. Find the **Annotations & Alerts (Built-in)** query.

   If it shows **Disabled** before the name of the query, then you'll need to select the query name to open it and update the setting.

To stop annotations from being fetched and drawn

1. Select the dashboard **settings** (gear) icon in the dashboard header to open the dashboard settings menu.
2. Select **Annotations**.
3. Select the **Annotations & Alerts (Built-in)** query.
4. Select the **Enabled** toggle to turn it off.

When you copy a dashboard using the **Save As** feature it will get a new dashboard id, so annotations created on the source dashboard will no longer be visible on the copy. You can still show them if you add a new **Annotation Query** and filter by tags. However, this only works if the annotations on the source dashboard had tags to filter by.

Filtering queries by tag

You can create new queries to fetch annotations from the built-in annotation query using the `Grafana` data source by setting **Filter by** to `Tags`.

For example, create an annotation query name `outages` and specify a tag `outage`. This query will show all annotations (from any dashboard or via API) with the `outage` tag. If multiple tags are defined in an annotation query, then Grafana will only show annotations matching all the tags. To modify the behavior, enable **Match any**, and Grafana will show annotations that contain any one of the tags you provided.

You can also use template variables in the tag query. This means if you have a dashboard showing stats for different services and a template variable that dictates which services to show, you can use the same template variable in your annotation query to only show annotations for those services.

Dashboard JSON model

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A dashboard in Grafana is represented by a JSON object, which stores metadata of its dashboard. Dashboard metadata includes dashboard properties, metadata from panels, template variables, and panel queries.

To view the JSON of a dashboard

1. Navigate to a dashboard.
2. In the top navigation menu, select the **Dashboard settings** (gear) icon.
3. Select **JSON Model**.

JSON fields

When a user creates a new dashboard, a new dashboard JSON object is initialized with the following fields.

Note

In the following JSON, id is shown as null, which is the default value assigned to it until a dashboard is saved. After a dashboard is saved, an integer value is assigned to the `id` field.

```
{
  "id": null,
  "uid": "cLV5GDCkz",
  "title": "New dashboard",
  "tags": [],
  "timezone": "browser",
  "editable": true,
  "graphTooltip": 1,
  "panels": [],
  "time": {
    "from": "now-6h",
    "to": "now"
  },
  "timepicker": {
    "time_options": [],
    "refresh_intervals": []
  },
  "templating": {
    "list": []
  },
  "annotations": {
    "list": []
  },
  "refresh": "5s",
  "schemaVersion": 17,
  "version": 0,
  "links": []
}
```

The following describes each field in the dashboard JSON.

Name	Usage
`id`	unique numeric identifier for the dashboard (generated by the db)

Name	Usage
`uid`	unique dashboard identifier that can be generated by anyone. string (8-40)
`title`	current title of dashboard
`tags`	tags associated with dashboard, an array of strings
`style`	theme of dashboard, such as `dark` or `light`
`timezone`	timezone of dashboard, such as `utc` or `browser`
`editable`	whether a dashboard is editable or not
`graphTooltip`	0 for no shared crosshair or tooltip (default), 1 for shared crosshair, 2 for shared crosshair and shared tooltip
`time`	time range for dashboard, such as `last 6 hours` or `last 7 days`
`timepicker`	timepicker metadata, see timepicker section for details
`templating`	templating metadata, see templating section for details
`annotations`	annotations metadata, see annotations for how to add them
`refresh`	auto-refresh interval
`schemaVersion`	version of the JSON schema (integer), incremented each time a Grafana update brings changes to this schema

Name	Usage
`version`	version of the dashboard (integer), incremented each time the dashboard is updated
`panels`	panels array (see the next section for detail)

Panels

Panels are the building blocks of a dashboard. It consists of data source queries, type of graphs, aliases, and more. Panel JSON consists of an array of JSON objects, each representing a different panel. Most of the fields are common for all panels but some fields depend on the panel type. The following is an example of panel JSON of a text panel.

```
"panels": [
  {
    "type": "text",
    "title": "Panel Title",
    "gridPos": {
      "x": 0,
      "y": 0,
      "w": 12,
      "h": 9
    },
    "id": 4,
    "mode": "markdown",
    "content": "# title"
  }
```

Panel size and position

The gridPos property describes the panel size and position in grid coordinates.

- `w` – 1 to 24 (the width of the dashboard is divided into 24 columns)
- `h` – In grid height units, each represents 30 pixels.
- `x` – The x position, in the same unit as `w`.
- `y` – The y position, in the same unit as `h`.

The grid has a negative gravity that moves up panels if there is empty space above a panel.

Timepicker

```
"timepicker": {
    "collapse": false,
    "enable": true,
    "notice": false,
    "now": true,
    "refresh_intervals": [
      "5s",
      "10s",
      "30s",
      "1m",
      "5m",
      "15m",
      "30m",
      "1h",
      "2h",
      "1d"
    ],
    "status": "Stable",
    "type": "timepicker"
  }
```

Templating

The `templating` field contains an array of template variables with their saved values along with some other metadata.

```
"templating": {
    "enable": true,
    "list": [
       {
        "allFormat": "wildcard",
        "current":  {
          "tags": [],
          "text": "prod",
          "value": "prod"
        },
        "datasource": null,
        "includeAll": true,
        "name": "env",
        "options": [
           {
```

```
            "selected": false,
            "text": "All",
            "value": "*"
          },
           {
            "selected": false,
            "text": "stage",
            "value": "stage"
          },
           {
            "selected": false,
            "text": "test",
            "value": "test"
          }
        ],
        "query": "tag_values(cpu.utilization.average,env)",
        "refresh": false,
        "type": "query"
      },
       {
        "allFormat": "wildcard",
        "current":  {
          "text": "apache",
          "value": "apache"
        },
        "datasource": null,
        "includeAll": false,
        "multi": false,
        "multiFormat": "glob",
        "name": "app",
        "options": [
           {
            "selected": true,
            "text": "tomcat",
            "value": "tomcat"
          },
           {
            "selected": false,
            "text": "cassandra",
            "value": "cassandra"
          }
        ],
        "query": "tag_values(cpu.utilization.average,app)",
        "refresh": false,
```

```
        "regex": "",
        "type": "query"
      }
    ]
  }
```

The following table describes the usage of the templating fields.

Name	Usage
`enable`	whether templating is enabled or not
`list`	an array of objects each representing one template variable
`allFormat`	format to use while fetching all values from data source, including `wildcard`, `glob`, `regex`, `pipe`.
`current`	shows current selected variable text/value on the dashboard
`datasource`	shows data source for the variables
`includeAll`	whether all value option is available or not
`multi`	whether multiple values can be selected or not from variable value list
`multiFormat`	format to use while fetching timeseries from data source
`name`	name of variable
`options`	array of variable text/value pairs available for selection on dashboard
`query`	data source query used to fetch values for a variable

Name	Usage
`refresh`	configures when to refresh a variable
`regex`	extracts part of a series name or metric node segment
`type`	type of variable, `custom`, `query`, or `interval`

Best practices for dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This section provides information about best practices for Grafana administrators and users about how to build and maintain Grafana dashboards.

For information about the different kinds of dashboards you can create, refer to the Grafana dashboards: A complete guide to all the different types you can build blog post on the Grafana Labs website.

Note

This section can help you create a strategy for your monitoring and dashboard maintenance. You know your systems best, and should use this section to guide your understanding. Ultimately, it is your responsibility to create the best strategy for your system.

Common observability strategies

When you have a lot to monitor, like a server farm, you need a strategy to decide what is important enough to monitor. This page describes several common methods for choosing what to monitor.

A logical strategy allows you to make uniform dashboards and scale your observability platform more easily.

Guidelines for strategies

- The USE method tells you how happy your machines are, the RED method tells you how happy your users are.
- USE reports on causes of issues.
- RED reports on user experience and is more likely to report symptoms of problems.
- Monitoring both is important to understanding your system. As a best practice, alert on symptoms rather than causes. Typically, alerting is configured on RED dashboards.

USE method

USE stands for:

- **Utilization** – Percent time the resource is busy, such as node CPU usage.
- **Saturation** – Amount of work a resource has to do, often queue length or node load.
- **Errors** – Count of error events.

This method is best for hardware resources in infrastructure, such as CPU, memory, and network devices. For more information, see Brendan Gregg's blog post The USE Method.

RED method

RED stands for:

- **Rate** – Requests per second
- **Errors** – Number of requests that are failing.
- **Duration** – Amount of time these requests take, distribution of latency measurements.

This method is most applicable to services, especially a microservices environment. For each of your services, instrument the code to expose these metrics for each component. RED dashboards are good for alerting and SLAs. A well-designed RED dashboard is a proxy for user experience.

For more information, see Tom Wilkie's blog post The RED method: How to instrument your services.

The Four Golden Signals

According to the Google SRE handbook, if you can only measure four metrics of your user-facing system, focus on these four.

This method is similar to the RED method, but it includes saturation.

- **Latency** – Time taken to serve a request.
- **Traffic** – How much demand is placed on your system.
- **Errors** – Rate of requests that are failing.
- **Saturation** – How "full" your system is,

Dashboard management maturity model

Dashboard management maturity refers to how well-designed and efficient your dashboard ecosystem is. We recommend periodically reviewing your dashboard setup to gauge where you are and how you can improve.

Broadly speaking, dashboard maturity can be defined as low, medium, or high.

Much of the content for this topic was taken from the KubeCon 2019 talk Fool-Proof Kubernetes Dashboards for Sleep-Deprived Oncalls.

Low – default state

At this stage, you have no coherent dashboard management strategy. Almost everyone starts here.

How can you tell you are here?

- Everyone can modify your dashboards.
- Lots of copied dashboards, little to no dashboard reuse.

- One-off dashboards that hang around forever.
- No version control (dashboard JSON in version control).
- Lots of browsing for dashboards, searching for the right dashboard. This means lots of wasted time trying to find the dashboard you need.
- Not having any alerts to direct you to the right dashboard.

Medium – methodical dashboards

At this stage, you are starting to manage your dashboard use with methodical dashboards. You might have laid out a strategy, but there are some things you could improve.

How can you tell you are here?

- Prevent sprawl by using template variables. For example, you don't need a separate dashboard for each node, you can use query variables. Even better, you can make the data source a template variable too, so you can reuse the same dashboard across different clusters and monitoring backends.

  Refer to the list of examples in Variables, for ideas.
- Methodical dashboards according to an observability strategy.
- Hierarchical dashboards with drill-downs to the next level.
- Dashboard design reflects service hierarchies. For example, you could use the RED method with one row per service. The row order could reflect the data flow, as you scroll down the dashboard.
- Compare like to like: split service dashboards when the magnitude differs. Make sure aggregated metrics don't drown out important information.
- Expressive charts with meaningful use of color and normalizing axes where you can.
  - Example of meaningful color: Blue means it's good, red means it's bad. Thresholds can help with that.
  - Example of normalizing axes: When comparing CPU usage, measure by percentage rather than raw number, because machines can have a different number of cores. Normalizing CPU usage by the number of cores reduces cognitive load because the viewer can trust that at 100% all cores are being used, without having to know the number of CPUs.
- Directed browsing cuts down on guessing.
  - Template variables make it harder to browse randomly or aimlessly.
  - Most dashboards should be linked to by alerts.

  - Browsing is directed with links. For more information, see Managing dashboard links.
- Version-controlled dashboard JSON.

High – optimized use

At this stage, you have optimized your dashboard management use with a consistent and thoughtful strategy. It requires maintenance, but the results are worth it.

- Actively reducing sprawl.
  - Regularly review existing dashboards to make sure they are still relevant.
  - Only approved dashboards added to master dashboard list.
  - Tracking dashboard use. You can take advantage of Usage insights.
- Consistency by design.
- Use scripting libraries to generate dashboards, ensure consistency in pattern and style.
  - grafonnet (Jsonnet)
  - grafanalib (Python)
- No editing in the browser. Dashboard viewers change views with variables.
- Browsing for dashboards is the exception, not the rule.
- Perform experimentation and testing in a separate Grafana instance dedicated to that purpose, not your production instance. When a dashboard in the test environment is proven useful, then add that dashboard to your main Grafana instance.

Best practices for creating dashboards

This section outlines some best practices to follow when creating Grafana dashboards.

Before you begin

Here are some principles to consider before you create a dashboard.

A dashboard should tell a story or answer a question

What story are you trying to tell with your dashboard? Try to create a logical progression of data, such as large to small or general to specific. What is the goal for this dashboard? (Hint: If the dashboard doesn't have a goal, then ask yourself if you really need the dashboard.)

Keep your graphs simple and focused on answering the question that you are asking. For example, if your question is "which servers are in trouble?", then maybe you don't need to show all the server data. Just show data for the ones in trouble.

Dashboards should reduce cognitive load, not add to it

Cognitive load is basically how hard you need to think about something in order to figure it out. Make your dashboard easy to interpret. Other users and future you (when you're trying to figure out what broke at 2AM) will appreciate it.

Ask yourself:

- Can I tell what exactly each graph represents? Is it obvious, or do I have to think about it?
- If I show this to someone else, how long will it take them to figure it out? Will they get lost?

Have a monitoring strategy

It's easy to make new dashboards. It's harder to optimize dashboard creation and adhere to a plan, but it's worth it. This strategy should govern both your overall dashboard scheme and enforce consistency in individual dashboard design.

Refer to Common observability strategies and Dashboard management maturity levels for more information.

Write it down

Once you have a strategy or design guidelines, write them down to help maintain consistency over time.

Best practices to follow

- When creating a new dashboard, make sure it has a meaningful name.
  - If you are creating a dashboard to play or experiment, then put the word `TEST` or `TMP` in the name.
  - Consider including your name or initials in the dashboard name or as a tag so that people know who owns the dashboard.
  - Remove temporary experiment dashboards when you are done with them.
- If you create many related dashboards, think about how to cross-reference them for easy navigation. For more information, see Best practices for managing dashboards, later in this section.

- Grafana retrieves data from a data source. A basic understanding of Connect to data sources in general, and your specific data sources is important.
- Avoid unnecessary dashboard refreshing to reduce the load on the network or backend. For example, if your data changes every hour, then you don't need to set the dashboard refresh rate to 30 seconds.
- Use the left and right Y-axes when displaying time series with different units or ranges.
- Add documentation to dashboards and panels.
  - To add documentation to a dashboard, add a Text panel visualization to the dashboard. Record things like the purpose of the dashboard, useful resource links, and any instructions users might need to interact with the dashboard.
  - To add documentation to a panel, edit the panel settings and add a description. Any text you add will appear if you hover your cursor over the small `i` in the top left corner of the panel.
- Reuse your dashboards and enforce consistency by using templates and variables.
- Be careful with stacking graph data. The visualizations can be misleading, and hide important data. We recommend turning it off in most cases.

Best practices for managing dashboards

This page outlines some best practices to follow when managing Grafana dashboards.

Before you begin

Here are some principles to consider before you start managing dashboards.

Strategic observability

There are several common observability strategies. You should research them and decide whether one of them works for you or if you want to come up with your own. Either way, have a plan, write it down, and stick to it.

Adapt your strategy to changing needs as necessary.

Maturity level

What is your dashboard maturity level? Analyze your current dashboard setup and compare it to the Dashboard management maturity model. Understanding where you are can help you decide how to get to where you want to be.

Best practices to follow

- Avoid dashboard sprawl, meaning the uncontrolled growth of dashboards. Dashboard sprawl negatively affects time to find the right dashboard. Duplicating dashboards and changing "one thing" (worse: keeping original tags) is the easiest kind of sprawl.
  - Periodically review the dashboards and remove unnecessary ones.
  - If you create a temporary dashboard, perhaps to test something, prefix the name with `TEST:`. Delete the dashboard when you are finished.
- Copying dashboards with no significant changes is not a good idea.
  - You miss out on updates to the original dashboard, such as documentation changes, bug fixes, or additions to metrics.
  - In many cases copies are being made to simply customize the view by setting template parameters. This should instead be done by maintaining a link to the master dashboard and customizing the view with URL parameters.
- When you must copy a dashboard, clearly rename it and *do not* copy the dashboard tags. Tags are important metadata for dashboards that are used during search. Copying tags can result in false matches.
- Maintain a dashboard of dashboards or cross-reference dashboards. This can be done in several ways:
  - Create dashboard links, panel, or data links. Links can go to other dashboards or to external systems. For more information, refer to Manage dashboard links.
  - Add a Dashboard list panel. You can then customize what you see by doing tag or folder searches.
  - Add a Text panel and use markdown to customize the display.

Managing dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

On the **Dashboards** page of your workspace (available by selecting **Dashboards** from the left menu), you can perform dashboard management tasks, including organizing your dashboards into folders.

For more information about creating dashboards, see Building dashboards.

Browse dashboards

On the **Dashboards** page, you can browse and manage folders and dashboards. This includes options to:

- Create folders and dashboards.
- Move dashboards between folders.
- Delete multiple dashboards and folders.
- Navigate to a folder.
- Manage folder permissions. For more information, see Dashboard and folder permissions.

Creating dashboard folders

Folders help you organize and group dashboards, which is useful when you have many dashboards or multiple teams using the same Grafana instance. Subfolders allow you to create a nested hierarchy in your dashboard organization.

Prerequisites

Ensure that you have Grafana Admin permissions. For more information about dashboard permissions, see Dashboard and folder permissions.

To create a dashboard folder

1. Sign in to Grafana.
2. On the left menu, select **Dashboards**.
3. On the **Dashboards** page, select **New** then choose **New folder** in the drop down.
4. Enter a unique name and click **Create**.

Note

When you save a dashboard, you can either select a folder for the dashboard to be saved in or create a new folder.

To edit the name of a folder

1. Select **Dashboards**in the left menu.
2. Select the folder to rename
3. Select the **Edit title** (pencil) icon in the header and update the name of the folder.

   The new folder name is automatically saved.

Folder permissions

You can assign permissions to a folder. Dashboard in the folder inherit any permissions that you've assigned to the folder. You can assign permissions to organization roles, teams, and users.

To modify permissions for a folder

1. Select **Dashboards** from the left menu.
2. Select the folder in the list.
3. On the folder's details page, select **Folder actions** and select **Manage permissions** in the drop down list.
4. Update the permissions as desired.

Changes are saved automatically.

Managing playlists

This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A *playlist* is a list of dashboards that are displayed in a sequence. You might use a playlist to build situational awareness or to present your metrics to your team or visitors. Grafana automatically scales dashboards to any resolution, which makes them perfect for large screens. You can access the playlist feature from Grafana's side menu in the **Dashboards** submenu.

Accessing, sharing, and controlling a playlist

Use the information in this section to access existing playlists. Start and control the display of a playlist using one of the five available modes.

To access a playlist

1. Select **Playlists** from the left menu.
2. Choose a playlist from the list of existing playlists.

Starting a playlist

You can start a playlist in five different view modes. View mode determines how the menus and navigation bar appear on the dashboards.

By default, each dashboard is displayed for the amount of time entered in the **Interval** field, which you set when you create or edit a playlist. After you start a playlist, you can control it with the navigation bar at the top of the page.

To start a playlist

1. Access the playlist page to see a list of existing playlists.
2. Find the playlist that you want to start, then click **Start playlist**.

   The start playlist dialog box will open.
3. Select one of the five playlist modes available based on the information in the following table.
4. Click **Start <playlist name>**.

The playlist displays each dashboard for the time specified in the `Interval` field, set when creating or editing a playlist. After a playlist starts, you can control it using the navigation bar at the top of your screen.

Mode	Description
Normal mode	• The side menu remains visible. • The navigation bar, row, and panel controls appear at the top of the screen.
TV mode	• The side menu and dashboard submenu (including variable dropdowns and dashboard links) are hidden or removed. • The navigation bar, row, and panel controls appear at the top of the screen. • Enabled automatically after one minute of user inactivity. • Enable it manually using the `d v` sequence shortcut, or by appending the parameter `?inactive` to the dashboard URL. • Disable it with any pointer movement or keyboard action.
TV mode (with auto fit panels)	• The navigation bar, row, and panel controls appear at the top of the screen. • Dashboard panels automatically adjust to optimize space on screen.
Kiosk mode	• The side menu, navigation bar, row and panel controls are completely hidden/re moved from view. • You can enable it manually using the `d v` sequence shortcut after the playlist has started. • You can disable it manually with the same shortcut.
Kiosk mode (with auto fit panels)	• The side menu, navigation bar, row, and panel controls are completely hidden/re moved from view.

Mode	Description
	• Dashboard panels automatically adjust to optimize space on screen.

Controlling a playlist

You can control a playlist in **Normal** or **TV** mode after it has started, using the navigation bar at the top of your screen. Press the `Esc` key in your keyboard to stop the playlist.

Button	Action
Next (double-right arrow)	Advances to the next dashboard.
Back (left arrow)	Returns to the previous dashboard.
Stop (square)	Ends the playlist, and exits to the current dashboard.
Cycle view mode (monitor icon)	Rotates the display of the dashboards in different view modes.
Time range	Displays data within a time range. It can be set to display the last 5 minutes up to 5 years ago, or a custom time range, using the down arrow.
Refresh (circle arrow)	Reloads the dashboard, to display the current data. It can be set to reload automatically every 5 seconds to 1 day, using the dropdown arrow.

Creating a playlist

You can create a playlist to present dashboards in a sequence with a set order and time interval between dashboards.

To create a playlist

1. Select **Dashboards** from the left menu.

2. Select **Playlists** on the playlist page.
3. Select **New playlist**.
4. Enter a descriptive name in the **Name** text box.
5. Enter a time interval in the **Interval** text box. The dashboards you add are listed in a sequential order.
6. In **Dashboards**, add existing dashboards to the playlist using the **Add by title** and **Add by tag** dropdown options.
7. Optionally:
   - Search for a dashboard by its name, a regular expression, or a tag.
   - Filter your results by starred status or tags.
   - Rearrange the order of the dashboards you have added using the up and down arrow icon.
   - Remove a dashboard from the playlist by clicking the **x** icon beside the dashboard.
8. Select **Save** to save your changes.

Saving a playlist

You can save a playlist and add it to your **Playlists** page, where you can start it.

> **Important**
>
> Ensure all the dashboards that you want to appear in your playlist are added when creating or editing the playlist before saving it.

To save a playlist

1. Select **Dashboards** in the left menu.
2. Select **Playlists** to view the playlists available to you.
3. Choose the playlist of your choice.
4. Edit the playlist.
5. Check that the playlist has a **Name**, **Interval**, and at least one **Dashboard** added to it.
6. Select **Save** to save your changes.

Editing or deleting a playlist

You can edit a playlist by updating its name, interval time, and by adding, removing, and rearranging the order of dashboards, or you can delete the playlist.

To edit a playlist

1. Select **Edit playlist** on the playlist page.
2. Update the name and time interval, then add or remove dashboards from the playlist using instructions in Create a playlist, above.
3. Select **Save** to save your changes.

To delete a playlist

1. Select **Playlists**.
2. Select **Remove** next to the playlist you want to delete.

To rearrange dashboard order in a playlist

1. Next to the dashboard you want to move, click the up or down arrow.
2. Select **Save** to save your changes.

To remove a dashboard

1. Select **Remove** to remove a dashboard from the playlist.
2. Select **Save** to save your changes.

Sharing a playlist in view mode

You can share a playlist by copying the link address on the view mode you prefer, and pasting the URL to your destination.

To share a playlist in view mode

1. From the **Dashboards** left side menu, choose **Playlists**.
2. Select **Start playlist** next to the playlist you want to share.
3. In the dropdown, right click the view mode you prefer.

4. Select **Copy Link Address** to copy the URL to your clipboard.
5. Paste the URL to your destination.

Sharing dashboards and panels

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana enables you to share dashboards and panels with other users within an organization and in certain situations, publicly on the Web. You can share using:

- A direct link
- A snapshot
- An export link (for dashboards only)

You must have an authorized viewer permission to see an image rendered by a direct link.

When you share a panel or dashboard as a snapshot, a snapshot (which is a panel or dashboard at the moment you take the snapshot) is publicly available on the web. Anyone with a link to it can access it. Because snapshots do not require any authorization to view, Grafana removes information related to the account it came from, as well as any sensitive data from the snapshot.

Sharing a dashboard

You can share a dashboard as a direct link or as a snapshot. You can also export a dashboard. The sharing workflow includes customizable shareable panel images and a streamlined interface.

Note

If you change a dashboard, ensure that you save the changes before sharing.

To Share a dashboard

1. Select **Dashboards** from the left menu in your workspace.
2. Choose the dashboard you want to share.
3. Select the share icon at the top of the screen.

   The share dialog box opens and shows the **Link** tab.

Sharing a direct link

The **Link** tab shows the current time range, template variables, and the default theme. You can also share a shortened URL.

To share a directly link

1. Select **Copy**. This action copies the default or the shortened URL to the clipboard.
2. Send the copied URL to a Grafana user with authorization to view the link.

Publishing a snapshot

A dashboard snapshot shares an interactive dashboard publicly. Grafana strips sensitive data such as queries (metric, template, and annotation) and panel links, leaving only the visible metric data and series names embedded in the dashboard. Dashboard snapshots can be accessed by anyone with the link.

You can publish snapshots to your local instance.

To publish a snapshot

1. Select the **Snapshot** tab.
2. Select the **Local Snapshot**.
3. Grafana generates a link of the snapshot. Copy the snapshot link, and share it either within your organization or publicly on the web.

Exporting a dashboard

Grafana dashboards can easily be exported and imported. For more information, see the import and export sections in Building dashboards.

Sharing a panel

You can share a panel as a direct link, or as a snapshot. You can also create library panels using the **Share** option on any panel.

To share a panel

1. Select the panel title of the panel you want to share. The panel menu opens.
2. Select **Share**. The share dialog box will open and show the **Link** tab.

Using a direct link

The **Link** tab shows the current time range, template variables, and the default theme. You can optionally enable a shortened URL to share.

To use a direct link

1. Select **Copy** to copy the default or the shortened URL to the clipboard.
2. Send the copied URL to a Grafana user with authorization to view the link.

Publishing a snapshot of a panel

A panel snapshot is a share of an interactive panel publicly. Grafana strips sensitive data leaving only the visible metric data and series names embedded in the dashboard. Panel snapshots can be accessed by anyone with the link.

You can publish snapshots to your local instance.

To publish a snapshot of a panel

1. In the **Share Panel** dialog box, select the **Snapshot** tab.
2. Select **Local Snapshot**. Grafana generates the link of the snapshot.
3. Copy the snapshot link, and share it either within your organization or publicly on the web.

If you created a snapshot by mistake, click **Delete snapshot** to remove the snapshot from your Grafana instance.

Creating a library panel

To create a library panel from the **Share Panel** dialog box.

To create a library panel

1. Select **Library panel**.
2. In **Library panel name**, enter the name.
3. In **Save in folder**, select the folder in which to save the library panel. By default, the root folder is selected.
4. Select **Create library panel** to save your changes.
5. Save the dashboard.

Variables

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A variable is a placeholder for a value. You can use variables in metric queries and in panel titles. So when you change the value, using the dropdown at the top of the dashboard, your panel's metric queries will change to reflect the new value.

Variables allow you to create more interactive and dynamic dashboards. Instead of hard-coding things like server, application, and sensor names in your metric queries, you can use variables in their place. Variables are displayed as dropdown lists at the top of the dashboard. These dropdowns make it easy to change the data being displayed in your dashboard.

These can be especially useful for administrators who want to allow Grafana viewers to quickly adjust visualizations but do not want to give them full editing permissions. Grafana Viewers can use variables.

Variables and templates also allow you to single-source dashboards. If you have multiple identical data sources or servers, you can make one dashboard and use variables to change what you are viewing. This simplifies maintenance and upkeep enormously.

Templates

A template is any query that contains a variable. For example, if you were administering a dashboard to monitor several servers, you could make a dashboard for each server, or you could create one dashboard and use panels with template queries, such as the following.

```
wmi_system_threads{instance=~"$server"}
```

Variable values are always synced to the URL using the syntax var-<varname>=value.

Examples

Variables are listed in dropdown lists across the top of the screen. Select different variables to see how the visualizations change.

To see variable settings, navigate to **Dashboard Settings > Variables**. Click a variable in the list to see its settings.

Variables can be used in titles, descriptions, text panels, and queries. Queries with text that starts with $ are templates. Not all panels will have template queries.

Variable best practices

- Variable dropdown lists are displayed in the order they are listed in the variable list in **Dashboard settings**.
- Put the variables that you will change often at the top, so they will be shown first (far left on the dashboard).
- Variables preselect the topmost value in the dropdown list by default. If you want to choose an empty value instead, change the variable settings, as follows:
  1. Select the **Include All Option** checkbox.
  2. In the **Custom all value** field, enter the value +.

Topics

- Add and manage variables
- Inspect Variables
- Variable syntax

Add and manage variables

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The following table lists the types of variables in Grafana.

Variable type	Description
Query	Query-generated list of values such as metric names, server names, sensor IDs, data centers, and so on. Query variables also support static options, giving you more control over variable values without requiring a data source query.
Custom	Define the variable options manually using a comma-separated list.
Text box	Display a free text input field with an optional default value.
Constant	Define a hidden constant.
Data source	Quickly change the data source for an entire dashboard.
Interval	Interval variables represent time spans.
Ad hoc filters	Key-value filters that are automatically added to all metric queries for a data source. Ad hoc filters provide improved support across data sources for dynamically filtering dashboard data.

Variable type	Description
Switch	Quickly toggle between values in queries, simplifying dashboard interactions.
Global variables	Built-in variables that can be used in expressio ns in the query editor.
Chained variables	Variable queries can contain other variables.

Topics

- Entering General options
- Adding a query variable
- Adding a custom variable
- Adding a text box variable
- Adding a constant variable
- Adding a data source variable
- Adding an interval variable
- Adding ad hoc filters
- Configure variable selection options
- Global variables
- Chained variables
- Manage variables
- Filter variables with regex

Entering General options

You must enter general options for any type of variable that you create.

To enter general options

1. Navigate to the dashboard you want to make a variable for and select the **Dashboard settings** (gear) icon at the top of the page.
2. On the **Variables** tab, select **New variable**.

3. Enter a **Name** for the variable.
4. In the **Type** list, select **Query**.
5. (Optional) In **Label**, enter the display name of the variable dropdown.

   If you don't enter a display name, then the dropdown label is the variable name.
6. Choose a **Hide** option:

   - **No selection (blank)** – The variable dropdown displays the variable **Name** or **Label** value.
   - **Label** – The variable dropdown only displays the selected variable value and a down arrow.
   - **Variable** – No variable dropdown is displayed on the dashboard.

Adding a query variable

Query variables enable you to write a data source query that can return a list of metric names, tag values, or keys. For example, a query variable might return a list of server names, sensor IDs, or data centers. The variable values change as they dynamically fetch options with a data source query.

Query variables are generally only supported for strings. If your query returns numbers or any other data type, you might need to convert them to strings in order to use them as variables. For the Azure data source, for example, you can use the tostring function for this purpose.

Query expressions can contain references to other variables and in effect create linked variables. Grafana detects this and automatically refreshes a variable when one of its linked variables change.

> **Note**
>
> Query expressions are different for each data source. For more information, refer to the documentation for your data source.

To add a query variable

1. Enter general options, as above.
2. In the **Data source** list, select the target data source for the query.
3. In the **Refresh** list, select when the variable should update options.

- **On Dashboard Load** – Queries the data source every time the dashboard loads. This slows down dashboard loading, because the variable query needs to be completed before dashboard can be initialized.
- **On Time Range Change** – Queries the data source when the dashboard time range changes. Only use this option if your variable options query contains a time range filter or is dependent on the dashboard time range.

4. In the **Query** field, enter a query.

   - The query field varies according to your data source. Some data sources have custom query editors.
   - The query must return values named `__text` and `__value`. For example, in SQL, you can use a query such as `SELECT hostname AS __text, id AS __value from MyTable`. Queries for other languages will vary depending on syntax.
   - If you need more room in a single input field query editor, then hover your cursor over the lines in the lower right corner of the field and drag downward to expand.

5. (Optional) In the **Regex** field, type a regex expression to filter or capture specific parts of the names returned by your data source query. To see examples, refer to Filter variables with regex.
6. In the **Sort** list, select the sort order for values to be displayed in the dropdown list. The default option, **Disabled**, means that the order of options returned by your data source query will be used.
7. (Optional) Enter Selection Options.
8. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
9. Select **Add** to add the variable to the dashboard.

Adding a custom variable

Use a *custom* variable for a value that does not change, such as a number or a string.

For example, if you have server names or Region names that never change, then you might want to create them as custom variables rather than query variables. Because they do not change, you might use them in chained variables rather than other query variables. That would reduce the number of queries Grafana must send when chained variables are updated.

To add a custom variable

1. Enter general options, as above.
2. In the **Values separated by comma** list, enter the values for this variable in a comma-separated list. You can include numbers, strings, or key-value pairs separated by a space and a colon. For example, `key1 : value1,key2 : value2`.
3. (Optional) Enter Selection Options.
4. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
5. Select **Add** to add the variable to the dashboard.

Adding a text box variable

Text box variables display a free text input field with an optional default value. This is the most flexible variable, because you can enter any value. Use this type of variable if you have metrics with high cardinality or if you want to update multiple panels in a dashboard at the same time.

To add a text box variable

1. Enter general options, as above.
2. (Optional) In the **Default value** field, select the default value for the variable. If you do not enter anything in this field, then Grafana displays an empty text box for users to type text into.
3. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
4. Select **Add** to add the variable to the dashboard.

Adding a constant variable

Constant variables enable you to define a hidden constant. This is useful for metric path prefixes for dashboards you want to share. When you export a dashboard, constant variables are converted to import options.

Constant variables are *not* flexible. Each constant variable only holds one value, and it cannot be updated unless you update the variable settings.

Constant variables are useful when you have complex values that you need to include in queries but don't want to retype in every query. For example, if you had a server path called `i-0b6a61efe2ab843gg`, then you could replace it with a variable called `$path_gg`.

To add a constant variable

1. Enter general options, as above.
2. In the **Value** field, enter the variable value. You can enter letters, numbers, and symbols. You can even use wildcards if you use raw format.
3. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
4. Select **Add** to add the variable to the dashboard.

Adding a data source variable

Data source variables enable you to quickly change the data source for an entire dashboard. They are useful if you have multiple instances of a data source, perhaps in different environments.

To add a data source variable

1. Enter general options, as above.
2. In the **Type** list, select the target data source for the variable.

   You can also choose **Open advanced data source picker** to see more options, including adding a data source (Admins only). For more information, see Connect to data sources.
3. (Optional) In **Instance name filter**, enter a regex filter for which data source instances to choose from in the variable value dropdown list. Leave this field empty to display all instances.
4. (Optional) Enter Selection Options.
5. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
6. Select **Add** to add the variable to the dashboard.

Adding an interval variable

Use an *interval* variable to represents time spans such as `1m`,`1h`, or `1d`. You can think of them as a dashboard-wide *group by time* command. Interval variables change how the data is grouped in

the visualization. You can also use the Auto Option to return a set number of data points per time span.

You can use an interval variable as a parameter to group by time (for InfluxDB), date histogram interval (for Elasticsearch), or as a summarize function parameter (for Graphite).

To add an interval variable

1. Enter general options, as above.
2. In the **Values** field, enter the time range intervals that you want to appear in the variable dropdown list. The following time units are supported: `s (seconds)`, `m (minutes)`, `h (hours)`, `d (days)`, `w (weeks)`, `M (months)`, and `y (years)`. You can also accept or edit the default values: `1m,10m,30m,1h,6h,12h,1d,7d,14d,30d`.
3. (Optional) Turn on the **Auto Option** if you want to add the `auto` option to the list. This option allows you to specify how many times the current time range should be divided to calculate the current `auto` time span. If you turn it on, then two more options appear:
   - **Step count** – Select the number of times the current time range will be divided to calculate the value, similar to the **Max data points** query option. For example, if the current visible time range is 30 minutes, then the `auto` interval groups the data into 30 one-minute increments. The default value is 30 steps.
   - **Min Interval** – The minimum threshold below which the step count intervals will not divide the time. To continue the 30 minute example, if the minimum interval is set to 2m, then Grafana would group the data into 15 two-minute increments.
4. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
5. Select **Add** to add the variable to the dashboard.

Interval variable examples

The following example shows a template variable `myinterval` in a Graphite function:

```
summarize($myinterval, sum, false)
```

Adding ad hoc filters

Ad hoc filters enable you to add key-value filters that are automatically added to all metric queries that use the specified data source. Unlike other variables, you do not use ad hoc filters in queries. Instead, you use ad hoc filters to write filters for existing queries.

> **Note**
>
> Ad hoc filter variables only work with Prometheus, Loki, InfluxDB, and Elasticsearch data sources.

1. Enter general options, as above.
2. In the **Data source** list, select the target data source.

   You can also choose **Open advanced data source picker** to see more options, including adding a data source (Admins only). For more information, see Connect to data sources.
3. Select **Add** to add the variable to the dashboard.

Create ad hoc filters

Ad hoc filters are one of the most complex and flexible variable options available. Instead of a regular list of variable options, this variable allows you to build a dashboard-wide ad hoc query. Filters you apply in this manner are applied to all panels on the dashboard.

Configure variable selection options

Selection Options are a feature you can use to manage variable option selections. All selection options are optional, and they are off by default.

Multi-value variables

Interpolating a variable with multiple values selected is tricky as it is not straight forward how to format the multiple values into a string that is valid in the given context where the variable is used. Grafana tries to solve this by allowing each data source plugin to inform the templating interpolation engine what format to use for multiple values.

> **Note**
>
> The **Custom all value** option on the variable must be blank for Grafana to format all values into a single string. If it is left blank, then Grafana concatenates (adds together) all the values in the query. For example, `value1,value2,value3`. If a custom `all` value is used, then instead the value will be `*` or `all`.

Multi-value variables with a Graphite data source

Graphite uses glob expressions. A variable with multiple values is, in this case, be interpolated as `{host1,host2,host3}` if the current variable value iss *host1*, *host2*, and *host3*.

Multi-value variables with a Prometheus or InfluxDB datasource

InfluxDB and Prometheus use regex expressions, so the same variable is interpolated as `(host1|host2|host3)`. Every value is also regex escaped. If it were not, a value with a regex control character would break the regex expression.

Multi-value variables with an Elastic data source

Elasticsearch uses lucene query syntax, so the same variable is formatted as `("host1" OR "host2" OR "host3")`. In this case, every value is escaped so that the value only contains lucene control words and quotation marks.

Troubleshoot multi-value variables

Automatic escaping and formatting can cause problems and it can be tricky to grasp the logic behind it. Especially for InfluxDB and Prometheus where the use of regex syntax requires that the variable is used in regex operator context.

If you do not want Grafana to do this automatic regex escaping and formatting, then you must do one of the following:

- Turn off the **Multi-value** or **Include All option** options.
- Use the raw format.

Include All option

Grafana adds an `All` option to the variable dropdown list. If a user selects this option, then all variable options are selected.

Custom all value

This option is only visible if the **Include All option** is selected.

Enter regex, globs, or lucene syntax in the **Custom all value** field to define the value of the `All` option.

By default the `All` value includes all options in a combined expression. This can become very long and can have performance problems. Sometimes it can be better to specify a custom all value, like a wildcard regex.

To have custom regex, globs, or lucene syntax in the **Custom all value** option, it is never escaped so you will have to think about what is a valid value for your data source.

Global variables

Grafana has global built-in variables that can be used in expressions in the query editor. This topic lists them in alphabetical order and defines them. These variables are useful in queries, dashboard links, panel links, and data links.

$__dashboard

This variable is the name of the current dashboard.

$__from and $__to

Grafana has two built-in time range variables: `$__from` and `$__to`. They are currently always interpolated as epoch milliseconds by default, but you can control date formatting.

Syntax	Example result	Description
`${__from}`	1594671549254	Unix millisecond epoch
`${__from:date}`	2020-07-13T20:19:09.254Z	No args, defaults to ISO 8601/RFC 3339
`${__from:date:iso}`	2020-07-13T20:19:09.254Z	ISO 8601/RFC 3339
`${__from:date:seconds}`	1594671549	Unix seconds epoch

Syntax	Example result	Description
`${__from:date:YYYY-MM}`	2020-07	Any custom date format that does not include the : character

The syntax above also works with `${__to}`.

$__interval

You can use the `$__interval` variable as a parameter to group by time (for InfluxDB, MySQL, Postgres, MSSQL), Date histogram interval (for Elasticsearch), or as a *summarize* function parameter (for Graphite).

Grafana automatically calculates an interval that can be used to group by time in queries. When there are more data points than can be shown on a graph, the queries can be made more efficient by grouping by a larger interval. For example, if you are looking at a graph of 3 months worth of data, you might not be able to see detail at the minute level. Grouping by the hour or day makes the query more efficient without affecting what the graph shows. The `$__interval` is calculated using the time range and the width of the graph (the number of pixels).

Approximate Calculation: `(to - from) / resolution`

For example, when the time range is 1 hour and the graph is full screen, then the interval might be calculated to `2m` - points are grouped in 2 minute intervals. If the time range is 6 months and the graph is full screen, then the interval might be `1d` (1 day) - points are grouped by day.

In the InfluxDB data source, the legacy variable `$interval` is the same variable. `$__interval` should be used instead.

The InfluxDB and Elasticsearch data sources have `Group by time interval` fields that are used to hard code the interval or to set the minimum limit for the `$__interval` variable (by using the > syntax, for example `>10m`).

$__interval_ms

This variable is the `$__interval` variable in milliseconds, not a time interval formatted string. For example, if the `$__interval` is `20m` then the `$__interval_ms` is `1200000`.

$__org

This variable is the ID of the current organization. `${__org.name}` is the name of the current organization.

$__user

`${__user.id}` is the ID of the current user. `${__user.login}` is the login handle of the current user. `${__user.email}` is the email for the current user.

$__range

Only supported for Prometheus and Loki data sources. This variable represents the range for the current dashboard. It is calculated by `to - from`. It has a millisecond and a second representation called `$__range_ms` and `$__range_s`.

$__rate_interval

Only supported for Prometheus data sources. The `$__rate_interval` variable is meant to be used in the rate function.

$timeFilter or $__timeFilter

The `$timeFilter` variable returns the currently selected time range as an expression. For example, the time range interval `Last 7 days` expression is `time > now() - 7d`.

This is used in several places, including:

- The WHERE clause for the InfluxDB data source. Grafana adds it automatically to InfluxDB queries when in Query Editor mode. You can add it manually in Text Editor mode: `WHERE $timeFilter`.
- Log Analytics queries in the Azure Monitor data source.
- SQL queries in MySQL, Postgres, and MSSQL.
- The `$__timeFilter` variable is used in the MySQL data source.

$__timezone

The `$__timezone` variable returns the currently selected time zone, either `utc` or an entry of the IANA time zone data base (for example, `America/New_York`).

If the currently selected time zone is *Browser Time*, Grafana will try to determine your browser time zone.

Chained variables

Chained variables, also called *linked variables* or *nested variables*, are query variables with one or more other variables in their variable query.

Chained variable queries are different for every data source, but the premise is the same for all. You can use chained variable queries in any data source that allows them.

Extremely complex linked templated dashboards are possible, 5 or 10 levels deep. Technically, there is no limit to how deep or complex you can go, but the more links you have, the greater the query load.

Best practices and tips

The following practices will make your dashboards and variables easier to use.

Creating new linked variables

- Chaining variables create parent/child dependencies. You can envision them as a ladder or a tree.
- The easiest way to create a new chained variable is to copy the variable that you want to base the new one on. In the variable list, click the **Duplicate variable** icon to the right of the variable entry to create a copy. You can then add on to the query for the parent variable.
- New variables created this way appear at the bottom of the list. You might need to drag it to a different position in the list to get it into a logical order.

Variable order

You can change the orders of variables in the dashboard variable list by clicking the up and down arrows on the right side of each entry. Grafana lists variable dropdowns left to right according to this list, with the variable at the top on the far left.

- List variables that do not have dependencies at the top, before their child variables.
- Each variable should follow the one it is dependent on.
- Remember there is no indication in the UI of which variables have dependency relationships. List the variables in a logical order to make it easy on other users (and yourself).

Complexity consideration

The more layers of dependency you have in variables, the longer it will take to update dashboards after you change variables.

For example, if you have a series of four linked variables (country, Region, server, metric) and you change a root variable value (country), then Grafana must run queries for all the dependent variables before it updates the visualizations in the dashboard.

Manage variables

The variables page lets you add variables and manage existing variables. It also allows you to inspect variables and identify whether a variable is being referenced (or used) in other variables or dashboard.

Move – You can move a variable up or down the list using drag and drop.

Clone – To clone a variable, click the clone icon from the set of icons on the right. This creates a copy of the variable with the name of the original variable prefixed with `copy_of_`.

Delete – To delete a variable, click the trash icon from the set of icons on the right.

Filter variables with regex

Using the Regex Query option, you filter the list of options returned by the variable query or modify the options returned.

This page shows how to use regex to filter/modify values in the variable dropdown.

Using the Regex Query Option, you filter the list of options returned by the Variable query or modify the options returned. For more information, refer to the Mozilla guide on Regular expressions.

The following examples show filtering on the following list of options

```
backend_01
backend_02
backend_03
backend_04
```

Filter so that only the options that end with `01` or `02` are returned

Regex:

```
/
(
01|02
)
$/
```

Result:

```
backend_01
backend_02
```

Filter and modify the options using a regex capture group to return part of the text

Regex:

```
/.*
(
01|02
)
/
```

Result:

```
01
02
```

Filter and modify - Prometheus Example

For this list of options:

```
up{instance="demo.robustperception.io:9090",job="prometheus"} 1 1521630638000
up{instance="demo.robustperception.io:9093",job="alertmanager"} 1 1521630638000
up{instance="demo.robustperception.io:9100",job="node"} 1 1521630638000
```

This regex:

```
/. *instance="
(
[^"]*
)
.*/
```

Returns these results:

```
demo.robustperception.io:9090
demo.robustperception.io:9093
demo.robustperception.io:9100
```

Filter and modify using named text and value capture groups

Using named capture groups, you can capture separate 'text' and 'value' parts from the options returned by the variable query. This allows the variable dropdown list to contain a friendly name for each value that can be selected.

For example, when querying the node_hwmon_chip_names Prometheus metric, the chip_name is a lot friendlier that the chip value. So the following variable query result:

```
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_0",chip_name="enp216s0f0np0"} 1
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_1",chip_name="enp216s0f0np1"} 1
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_2",chip_name="enp216s0f0np2"} 1
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_3",chip_name="enp216s0f0np3"} 1
```

Passed through the following Regex:

```
/chip_name="(?<text>[ ^ " ] + ) |chip=" (?<value >[ ^ " ] + )/g
```

Would produce the following dropdown list:

```
Display Name          Value
------------          -------------------------
enp216s0f0np0         0000:d7:00_0_0000:d8:00_0
enp216s0f0np1         0000:d7:00_0_0000:d8:00_1
enp216s0f0np2         0000:d7:00_0_0000:d8:00_2
enp216s0f0np3         0000:d7:00_0_0000:d8:00_3
```

Only text and value capture group names are supported.

Inspect Variables

The variables page lets you easily identify whether a variable is being referenced (or used) in other variables or dashboard.

Any variable that is referenced or used has a green check mark next to it, while unreferenced variables have an orange caution icon next to them. In addition, all referenced variables have a

dependency icon next to the green check mark. You can select the icon to view the dependency map. The dependency map can be moved. You can zoom in or out with the mouse wheel or equivalent.

Variable syntax

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Panel titles and metric queries can refer to variables using two different syntaxes.

- `$varname` – This syntax is easy to read, but it does not allow you to use a variable in the middle of a word.

  Example: `apps.frontend.$server.requests.count`
- `${var_name}` – Use this syntax when you want to use a variable in the middle of an expression.
- `${var_name:<format>}` – This format gives you more control over how Grafana interprets values. For more information, see *Advanced variable format options*, following this list.
- `[[varname]]` – Do not use. This syntax is old and has been deprecated. It will be removed in a future release.

Before queries are sent to your data source the query is *interpolated*, meaning the variable is replaced with its current value. During interpolation, the variable value might be *escaped* in order to conform to the syntax of the query language and where it is used. For example, a variable used in a regex expression in an InfluxDB or Prometheus query will be regex escaped.

Advanced variable format options

The formatting of the variable interpolation depends on the data source, but there are some situations where you might want to change the default formatting.

For example, the default for the MySQL data source is to join multiple values as comma-separated with quotes: `'server01','server02'`. In some cases, you might want to have a comma-

separated string without quotes: `server01,server02`. You can make that happen with advanced variable formatting options listed below.

General syntax

Syntax: `${var_name:option}`

If any invalid formatting option is specified, then `glob` is the default/fallback option.

CSV

Formats variables with multiple values as a comma-separated string.

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:csv}'
Interpolation result:  'test1,test2'
```

Distributed - OpenTSDB

Formats variables with multiple values in custom format for OpenTSDB.

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:distributed}'
Interpolation result:  'test1,servers=test2'
```

Doublequote

Formats single- and multi-valued variables into a comma-separated string, escapes " in each value by \" and quotes each value with ".

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:doublequote}'
Interpolation result:  '"test1","test2"'
```

Glob - Graphite

Formats variables with multiple values into a glob (for Graphite queries).

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:glob}'
Interpolation result:  '{test1,test2}'
```

JSON

Formats variables with multiple values as a comma-separated string.

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:json}'
Interpolation result:  '["test1", "test2"]'
```

Lucene - Elasticsearch

Formats variables with multiple values in Lucene format for Elasticsearch.

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:lucene}'
Interpolation result:  '("test1" OR "test2")'
```

Percentencode

Formats single and multivalued variables for use in URL parameters.

```
servers = [ 'foo()bar BAZ',  'test2' ]
String to interpolate:  '${servers:percentencode}'
Interpolation result:  'foo%28%29bar%20BAZ%2Ctest2'
```

Pipe

Formats variables with multiple values into a pipe-separated string.

```
servers = [ 'test1.',  'test2' ]
String to interpolate:  '${servers:pipe}'
Interpolation result:  'test1.|test2'
```

Raw

Turns off data source-specific formatting, such as single quotes in an SQL query.

```
servers = [ 'test.1',  'test2' ]
String to interpolate:  '${var_name:raw}'
Interpolation result:  'test.1,test2'
```

Regex

Formats variables with multiple values into a regex string.

```
servers = [ 'test1.',  'test2' ]
String to interpolate:  '${servers:regex}'
Interpolation result:  '(test1\.|test2)'
```

Singlequote

Formats single- and multi-valued variables into a comma-separated string, escapes `'` in each value by `\'` and quotes each value with `'`.

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:singlequote}'
Interpolation result:  "'test1','test2'"
```

Sqlstring

Formats single- and multi-valued variables into a comma-separated string, escapes `'` in each value by `''` and quotes each value with `'`.

```
servers = [ "test'1",  "test2" ]
String to interpolate:  '${servers:sqlstring}'
Interpolation result:  "'test''1','test2'"
```

Text

Formats single- and multi-valued variables into their text representation. For a single variable, it will just return the text representation. For multi-valued variables, it will return the text representation combined with +.

```
servers = [ "test1",  "test2" ]
String to interpolate:  '${servers:text}'
Interpolation result:  "test1 + test2"
```

Query parameters

Formats single- and multi-valued variables into their query parameter representation. Example: `var-foo=value1&var-foo=value2`

```
servers = [ "test1",  "test2" ]
String to interpolate:  '${servers:queryparam}'
```

```
Interpolation result:  "var-servers=test1&var-servers=test2"
```

Assessing dashboard usage

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

For every dashboard and data source, you can access usage information.

Dashboard insights

To see dashboard usage information, select **Dashboard insights** in the top bar.

Dashboard insights show the following information.

- **Stats** – The number of daily queries and errors for the past 30 days.
- **Users & activity** – The daily view count for the last 30 days; last activities on the dashboard and recent users (with a limit of 20).

Data source insights

Data source insights provide information about how a data source has been used in the past 30 days, such as:

- Queries per day
- Errors per day
- Query load time per day (averaged in ms)

To find data source insights

1. Select **Connections** in the main navigation of your workspace.
2. Select **Data sources**.

3. Choose a data source.
4. Select the **Insights** tab.

Presence indicator

When you are signed in and look at a dashboard, you can know who is looking at the same dashboard as you are through a presence indicator, which displays avatars of users who have recently interacted with the dashboard. The default timeframe is 10 minutes. To see the user's name, hover over the user's avatar. The avatars come from Gravatar based on the user's email.

When there are more active users on a dashboard than can fit within the presence indicator, click the **+X** icon. Doing this will open dashboard insights, which contain more details about recent user activity.

Sorting dashboards by using insights data

In the search view, you can use insights data to help you find most-used, broken, and unused dashboards. You can sort dashboards by the following.

- Views
- Errors
- Views
- Created time
- Updated time

Troubleshoot dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use the following strategies to help you troubleshoot common dashboard problems.

Dashboard is slow

- Are you trying to render dozens (or hundreds or thousands) of time series on a graph? This can cause the browser to lag. Try using functions like `highestMax` (in Graphite) to reduce the number of returned series.
- Sometimes series names can be very large. This causes larger response sizes. Try using `alias` to reduce the size of the returned series names.
- Are you querying many time series or a long time range? Both of these conditions can cause Grafana or your data source to pull in a lot of data, which may slow the dashboard down. Try reducing one or both of these.
- There could be high load on your network infrastructure. If the slowness isn't consistent, this may be the problem.

Dashboard refresh rate issues

By default, Grafana queries your data source every 30 seconds. However, setting a low refresh rate on your dashboards puts unnecessary stress on the backend. In many cases, querying this frequently isn't necessary because the data source isn't sending data often enough for there to be changes every 30 seconds.

We recommend the following:

- Only enable auto-refreshing on dashboards, panels, or variables if necessary. Users can refresh their browser manually.
- If you require auto-refreshing, then set the refresh rate to a longer time period that makes sense, such as once a minute, every 10 minutes, or every hour.
- Check the time range of your dashboard. If your dashboard has a longer time range, such as a week, then you really don't need automated refreshing and you should disable it.

Handling or rendering null data is wrong or confusing

Some applications publish data intermittently; for example, they only post a metric when an event occurs. By default, Grafana graphs connect lines between the data points, but this can be deceptive.

Graphs that have the **Connect null values** option set to **Always**, will connect lines where there are missing values.

One way to fix this is to use bars instead of lines and have the **No value** option (under **Standard options**) set to `0`. In this case, the missing data will show up as areas of the graph with no data.

Searching Dashboards in Grafana version 12

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can search for dashboards by dashboard name and by panel title. When you search for dashboards, the system returns all dashboards available within the Grafana instance, even if you do not have permission to view the contents of the dashboard.

Search dashboards using dashboard name

Enter any part of the dashboard name in the search bar. The search returns results for any partial string match in real-time, as you type.

Dashboard search is:

- Real-time
- *Not* case sensitive
- Functional across stored and file based dashboards.

> **Tip**
> You can use your keyboard arrow keys to navigate the results and press `Enter` to open the selected dashboard.

Search dashboards using panel title

You can search for a dashboard by the title of a panel that appears in a dashboard. If a panel's title matches your search query, the dashboard appears in the search results.

Filter dashboard search results by tags

Tags are a great way to organize your dashboards, especially as the number of dashboards grow. You can add and manage tags in dashboard **Settings**.

When you select multiple tags, Grafana shows dashboards that include all selected tags.

To filter dashboard search result by a tag, complete one of the following steps:

- To filter dashboard search results by tag, choose a tag that appears in the right column of the search results.

  You can continue filtering by choosing additional tags.
- To see a list of all available tags, click the **Filter by tags** dropdown menu and select a tag.

  All tags will be shown, and when you select a tag, the dashboard search will be instantly filtered.

> **Tip**
>
> When using only a keyboard, press the `tab` key and navigate to the **Filter by tag** dropdown menu, press the down arrow key to activate the menu and locate a tag, and press `Enter` to select the tag.

Command palette

You can use the command palette to do the following:

- Search for and open dashboards and folders.
- Create dashboards and alert rules.
- Locate pages within Grafana.
- Change the theme to dark or light.

To open the command palette, enter `ctrl+k` (`cmd+k` in MacOS). You can also select the search input in the Grafana navigation bar.

Note

To go to the previous step, press `backspace` with the command palette empty.

Panels and visualizations in Grafana version 12

This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The *panel* is the basic visualization building block in Grafana. Each panel has a query editor specific to the data source selected in the panel. The query editor allows you to build a query that returns the data you want to visualize.

There are a wide variety of styling and formatting options for each panel. Panels can be dragged, dropped, and resized to rearrange them on the dashboard.

Before you add a panel, ensure that you have configured a data source.

Additional panel types might be available by installing additional plugins to your workspace.

For details about using specific data sources, see Connect to data sources.

You can add interactive actions to visualizations that trigger API calls or navigate to URLs, with support for custom variables and one-click data links.

Topics

- Panel editor overview
- The panel inspect view
- Query and transform data
- Configure panel options

- Configure standard options
- Configure a legend
- Configure data links
- Configure value mappings
- Configure thresholds
- Configure field overrides
- Visualizations available in Grafana version 12

Panel editor overview

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

In the panel editor, you can update all the elements of a visualization, including the data source, queries, time range, and display options.

To add a panel to a new dashboard, select **+ Add visualization** in the middle of the dashboard. To add a panel to an existing dashboard, select **Add** in the dashboard header and select **Visualization** in the drop down. You can also copy and paste an existing panel from the same or a different dashboard.

Panel menu

To access the panel editor, hover over the top-right corner of any panel. Select the panel menu icon that appears and select **Edit**.

The panel menu also gives you access to the following actions:

- **View** – View the panel in full screen.
- **Edit** – Open the panel editor to edit panel and visualization options.

- **Share** – Share the panel as a link, or library panel.
- **Explore** – Open the panel in **Explore**, where you can focus on your query.
- **Inspect** – Open the **Inspect** drawer, where you can review the panel data, stats, metadata, JSON, and query.
  - **Data** – Open the **Inspect** drawer in the **Data** tab.
  - **Query** – Open the **Inspect** drawer in the **Query** tab.
  - **Panel JSON** – Open the **Inspect** drawer in the **JSON** tab.
- **Extensions** – Access other actions provided by installed applications, such as declaring an incident. This option only appears if you have app plugins installed which contribute an extension to the panel menu.
- **More** – Access other panel actions.
  - **Duplicate** – Make a copy of the panel. Duplicated panels query data separately from the original panel. If you want to use the same query results, you can use the `Dashboard` data source in the second panel.
  - **Copy** – Copy the panel to the clipboard.
  - **Create library panel** – Create a panel that can be imported into other dashboards.
  - **Create alert** – Open the alert rule configuration page in **Alerting**, where you can create a Grafana-managed alert based on the panel queries.
  - **Hide legend** – Hide the panel legend.
  - **Get help** – Send a snapshot or panel data to Grafana Labs Technical Support.
- **Remove** – Remove the panel from the dashboard.

Panel editor

This section describes the areas of the Grafana panel editor.

- Panel header – The header section lists the dashboard in which the panel appears and the following controls:
  - **Discard** – Discards changes you have made to the panel since you last saved the dashboard.
  - **Save** – Saves changes you made to the panel.
  - **Apply** – Applies changes you made and closes the panel editor, returning you to the dashboard. You will have to save the dashboard to persist the applied changes.
- Visualization preview – The visualization preview section contains the following options:

  - **Table view** – Convert any visualization to a table so you can see the data. Table views are helpful for troubleshooting. This view only contains the raw data. It does not include transformations you might have applied to the data or the formatting options available in the Table visualization.
  - **Fill** – The visualization preview fills the available space. If you change the width of the side pane or height of the bottom pane the visualization changes to fill the available space.
  - **Actual** – The visualization preview will have the exact size as the size on the dashboard. If not enough space is available, the visualization will scale down preserving the aspect ratio.
  - **Time range controls** – **Default** is either the browser local timezone or the timezone selected at a higher level.
- Data section – The data section contains tabs where you enter queries, transform your data, and create alert rules (if applicable).
  - **Query tab** – Select your data source and enter queries here. For more information, see Query and transform data. When you initially create a dashboard, you are prompted to select a data source. You can update the data source or the query in this tab.
  - **Transform tab** – Apply data transformations. For more information, see Query and transform data.
  - **Alert tab** – Write alert rules. For more information, see Alerts in Grafana version 12.
- Panel display options – The display options section contains tabs where you configure almost every aspect of your data visualization. The details vary based on the visualization type selected.

Panel inspect drawer

The inspect drawer helps you understand and troubleshoot your panels. You can view the raw data for any panel, export that data to a comma-separated values (CSV) file, view query requests, and export panel and data JSON.

> **Note**
>
> Not all panel types include all tabs. For example, dashboard list panels do not have raw data to inspect, so they do not display the Stats, Data, or Query tabs.

The panel inspector consists of the following options:

- The panel inspect drawer displays as a drawer on the right side. Select the arrow in the upper right corner to expand or reduce the drawer pane.
- **Data tab** – Shows the raw data returned by the query with transformations applied. Field options such as overrides and value mappings are not applied by default.
- **Stats tab** – Shows how long your query takes and how much it returns.
- **JSON tab** – Allows you to view and copy the panel JSON, panel data JSON, and data frame structure JSON. This is useful if you are provisioning or administering Grafana.
- **Query tab** – Shows you the requests to the server sent when Grafana queries the data source.
- **Error tab** – Shows any errors returned by a query. The tab is only visible when a query returns an error.

The panel inspect view

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The panel inspect view, which you can open via the panel menu, helps you understand and troubleshoot your panels. You can inspect the raw data for any Amazon Managed Grafana panel, export that data to a comma-separated values (CSV) file, view query requests, and export panel and data JSON.

Note

Not all panel types include all tabs. For example, dashboard list panels do not have raw data to inspect, so they do not display the Stats, Data, or Query tabs.

The panel inspector consists of the following options:

1. The panel inspector displays **Inspect:** at the top of the pane. Select the arrow in the upper right corner to expand or reduce the pane.

2. **Data tab** – Shows the raw data returned by the query with transformations applied. Field options such as overrides and value mappings are not applied by default.
3. **Stats tab** – Shows how long your query takes and how much it returns.
4. **JSON tab** – Allows you to view and copy the panel JSON, panel data JSON, and data frame structure JSON. This is useful if you are provisioning or administering Grafana.
5. **Query tab** – Shows you the requests sent to the server when Grafana queries the data source.
6. **Error tab** – Shows any errors. Only visible when a query returns an error.

Downloading raw query results

Amazon Managed Grafana generates a CSV file that contains your data, including any transformations to that data. You can choose to view the data before or after the panel applies field options or field option overrides.

To download raw query results

1. Edit the panel that contains the query data you want to download.
2. In the query editor, select **Query Inspector**.
3. Select **Data**.

   If your panel contains multiple queries or queries multiple nodes, then you have additional options.

   - **Select result** – Choose which result set data you want to view.
   - **Transform data**
   - **Join by time** – View raw data from all your queries at once, one result set per column. Select a column heading to reorder the data.
4. To see data before the system applies field overrides, select the **Formatted data** toggle.
5. To download a CSV file specifically formatted for Excel, select the **Download for Excel** toggle.
6. Select **Download CSV**.

Inspecting query performance

The **Stats** tab displays statistics that tell you how long your query takes, how many queries you send, and the number of rows returned. This information can help you troubleshoot your queries, especially if any of the numbers are unexpectedly high or low.

To inspect query performance

1. Edit the panel that contains the query with performance you want to inspect.
2. In the query editor, select **Query Inspector**.
3. Select **Stats**.

Statistics are displayed in read-only format.

Inspecting query request and response

You can inspect query request and response data when you want to troubleshoot a query that returns unexpected results, or fails to return expected results.

1. Edit the panel that contains the query you want to export.
2. In the query editor, select **Query Inspector**.
3. Select **Refresh**.

   The panel populates with response data.
4. Make adjustments, as necessary and re-run the query.
5. To download the query request and response data, click the **Copy to clipboard** icon and paste the results into another application.

Query and transform data

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Amazon Managed Grafana supports many types of data sources. Data source *queries* return data that Grafana can *transform* and visualize. Each data source uses its own query language, and data source plugins each implement a query-building user interface called a query editor.

Topics

- About queries
- Navigate the query tab
- Adding a query
- Manage queries
- Query options
- Write expression queries
- Share query results with another panel
- Transform data
- Troubleshoot queries
- Calculation types

About queries

Grafana panels communicate with data sources via queries, which retrieve data for the visualization. A query is a question written in the query language used by the data source.

You can configure query frequency and data collection limits in the panel's data source options. Grafana supports up to 26 queries per panel.

You can find more information about each data source's query language in the Data sources section.

Query editors

Each data source's *query editor* provides a customized user interface that helps you write queries that take advantage of its unique capabilities.

Because of the differences between query languages, each data source query editor looks and functions differently. Depending on your data source, the query editor might provide auto-completion features, metric names, variable suggestions, or a visual query-building interface.

For details on a specific data source's unique query editor features, including information about queries and syntax, see the data source documentation:

- For data sources included with Amazon Managed Grafana, see Built-in data sources.
- For data sources included with Grafana Enterprise, see Connect to Enterprise data sources.

- For other data source plugins that you install via the Find plugins with the plugin catalog, the documentation is linked within the listing in the plugin catalog.

Query syntax

Data sources use different query languages to request data. For details on a specific data source's unique query language, refer to its documentation.

PostgreSQL example:

```
SELECT hostname FROM host WHERE region IN($region)
```

PromQL example:

```
query_result(max_over_time(<metric>[${__range_s}s]) != <state>)
```

Special data sources

Grafana also includes three special data sources: **Grafana**, **Mixed**, and **Dashboard**. For more information, see Connect to data sources.

Navigate the query tab

A panel's **Query** tab consists of the following elements:

- **Data source selector** – Selects the data source to query.
- **Query options** – Sets maximum data retrieval parameters and query run time intervals.
- **Query inspector button** – Opens the query inspector panel, where you can view and optimize your query.
- **Query editor list** – Lists the queries you've written.
- **Expressions** – Uses the expression builder to create alert expressions. For more information about expressions, see Write expression queries.

Adding a query

A query returns data that Grafana visualizes in dashboard panels. When you create a panel, Grafana automatically selects the default data source.

To add a query

1. Edit the panel to which you're adding a query.
2. Choose the **Query** tab.
3. Choose the **Data source** dropdown menu and select a data source.
4. Choose **Query options** to configure the maximum number of data points you need. For more information about query options, see Query options.
5. Write the query using the query editor.
6. Choose **Apply**.

Grafana queries the data source and visualizes the data.

Manage queries

Grafana organizes queries in collapsible query rows. Each query row contains a query editor and is identified with a letter (A, B, C, and so on).

To manage your queries, you can copy queries, hide queries, remove queries, reorder queries, and toggle help for the query editor.

Query options

Choose **Query options** next to the data source selector to see settings for the selected data source. Changes you make here affect only queries made in this panel.

Grafana sets defaults that are shown in dark gray text. Changes are displayed in white text. To return a field to the default setting, delete the white text from the field.

Panel data source query options include:

- **Max data points** – If the data source supports it, this sets the maximum number of data points for each series returned. If the query returns more data points than the max data points setting, then the data source reduces the number of points returned by aggregating them together by average, max, or another function.

  You can limit the number of points to improve query performance or smooth the visualized line. The default value is the width (or number of pixels) of the graph, because you can only visualize as many data points as the graph panel has room to display.

With streaming data, Grafana uses the max data points value for the rolling buffer. Streaming is a continuous flow of data, and buffering divides the stream into chunks.

- **Min interval** – Sets a minimum limit for the automatically calculated interval, which is typically the minimum scrape interval. If a data point is saved every 15 seconds, you don't benefit from having an interval lower than that. You can also set this to a higher minimum than the scrape interval to retrieve queries that are more coarse-grained and well-functioning.
- **Interval** – Sets a time span that you can use when aggregating or grouping data points by time.

  Grafana automatically calculates an appropriate interval that you can use as a variable in templated queries. The variable is measured in either seconds (`$__interval`) or milliseconds (`$__interval_ms`).

  Intervals are typically used in aggregation functions like sum or average.
  For example, this is a Prometheus query that uses the interval variable:
  `rate(http_requests_total[$__interval])`.

  This automatic interval is calculated based on the width of the graph. As the user zooms out on a visualization, the interval grows, resulting in a more coarse-grained aggregation. Likewise, if the user zooms in, the interval decreases, resulting in a more fine-grained aggregation.

  For more information, see Global variables.
- **Relative time** – Overrides the relative time range for individual panels, which causes them to be different than what is selected in the dashboard time picker in the top-right corner of the dashboard. You can use this to show metrics from different time periods or days on the same dashboard.

  > **Note**
  >
  > Panel time overrides have no effect when the dashboard's time range is absolute.

Example	Relative time field
Last 5 minutes	`now-5m`
The day so far	`now/d`

Example	Relative time field
Last 5 days	`now-5d/d`
This week so far	`now/w`
Last 2 years	`now-2y/y`

- **Time shift** – Overrides the time range for individual panels by shifting its start and end relative to the time picker. For example, you can shift the time range for the panel to be two hours earlier than the dashboard time picker.

> **Note**
>
> Panel time overrides have no effect when the dashboard's time range is absolute.

Example	Time shift field
Last entire week	`1w/w`
Two entire weeks ago	`2w/w`
Last entire month	`1M/M`
This entire year	`1d/y`
Last entire year	`1y/y`

- **Cache timeout** – *(Visible only if available in the data source)* Overrides the default cache timeout if your time series store has a query cache. Specify this value as a numeric value in seconds.

Write expression queries

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Server-side expressions enable you to manipulate data returned from queries with math and other operations. Expressions create new data and do not manipulate the data returned by data sources.

About expressions

Server-side expressions allow you to manipulate data returned from queries with math and other operations. Expressions create new data and do not manipulate the data returned by data sources, aside from some minor data restructuring to make the data acceptable input for expressions.

Using expressions

Expressions are most commonly used by Grafana alerting. The processing is done server-side, so expressions can operate without a browser session. However, expressions can also be used with backend data sources and visualization.

Note

Expressions do not work with legacy dashboard alerts.

Expressions are meant to augment data sources by enabling queries from different data sources to be combined or by providing operations unavailable in a data source.

Note

When possible, you should do data processing inside the data source. Copying data from storage to the Grafana server for processing is inefficient, so expressions are targeted at lightweight data processing.

Expressions work with data source queries that return time series or numeric data. They also operate on multiple-dimensional data. For example, a query that returns multiple series, where each series is identified by labels or tags.

An individual expression takes one or more queries or other expressions as input and adds data to the result. Each individual expression or query is represented by a variable that is a named identifier known as its RefID (e.g., the default letter `A` or `B`).

To reference the output of an individual expression or a data source query in another expression, this identifier is used as a variable.

Types of expressions

Expressions work with two types of data.

- A collection of time series.
- A collection of numbers, where each number is an item.

Each collection is returned from a single data source query or expression and represented by the RefID. Each collection is a set, where each item in the set is uniquely identified by its dimensions which are stored as labels or key-value pairs.

Data source queries

Server-side expressions only support data source queries for backend data sources. The data is generally assumed to be labeled time series data.

Data source queries, when used with expressions, are run by the expression engine. When it does this, it restructures data to be either one time series or one number per data frame. So for example if using a data source that returns multiple series on one frame in the table view, you might notice it looks different when run with expressions.

Currently, the only non-time series format (number) is supported when using data frames are you have a table response that returns a data frame with no time, string columns, and one number column:

The following example table produces a number that works with expressions. The string columns become labels and the number column the corresponding value. For example `{"Loc": "MIA", "Host": "A"}` with a value of 1.

Loc	Host	Avg_CPU
MIA	A	1

Loc	Host	Avg_CPU
NYC	B	2

Operations

You can use the following operations in expressions: math, reduce, and resample.

Math

Math is for free-form math formulas on time series or number data. Math operations take numbers and time series as input and changes them to different numbers and time series.

Data from other queries or expressions are referenced with the RefID prefixed with a dollar sign, for example `$A`. If the variable has spaces in the name, then you can use a brace syntax like `${my variable}`.

Numeric constants can be in decimal (`2.24`), octal (with a leading zero like `072`), or hex (with a leading 0x like `0x2A`). Exponentials and signs are also supported (for example, `-0.8e-2`).

Operators

The arithmetic (+, binary and unary `-`, `*`, `/`, `%`, exponent `**`), relational (<, >, ==, `!=`, >=, <=), and logical (&&, `||`, and unary `!`) operators are supported.

How the operation behaves with data depends on if it is a number or time series data.

With binary operations, such as `$A + $B` or `$A || $B`, the operator is applied in the following ways depending on the type of data:

- If both `$A` and `$B` are a number, then the operation is performed between the two numbers.
- If one variable is a number, and the other variable is a time series, then the operation between the value of each point in the time series and the number is performed.
- If both `$A` and `$B` are time series data, then the operation between each value in the two series is performed for each time stamp that exists in both `$A` and `$B`. The `Resample` operation can be used to line up time stamps.

Summary:

- Number <Operation> number = number
- Number <Operation> series = series
- Series <Operation> series = series

Because expressions work with multiple series or numbers represented by a single variable, binary operations also perform a union (join) between the two variables. This is done based on the identifying labels associated with each individual series or number.

So if you have numbers with labels like `{host=web01}` in `$A` and another number in `$B` with the same labels then the operation is performed between those two items within each variable, and the result will share the same labels. The rules for the behavior of this union are as follows:

- An item with no labels will join to anything.
- If both `$A` and `$B` each contain only one item (one series, or one number), they will join.
- If labels are exact match they will join.
- If labels are a subset of the other, for example an item in `$A` is labeled `{host=A,dc=MIA}` and an item in `$B` is labeled `{host=A}` they will join.
- If within a variable such as `$A` there are different tag keys for each item, the join behavior is undefined.

The relational and logical operators return 0 for false 1 for true.

Math Functions

While most functions exist in the own expression operations, the math operation does have some functions that similar to math operators or symbols. When functions can take either numbers or series, than the same type as the argument will be returned. When it is a series, the operation of performed for the value of each point in the series.

abs

abs returns the absolute value of its argument which can be a number or a series. For example `abs(-1)` or `abs($A)`.

is_inf

is_inf takes a number or a series and returns `1` for `Inf` values (negative or positive) and `0` for other values. For example `is_inf($A)`.

> **Note**
>
> If you need to specifically check for negative infinity for example, you can do a comparison like `$A == infn()`.

is_nan

is_nan takes a number or a series and returns `1` for `NaN` values and `0` for other values. For example `is_nan($A)`. This function is required for this check because `NaN` is not equal to `NaN`.

is_null

is_null takes a number or a series and returns `1` for `null` values and `0` for other values. For example `is_null($A)`.

is_number

is_number takes a number or a series and returns `1` for all real number values and `0` for other values (which are `null`, `Inf+`, `Inf-`, and `NaN`). For example `is_number($A)`.

log

Log returns the natural logarithm of its argument which can be a number or a series. If the value is less than 0, `NaN` is returned. For example `log(-1)` or `log($A)`.

inf, infn, nan, and null

The inf, infn, nan, and null functions all return a single value of the name. They primarily exist for testing. Example: `null()`.

round

Round returns a rounded integer value. For example, `round(3.123)` or `round($A)`.

ceil

Ceil rounds the number up to the nearest integer value. For example, `ceil(3.123)` returns `4`.

floor

Floor rounds the number down to the nearest integer value. For example, `floor(3.123)` returns 3.

Reduce

Reduce takes one or more time series returned from a query or an expression and turns each series into a single number. The labels of the time series are kept as labels on each outputted reduced number.

Fields:

- **Function** – The reduction function to use
- **Input** – The variable (refID (such as A)) to resample
- **Mode** – Allows control behavior of reduction function when a series contains non-numerical values (null, NaN, +-Inf)

Reduction Functions

Count

Count returns the number of points in each series.

Mean

Mean returns the total of all values in each series divided by the number of points in that series. In `strict` mode if any values in the series are null or nan, or if the series is empty, NaN is returned.

Min and Max

Min and Max return the smallest or largest value in the series respectively. In `strict` mode if any values in the series are null or nan, or if the series is empty, NaN is returned.

Sum

Sum returns the total of all values in the series. If series is of zero length, the sum will be 0. In `strict` mode if there are any NaN or Null values in the series, NaN is returned.

Last

Last returns the last number in the series. If the series has no values then returns NaN.

Reduction Modes

Strict

In Strict mode the input series is processed as is. If any values in the series are non-numeric (null, NaN or +-Inf), NaN is returned.

Drop Non-Numeric

In this mode all non-numeric values (null, NaN or +-Inf) in the input series are filtered out before running the reduction function.

Replace Non-Numeric

In this mode all non-numeric values are replaced by a pre-defined value.

Resample

Resample changes the time stamps in each time series to have a consistent time interval. The main use case is so you can resample time series that do not share the same timestamps so math can be performed between them. This can be done by resample each of the two series, and then in a Math operation referencing the resampled variables.

Fields:

- **Input** – The variable of time series data (refID (such as `A`)) to resample
- **Resample** to – The duration of time to resample to, for example `10s`. Units can be `s` for seconds, `m` for minutes, `h` for hours, `d` for days, `w` for weeks, and `y` for years.
- **Downsample** – The reduction function to use when there are more than one data point per window sample. See the reduction operation for behavior details.
- **Upsample** – The method to use to fill a window sample that has no data points.
  - **pad** fills with the last knowm value
  - **backfill** with next known value
  - **fillna** to fill empty sample windows with NaNs

Write an expression

If your data source supports them, then Grafana displays the **Expression** button and shows any existing expressions in the query editor list.

To write an expression

1. Open the panel.
2. Below the query, choose **Expression**.
3. In the **Operation** field, select the type of expression you want to write.
4. Write the expression.
5. Choose **Apply**.

Special cases

When any queried data source returns no series or numbers, the expression engine returns `NoData`. For example, if a request contains two data source queries that are merged by an expression, if `NoData` is returned by at least one of the data source queries, then the returned result for the entire query is `NoData`. For more information about how Grafana Alerting processes `NoData` results, see Configure Grafana managed alert rules.

In the case of using an expression on multiple queries, the expression engine requires that all of the queries return an identical timestamp. For example, if using math to combine the result of multiple SQL queries which each use `SELECT NOW() AS "time"`, the expression will only work if all queries evaluate `NOW()` to an identical timestamp, which does not always happen. To resolve this, you can replace `NOW()` with an arbitrary time, such as `SELECT 1 AS "time"`, or any other valid UNIX timestamp.

Share query results with another panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana lets you use the query result from one panel for any other panel in the dashboard. Sharing query results across panels reduces the number of queries made to your data source, which can improve the performance of your dashboard.

The *Dashboard* data source lets you select a panel in your dashboard that contains the queries you want to share the results for. Instead of sending a separate query for each panel, Grafana sends one query and other panels use the query results to construct visualizations.

This strategy can drastically reduce the number of queries being made when you for example have several panels visualizing the same data.

To share query results

1. Create a dashboard.
2. Change the title to `Source panel`. You'll use this panel as a source for the other panels.
3. Define the query or queries that you want to share.

   If you don't have a data source available, use the **Grafana** data source, which returns a random time series that you can use for testing.
4. Add a second panel and select the **Dashboard** data source in the query editor.
5. In the **Use results from panel list**, select the first panel you created.

All queries defined in the source panel are now available to the new panel. Queries made in the source panel can be shared with multiple panels.

You can click on any of the queries to go to the panel where they are defined.

Transform data

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Transformations are a powerful way to manipulate data returned by a query before the system applies a visualization. Using transformations, you can:

- Rename fields

- Join time series data
- Perform mathematical operations across queries
- Use the output of one transformation as the input to another transformation

For users that rely on multiple views of the same dataset, transformations offer an efficient method of creating and maintaining numerous dashboards.

You can also use the output of one transformation as the input to another transformation, which results in a performance gain.

> **Note**
>
> Sometimes the system cannot graph transformed data. When that happens, click the **Table view** toggle above the visualization to switch to a table view of the data. This can help you understand the final result of your transformations.

Transformation types

Grafana provides a number of ways that you can transform data. There is a complete list of transformation functions below.

Order of transformations

When there are multiple transformations, Grafana applies them in the order they are listed. Each transformation creates a result set that then passes on to the next transformation in the processing pipeline.

The order in which Grafana applies transformations directly impacts the results. For example, if you use a Reduce transformation to condense all the results of one column into a single value, then you can only apply transformations to that single value.

Add a transformation function to data

The following steps guide you in adding a transformation to data. This documentation does not include steps for each type of transformation.

To add a transformation to a panel

1. Navigate to the panel where you want to add one or more transformations.

2. Hover over any part of the panel to display the actions menu on the top right corner.
3. From the actions menu choose **Edit**.
4. Select the **Transform** tab.
5. Select a transformation. A transformation row appears where you configure the transformation options.
6. To apply another transformation, Choose **Add transformation**. This transformation acts on the result set returned by the previous transformation.

Debug a transformation

To see the input and the output result sets of the transformation, choose the debug (bug) icon on the right side of the transformation row. This will display the input data, and the result of the transformation as output.

The input and output results sets can help you debug a transformation.

Disable a transformation

You can disable or hide a transformation by choosing the show (eye) icon on the top right of the transformation row. This disables the applied actions of that specific transformation and can help to identify issues when you change several transformations one after another.

Filter a transformation

If your transformation uses more than one query, you can filter these, and apply the selected transformation to only one of the queries. To do this, choose the filter icon on the top right of the transformation row. This opens a dropdown with a list of queries used on the panel. From here, you can select the query you want to transform.

You can also filter by annotations (which includes exemplars) to apply transformations to them. When you do so, the list of fields changes to reflect those in the annotation or exemplar tooltip.

The filter icon is always displayed if your panel has more than one query or source of data (that is panel or annotation data), but it may not work if previous transformations for merging the queries' outputs are applied. This is because one transformation takes the output of the previous one.

Delete a transformation

We recommend that you remove transformations that you don’t need. When you delete a transformation, you remove the data from the visualization.

Prerequisites:

Identify all dashboards that rely on the transformation and inform impacted dashboard users.

To delete a transformation

1. Open a panel for editing.
2. Select the **Transform** tab.
3. Choose the trash icon next to the transformation you want to delete.

Transformation functions

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can perform the following transformations on your data.

Add field from calculation

Use this transformation to add a new field calculated from two other fields. Each transformation allows you to add one new field.

- **Mode** - Select a mode:
  - **Reduce row** – Apply selected calculation on each row of selected fields independently.
  - **Binary operation** – Apply basic binary operations (for example, sum or multiply) on values in a single row from two selected fields.
  - **Unary operation** – Apply basic unary operations on values in a single row from a selected field. The available operations are:
    - **Absolute value (abs)** – Returns the absolute value of a given expression. It represents the distance from zero as a positive number.
    - **Natural exponential (exp)** – Returns *e* raised to the power of a given expression.
    - **Natural logarithm (ln)** – Returns the natural logarithm of a given expression.

    - **Floor (floor)** – Returns the largest integer less than or equal to a given expression.
    - **Ceiling (ceil)** – returns the smallest integer greater than or equal to a given expression.
  - **Cumulative functions** – Apply functions on teh current row and all preceding rows.
    - **Total** – Calculates the cumulative total up to and including the current row.
    - **Mean** – Calculates the mean up to and including the current row.
  - **Window functions** – Apply window functions. The window can either be *trailing* or *centered*. With a trailing window the current row will be the last row in the window. With a centered window, the window will be centered on the current row. For even windows sizes, the window will be centered between the current row and the previous row.
    - **Mean** – Calculates the moving mean or running average.
    - – Calculates the moving standard deviation.
    - **Variance** – Calculate the moving variance.
  - **Row index** – Insert a field with the row index.
- **Field name** – Select the names of fields you want to use in the calculation for the new field.
- **Calculation** – If you select **Reduce row** mode, then the **Calculation** field appears. Select the field to see a list of calculation choices you can use to create the new field. For information about available calculations, refer to Calculation types.
- **Operation** – If you select **Binary operation** or **Unary operation** mode, then the **Operation** fields appear. These fields allow you to do basic math operations on values in a single row from two selected fields. You can also use numerical values for binary operations.
- **As percentile** – If you select **Row index** mode, then the **As percentile** switch appears. This switch allows you to transform the row index as a percentage of the total number of rows.
- **Alias** – (Optional) Enter the name of your new field. If you leave this blank, then the field will be named to match the calculation.
- **Replace all fields** – (Optional) Select this option if you want to hide all other fields and display only your calculated field in the visualization.

> **Note**
>
> **Cumulative functions** and **Window functions** are current in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

Concatenate fields

Concatenate fields

Use this transformation to combines all fields from all frames into one result.

For example, if you have separate queries retrieving temperature and uptime data (query A) and air quality index and error information (query b), applying the concatenate transformation yields a consolidated data frame with all relevant information in one view.

Consider these two following.

Query A:

Temp	Uptime
15.4	1230233

Query B:

AQI	Errors
3.2	5

After you concatenate the fields, the data frame would be:

Temp	Uptime	AQI	Errors
15.4	1230233	3.2	5

This transformation simplifies the process of merging data from different sources, providing a comprehensive view for analysis and visualization.

Config from query results

Config from query results

Use this transformation to select one query and extract standard options like **Min**, **Max**, **Unit** and **Thresholds** and apply them to other query results. This enables dynamic visualization configuration, based on the data returned by a specific query.

Options

- **Config query** – Select the query that returns the data you want to use as configuration.
- **Apply to** – Select the fields or series to which the configuration should be applied.
- **Apply to options** – Specify a field type or use a field name regex, depending on your selection in **Apply to**.

Field mapping table

Below the configuration options, you'll find the field mapping table. This table lists all fields found in the data returned by the config query, along with the **Use as** and **Select** options. It provides control over mapping fields to config properties, and for multiple rows, it allows you to choose which value to select.

The following example shows an input query and a query used as field configuration.

Input query

Time	Value
1626178119127	10
1626178119129	30

Config query

Time	Value
1626178119127	100
1626178119129	100

Output query (same as input, but now with config on the value field)

Time	Value (config: Max=100)
1626178119127	10
1626178119129	30

Each field now has a maximum configuration option set. Options such as **Min**, **Max**, **Unit**, and **Thresholds** are part of teh field configuration. If set, they are used by the visualization instead of any options manually configured in the panel editor options pane.

Value mappings

You can also transform a query result into value mappings. With this option, every row in the configuration query result defines a single value mapping row. See the following example.

Config query result

Value	Text	Color
L	Low	blue
M	Medium	green
H	High	red

In the field mapping, specify:

Field	Use as	Select
Value	Value mappings / Value	All values
Text	Value mappings / Text	All values
Color	Value mappings / Color	All values

Grafana builds value mappings from your query result and applies them to the real data query results. You should see values being mapped and colored according to the config query results.

Convert field type

Use this transformation to modify the field type of the specified field.

This transformation has the following options:

- **Field** – Select from available fields.
- **as** – Select the FieldType to convert to.
  - **Numeric** – attempts to make the values numbers.
  - **String** – will make the values strings.
  - **Time** – attempts to parse the values as time.
    - Will show an option to specify a DateFormat as input by a string, like `yyyy-mm-dd` or `DD MM YYYY hh:mm:ss`.
  - **Boolean** – will make the values Boolean.
  - **Enum** – will make the values enums.
    - Will show a table to manage the enums.
  - **Other** – attempts to parse the values as json.

For example consider the following query that could be modified by selecting the time field, as **Time**, and Date Format as YYYY.

Time	Mark	Value
2017-07-01	above	25
2018-08-02	below	22
2019-09-02	below	29
2020-10-04	above	22

The result:

Time	Mark	Value
2017-01-01 00:00:00	above	25

Time	Mark	Value
2018-01-01 00:00:00	below	22
2019-01-01 00:00:00	below	29
2020-01-01 00:00:00	above	22

This transformation allows you to flexibly adapt your data types, ensuring compatibility and consistency in your visualizations.

Extract Fields

Use this transformation to select a source of data and extract content from it in different formats. This transformation has the following fields:

- **Source** – Select the field for the source of data.
- **Format** – Choose one of the following:
  - **JSON** – Parse JSON content from the source.
  - **Key+value pairs** – Parse content in the format a=b or c : d from the source.
  - **Auto** – Discover fields automatically.
  - **Regex** – Extract fields using regular expressions for more powerful field extraction from text data.
- **Replace All Fields** – (Optional) Select this option to hide all other fields and display only your calculated field in the visualization.
- **Keep Time** - (Optional) Available only if **Replace All Fields** is true. Keeps the time field in the output.

Consider the following dataset:

Dataset Example

Timestamp	json_data
1636678740000000000	{"value": 1}

Timestamp	json_data
1636678680000000000	{"value": 5}
1636678620000000000	{"value": 12}

You could prepare the data to be used by a Time series panel with this configuration:

- Source: json_data
- Format: JSON
  - Field: value
  - Alias: my_value
- Replace all fields: true
- Keep time: true

This will generate the following output:

Transformed Data

Timestamp	my_value
1636678740000000000	1
1636678680000000000	5
1636678620000000000	12

With this transformation, you can extract and format data in various ways. You can customize the extraction format based on your specific data needs.

Lookup fields from resource

Use this transformation to enrich a field value by looking up additional fields from an external source.

This transformation has the following fields:

- **Field** – Select a text field from your dataset.
- **Lookup** – Choose from **Countries**, **USA States**, and **Airports**.

> **Note**
>
> This transformation only supports spatial data.

For example, if you have this data:

Dataset Example

Location	Values
AL	0
AK	10
Arizona	5
Arkansas	1
Somewhere	5

With this configuration:

- Field: location
- Lookup: USA States

You'll get the following output:

Transformed Data

Location	ID	Name	Lng	Lat	Values
AL	AL	Alabama	-80.891064	12.448457	0

Location	ID	Name	Lng	Lat	Values
AK	AK	Arkansas	-100.891064	24.448457	10
Arizona					5
Arkansas					1
Somewhere					5

This transformation lets you augment your data by fetching additional information from external sources, providing a more comprehensive dataset for analysis and visualization.

Filter data by query refId

Use this transformation to hide one or more queries in panels that have multiple queries.

Grafana displays the query identification letters in dark gray text. Choose a query identifier to toggle filtering. If the query letter is white, then the results are displayed. If the query letter is dark, then the results are hidden.

> **Note**
>
> This transformation is not available for Graphite because this data source does not support correlating returned data with queries.

Filter data by values

Use this transformation to selectively filter data points directly within your visualization. This transformation provides options to include or exclude data based on one or more conditions applied to a selected field.

This transformation is very useful if your data source does not natively filter by values. You might also use this to narrow values to display if you are using a shared query.

The available conditions for all fields are:

- **Regex** – Match a regex expression.
- **Is Null** – Match if the value is null.

- **Is Not Null** – Match if the value is not null.
- **Equal** – Match if the value is equal to the specified value.
- **Different** – Match if the value is different than the specified value.
- **Substring** – Match if the value contains the specified text.

Additional available conditions for number fields are:

- **Greater** – Match if the value is greater than the specified value.
- **Lower** – Match if the value is lower than the specified value.
- **Greater or equal** – Match if the value is greater or equal.
- **Lower or equal** – Match if the value is lower or equal.
- **Range** – Match a range between a specified minimum and maximum, min and max included.

Consider the following dataset:

Time	Temperature	Altitude
2020-07-07 11:34:23	32	101
2020-07-07 11:34:22	28	125
2020-07-07 11:34:21	26	110
2020-07-07 11:34:20	23	98
2020-07-07 10:32:24	31	95
2020-07-07 10:31:22	20	85
2020-07-07 09:30:57	19	101

If you **Include** the data points that have a temperature below 30°C, the configuration will look as follows:

- Filter Type: 'Include'
- Condition: Rows where 'Temperature' matches 'Lower Than' '30'

And you will get the following result, where only the temperatures below 30°C are included:

Transformed Data

Time	Temperature	Altitude
2020-07-07 11:34:22	28	125
2020-07-07 11:34:21	26	110
2020-07-07 11:34:20	23	98
2020-07-07 10:31:22	20	85
2020-07-07 09:30:57	19	101

You can add more than one condition to the filter. For example, you might want to include the data only if the altitude is greater than 100. To do so, add that condition to the following configuration:

- Filter type: 'Include' rows that 'Match All' conditions
- Condition 1: Rows where 'Temperature' matches 'Lower' than '30'
- Condition 2: Rows where 'Altitude' matches 'Greater' than '100'

When you have more than one condition, you can choose if you want the action (include/exclude) to be applied on rows that **Match all** conditions or **Match any** of the conditions you added.

In the example above, we chose **Match all** because we wanted to include the rows that have a temperature lower than 30°C *AND* an altitude higher than 100. If we wanted to include the rows that have a temperature lower than 30°C *OR* an altitude higher than 100 instead, then we would select **Match any**. This would include the first row in the original data, which has a temperature of 32°C (does not match the first condition) but an altitude of 101 (which matches the second condition), so it is included.

Conditions that are invalid or incompletely configured are ignored.

This versatile data filtering transformation lets you to selectively include or exclude data points based on specific conditions. Customize the criteria to tailor your data presentation to meet your unique analytical needs.

Filter fields by name

Use this transformation to remove parts of your query results. There are three ways to filter field names:

- Enter a regular expression.
- Manually select included fields.
- Use a dashboard variable.

Use a regular expression

When you filter using a regular expression, field names that match the regular expression are included. For example, using the regular expression `'prod.*'` would return only the fields that start with `prod`

The regular expression can include an interpolated dashboard variable using the `${variableName}` syntax.

Manually select included fields

Select or deselect field names to remove them from the result. If a regular expression is also included, fields that are matched by the expression are included, even if they're unchecked.

Use a dashboard variable

Select **From variable** to let you select a dashboard variable that's used to include fields. By setting up a dashboard variable with multiple choices, the same fields can be displayed across multiple visualizations.

This transformation provides flexibility in tailoring your query results to focus on the specific fields you need for effective analysis and visualization.

Format string

Use this transformation to customize the output of a string field. This transformation has the following fields:

- **Upper case** – Formats the entire string in uppercase characters.
- **Lower case** – Formats the entire string in lowercase characters.
- **Sentence case** – Formats the first character of the string in uppercase.

- **Title case** – Formats the first character of each word in the string in uppercase.
- **Pascal case** – Formats the first character of each word in the string in uppercase and doesn't include spaces between words.
- **Camel case** – Formats the first character of each word in the string in uppercase, except the first word, and doesn't include spaces between words.
- **Snake case** – Formats all characters in the string in lowercase and uses underscores instead of spaces between words.
- **Kebab case** – Formats all characters in the string in lowercase and uses dashes instead of spaces between words.
- **Trim** – Removes all leading and trailing spaces from the string.
- **Substring** – Returns a substring of the string, using the specified start and end positions.

This transformation provides a convenient way to standardize and tailor the presentation of string data for better visualization and analysis.

> **Note**
>
> This transformation is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

Format time

Use this transformation to customize the output of a time field. Output can be formatted using Moment.js format strings. For example, if you want to display only the year of a time field, the format string 'YYYY' can be used to show the calendar year (for example, 1999 or 2012).

Before Transformation:

Timestamp	Event
1636678740000000000	System Start
1636678680000000000	User Login
1636678620000000000	Data Updated

After applying 'YYYY-MM-DD HH:mm:ss':

Timestamp	Event
2021-11-12 14:25:40	System Start
2021-11-12 14:24:40	User Login
2021-11-12 14:23:40	Data Updated

This transformation lets you tailor the time representation in your visualizations, providing flexibility and precision in displaying temporal data.

> **Note**
>
> This transformation is available in workspaces compatbile with Grafana v12 as an alpha feature.

Group by

This transformation groups the data by a specified field (column) value and processes calculations on each group. Select to see a list of calculation choices.

Here's an example of original data.

Time	Server ID	CPU Temperature	Server Status
7/7/2020 11:34:20 AM	server 1	80	Shutdown
7/7/2020 11:34:20 AM	server 3	62	OK
7/7/2020 10:32:20 AM	server 2	90	Overload
7/7/2020 10:31:22 AM	server 3	55	OK

Time	Server ID	CPU Temperature	Server Status
7/7/2020 9:30:57 AM	server 3	62	Rebooting
7/7/2020 9:30:05 AM	server 2	88	OK
7/7/2020 9:28:06 AM	server 1	80	OK
7/7/2020 9:25:05 AM	server 2	88	OK
7/7/2020 9:23:07 AM	server 1	86	OK

This transformation goes in two steps. First you specify one or multiple fields to group the data by. This will group all the same values of those fields together, as if you sorted them. For instance if we group by the Server ID field, then it would group the data this way:

Time	Server ID	CPU Temperature	Server Status
7/7/2020 11:34:20 AM	server 1	80	Shutdown
7/7/2020 9:28:06 AM	server 1	80	OK
7/7/2020 9:23:07 AM	server 1	86	OK
7/7/2020 10:32:20 AM	server 2	90	Overload
7/7/2020 9:30:05 AM	server 2	88	OK
7/7/2020 9:25:05 AM	server 2	88	OK
7/7/2020 11:34:20 AM	server 3	62	OK
7/7/2020 10:31:22 AM	server 3	55	OK
7/7/2020 9:30:57 AM	server 3	62	Rebooting

All rows with the same value of Server ID are grouped together.

After choosing which field you want to group your data by, you can add various calculations on the other fields, and apply the calculation to each group of rows. For instance, we could want to calculate the average CPU temperature for each of those servers. So we can add the *mean* calculation applied on the CPU Temperature field to get the following:

Server ID	CPU Temperature (mean)
server 1	82
server 2	88.6
server 3	59.6

And we can add more than one calculation. For instance:

- For field Time, we can calculate the *Last* value, to know when the last data point was received for each server
- For field Server Status, we can calculate the *Last* value to know what is the last state value for each server
- For field Temperature, we can also calculate the *Last* value to know what is the latest monitored temperature for each server

We would then get:

Server ID	CPU Temperature (mean)	CPU Temperature (last)	Time (last)	Server Status (last)
server 1	82	80	7/7/2020 11:34:20 AM	Shutdown
server 2	88.6	90	7/7/2020 10:32:20 AM	Overload
server 3	59.6	62	7/7/2020 11:34:20 AM	OK

This transformation enables you to extract key information from your time series and display it in a convenient way.

Grouping to matrix

Use this transformation to combine three fields—which are used as input for the **Column**, **Row**, and **Cell value** fields from the query output—and generate a matrix. The matrix is calculated as follows:

Original data

Server ID	CPU Temperature	Server Status
server 1	82	OK
server 2	88.6	OK
server 3	59.6	Shutdown

We can generate a matrix using the values of `Server Status` as column names, the `Server ID` values as row names, and the `CPU Temperature` as content of each cell. The content of each cell will appear for the existing column (`Server Status`) and row combination (`Server ID`). For the rest of the cells, you can select which value to display between: **Null**, **True**, **False**, or **Empty**.

Output

Server IDServer Status	OK	Shutdown
server 1	82	
server 2	88.6	
server 3		59.6

Use this transformation to construct a matrix by specifying fields from your query results. The matrix output reflects the relationships between the unique values in these fields. This helps you present complex relationships in a clear and structured matrix format.

Group to nested table

Use this transformation to group the data by a specified field (column) value and process calculation on each group. Records are generated that share the same grouped field value, to be displayed in a nested table.

To calculate a statistic for a field, select the box next to the field and choose the **Calculate** option. This will add another selection box with statistics to be selected.

The following table shows sample data.

Time	Server ID	CPU Temperature	Server Status
7/7/2020 11:34:20 AM	server 1	80	Shutdown
7/7/2020 11:34:20 AM	server 3	62	OK
7/7/2020 10:32:20 AM	server 2	90	Overload
7/7/2020 10:31:22 AM	server 3	55	OK
7/7/2020 9:30:57 AM	server 3	62	Rebooting
7/7/2020 9:30:05 AM	server 2	88	OK
7/7/2020 9:28:06 AM	server 1	80	OK
7/7/2020 9:25:05 AM	server 2	88	OK
7/7/2020 9:23:07 AM	server 1	86	OK

This transformation has two steps. First, specify one or more fields by which to group the data. This groups all the same values of those fields together, as if you sorted them For instance, if you group by the `Server ID` field, Grafana groups the data this way:

<table>
<tr><th>Server ID</th><th>Data</th></tr>
<tr><td>server 1</td><td>
<table>
<tr><th>Time</th><th>CPU Temperature</th><th>Server Status</th></tr>
<tr><td>7/7/2020 11:34:20 AM</td><td>80</td><td>Shutdown</td></tr>
<tr><td>7/7/2020 9:28:06 AM</td><td>80</td><td>OK</td></tr>
<tr><td>7/7/2020 9:23:07 AM</td><td>86</td><td>OK</td></tr>
</table>
</td></tr>
<tr><td>server 2</td><td>
<table>
<tr><th>Time</th><th>CPU Temperature</th><th>Server Status</th></tr>
<tr><td>7/7/2020 10:32:20 AM</td><td>90</td><td>Overload</td></tr>
<tr><td>7/7/2020 9:30:05 AM</td><td>88</td><td>OK</td></tr>
<tr><td>7/7/2020 9:25:05 AM</td><td>88</td><td>OK</td></tr>
</table>
</td></tr>
<tr><td>server 3</td><td>
<table>
<tr><th>Time</th><th>CPU Temperature</th><th>Server Status</th></tr>
<tr><td>7/7/2020 11:34:20 AM</td><td>62</td><td>OK</td></tr>
</table>
</td></tr>
</table>

Server ID	Data		
	Time	**CPU Temperature**	**Server Status**
	7/7/2020 10:31:22 AM	55	OK
	7/7/2020 9:30:57 AM	62	Rebooting

After choosing the field by which you want to group your data, you can add various calculations on the other fields and apply the calculation to each group of rows. For instance, you might want to calculate the average CPU temperature for each of those servers. To do so, add the mean calculation applied on the CPU Temperature field to get the following result:

Server ID	CPU Temperatute (mean)		
server 1	82	**Time**	**Server Status**
		7/7/2020 11:34:20 AM	Shutdown
		7/7/2020 9:28:06 AM	OK
		7/7/2020 9:23:07 AM	OK

Server ID	CPU Temperatute (mean)	
server 2	88.6	<table><tr><th>Time</th><th>Server Status</th></tr><tr><td>7/7/2020 10:32:20 AM</td><td>Overload</td></tr><tr><td>7/7/2020 9:30:05 AM</td><td>OK</td></tr><tr><td>7/7/2020 9:25:05 AM</td><td>OK</td></tr></table>
server 3	59.6	<table><tr><th>Time</th><th>Server Status</th></tr><tr><td>7/7/2020 11:34:20 AM</td><td>OK</td></tr><tr><td>7/7/2020 10:31:22 AM</td><td>OK</td></tr><tr><td>7/7/2020 9:30:57 AM</td><td>Rebooting</td></tr></table>

Create heatmap

Use this transformation to prepare histogram data for visualizing trends over time. Similar to the heatmap visualization, this transformation converts histogram metrics into temporal buckets.

X Bucket

This setting determines how the x-axis is split into buckets.

- **Size** – Specify a time interval in the input field. For example, a time range of 1h creates cells one hour wide on the x-axis.
- **Count** – For non-time-related series, use this option to define the number of elements in a bucket.

Y Bucket

This setting determines how the y-axis is split into buckets.

- **Linear**
- **Logarithmic** – Choose between log base 2 or log base 10.
- **Symlog** – Uses a symmetrical logarithmic scale. Choose between log base 2 or log base 10, allowing for negative values.

Assume you have the following dataset:

Timestamp	Value
2023-01-01 12:00:00	5
2023-01-01 12:15:00	10
2023-01-01 12:30:00	15
2023-01-01 12:45:00	8

- With X Bucket set to `Size: 15m` and Y Bucket as `Linear`, the histogram organizes values into time intervals of 15 minutes on the x-axis and linearly on the y-axis.
- For X Bucket as `Count: 2` and Y Bucket as `Logarithmic (base 10)`, the histogram groups values into buckets of two on the x-axis and use a logarithmic scale on the y-axis.

Histogram

Use this transformation to generate a histogram based on input data, allowing you to visualize the distribution of values.

- **Bucket size** – The range between the lowest and highest items in a bucket (xMin to xMax).

- **Bucket offset** – The offset for non-zero-based buckets.
- **Combine series** – Create a unified histogram using all available series.

Original data

Series 1:

A	B	C
1	3	5
2	4	6
3	5	7
4	6	8
5	7	9

Series 2:

C
5
6
7
8
9

Output

xMin	xMax	A	B	C	C
1	2	1	0	0	0

xMin	xMax	A	B	C	C
2	3	1	0	0	0
3	4	1	1	0	0
4	5	1	1	0	0
5	6	1	1	1	1
6	7	0	1	1	1
7	8	0	1	1	1
8	9	0	0	1	1
9	10	0	0	1	1

Visualize the distribution of values using the generated histogram, providing insights into the data's spread and density.

Join by field

Use this transformation to merge multiple results into a single table, enabling the consolidation of data from different queries.

The is especially useful for converting multiple time series results into a single wide table with a shared time field.

Inner join

An inner join merges data from multiple tables where all tables share the same value from the selected field. This type of join excludes data where values do not match in every result.

Use this transformation to combine the results from multiple queries (combining on a passed join field or the first time column) into one result, and drop rows where a successful join cannot occur.

In the following example, two queries return table data. It is visualized as two separate tables before applying the inner join transformation.

Query A:

Time	Job	Uptime
7/7/2020 11:34:20 AM	node	25260122
7/7/2020 11:24:20 AM	postgre	123001233
7/7/2020 11:14:20 AM	postgre	345001233

Query B:

Time	Server	Errors
7/7/2020 11:34:20 AM	server 1	15
7/7/2020 11:24:20 AM	server 2	5
7/7/2020 11:04:20 AM	server 3	10

The result after applying the inner join transformation looks like the following:

Time	Job	Uptime	Server	Errors
7/7/2020 11:34:20 AM	node	25260122	server 1	15
7/7/2020 11:24:20 AM	postgre	123001233	server 2	5

Outer join

An outer join includes all data from an inner join and rows where values do not match in every input. While the inner join joins Query A and Query B on the time field, the outer join includes all rows that don't match on the time field.

In the following example, two queries return table data. It is visualized as two tables before applying the outer join transformation.

Query A:

Time	Job	Uptime
7/7/2020 11:34:20 AM	node	25260122
7/7/2020 11:24:20 AM	postgre	123001233
7/7/2020 11:14:20 AM	postgre	345001233

Query B:

Time	Server	Errors
7/7/2020 11:34:20 AM	server 1	15
7/7/2020 11:24:20 AM	server 2	5
7/7/2020 11:04:20 AM	server 3	10

The result after applying the outer join transformation looks like the following:

Time	Job	Uptime	Server	Errors
7/7/2020 11:04:20 AM			server 3	10
7/7/2020 11:14:20 AM	postgre	345001233		
7/7/2020 11:34:20 AM	node	25260122	server 1	15
7/7/2020 11:24:20 AM	postgre	123001233	server 2	5

Join by labels

Use this transformation to join multiple results into a single table.

This is especially useful for converting multiple time series results into a single wide table with a shared **Label** field.

- **Join** – Select the label to join by between the labels available or common across all time series.
- **Value** – The name for the output result.

Example

Input 1: `series1{what='Temp', cluster='A', job='J1'}`

Time	Value
1	10
2	200

Input 2: `series2{what='Temp', cluster='B', job='J1'}`

Time	Value
1	10
2	200

Input 3: `series3{what='Speed', cluster='B', job='J1'}`

Time	Value
22	22
28	77

Config:

```
value: 'what'
```

Output:

cluster	job	Temp	Speed
A	J1	10	
A	J1	200	
B	J1	10	22
B	J1	200	77

Combine and organize time series data effectively with this transformation for comprehensive insights.

Labels to fields

Use this transformation to convert time series results with labels or tags into a table, including each label's keys and values in the result. Display labels as either columns or row values for enhanced data visualization.

Given a query result of two time series:

- Series 1 – labels `Server=Server A`, `Datacenter=EU`
- Series 2 – labels `Server=Server B`, `Datacenter=EU`

In **Columns** mode, the result looks like this:

Time	Server	Datacenter	Value
7/7/2020 11:34:20 AM	Server A	EU	1
7/7/2020 11:34:20 AM	Server B	EU	2

In “Rows” mode, the result has a table for each series and show each label value like this:

label	value
Server	Server A
Datacenter	EU

label	value
Server	Server B
Datacenter	EU

Value field name

If you selected Server as the **Value field name**, then you would get one field for every value of the Server label.

Time	Datacenter	Server A	Server B
7/7/2020 11:34:20 AM	EU	1	2

Merging behavior

The labels to fields transformer is internally two separate transformations. The first acts on single series and extracts labels to fields. The second is the merge transformation that joins all the results into a single table. The merge transformation tries to join on all matching fields. This merge step is required and cannot be turned off.

To illustrate this, here is an example where you have two queries that return time series with no overlapping labels.

- Series 1 – labels `Server=ServerA`
- Series 2 – labels `Datacenter=EU`

This will first result in these two tables:

Time	Server	Value
7/7/2020 11:34:20 AM	ServerA	10

Time	Datacenter	Value
7/7/2020 11:34:20 AM	EU	20

After merge:

Time	Server	Value	Datacenter
7/7/2020 11:34:20 AM	ServerA	10	
7/7/2020 11:34:20 AM		20	EU

Limit

Use this transformation to restrict the number of rows displayed, providing a more focused view of your data. This is particularly useful when dealing with large datasets.

The following is an example illustrating the impact of the **Limit** transformation on a response from a data source:

Time	Metric	Value
7/7/2020 11:34:20 AM	Temperature	25
7/7/2020 11:34:20 AM	Humidity	22
7/7/2020 10:32:20 AM	Humidity	29

Time	Metric	Value
7/7/2020 10:31:22 AM	Temperature	22
7/7/2020 9:30:57 AM	Humidity	33
7/7/2020 9:30:05 AM	Temperature	19

Here is the result after adding a Limit transformation with a value of '3':

Time	Metric	Value
7/7/2020 11:34:20 AM	Temperature	25
7/7/2020 11:34:20 AM	Humidity	22
7/7/2020 10:32:20 AM	Humidity	29

This transformation helps you tailor the visual presentation of your data to focus on the most relevant data.

Merge series/tables

Use this transformation to combine the result from multiple queries into a single result, which is particularly useful when using the table panel visualization. The transformation merges values into the same row if shared fields contain the same data.

Here's an exampleillustration the impact of the **Merge series/tables** transformation on two queries returning table data:

Query A:

Time	Job	Uptime
7/7/2020 11:34:20 AM	node	25260122
7/7/2020 11:24:20 AM	postgre	123001233

Query B:

Time	Job	Errors
7/7/2020 11:34:20 AM	node	15
7/7/2020 11:24:20 AM	postgre	5

Here is the result after applying the Merge transformation:

Time	Job	Errors	Uptime
7/7/2020 11:34:20 AM	node	15	25260122
7/7/2020 11:24:20 AM	postgre	5	123001233

This transformation combines values from Query A and Query B into a unified table, enhancingthe presentation for better insights.

Organize fields by name

Use this transformation to rename, reorder, or hide fields returned by a single query in your panel. This transformation only works in panels with a single query. If your panel has multiple queries, then you must either apply an *Outer join* transformation or remove the extra queries.

Transforming fields

Grafana displays a list of fields returned by the query. You can:

- **Change field order** – Drag a field to a new location in the list.
- **Hide or show a field** – Use the eye icon next to the field name to toggle the visibility of a field.
- **Rename fields** – Type a new name into the **Rename** box.

Example

Given this initial query result:

Time	Metric	Value
2020-07-07 11:34:20	Temperature	25
2020-07-07 11:34:20	Humidity	22
2020-07-07 10:32:20	Humidity	29

You could apply a rename field override to create:

Time	Sensor	Reading
2020-07-07 11:34:20	Temperature	25
2020-07-07 11:34:20	Humidity	22
2020-07-07 10:32:20	Humidity	29

This transformation lets you tailor the display of query results, ensuring a clear and insightful representation of your data in Grafana.

Partition by values

Use this transformation to streamline the process of graphing multiple series without the need for multiple queries with different `WHERE` clauses.

> **Note**
>
> This function is available in workspaces compatible with Grafana version 9 and above.

This is particularly useful when using a metrics SQL table, as in this example:

Time	Region	Value
10/20/2022 12:00:00 PM	US	1520

Time	Region	Value
10/20/2022 12:00:00 PM	EU	2936
10/20/2022 1:00:00 AM	US	1327
10/20/2022 1:00:00 AM	EU	912

With the *Partition by values* transformer, you can issue a single query and split the results by unique values in one or more columns (fields) of your choosing. The following example uses `Region`.

```
SELECT Time, Region, Value FROM metrics WHERE Time > '2022-10-20'
```

Time	Region	Value
10/20/2022 12:00:00 PM	US	1520
10/20/2022 1:00:00 AM	US	1327

Time	Region	Value
10/20/2022 12:00:00 PM	EU	2936
10/20/2022 1:00:00 AM	EU	912

This transformation simplifies the process and enhances the flexibility of visualizing multiple series within the same time series visualization.

Prepare times series

Use this transformation to address issues when a data source returns time series data in a format that isn't compatible with the desired visualization. This transformation allows you to convert time series data from between wide and long formats.

Multi-frame time series

Use this option to transform the time series data frame from the wide format to the long format. This is particularly helpful when your data source delivers time series information in a format that needs to be reshaped for optimal compatibility with your visualization.

Example

This input:

Timestamp	Value1	Value2
2023-01-01 00:00:00	10	20
2023-01-01 01:00:00	15	25

Could be transformed to:

Timestamp	Variable	Value
2023-01-01 00:00:00	Value1	10
2023-01-01 00:00:00	Value2	20
2023-01-01 01:00:00	Value1	15
2023-01-01 01:00:00	Value2	25

Wide time series

Use this option to transform the time series data frame from the long format to the wide format. This is particularly helpful when your data source delivers time series data in a long format and your visualization requires a wide format.

Example

This input:

Timestamp	Variable	Value
2023-01-01 00:00:00	Value1	10

Timestamp	Variable	Value
2023-01-01 00:00:00	Value2	20
2023-01-01 01:00:00	Value1	15
2023-01-01 01:00:00	Value2	25

Could be transformed to:

Timestamp	Value1	Value2
2023-01-01 00:00:00	10	20
2023-01-01 01:00:00	15	25

Reduce

Use this transformation applies a calculation to each field in the data frame and return a single value. This transformation is particularly useful for consolidating mulitiple time series data into a more compact, summarized format. Time fields are removed when applying this transformation.

Consider the input:

Query A:

Time	Temp	Uptime
2020-07-07 11:34:20	12.3	256122
2020-07-07 11:24:20	15.4	1230233

Query B:

Time	AQI	Errors
2020-07-07 11:34:20	6.5	15

Time	AQI	Errors
2020-07-07 11:24:20	3.2	5

The reduce transformer has two modes:

- **Series to rows** – Creates a row for each field and a column for each calculation.
- **Reduce fields** – Keeps the existing frame structure, but collapses each field into a single value.

For example, if you used the **First** and **Last** calculation with a **Series to rows** transformation, then the result wouldbe:

Field	First	Last
Temp	12.3	15.4
Uptime	256122	1230233
AQI	6.5	3.2
Errors	15	5

The **Reduce fields** with the **Last** calculation, results in two frames, each with one row:

Query A:

Temp	Uptime
15.4	1230233

Query B:

AQI	Errors
3.2	5

Rename by regex

Use this transformation to rename parts of the query results using a regular expression and replacement pattern.

You can specify a regular expression, which is only applied to matches, along with a replacement pattern that support back references. For example, let's imagine you're visualizing CPU usage per host and you want to remove the domain name. You could set the regex to `([^\.]+)\..+` and the replacement pattern to `$1`, `web-01.example.com` would become `web-01`.

This transformation lets you tailor your data to meet your visualization needs, making your dashboards more informative and user-friendly.

Rows to fields

Use this transformation to convert rows into separate fields. This can be useful because fields can be styled and configured individually. It can also use additional fields as sources for dynamic field configuration or map them to field labels. The additional labels can then be used to define better display names for the resulting fields.

This transformation includes a field table which lists all fields in the data returned by the configuration query. This table gives you control over what field should be mapped to each configuration property (the **Use as** option). You can also choose which value to select if there are multiple rows in the returned data.

This transformation requires:

- One field to use as the source of field names.

  By default, the transform uses the first string field as the source. You can override this default setting by selecting **Field name** in the **Use as** column for the field you want to use instead.
- One field to use as the source of values.

  By default, the transform uses the first number field as the source. But you can override this default setting by selecting **Field value** in the **Use as** column for the field you want to use instead.

Useful when visualizing data in:

- Gauge

- Stat
- Pie chart

Map extra fields to labels

If a field does not map to config property Grafana will automatically use it as source for a label on the output field.

Example:

Name	DataCenter	Value
ServerA	US	100
ServerB	EU	200

Output:

ServerA (labels: DataCenter: US)	ServerB (labels: DataCenter: EU)
100	200

The extra labels can now be used in the field display name to provide more complete field names.

If you want to extract config from one query and apply it to another you should use the *config from query results* transformation.

Example

Input:

Name	Value	Max
ServerA	10	100
ServerB	20	200
ServerC	30	300

Output:

ServerA (config: max=100)	ServerB (config: max=200)	ServerC (config: max=300)
10	20	30

As you can see each row in the source data becomes a separate field. Each field now also has a max config option set. Options like **Min**, **Max**, **Unit** and **Thresholds** are all part of field configuration and if set like this will be used by the visualization instead of any options manually configured in the panel editor options pane.

This transformation enables the conversion of rows into individual fields, facilitates dynamic field configuration, and maps additional fields to labels.

Series to rows

Use this transformation to combine the result from multiple time series data queries into one single result. This is helpful when using the table panel visualization.

The result from this transformation will contain three columns: Time, Metric, and Value. The Metric column is added so you easily can see from which query the metric originates from. Customize this value by defining Label on the source query.

In the example below, we have two queries returning time series data. It is visualized as two separate tables before applying the transformation.

Query A:

Time	Temperature
2020-07-07 11:34:20	25
2020-07-07 10:31:22	22
2020-07-07 09:30:05	19

Query B:

Time	Humidity
2020-07-07 11:34:20	24
2020-07-07 10:32:20	29
2020-07-07 09:30:57	33

Here is the result after applying the Series to rows transformation.

Time	Metric	Value
2020-07-07 11:34:20	Temperature	25
2020-07-07 11:34:20	Humidity	22
2020-07-07 10:32:20	Humidity	29
2020-07-07 10:31:22	Temperature	22
2020-07-07 09:30:57	Humidity	33
2020-07-07 09:30:05	Temperature	19

This transformation facilitates the consolidation of results from multiple time series queries, providing a streamlined and unified dataset for efficient analysis and visualization in a tabular format.

Sort by

Use this transformation to sort each frame within a query result based on a specified field, making your data easier to understand and analyze. By configuring the desired field for sorting, you can control the order in which the data is presented in the table or visualization.

Use the **Reverse** switch to inversely order the values within the specified field. This functionality is particularly useful when you want to quickly toggle between ascending and descending order to suit your analytical needs.

For example, in a scenario where time-series data is retrieved from a data source, the **Sort by** transformation can be applied to arrange the data frames based on the timestamp, either in ascending or descending order, depending on the analytical requirements. This capability ensures that you can easily navigate and interpret time-series data, gaining valuable insights from the organized and visually coherent presentation.

Spatial

Use this transformation to apply spatial operations to query results.

- **Action** – Select an action:
  - **Prepare spatial field** – Set a geometry field based on the results of other fields.
    - **Location mode** – Select a location mode (these options are shared by the **Calculate value** and **Transform** modes):
      - **Auto** – Automatically identify location data based on default field names.
      - **Coords** – Specify latitude and longitude fields.
      - **Geohash** – Specify a geohash field.
      - **Lookup** – Specify Gazetteer location fields.
  - **Calculate value** – Use the geometry to define a new field (heading/distance/area).
    - **Function** – Choose a mathematical operation to apply to the geometry:
      - **Heading** – Calculate the heading (direction) between two points.
      - **Area** – Calculate the area enclosed by a polygon defined by the geometry.
      - **Distance** – Calculate the distance between two points.
  - **Transform** – Apply spatial operations to the geometry.
    - **Operation** – Choose an operation to apply to the geometry:
      - **As line** – Create a single line feature with a vertex at each row.
      - **Line builder** – Create a line between two points.

This transformation allows you to manipulate and analyze geospatial data, enabling operations such as creating lines between points, calculating spatial properties, and more.

Time series to table transform

Use this transformation to convert time series results into a table, transforming a time series data frame into a **Trend** field. The **Trend** field can then be rendered using the sparkline cell type,

generating an inline sparkline for each table row. If there are multiple time series queries, each will result in a separate table data frame. These can be joined using join or merge transforms to produce a single table with multiple sparklines per row.

For each generated **Trend** field value, a calculation function can be selected. The default is **Last non-null value**. This value is displayed next to the sparkline and used for sorting table rows.

Regression analysis

Use this transformation to create a new data frame containing values predicted by a statistical model. This is useful for finding a trend in chaotic data. It works by fitting a mathematical function to the data, using either linear or polynomial regression. The data frame can then be used in a visualization to display a trendline.

There are two different models:

- **Linear regression** – Fits a linear function to the data.
- **Polynomial regression** – Fits a polynomial function to the data.

> **Note**
>
> This transformation is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

Troubleshoot queries

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This page provides information to solve common dashboard problems.

I get different results when I rearrange my functions

Function order is very important. Just like in math, the order that you place your functions can affect the result.

Inspect your query request and response

The most common problems are related to the query and response from your data source. Even if it looks like a bug or visualization issue in Grafana, it is almost always a problem with the data source query or the data source response. Start by inspecting your panel query and response.

For more information, refer to Inspect request and response data.

My query is slow

How many data points is your query returning? A query that returns lots of data points will be slow. Try this:

- In **Query options**, limit the **Max data points** returned.
- In **Query options**, increase the **Min interval** time.
- In your query, use a `group by` function.

Calculation types

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The following table contains a list of calculations you can perform in Grafana. You can find these calculations in the **Transform** tab and in the bar gauge, gauge, and stat visualizations.

Calculation	Description
All nulls	True when all values are null
All values	Array with all values

Calculation	Description
All unique values	Array with all unique values
All zeros	True when all values are 0
Change count	Number of times the field's value changes
Count	Number of values in a field
Delta	Cumulative change in value, only counts increments
Difference	Difference between first and last value of a field
Difference percent	Percentage change between first and last value of a field
Distinct count	Number of unique values in a field
First	First value in a field
First* (not null)	First, not null value in a field (also excludes NaNs)
Last	Last value in a field
Last* (not null)	Last, not null value in a field (also excludes NaNs)
Max	Maximum value of a field
Mean	Mean value of all values in a field
Variance	Variance of all values in a field
StdDev	Standard deviation of all values in a field
Min	Minimum value of a field

Calculation	Description
Min (above zero)	Minimum, positive value of a field
Range	Difference between maximum and minimum values of a field
Step	Minimal interval between values of a field
Total	Sum of all values in a field

Configure panel options

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A Grafana panel is a visual representation of data that you can customize by defining a data source query, transforming and formatting data, and configuring visualization settings.

A panel editor includes a query builder and a series of options that you can use to transform data and add information to your panels.

This topic describes how to:

- Open a panel for editing
- Add a panel title and description
- View a panel JSON model
- Configure repeating rows and panels

Editing a panel

After you add a panel to a dashboard, you can open it at any time to change or update queries, add data transformation, and change visualization settings.

To edit a panel

1. Open the dashboard that contains the panel you want to edit.
2. Hover over any part of the panel to display the actions menu in the top right corner.
3. Choose the menu and select **Edit**.

   To use a keyboard shortcut to open the panel, hover over the panel and press e.

   The panel opens in edit mode.

Add a title and description to a panel

Add a title and description to a panel to share with users any important information about the visualization. For example, use the description to document the purpose of the visualization.

1. Edit a panel.
2. In the panel display options pane, locate the **Panel options** section.
3. Enter a **Title**.

   Text entered in this field appears in a tooltip in the panel editor and in the dashboard.
4. Write a description of the panel and the data you are displaying.

   Text entered in this field appears in a tooltip in the upper-left corner of the panel.

   You can use variables you have defined in the **Title** and **Description** field, but not global variables.

Viewing a panel JSON model

Explore and export panel, panel data, and data frame JSON models.

To view a panel JSON model

1. Open the dashboard that contains the panel.

2. Hover over any part of the panel to display the actions menu on the top right corner.
3. From the menu, select **Inspect > Panel JSON**.
4. In the **Select source** field, choose one of the following options:

   - **Panel JSON** – Displays a JSON object representing the panel.
   - **Panel data** – Displays a JSON object representing the data that was passed to the panel.
   - **DataFrame structure** – Displays the data structure of the panel, including any transformations, field configurations, and override configurations that have been applied.
5. To explore the JSON, choose **>** to expand or collapse portions of the JSON model.

Configuring repeating panels

You can configure Grafana to dynamically add panels or rows to a dashboard. A dynamic panel is a panel that the system creates based on the value of a variable. Variables dynamically change your queries across all panels in a dashboard. For more information about repeating panels, see Creating dashboards.

> **Note**
>
> Repeating panels require variables to have one or more items selected; you cannot repeat a panel zero times to hide it.

Prequisites

- Ensure that the query includes a multi-value variable.

To configure repeating panels

1. Edit the panel you want to repeat.
2. On the display options pane, choose **Panel options > Repeat options**.
3. Select a **direction**.

   - Choose **horizontal** to arrange panels side-by-side. Grafana adjusts the width of a repeated panel. You cannot mix other panels on a row with a repeated panel.
   - Choose **vertical** to arrange panels in a column. The width of repeated panels is the same as the original, repeated panel.

4. To propagate changes to all panels, reload the dashboard.

Configure standard options

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The data model used in Grafana is a columnar-oriented table structure that unifies both time series and table query results. Each column within this structure is called a *field*. A field can represent a single time series or table column.

Field options allow you to change how the data is displayed in your visualizations. Options and overrides that you apply do not change the data, they change how Grafana displays the data. When you change an option, it is applied to all fields, meaning all series or columns. For example, if you change the unit to percentage, then all fields with numeric values are displayed in percentages.

A complete list of field formatting options is included later in this topic.

> **Note**
>
> You can apply standard options to most built-in Grafana panels. Some older panels and community panels that have not updated to the new panel and data model will be missing either all or some of these field options.

To configure standard options

1. Open a dashboard. Hover over any part of the panel to display the actions menu at the top right corner of the dashboard.
2. From the actions menu, choose **Edit**.
3. In the panel display options pane, locate the **Standard options** section.
4. Select the standard options you want to apply.

5. To preview your change, select outside of the field option box you are editing or press **Enter**.

Standard options definitions

This section explains all available standard options.

You can apply standard options to most built-in Grafana panels. Some older panels and community panels that have not updated to the new panel and data model will be missing either all or some of these field options.

Most field options will not affect the visualization until you click outside of the field option box you are editing or press **Enter**.

> **Note**
>
> Grafana Labs is constantly working to add and expand options for all visualization, so all options might not be available for all visualizations.

Unit

Lets you choose what unit a field should use. Choose the **Unit** field, then drill down until you find the unit you want. The unit you select is applied to all fields except time.

Custom units

You can use the unit dropdown to also specify custom units, custom prefix or suffix and date time formats.

To select a custom unit enter the unit and select the last **Custom: xxx** option in the dropdown.

- **suffix:<suffix>** for a custom unit that should go after the value.
- **prefix:<prefix>** for a custom unit that should go before the value.
- **time:<format>** For custom date time formats, type for example `time:YYYY-MM-DD`. See format in the *Moment.js Documentation* for the format syntax and options.
- **si:<base scale><unit characters>** for custom SI units. For example: `si: mF`. This is a bit more advanced as you can specify both a unit and the source data scale. So if your source data is represented as milli (thousands of) something prefix the unit with that SI scale character.

- **count:<unit>** for a custom count unit.
- **currency:<unit>** for custom a currency unit.

You can also paste a native emoji in the unit picker and pick it as a custom unit.

String units

Grafana can sometimes be too aggressive in parsing strings and displaying them as numbers. To configure Grafana to show the original string value, create a field override and add a unit property with the **String** unit.

Scale units

By default, Grafana automatically scales the unit based on the magnitude of the value. For example, if you have a value of 0.14 kW, Grafana will display it as 140 W. Another example is that 3000 kW will be displayed as three MW. If you want to disable this behavior, you can turn off the **Scale units** switch.

Min

Lets you set the minimum value used in percentage threshold calculations. Leave blank to automatically calculate the minimum.

Max

Lets you set the maximum value used in percentage threshold calculations. Leave blank to automatically calculate the maximum.

Field min/max

By default the calculated min and max will be based on the minimum and maximum, in all series and fields. Turning field min/max on will calculate the min or max on each field individually, based on the minimum or maximum of that field.

Decimals

Specify the number of decimals Grafana includes in the rendered value. If you leave this field blank, Grafana automatically truncates the number of decimals based on the value. For example 1.1234 will display as 1.12 and 100.456 will display as 100.

To display all decimals, set the unit to **String**.

Display name

Lets you set the display title of all fields. You can use variables in the field title.

When multiple stats, fields, or series are shown, this field controls the title in each stat. You can use expressions like **${__field.name}** to use only the series name or the field name in the title.

Given a field with a name of `Temp`, and labels of `{"Loc"="PBI", "Sensor"="3"}`:

Expression syntax	Example	Renders to	Explanation
`${__field .displayName}`	Same as syntax	`Temp {Loc="PBI ", Sensor="3"}`	Displays the field name, and labels in {} if they are present. If there is only one label key in the response, then for the label portion, Grafana displays the value of the label without the enclosing braces.
`${__field.name}`	Same as syntax	`Temp`	Displays the name of the field (without labels).
`${__field .labels}`	Same as syntax	`Loc="PBI", Sensor="3"`	Displays the labels without the name.
`${__field .labels.` *X*`}`	`${__field .labels.Loc}`	`PBI`	Displays the value of the specified label key.
`${__field .labels._ _values}`	Same as Syntax	`PBI, 3`	Displays the values of the labels separated

Expression syntax	Example	Renders to	Explanation
			by a comma (without label keys).

If the value is an empty string after rendering the expression for a particular field, then the default display method is used.

Color scheme

The color options and their effect on the visualization depends on the visualization you are working with. Some visualizations have different color options.

You can specify a single color, or select a continuous (gradient) color scheme, based on a value. Continuous color interpolates a color using the percentage of a value relative to min and max.

Select one of the following palettes:

Color mode	Description
Single color	Specify a single color, useful in an override rule
Shades of a color	Selects shades of a single color, useful in an override rule
From thresholds	Informs Grafana to take the color from the matching threshold
Classic palette	Grafana will assign color by looking up a color in a palette by series index. Useful for Graphs and pie charts and other categorical data visualizations
Classic palette (by series name)	Grafana will assign color based on the name of the series. Useful when the series names to be vsiualized depend on the available data.
Green-Yellow-Red (by value)	Continuous color scheme

Color mode	Description
Red-Yellow-Green (by value)	Continuous color scheme
Blue-Yellow-Red (by value)	Continuous color scheme
Yellow-Red (by value)	Continuous color scheme
Blue-Purple (by value)	Continuous color scheme
Yellow-Blue (by value)	Continuous color scheme
Blues (by value)	Continuous color scheme (panel background to blue)
Reds (by value)	Continuous color scheme (panel background color to red)
Greens (by value)	Continuous color scheme (panel background color to green)
Purple (by value)	Continuous color scheme (panel background color to purple)

No value

Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

Configure a legend

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A panel includes a legend that you can use to interpret data displayed in a visualization. Each legend option adds context and clarity to the data illustrated in a visualization.

Legends are supported for the following visualizations:

- Bar chart
- Candlestick
- Histogram
- Pie chart
- State timeline
- Status history
- Time series
- Trend

Geomaps and Heatmaps also have legends, but they only provide the choice to display or not display a legend, and don't support other legend options.

Legend options

You can find the following options under the **Legend** section in the panel edit pane.

> **Note**
>
> Not all of the options listed apply to all visualizations with legends.

Visibility

Set whether the legend is displayed or not. Use the switch to toggle a legend on or off.

Mode

Set the format in which the legend is displayed. Choose from:

- **List**
- **Table**

When you format a legend as a table, other information about the legend, such as associated values or where it's located in the visualization, might be displayed as well.

Placement

Set where on the visualization a legend is displayed. Choose from:

- **Bottom**
- **Right**

Width

If you set the legend placement to **Right**, the **Width** option becomes available. Leave the field empty to allow Grafana to automatically set the legend width, or enter a vale in the field.

Values

You can add more context to a visualization by adding series data values or calculations to a legend. You can add as many values as you like. After you apply your changes, you can scroll the legend to see all values.

Changing a series color

By default, Grafana sets the color of your series data, but you can change them through the panel legend.

To change a series color

1. Navigate to the panel you want to update.
2. In the legend, select the color bar associated with the series.
3. Select a pre-set color in the **Colors** tab or set a custom color in the **Custom** tab, using the picker or RGB values.
4. Save the dashboard.

Isolating series data in a visualization

Visualizations can often be visually complex, and include many data series. You can simplify the view by removing series data from the visualization through the legend, which isolates the data

you want to see. When you do this, Grafana automatically creates a new override in the **Override** tab.

To isolate series data in a visualization

1. Navigate to the panel you want to update.
2. In the legend, select the label of the series you want to isolate.

   The system removes from view all other series data.
3. To incrementally add series data back to an isolated series, press the **Ctrl** or **Command** key and select the label of the series you want to add.
4. To save your changes so that they appear to all viewers of the panel, save the dashboard.

To revert back to the default view that includes all data, click any series label twice.

Sorting series

When you format a legend as a table and add values to it, you can sort the series in the table by those values.

To sort series

1. Navigate to the panel you want to update.
2. Hover over any part of the panel you want to work on to display the menu on the top right corner of the panel.
3. From the menu, choose **Edit**.
4. Scroll to the **Legend** section of the panel edit pane.
5. Under **Values**, select the value or calculation that you want to show.

   The legend now displays values.
6. Choose the calculation name header in the legend table to sort the values in the table in ascending or descending order.

> **Note**
>
> This feature is only supported in these panels: bar chart, histogram, time series.

Configure data links

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Data links allow you to provide more granular context to your links. You can create links that include the series name or even the value under the cursor. For example, if your visualization shows four servers, you can add a datalink to one or two of them. You can also link panels using data links.

The link itself is accessible in different ways depending on the visualization. For the time series visualization, for example, you choose a data point or line. For large area visualizations, like stat, gauge, or bar gauge, you can choose anywhere on the visualization to open the context menu.

If there is only one data link in the visualization, choosing anywhere on the visualization opens the link rather than the context menu.

Supported visualizations

- Bar chart
- Bar gauge
- Candlestick
- Canvas
- Gauge
- Geomap
- Heatmap
- Histogram
- Pie chart
- Stat
- State timeline

- Status history
- Table
- Time series
- Trend

Data link variables

Variables in data links let you send people to a detailed dashboard with preserved data filters. For example, you could use variable to specify a label, time range, series, or variable selection.

To see a list of available variables, type $ in the data link **URL** field.

You can also use template variables in your data links URLs, see Variables.

Time range panel variables

These variables allow you to include the current time range in the data link URL.

- `__url_time_range` – current dashboard's time range (i.e. `?from=now-6h&to=now`)
- `$__from` – For more information, see Global variables.
- `$__to` – For more information, see Global variables.

Series variables

Series specific variables are available under `__series` namespace:

- `__series.name` – series name to the URL

Field variables

Field-specific variables are available under `__field` namespace:

- `__field.name` – the name of the field
- `__field.labels.<LABEL>` – label's value to the URL. If your label contains dots, then use `__field.labels["<LABEL>"]` syntax.

Value variables

Value-specific variables are available under `__value` namespace:

- `__value.time` – value's timestamp (Unix ms epoch) to the URL (i.e. `?time=1560268814105`)
- `__value.raw` – raw value
- `__value.numeric` – numeric representation of a value
- `__value.text` – text representation of a value
- `__value.calc` – calculation name if the value is result of calculation

Using value-specific variable in data links can show different result depending on the set option of Tooltip mode.

Data variables

To access values from other fields use:

- `__data.fields[i]` – Value of field `i` (on the same row).
- `__data.fields["NameOfField"]` – Value of field using name instead of index.
- `__data.fields[i].labels.cluster` – Access the labels of another field.

Template variables

When linking to another dashboard that uses template variables, select variable values for whoever clicks the link.

`${var-myvar:queryparam}` – where `var-myvar` is the name of the template variable that matches one in the current dashboard that you want to use.

Variable state	Result in the created URL
selected one value	`var-myvar=value1`
selected multiple values	`var-myvar=value1&var-myvar=value2`
selected `All`	`var-myvar=All`

If you want to add all of the current dashboard's variables to the URL, then use `${__all_variables}`.

Adding a data link

You can add data links to your panels.

1. Navigate to the panel to which you want to add the data link.
2. Hover over the panel to display the menu icon in the upper-right corner.
3. From the menu, choose **Edit** to open the panel editor.
4. In the **Panel edit** pane, scroll down to the **Data links** section and expand it.
5. Choose **Add link**.
6. In the dialog box that opens, enter a **Title**. This is a human-readable label for the link, which will be displayed in the UI.
7. Enter the **URL** or variable you want to link to.

   To add a data link variable, select the **URL** field and then enter $ or press Ctrl+Space or Cmd +Space to see a list of available variables.
8. If you want the link to open in a new tab, then select **Open in a new tab**.
9. Choose **Save** to save changes and close the dialog box.
10. Save your changes to the dashboard.

Configure value mappings

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

In addition to field overrides, value mapping is a technique that you can use to change how data appears in a visualization.

Values mappings bypass the unit formatting set in the Standard options of the panel editor, like color or number of decimal places displayed. When value mappings are present in a panel, Grafana displays a summary of them in the **Value mappings** section of the panel editor.

Supported visualizations

- Bar chart
- Bar gauge
- Candlestick
- Canvas
- Gauge
- Geomap
- Histogram
- Pie chart
- Stat
- State timeline
- Status history
- Table
- Time series
- Trend

Types of value mappings

Grafana supports the following value mapping types:

- **Value** – Maps specific values to a text and a color. For example, you can configure a value mapping so that all instances of the value `10` appear as `Perfection!` rather than the number. Use a **Value** mapping when you want to format a single value.
- **Range** – Maps numerical ranges to text and a color. For example, if a value is within a certain range, you can configure a range value mapping to display `Low` or `High` rather than the number. Use **Range** mapping when you want to formate multiple continuous values.
- **Regex** – Maps regular expressions to text and a color. For example, if a value is `www.example.com`, you can configure a regex value mapping so that Grafana displays `www` and truncates the domain. Use the **Regex** mapping when you want to format the text and color of a regular expression value.
- **Special** – Maps special values like `Null`, `NaN` (not a number), and Boolean values like `true` and `false` to a text and color. For example, you can configure a special value mapping so that `null`

values appear as N/A. Use the **Special** mappiong when you want to format uncommon, Boolean, or empty value.

Adding a value mapping

You can add value mappings to your panels.

To add a value mapping

1. Navigate to the panel you want to update.
2. Hover over any part of the panel to display a menu at the top right corner of the panel.
3. From the menu, choose **Edit**.
4. In the **Value mappings** section, choose **Add value mappings**.
5. Choose **Add a new mapping**, and then select one of the following:
   - **Value** – Enter a single value to match.
   - **Range** – Enter the beginning and ending values of a range to match.
   - **Regex** – Enter a regular expression pattern to match.
   - **Special** – Choose a special value to match.
6. (Optional) Enter display text.
7. (Optional) Set the color.
8. Choose **Update** to save the value mapping.

After you've added a mapping, the **Edit value mappings** button replaces the **Add value mappings** button. Choose the edit button to add or update mappings.

Configure thresholds

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

In dashboards, a threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

You can use thresholds to:

- Color grid lines or grid ares areas in a time-series visualization.
- Color the background or value text in a stat visualization.
- Define regions and region colors in a state timeline.
- Color lines in a time-series visualization.
- Color the gauge and threshold markers in a gauge.
- Color markers in a geomap.
- Color cell text or background in a table.

Supported visualizations

- Bar chart
- Bar gauge
- Candlestick
- Canvas
- Gauge
- Geomap
- Histogram
- Stat
- State timeline
- Status history
- Table
- Time series
- Trend

Default thresholds

On visualizations that support thresholds, Grafana has the following default threshold settings:

- 80 = red
- Base = green
- Mode = Absolute
- Show thresholds = Off (for some visualizations)

  For more information, see Show thresholds.

Thresholds options

You can set the following options to further define how thresholds look.

Thresholds value

This number is the value that triggers the threshold. You can also set the color associated with the threshold in this field.

The **Base** value represents minus infinity. By default, it's set to the color green, which is generally the "good" color.

Thresholds mode

There are two threshold modes:

- **Absolute** thresholds are defined by a number. For example, 80 on a scale of 1 to 150.
- **Percentage** thresholds are defined relative to minimum or maximum. For example, 80 percent.

Show thresholds

> **Note**
>
> This option is only supported for the bar chart, candlestick, time series, and trend visualizations.

Set if and how thresholds are shown on the visualization with the following options.

- **Off** – The thresholds are not shown.
- **As lines** – The thresholds are shown as horizontal lines on the visualization.
- **As lines (dashed)** – The thresholds are shown as dashed horizontal lines.

- **As filled regions** – The thresholds are shown as horizontal regions.
- **As filled regions and lines** – The thresholds are shown as horizontal regions separated by lines.
- **As filled regions and lines (dashed)** – The thresholds are shown as horizontal regions separated by dashed lines.

Adding a threshold

You can add as many thresholds to a visualization as you want. Grafana automatically sorts thresholds values from highest to lowest.

To add a threshold

1. Navigate to the panel you want to update.
2. Hover over any part of the panel to display the menu at the top right corner.
3. From the menu, select **Edit**.
4. Scroll to the **Thresholds** section, or enter `Thresholds` in the search bar at the top of the panel edit pane.
5. Choose **+ Add threshold**.
6. Enter a new threshold value, or use the up and down arrows at the right side of the field to increase or decrease the value incrementally.
7. Click the colored circle to the left of the threshold value to open the color picker, where you can update the threshold color.
8. Under **Thresholds mode**, select either **Absolute** or **Percentage**.
9. Under **Show thresholds**, set how the threshold is displayed, or turn it off.

To delete a threshold, navigate to the panel that contains the threshold and choose the trash icon next to the threshold you want to remove.

Configure field overrides

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Overrides allow you to customize visualization settings for specific fields or series. This is accomplished by adding an override rule that targets a particular set of fields and that can each define multiple options.

For example, you set the unit for all fields that include the text `bytes` by adding an override using the **Fields with name matching regex** matcher and then add the Unit option to the override rule.

Example 1: Format temperature

Let's assume that our result set is a data frame that consists of two fields: `time` and `temperature`.

time	temperature
2020-01-02 03:04:00	45.0
2020-01-02 03:05:00	47.0
2020-01-02 03:06:00	48.0

Each field (column) of this structure can have field options applied that alter the way its values are displayed. This means that you can, for example, set the Unit to Temperature > Celsius, resulting in the following table:

time	temperature
2020-01-02 03:04:00	45.0 °C
2020-01-02 03:05:00	47.0 °C
2020-01-02 03:06:00	48.0 °C

In addition, the decimal place is not required, so we can remove it. You can change the Decimals from **auto** to zero (**0**), resulting in the following table:

time	temperature
2020-01-02 03:04:00	45 °C
2020-01-02 03:05:00	47 °C
2020-01-02 03:06:00	48 °C

Example 2: Format temperature and humidity

Let's assume that our result set is a data frame that consists of four fields: `time`, `high temp`, `low temp`, and `humidity`.

time	high temp	low temp	humidity
2020-01-02 03:04:00	45.0	30.0	67
2020-01-02 03:05:00	47.0	34.0	68
2020-01-02 03:06:00	48.0	31.0	68

Let's add the Celsius unit and get rid of the decimal place. This results in the following table:

time	high temp	low temp	humidity
2020-01-02 03:04:00	45 °C	30 °C	67 °C
2020-01-02 03:05:00	47 °C	34 °C	68 °C
2020-01-02 03:06:00	48 °C	31 °C	68 °C

The temperature fields look good, but the humidity must now be changed. We can fix this by applying a field option override to the humidity field and change the unit to Misc > percent (0-100).

time	high temp	low temp	humidity
2020-01-02 03:04:00	45 °C	30 °C	67%
2020-01-02 03:05:00	47 °C	34 °C	68%
2020-01-02 03:06:00	48 °C	31 °C	68%

Adding a field override

A field override rule can customize the visualization settings for a specific field or series.

To add a field override

1. Edit the panel to which you want to add an override.
2. In the panel options side pane, choose **Add field override** at the bottom of the pane.
3. Select which fields an override rule will be applied to:
   - **Fields with name** – Select a field from the list of all available fields. Properties you add to a rule with this selector are only applied to this single field.
   - **Fields with name matching regex** – Specify fields to override with a regular expression. Properties you add to a rule with this selector are applied to all fields where the field name match the regex. This selects fields to override, but doesn't rename the fields; to do this, use the Rename by regex transformation.
   - **Fields with type** – Select fields by type, such as string, numeric, and so on. Properties you add to a rule with this selector are applied to all fields that match the selected type.
   - **Fields returned by query** – Select all fields returned by a specific query, such as A, B, or C. Properties you add to a rule with this selector are applied to all fields returned by the selected query.
4. Choose **Add override property**.
5. Select the field option that you want to apply.
6. Enter options by adding values in the fields. To return options to default values, delete the white text in the fields.
7. Continue to add overrides to this field by choosing **Add override property**, or you can choose **Add override** and select a different field to add overrides to.
8. When finished, choose **Save** to save all panel edits to the dashboard.

Deleting a field override

Delete a field override when you no longer need it. When you delete an override, the appearance of value defaults to its original format. This change impacts dashboards and dashboard users that rely on an affected panel.

To delete a field override

1. Edit the panel that contains the override you want to delete.
2. In panel options side pane, scroll down until you see the overrides.
3. Choose the override you want to delete and then choose the associated trash icon.

Viewing field overrides

You can view field overrides in the panel display options.

1. Edit the panel that contains the overrides you want to view.
2. In panel options side pane, scroll down until you see the overrides.

> **Note**
>
> The override settings that appear on the **All** tab are the same as the settings that appear on the **Overrides** tab.

Editing a field override

Edit a field override when you want to make changes to an override setting. The change you make takes effect immediately.

To edit a field override

1. Edit the panel that contains the overrides you want to edit.
2. In panel options side pane, scroll down until you see the overrides.
3. Locate the override that you want to change.
4. Perform any of the following:

   - Edit settings on existing overrides or field selection parameters.

- Delete existing override properties by choosing the **X** next to the property.
- Add an override properties by choosing **Add override property**.

Visualizations available in Grafana version 12

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana offers a variety of visualizations to support different use cases. This section of the documentation highlights the built-in visualizations, their options and typical usage.

A common panel to get started with, and to learn the basics of using panels, is the Time series panel.

Note

If you are unsure which visualization to pick, Grafana can provide visualization suggestions based on the panel query. When you select a visualization, Grafana will show a preview with that visualization applied.

- Graphs & charts
  - Time series is the default and main Graph visualization.
  - State timeline for state changes over time.
  - Status history for periodic state over time.
  - Bar chart shows any categorical data.
  - Histogram calculates and shows value distribution in a bar chart.
  - Heatmap visualizes data in two dimensions, used typically for the magnitude of a phenomenon.
  - Pie chart is typically used where proportionality is important.

  - Candlestick is typically for financial data where the focus is price/data movement.
  - Gauge is the traditional rounded visual showing how far a single metric is from a threshold.
  - Trend for datasets that have a sequential, numeric x that is not time.
  - XY Chart provides a way to visualize arbitrary x and y values in a graph.
- Stats & numbers
  - Stat for big stats and optional sparkline.
  - Bar gauge is a horizontal or vertical bar gauge.
- Misc
  - Table is the main and only table visualization.
  - Logs is the main visualization for logs.
  - Node graph for directed graphs or networks.
  - Traces is the main visualization for traces.
  - Flame graph is the main visualization for profiling.
  - Geomap helps you visualize geospatial data.
  - Datagrid allows you to create and manipulate data, and acts as a data source for other panels.
- Widgets
  - Dashboard list can list dashboards.
  - Alert list can list alerts.
  - Text can show markdown and html.
  - News can show RSS feeds.

Get more

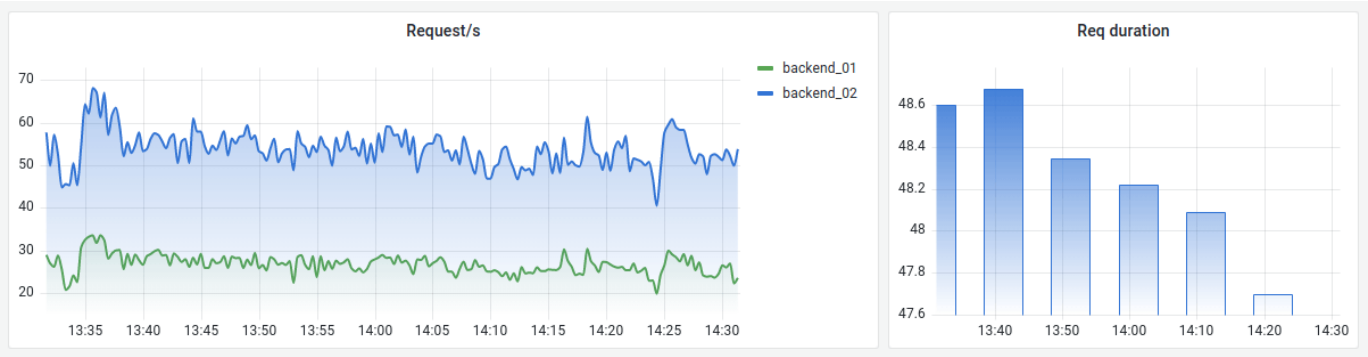
You can add more visualization types by installing panel plugins from the Find plugins with the plugin catalog.

Examples

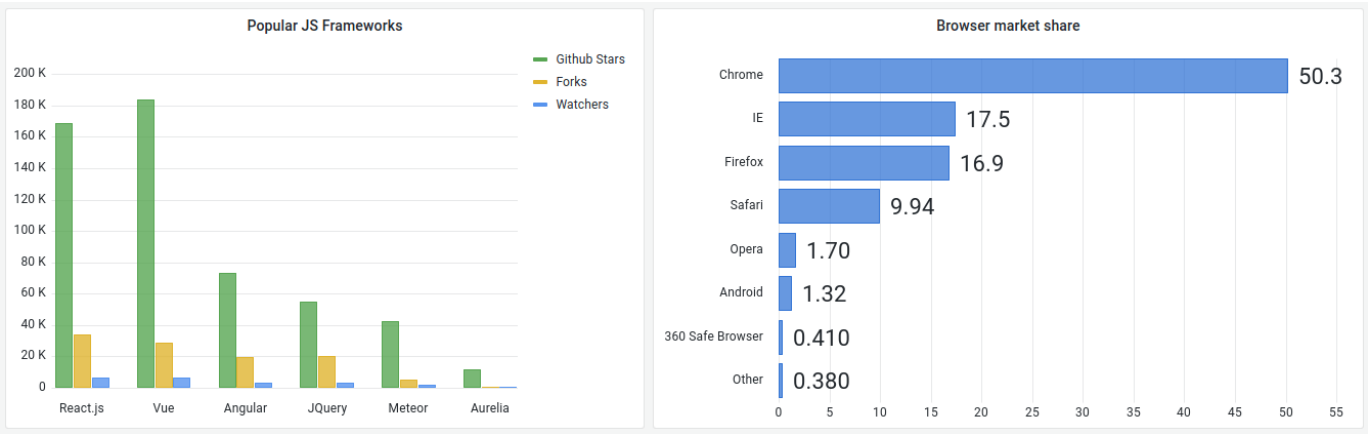
In the following sections you can find visualizations examples.

Graphs

For time based line, area, and bar charts, we recommend the default time series visualization.

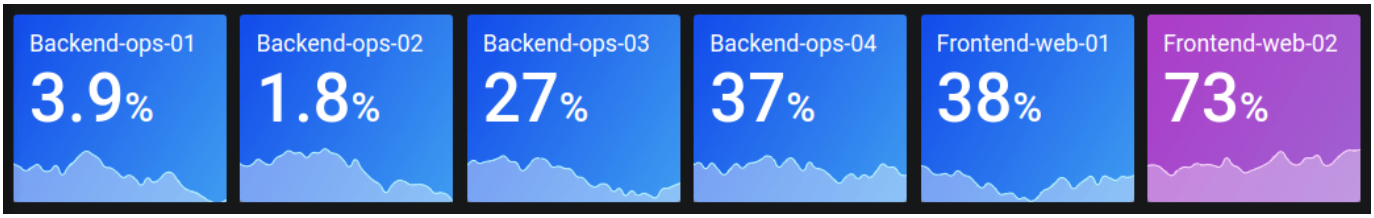


For categorical data, use a bar chart.



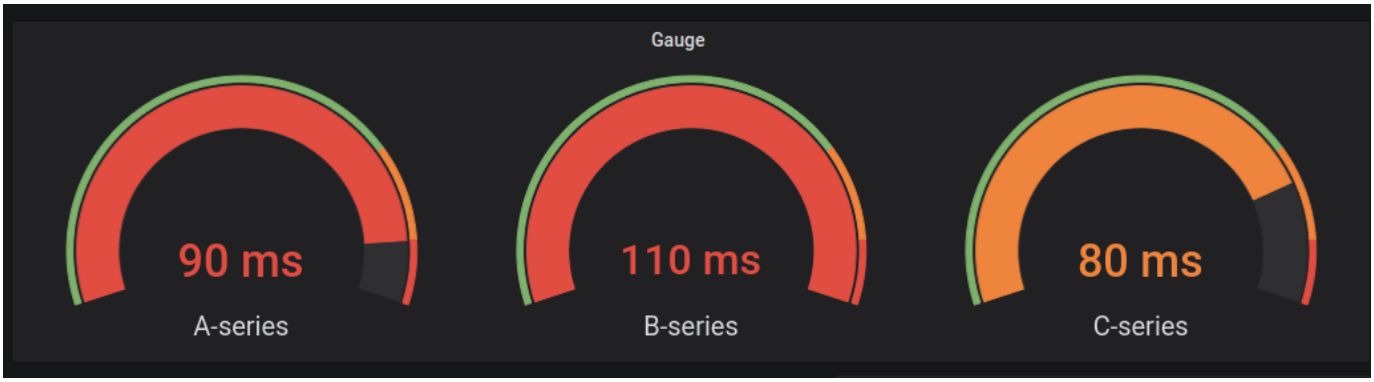
Big numbers & stats

A stat visualization shows one large stat value with an optional graph sparkline. You can control the background or value color using thresholds or color scales.

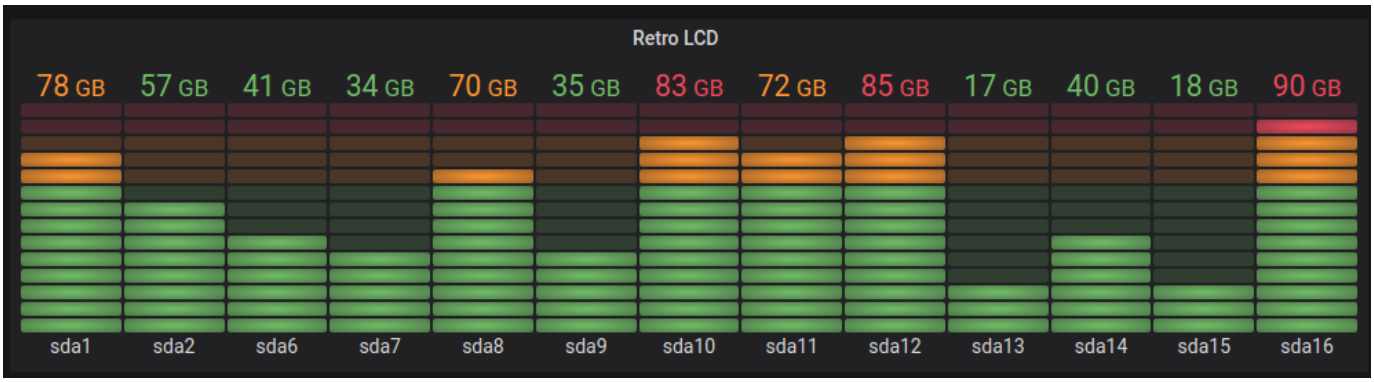


Gauge

If you want to present a value as it relates to a min and max value, you have two options. First a standard radial gauge:

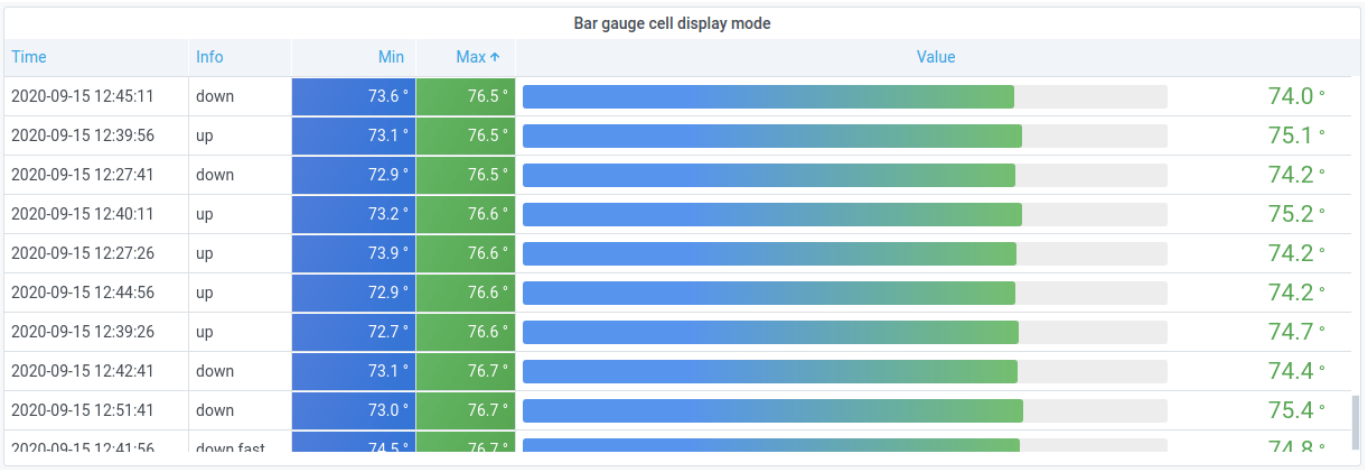


Secondly, Grafana also has a horizontal or vertical bar gauge with three distinct display modes.



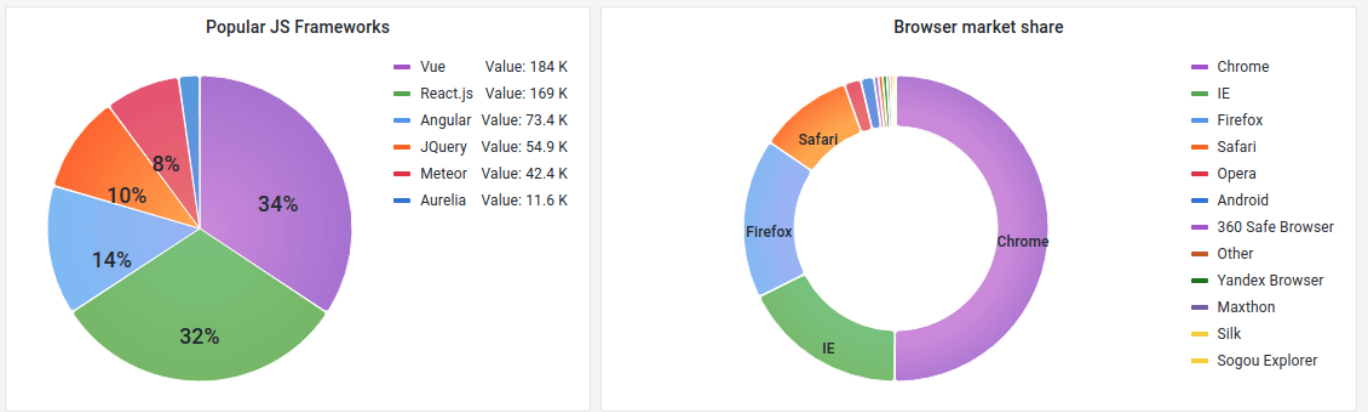
Table

To show data in a table layout, use a table visualization.



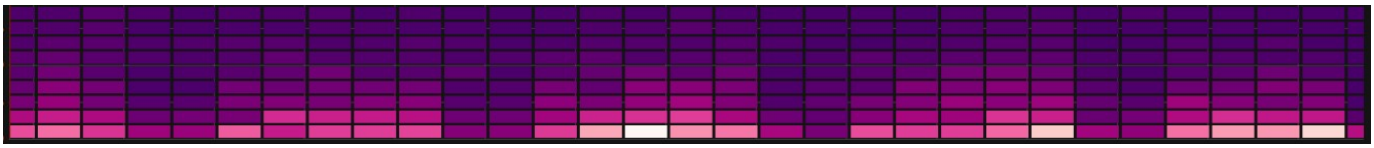
Pie chart

To display reduced series, or values in a series, from one or more queries, as they relate to each other, use a pie chart visualization.



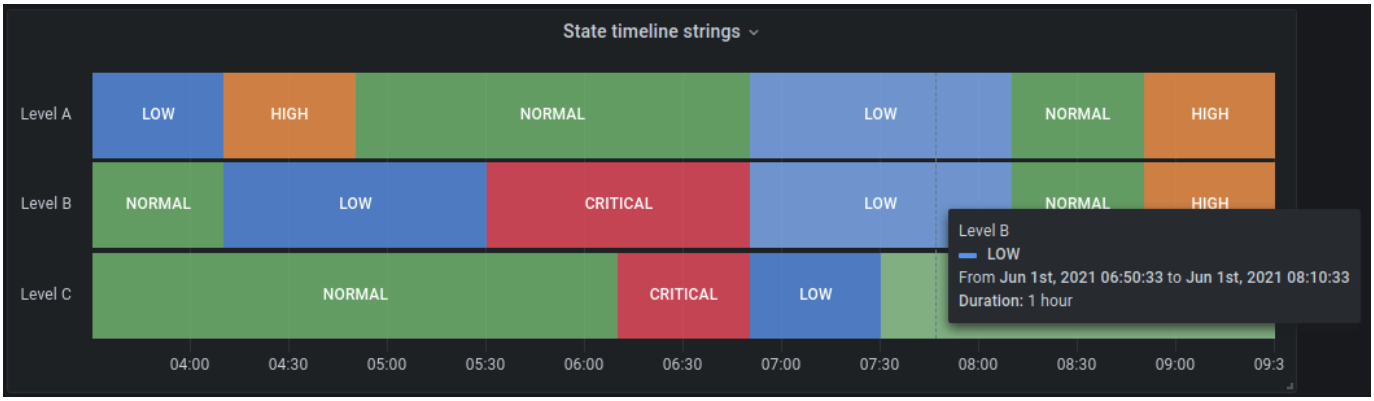
Heatmaps

To show value distribution over time, use a heatmap visualization.



State timeline

A state timeline shows discrete state changes over time. When used with time series, thresholds are used to turn numerical values into discrete state regions.



Alert list

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use alert lists to display your alerts. You can configure the list to show current state. For more information about alerts, see Alerts in Grafana version 12.

Use these settings to refine your visualization.

Options

- **Group mode** – Choose between **Default grouping** to show alert instances grouped by their alert rule, and **Custom grouping** to group alert instances by a custom set of labels.
- **Max Items** – Set the maximum number of alerts to list.
- **Sort order** – Select how to order the alerts displayed.
  - **Alphabetical (asc)** – Alphabetical order
  - **Alphabetical (desc)** – Reverse alphabetical order
  - **Importance** – By importance according to the following values, with 1 being the highest:
    - `alerting` or `firing`: 1
    - `no_data`: 2
    - `pending`: 3
    - ok: 4
    - `paused` or `inactive`: 5
  - **Time (asc)** – Newest active alert instances first.
  - **Time (desc)** – Oldest active alert instances first.
- **Alerts from this dashboard** – Show alerts only from the dashboard that the alert list is in.

Filter

These options allow you to limit alerts shown to only those that match the query, folder, or tags that you choose:

- **Alert name** – Enter an alert name query.
- **Alert instance label** – Filter alert instances using label querying. For example, `{severity="critical", instance=~"cluster-us-.+"}`.
- **Folder** – Select a folder. Only alerts from dashboards in the selected folder will be displayed.

- **Datasource** – Filter alerts from the selected data source.

State filter

Choose which alert states to display in this panel.

- Alerting / Firing
- Pending
- No data
- Normal
- Error

Annotations list

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The Annotations list shows a list of available annotations you can use to view annotated data. Various options are available to filter the list based on tags and on the current dashboard.

Annotation query

The following options control the source query for the list of annotations.

Query Filter

Use the query filter to create a list of annotations from all dashboards in your organization or the current dashboard in which this panel is located. It has the following options:

- All dashboards - List annotations from all dashboards in the current organization.
- This dashboard - Limit the list to the annotations on the current dashboard.

Time Range

Use the time range option to specify whether the list should be limited to the current time range. It has the following options:

- None - no time range limit for the annotations query.
- This dashboard - Limit the list to the time range of the dashboard where the annotation list panel is available.

Tags

Use the tags option to filter the annotations by tags. You can add multiple tags in order to refine the list.

> **Note**
>
> Optionally, leave the tag list empty and filter on the fly by selecting tags that are listed as part of the results on the panel itself.

Limit

Use the limit option to limit the number of results returned.

Display

These options control additional metadata included in the annotations panel display.

Show user

Use this option to show or hide which user created the annotation.

Show time

Use this option to show or hide the annotation creation time.

Show Tags

Use this option to show or hide the tags associated with an annotation. *NB*: You can use the tags to live-filter the annotation list on the visualization itself.

Link behavior

Link target

Use this option to chose how to view the annotated data. It has the following options.

- Panel - This option will take you directly to a full-screen view of the panel with the corresponding annotation
- Dashboard - This option will focus the annotation in the context of a complete dashboard

Time before

Use this option to set the time range before the annotation. Use duration string values like “1h” = 1 hour, “10m” = 10 minutes, etc.

Time after

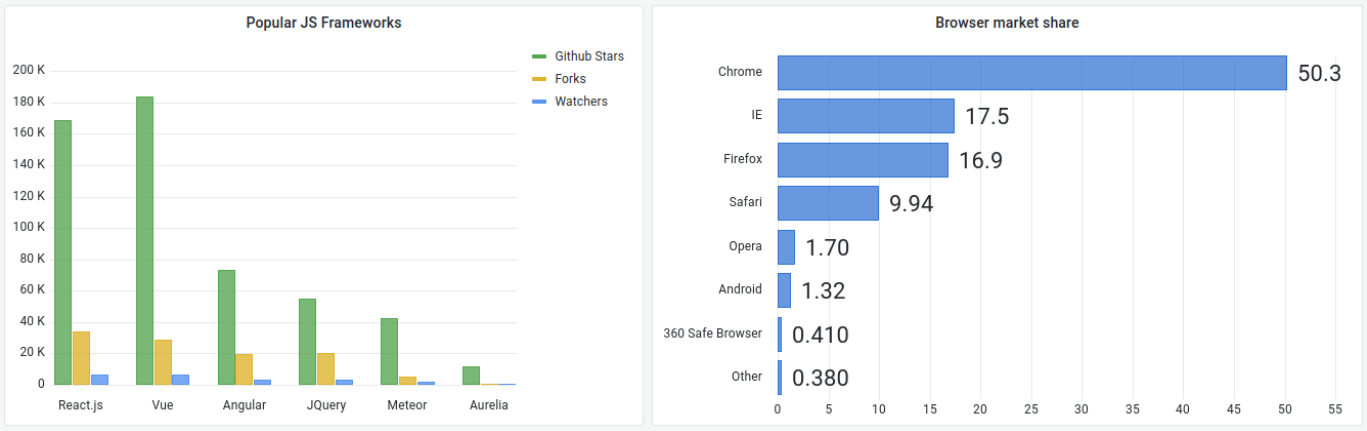
Use this option to set the time range after the annotation.

Bar chart



This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Bar charts allow you to graph categorical data.



Supported data formats

Only one data frame is supported and it needs to have at least one string field that will be used as the category for an X or Y axis and one or more numerical fields. The following is an example of data formats:

Browser	Market share
Chrome	50
Internet Explorer	17.5

If you have more than one numerical field, the panel shows grouped bars.

Visualizing time series or multiple result sets

If you have multiple time series or tables, you first need to join them using a join, or reduce transform. For example, if you have multiple time series and you want to compare their last and max value, add the **Reduce** transform and specify **Max** and **Last** as options under **Calculations**.

Bar chart options

Use these options to refine your visualization.

Orientation

- **Auto** – Grafana decides the bar orientation based on the panel dimensions.
- **Horizontal** – Makes the X axis the category axis.
- **Vertical** – Makes the Y axis the category axis.

Rotate x-axis tick labels

When the graph is vertically oriented, this setting rotates the labels under the bars. This setting is useful when bar chart labels are long and overlap.

X-axis tick label maximum length

Sets the maximum length of bar chart labels. Labels longer than the maximum length are truncated with ellipses.

Bar labels minimum spacing

Sets the minimum spacing between bar labels.

Show values

Controls whether values are shown on top of or to the left of bars.

- **Auto** – Values are shown if there is space.
- **Always** – Always show values.
- **Never** – Never show values.

Stacking

Controls bar chart stacking.

- **Off** – Bars will not be stacked.
- **Normal** – Bars will be stacked on top of each other.
- **Percent** – Bars will be stacked on top of each other, and the height of each bar is the percentage of the total height of the stack.

Group width

Controls the width of groups.

- `0 = Minimum width`
- `1 = Maximum width`

Bar width

Controls the width of bars.

- `0 = Minimum width`
- `1 = Maximum width`

Bar radius

Controls the radius of the bars.

- `0 = Minimum radius`
- `0.5 = Maximum radius`

Highlight full area on hover

Controls if the entire surrounding area of the bar is highlighted when you hover over the bar.

Line width

Controls line width of the bars.

Fill opacity

Controls the fill opacity of the bars.

Gradient mode

Sets the mode of the gradient fill. Fill gradient is based on the line color. To change the color, use the standard color scheme field option.

Gradient appearance is influenced by the **Fill opacity** setting.

- **None** – no gradient fill. This is the default setting.
- **Opacity** – Transparency of the gradient is calculated based on the values on the y-axis. Opacity of the fill is increasing with the values on the Y-axis.
- **Hue** – Gradient color is generated based on the hue of the line color.

Tooltip mode

When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.

- **Single** – The hover tooltip shows only a single series, the one that you are hovering over on the visualization.
- **All** – The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden** – Do not display the tooltip when you interact with the visualization.

> **Note**
>
> You can use an override to hide individual series from the tooltip.

Text size

Enter a value to change the size of the text on your bar chart.

Legend options

Legend mode

Use these settings to define how the legend appears in your visualization. For more information, see Configure a legend.

- **List** – Displays the legend as a list. This is a default display mode of the legend.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement

Choose where to place the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend values

Choose series data values or standard calculations to show in the legend. You can have more than one. For more information, see Configure a legend.

Axis options

Use the following field settings to refine how your axes display. Some field options will not affect the visualization until you click outside of the field option box you are editing or press Enter.

Placement

Sets the placement of the Y-axis.

- **Auto** – Grafana automatically assigns Y-axis to the series. When there are two or more series with different units, then Grafana assigns the left axis to the first unit and right to the following units.
- **Left** – Display all Y-axes on the left side.
- **Right** – Display all Y-axes on the right side.
- **Hidden** – Hide all Y-axes.

To selectively hide axes, adding field overrides that targets specific fields.

Label

Set a Y-axis text label. If you have more than one Y-axis, then you can assign different labels with an override.

Width

Set a fixed width of the axis. By default, Grafana dynamically calculates the width of an axis.

By setting the width of the axis, data with different axes types can share the same display proportions. This makes it easier to compare more than one graph's worth of data because the axes are not shifted or stretched within visual proximity of each other.

Soft min and soft max

Set a soft min or soft max option for better control of Y-axis limits. By default, Grafana sets the range for the Y-axis automatically based on the dataset.

Soft min and soft max settings can prevent blips from turning into mountains when the data is mostly flat, and hard min or max derived from standard min and max field options can prevent intermittent spikes from flattening useful detail by clipping the spikes past a defined point.

You can set standard min/max options to define hard limits of the Y-axis. For more information, see Configure standard options.

Display multiple y-axes

In some cases, you may want to display multiple y-axes. For example, if you have a dataset showing both temperature and humidity over time, you may want to show two y-axes with different units for these two series.

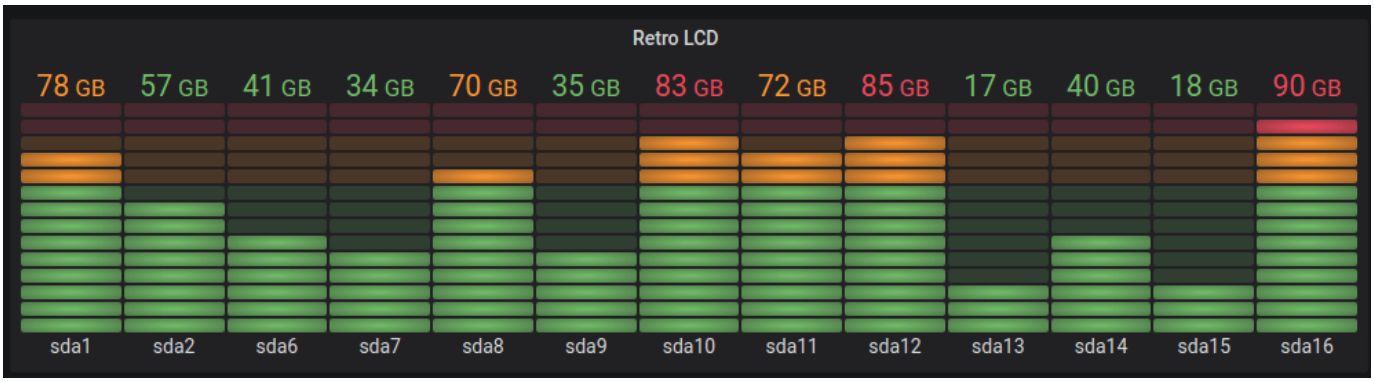
You can do this by adding field overrides. Follow the steps as many times as required to add as many y-axes as you need.

Bar gauge

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Bar gauges simplify your data by reducing every field to a single value. You choose how Grafana calculates the reduction.

This panel can show one or more bar gauges depending on how many series, rows, or columns your query returns. Bar gauge visualizations also support legends, improving readability when displaying multiple series.



Value options

Use the following options to refine how your visualization displays the value:

Show

Choose how Grafana displays your data.

Calculate

Show a calculated value based on all rows.

- **Calculation** – Select a reducer function that Grafana will use to reduce many fields to a single value. For a list of available calculations, refer to Calculation types.
- **Fields** – Select the fields display in the panel.

All values

Show a separate stat for every row. If you select this option, then you can also limit the number of rows to display.

- **Limit** – The maximum number of rows to display. Default is 5,000.
- **Fields** – Select the fields display in the panel.

Bar gauge options

Adjust how the bar gauge is displayed.

Orientation

Choose a stacking direction.

- **Auto** – Grafana selects what it thinks is the best orientation.
- **Horizontal** – Bars stretch horizontally, left to right.
- **Vertical** – Bars stretch vertically, bottom to top.

Display mode

Choose a display mode.

- **Gradient** – Threshold levels define a gradient.
- **Retro LCD** – The gauge is split into small cells that are lit or unlit.
- **Basic** – Single color based on the matching threshold.

Value display

Choose a value display mode.

- **Value color** – Value color is determined by value.
- **Text color** – Value color is the default text color.
- **Hidden** – Values are hidden.

Name placement

Choose a name placement mode.

> **Note**
>
> This option only applies when the orientation of the bar gauge is horizontal. When the bar gauge is in the vertical orientation, names are always placed at the bottom of each bar gauge.

- **Auto** – Grafana determines the best placement.
- **Top** – Names are placed on the top of each bar gauge.
- **Left** – Names are placed to the left of each bar gauge.

Show unfilled area

Select this if you want to render the unfilled region of the bars as dark gray. Not applicable to Retro LCD display mode.

Bar size

Choose a bar size mode.

- **Auto** – Grafana determines the best bar gauge size.
- **Manual** – Manually configure the bar gauge size.

Min width

Limit the minimum width of the bar column when the gauge is oriented vertically.

Automatically show x-axis scrollbar when there is a large amount of data.

> **Note**
>
> This option only applies when bar size is set to manual.

Min height

Limit the minimum height of the bar row when the gauge is oriented horizontally.

Automatically show y-axis scrollbar when there is a large amount of data.

> **Note**
>
> This option only applies when bar size is set to manual.

Max height

Limit the maximum height of the bar row when the gauge is oriented horizontally.

Automatically show y-axis scrollbar when there is a large amount of data.

> **Note**
>
> This option only applies when bar size is set to manual.

Candlestick

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The Candlestick visualization allows you to visualize data that includes a number of consistent dimensions focused on price movement. The Candlestick panel includes an Open-High-Low-Close (OHLC) mode, as well as support for additional dimensions based on time series data.



Candlestick visualizations build upon the foundation of the Time series and includes many common configuration settings.

Mode

The mode options allow you to toggle which dimensions are used for the visualization.

- **Candles** – Limit the panel dimensions to the open, high, low, and close dimensions used by candlestick visualizations.
- **Volume** – Limit the panel dimension to the volume dimension.
- **Both** – The default behavior for the candlestick panel. It includes both candlestick and volume visualizations.

Candle style

- **Candles** – The default display style, creates candle-style visualizations between the open and close dimensions.

- **OHLC Bars** – Display the four core dimensions open, high, low, and close values.

Color strategy

- **Since Open** – The default behavior. This mode will utilize the *Up* color (below) if the intra-period price movement is positive. In other words, if the value on close is greater or equal to the value on open, the *Up* color is used.
- **Since Prior Close** – An alternative display method where the color of the candle is based on the inter-period price movement or change in value. In other words, if the value on open is greater than the previous value on close, the *Up* color is used. If the value on open is lower than the previous value on close, the *Down* color is used. *This option also triggers the hollow candlestick visualization mode*. Hollow candlesticks indicate that the intra-period movement is positive (value is higher on close than on open), filled candlesticks indicate the intra-period change is negative (value is lower on close than on open). To learn more, see the explanation of the differences.

Up & down colors

The **Up color** and **Down color** options select which colors are used when the price movement is up or down. The *Color strategy* above will determine if intra-period or inter-period price movement is used to select the candle or OHLC bar color.

Open, high, low, close

The candlestick panel will attempt to map fields to the appropriate dimension.

- **Open** corresponds to the starting value of the given period.
- **High** corresponds to the highest value of the given period.
- **Low** corresponds to the lowest value of the given period.
- **Close** corresponds to the final (end) value of the given period.
- **Volume** corresponds to the sample count in the given period. (e.g. number of trades)

> **Note**
>
> The candlestick legend doesn't display these values.

To properly map these dimensions, the query results table from your data must include *at least* the following columns.

- `timestamp`
- `open`
- `high`
- `low`
- `close`

If your data can't be mapped to these dimensions for some reason (for example, because the column names aren't the same), you can map them manually using the **Open**, **High**, **Low**, and **Close** fields under the **Candlestick** options in the panel editor.

Additional fields

Additional fields

The candlestick panel is based on the time series visualization. It can visualize additional data dimensions beyond open, high, low, close, and volume. The **Include** and **Ignore** options allow it to visualize other included data such as simple moving averages, Bollinger bands and more, using the same styles and configurations available in the Time series.

Canvas

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Canvases combine the power of Grafana with the flexibility of custom elements. Canvases are extensible form-built panels that allow you to explicitly place elements within static and dynamic layouts. This empowers you to design custom visualizations and overlay data in ways that aren't possible with standard Grafana panels, all within Grafana's UI. If you've used popular UI and web design tools, then designing Canvas panels will feel very familiar.

Canvas panels support one-click data links and actions, dynamic connection directions, the ability to disable tooltips, and pan and zoom controls.

Elements

You can add these elements on your canvas. Adding multiple, and different kinds, of elements lets you customize a visualization in ways that are not possible with any other visualization.

Metric value

The metric value element lets you easily select the data you want to display on canvas. This element has a unique *edit* mode that can be triggered either through the context menu **Edit** option or by double-clicking panel. When in edit mode you can select which field data that you want to display.

Text

The text element lets you easily add text to the canvas. The element also supports an editing mode, triggered via either double-clicking or the edit menu option in the context menu.

Ellipse

The ellipse element lets you add a basic ellipse to the canvas. An ellipse element can display text (both fixed and field data) and its background color can be changed based on data thresholds.

Rectangle

The rectangle element lets you to add a basic rectangle to the canvas. A rectangle elements can display text (both fixed and field data) and its background color can be changed based on data thresholds.

Icon

The icon element lets you add a supported icon to the canvas. Icons can have their color set based on thresholds or value mappings.

Server

The server element lets you easily represent a single server, a stack of servers, a database, or a terminal. Server elements support status color, bulb color, and a bulb blink rate, all configurable by fixed or field values.

Button

The button element lets you add a basic button to the canvas. Button elements support triggering basic, unauthenticated API calls. API settings are found in the button element editor. You can also pass template variables in the API editor.

> **Note**
>
> Choosing a button will only trigger an API call when inline editing is disabled. See Canvas editing.

Connections

When building a canvas, you can connect elements together to create more complex visualizations. You can create connections by dragging from the connection anchor of one element to the connection anchor of another element. You can also create connections to the background of the canvas. Connection anchors are displayed when you hover over an element and inline editing is turned on. To remove a connection, select the connection and then press `Delete` or `Backspace`.

You can set both the size and color of connections based on fixed or field values. To do so, enter into panel edit mode, select the connection, and modify the connection's properties in the panel editor.

Canvas editing

Inline editor

You can edit your canvas inline while in the context of dashboard mode.

Pan and zoom

You and turn on panning and zooming in a canvas. This allows you to both create and navigate more complex designs.

> **Note**
>
> Pan and zoom is currently in preview by Grafana Labs. Support is limited, and breaking changes might occur before general availability.

Context menu

The context menu lets you perform common tasks quickly and efficiently. Supported functionality includes opening and closing the inline editor, duplicating an element, deleting an element, and more.

The context menu is triggered by a right click action (or equivalent) over the panel or a given canvas element.

When right clicking *the panel*, you are able to set a background image and easily add elements to the canvas.

When right clicking *an element*, you are able to edit, delete, and duplicate the element, or modify the element's layer positioning.

Canvas Options

Inline editing

The inline editing toggle lets you lock or unlock the canvas panel. When turned off, the canvas panel becomes *locked*, freezing elements in place and preventing unintended modifications.

Data links

Canvases support data links. You can create a data link for a metric-value element and display it for all elements that use the field name by following these steps.

To create a data link for an element

1. Set an element to be tied to a field value.
2. Turn off the inline editing toggle.
3. Create an override for **Fields with name** and select the element field name from the list.
4. Choose the **+ Add override property** button.
5. Select **Datalinks > Datalinks** from the list.
6. Choose **+ Add link**, add a title and URL for the data link.
7. Hover over the element to display the data link tooltip.
8. Choose the element to be able to open the data link.

If multiple elements use the same field name, and you want to control which elements display the data link, you can create a unique field name using the Add field from calculation transform. The alias you create in the transform will appear as a field you can use with an element.

Dashboard list

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Dashboard lists allow you to display dynamic links to other dashboards. The list can be configured to use starred dashboards, recently viewed dashboards, a search query, and dashboard tags.

On each dashboard load, this panel queries the dashboard list, always providing the most up-to-date results.

Options

Use these options to refine your visualization.

- **Include current time range** – Select this option to propagate the time range of the current dashboard to the dashboard links. When the user selects a link, the linked dashboard opens with the indicated time range already set.
- **Include current template variable values** – Select this option to include template variables currently used as query parameters in a link. When the user selects a link, any matching templates in the linked dashboard are set to the values from the list. For more information, see Dashboard URL variables.
- **Starred** – Display starred dashboards in alphabetical order.
- **Recently viewed** – Display recently viewed dashboards in alphabetical order.
- **Search** – Display dashboards by search query or tags. You must enter at least one value in **Query** or **Tags**. For the **Query** and **Tags** fields, variable interpolation is supported, for example, `$my_var` or `${my_var}`.
- **Show headings** – The chosen list selection (Starred, Recently viewed, Search) is shown as a heading.
- **Max items** – Sets the maximum number of items to list per section. For example, if you left this at the default value of 10 and displayed Starred and Recently viewed dashboards, then the panel would display up to 20 total dashboards, ten in each section.

Search

These options only apply if the **Search** option is selected.

- **Query** – Enter the query you want to search by. Queries are case-insensitive, and partial values are accepted.
- **Folder** – Select the dashboard folders that you want to display.
- **Tags** – Here is where you enter the tags you want to search by. Existing tags will not appear as you type, and they *are* case sensitive.

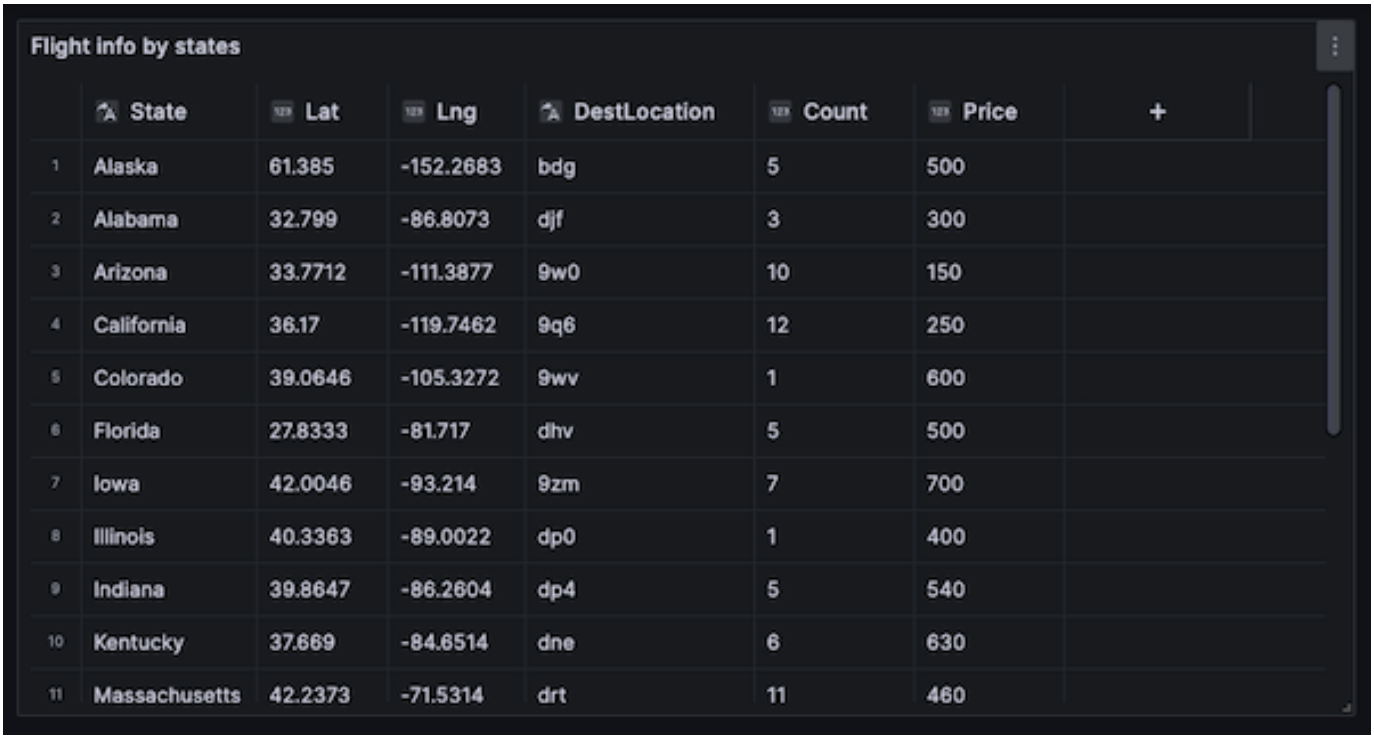
> **Note**
>
> When multiple tags and strings appear, the dashboard list displays those matching *all* conditions.

Datagrid

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

> **Note**
>
> The data grid visualization is currently in preview by Grafana Labs. Support is limited, and breaking changes might occur before general availability.

Datagrids offer you the ability to create, edit, and fine-tune data within Grafana. As such, this panel can act as a data source for other panels inside a dashboard.



	State	Lat	Lng	DestLocation	Count	Price
1	Alaska	61.385	-152.2683	bdg	5	500
2	Alabama	32.799	-86.8073	djf	3	300
3	Arizona	33.7712	-111.3877	9w0	10	150
4	California	36.17	-119.7462	9q6	12	250
5	Colorado	39.0646	-105.3272	9wv	1	600
6	Florida	27.8333	-81.717	dhv	5	500
7	Iowa	42.0046	-93.214	9zm	7	700
8	Illinois	40.3363	-89.0022	dp0	1	400
9	Indiana	39.8647	-86.2604	dp4	5	540
10	Kentucky	37.669	-84.6514	dne	6	630
11	Massachusetts	42.2373	-71.5314	drt	11	460

Through it, you can manipulate data queried from any data source, you can start from a blank slate, or you can pull data from a dragged and dropped file. You can then use the panel as a simple tabular visualization, or you can modify the data—and even remove it altogether—to create a blank slate.

Editing the dataset changes the data source to use the built-in **Grafana** data source, replacing the old data source settings and related queries, while also copying the current dataset into the dashboard model.

You can then use the panel as a data source for other panels, by using the built-in **Dashboard** data source to pull the datagrid data. This provides an interactive dashboard experience, where you can modify the data and see the changes reflected in other panels.

For more information about the **Grafana** and **Dashboard** data sources, see Special data sources.

Context menu

To provide a more streamlined experience, the datagrid has a context menu that can be accessed by right-clicking on a cell, column header, or row selector. Depending on the state of your datagrid, the context menu offers different options including the following.

- Delete or clear all rows and columns.
- Remove all existing data (rendering your datagrid blank).

- Trigger search functionality, which allows you to find keywords within the dataset.

Deleting a row or column will remove the data from the datagrid, while clearing a row or column will only remove the data from the cells, leaving the row or column intact.

Header menu

You can also access a header menu by choosing the dropdown icon next to the header title. From here, you can not only delete or clear a column, but also rename it, freeze it, or convert the field type of the column.

Selecting series

If there are multiple series, you can set the datagrid to display the preferred dataset using the **Select series** dropdown in the panel options.

Using datagrids

Datagrids offer various ways of interacting with your data. You can edit, move, clear, and remove rows and columns; use the built-in search functionality to find specific data; and convert field types or freeze horizontal scroll on a specific column.

Adding data

You can add data to a datagrid by creating a new column or row.

To add a new column

1. In an existing panel, choose the **+** button in the table header after the last column.
2. Add a name for the new column.
3. Select anywhere outside the field or press `Enter` to save the column.

Now you can add data in each cell.

To add a new row, choose a **+** button after the last row. The button is present in each cell after the last row, and choosing it triggers the creation of a new row while also activating the cell that you chose.

Editing data

You can move columns and rows as needed.

To move a column

1. Press and hold the header of the column that needs to be moved.
2. Drag the column to the desired location.
3. Release the column to finalize the move.

To move a row, select and hold the row selector from the number column situated on the far left side of the grid, and drag it to the desired location. Release the row to finalize the move.

Selecting multiple cells

You can select multiple cells by choosing a single cell, and dragging across others. This select can be used to copy the data from the selected cells or to delete them using the `Delete` key.

Deleting or clearing multiple rows or columns

To delete or clear multiple rows, you can do the following.

To delete or clear multiple rows or columns

1. Hover over the number column (to the left of the first column in the grid) to display row checkbox.
2. Select the checkboxes for the rows you want to work with. To select multiple consecutive rows, press and hold the `Shift` key while clicking the first and last row. To select non-consecutive rows, press and hold the `Ctrl` (or `Cmd`) key while clicking the desired rows.
3. Right-click (or equiavalent) to access the context menu.
4. Select **Delete rows** or **Clear rows**.

The same rules apply to columns by clicking the column headers.

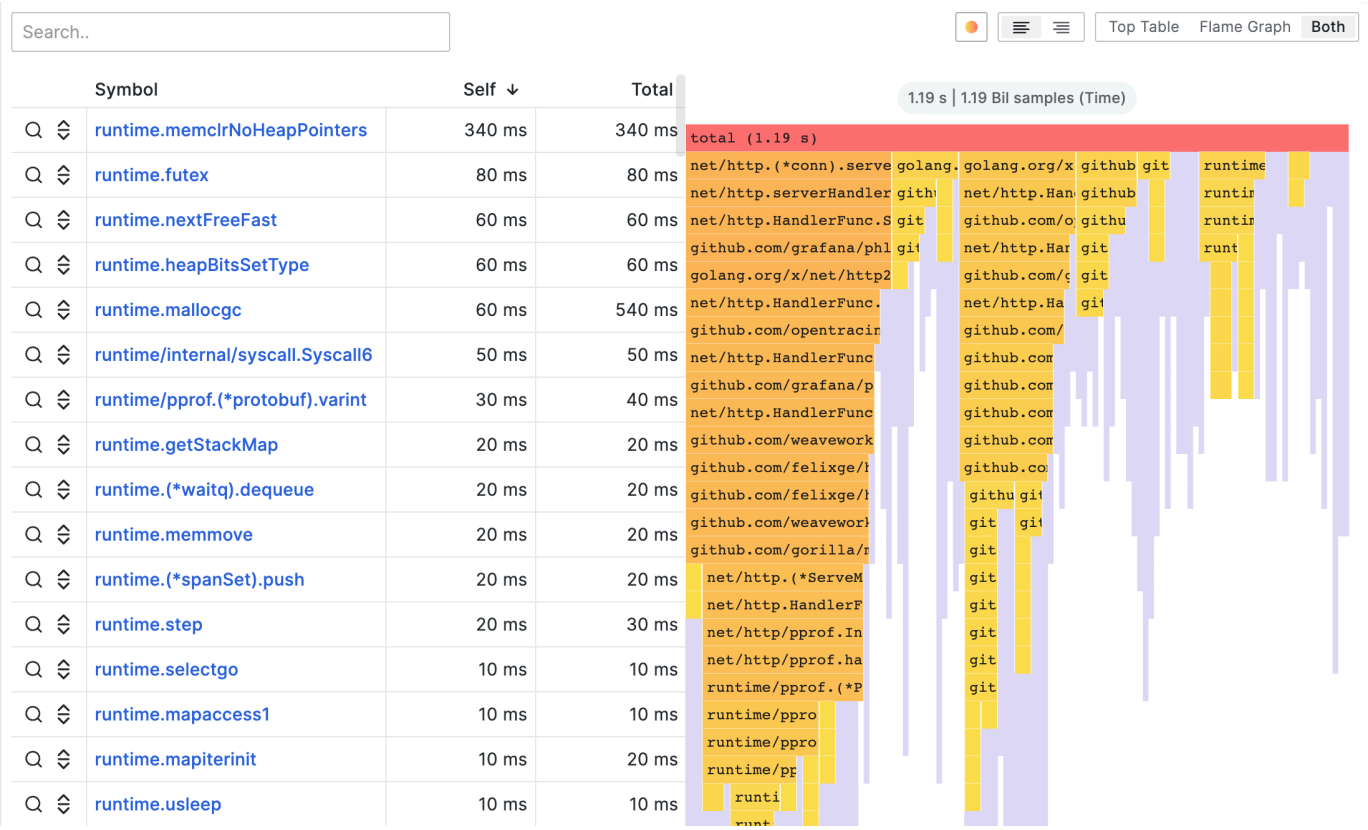
To delete all rows, use the **Select all** checkbox at the top left corner of the datagrid. This select all rows and allows you to delete them using the context menu.

Flame graph

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Flame graphs let you visualize profiling data. Using this visualization, the profile can be represented as a flame graph, table, or both.



Flame graph mode

A flame graph takes advantage of the hierarchical nature of profiling data. It condenses data into a format that allows you to easily see which code paths are consuming the most system resources, such as CPU time, allocated objects, or space when measuring memory. Each block in the flame graph represents a function call in a stack and its width represents its value.

Grayed-out sections are a set of functions that represent a relatively small value and they are collapsed together into one section for performance reasons.

You can hover over a specific function to view a tooltip that shows you additional data about that function, like the function's value, percentage of total value, and the number of samples with that function.

Drop-down actions

You can click a function to show a drop-down menu with additional actions:

- **Focus block** – When you choose **Focus block**, the block, or function, is set to 100% of the flame graph's width and all its child functions are shown with their widths updated relative to the width of the parent function. This makes it easier to drill down into smaller parts ofthe flame graph.
- **Copy function name** – When you choose **Copy function name**, the full name of the function that the block represents is copied.
- **Sandwich view** – The sandwich view allows you to show the context of the clicked function. It shows all the function's callers on the top and all the callees at the bottom. This shows the aggregated context of the function so if the function exists in multiple places in the flame graph, all the contexts are shown and aggregated in the sandwich view.

Status bar

The status bar shows metadata about the flame graph and currently applied modifications, like what part of the graph is in focus or what function is shown in sandwich view. Click the **X** in the status bar pill to remove that modification.

Toolbar

Search

You can use the search field to find functions with a particular name. All the functions in the flame graph that match the search will remain colored while the rest of the functions are grayed-out.

Color schema picker

You can switch between coloring functions by their value or by their package name to visually tie functions from the same package together.

Text align

Align text either to the left or to the right to show more important parts of the function name when it does not fit into the block.

Visualization picker

You can choose to show only the flame graph, only table, or both at the same time.

Top table mode

The top table shows the functions from the profile in table format. The table has three columns: symbols, self, and total. The table is sorted by self time by default, but can be reordered by total time or symbol name by clicking the column headers. Each row represents aggregated values for the given function if the function appears in multiple places in the profile.

There are also action buttons on the left for each row. The first button searches for the function name while second button shows the sandwich view of the function.

Data API

In order to render the flame graph, you must format the data frame data using a nested set model.

A nested set model ensures each item of the flame graph is encoded just by its nesting level as an integer value, its metadata, and by its order in the data frame. This means that the order of items is significant and needs to be correct. The ordering is a depth-first traversal of the items in the flame graph which recreates the graph without needing variable-length values in the data frame like in a children's array.

Required fields:

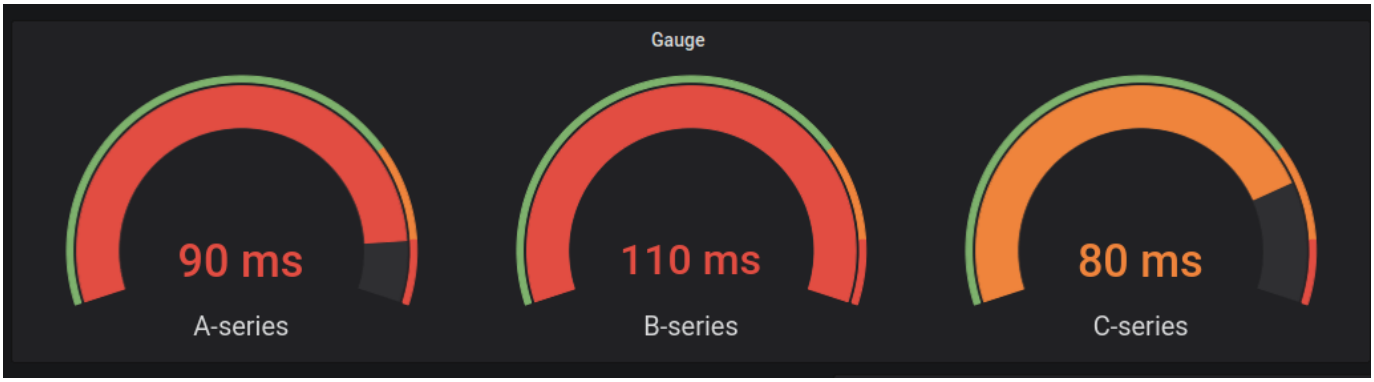
Field name	Type	Description
level	number	The nesting level of the item. In other words how many items are between this item and the top item of the flame graph.
value	number	The absolute or cumulativ e value of the item. This translates to the width of the item in the graph.
label	string	Label to be shown for the particular item.

Field name	Type	Description
self	number	Self value which is usually the cumulative value of the item minus the sum of cumulativ e values of its immediate children.

Gauge

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Gauges are single-value visualizations that can repeat a gauge for every series, column or row.



Value options

Use the following options to refine how your visualization displays the value:

Show

Choose how Grafana displays your data.

Calculate

Show a calculated value based on all rows.

- **Calculation** – Select a reducer function that Grafana will use to reduce many fields to a single value. For a list of available calculations, refer to Calculation types.
- **Fields** – Select the fields to display in the panel.

All values

Show a separate stat for every row. If you select this option, then you can also limit the number of rows to display.

- **Limit** – The maximum number of rows to display. Default is 5,000.
- **Fields** – Select the fields to display in the panel.

Gauge

Adjust how the gauge is displayed.

Orientation

Choose a stacking direction.

- **Auto** – Gauges display in rows and columns.
- **Horizontal** – Gauges display top to bottom.
- **Vertical** – Gauges display left to right.

Show threshold labels

Controls if threshold values are shown.

Show threshold markers

Controls if a threshold band is shown outside the inner gauge value band.

Gauge size

Choose a guage size mode

- **Auto** – Grafana determines the best gauge size.
- **Manual** – Manually configure the gauge size.

Min width

Set the minimum width of vertically-oriented gauges.

If you set a minimum width, the x-axis scrollbar is automatically displayed when there is a large amount of data.

> **Note**
>
> This option only applies when gauge size is set to manual.

Min height

Se the minimum height of horizontally-oriented gauges.

If you set a minimum height, the y-axis scrollbar is automatically displayed when there is a large amount of data.

> **Note**
>
> This option only applies when gauge size is set to manual.

Neutral

Set the starting value from which every gauge will be filled.

Text size

Adjust the sizes of the gauge text.

- **Title** – Enter a numeric value for the gauge title size.
- **Value** – Enter a numeric value for the gauge value size.

Geomap

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Geomaps allow you to view and customize the world map using geospatial data. You can configure various overlay styles and map view settings to easily focus on the important location-based characteristics of the data. Geomap visualizations support keyboard navigation for improved accessibility.

Note

You can add your own geospatial data on top of basemap layers provides by AWS. The basemap layers must all come from https://tiles.maps.search-services.aws.a2z.com.



Map View

The map view controls the initial view of the map when the dashboard loads.

Initial View

The initial view configures how the GeoMap panel renders when the panel is first loaded.

- **View** sets the center for the map when the panel first loads.
  - **Fit to data** fits the map view based on the data extents of Map layers and updates when data changes.

    - **Data** option allows selection of extent based on data from *All layers*, a single *Layer*, or the *Last value* from a selected layer.
    - **Layer** can be selected if fitting data from a single *Layer* or the *Last value* of a layer.
    - **Padding** sets padding in relative percent beyond data extent (not available when looking at *Last value* only).
    - **Max Zoom** sets the maximum zoom level when fitting data.
- **Coordinates** sets the map view based on:
    - **Latitude**
    - **Longitude**
- Default views are also available including:
    - **(0°, 0°)**
    - **North America**
    - **South America**
    - **Europe**
    - **Africa**
    - **West Asia**
    - **South Asia**
    - **South-East Asia**
    - **East Asia**
    - **Australia**
    - **Oceania**
- **Zoom** sets the initial zoom level.

Map layers

Geomaps support showing multiple layers. Each layer determines how you visualize geospatial data on top of the base map.

Types

There are three map layer types to choose from in the Geomap visualization.

- Markers layer renders a marker at each data point.
- Heatmap layer visualizes a heatmap of the data.

- GeoJSON layer renders static data from a GeoJSON file.
- Night / Day layer (Alpha) renders a night or day region.
- Route layer (preview) render data points as a route.
- Photos layer (preview) renders a photo at each data point.
- Network layer (preview) visualizes a network graph from the data.

There are also two experimental (or alpha) layer types.

- **Icon at last point (alpha)** renders an icon at the last data point.
- **Dynamic GeoJSON (alpha)** styles a GeoJSON file based on query results.

> **Note**
>
> Layers marked *preview* or *alpha* in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

Layer Controls

The layer controls allow you to create layers, change their name, reorder and delete layers.

- **Add layer** creates an additional, configurable data layer for the geomap. When you add a layer, you are prompted to select a layer type. You can change the layer type at any point during panel configuration.
- The layer controls allow you to rename, delete, and reorder the layers of the panel.
  - **Edit layer name** (pencil icon) renames the layer.
  - **Trash Bin** deletes the layer.
  - **Reorder** (six dots/grab handle) allows you to change the layer order. Data on higher layers will appear above data on lower layers. The visualization will update the layer order as you drag and drop to help simplify choosing a layer order.

You can add multiple layers of data to a single geomap panel in order to create rich, detailed visualizations.

Location

Geomaps needs a source of geographical data. This data comes from a database query, and there are four mapping options for your data.

- **Auto** automatically searches for location data. Use this option when your query is based on one of the following names for data fields.
  - *geohash*: `geohash`
  - *latitude*: `latitude`, `lat`
  - *longitude*: `longitude`, `lng`, `lon`
  - *lookup*: `lookup`
- **Coords** specifies that your query holds coordinate data. You will getprompted to select numeric data fields for latitude and longitude from your database query.
- **Geohash** specifies that your query holds geohash data. You will be prompted to select a string data field for the geohash from your database query.
- **Lookup** specifies that your query holds location name data that needs to be mapped to a value. You will be prompted to select the lookup field from your database query and a gazetteer. The gazetteer is the directory that is used to map your queried data to a geographical point.

Markers layer

The markers layer allows you to display data points as different marker shapes such as circles, squares, triangles, stars, and more.

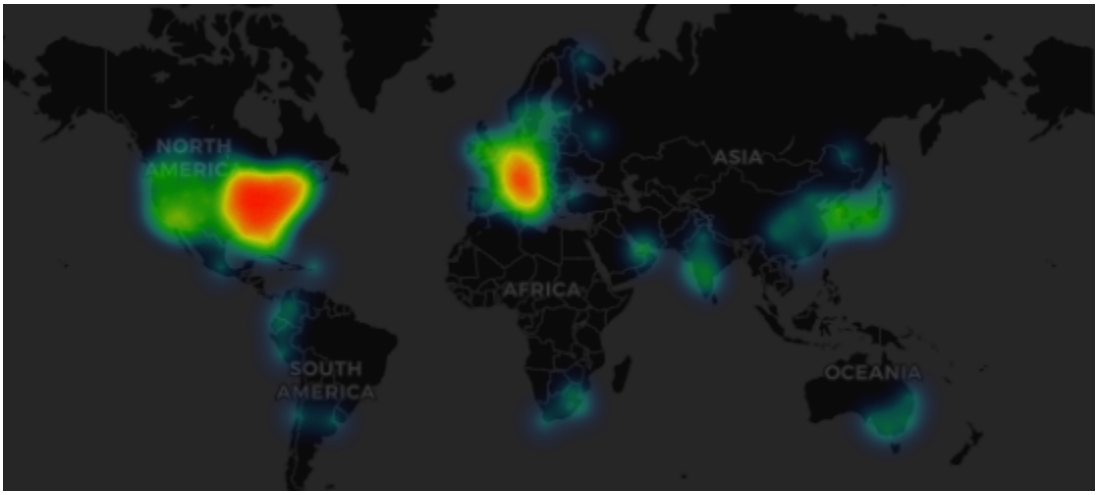
Markers have many customization options.

- **Size** configures the size of the markers. The default is `Fixed size`, which makes all marker sizes the same regardless of the data; however, there is also an option to size the markers based on the data corresponding to selected field. `Min` and `Max` marker size has to be set such that the Marker layer can scale within this range.
- **Symbol** allows you to choose the symbol, icon, or graphic to aid in providing additional visual context to your data. Choose from assets that are included with Grafana such as simple symbols or the Unicon library. You can also specify a URL containing an image asset. The image must be a scalable vector graphic (SVG).
- **Symbol Vertical Align** configures the vertical alignment of the symbol relative to the data point. Note that the symbol's rotation angle is applies first around the data point, then the vertical alignment is applied relative to the rotation of the symbol.

- **Symbol Horizontal Align** configures the horizontal alignment of the symbol relative to the data point. Note that the symbol's rotation angle is applies first around the data point, then the horizontal alignment is applied relative to the rotation of the symbol.
- **Color** configures the color of the markers. The default `Fixed color` sets all markers to a specific color. There is also an option to have conditional colors depending on the selected field data point values and the color scheme set at the **Standard options** section.
- **Fill opacity** configures the transparency of each marker.
- **Rotation angle** configures the rotation angle of each marker. The default is **Fixed value**, which makes all markers rotate to the same angle regardless of the data; however, there is also an option to set the rotation of the markers based on data corresponding to a selected field.
- **Text label** configures a text label for each marker.
- **Show legend** lets you toggle the legend for the layers.
- **Display tooltip** lets you toggle tooltips for the layer.

Heatmap layer

The heatmap layer clusters various data points to visualize locations with different densities.



To add a heatmap layer:

Choose on the dropdown menu under Data Layer and select `Heatmap`.

Similar to `Markers`, you are prompted with options to determine which data points to visualize and how you want to visualize them.

- **Weight values** configure the intensity of the heatmap clusters. **Fixed value** keeps a constant weight value throughout all data points. This value should be in the range of 0-1. Similar to Markers, there is an alternate option in the dropdown to automatically scale the weight values depending on data values.
- **Radius** configures the size of the heatmap clusters.
- **Blur** configures the amount of blur on each cluster.
- **Opacity** configures the opacity of each cluster.
- **Display tooltip** lets you toggle tooltips for the layer.

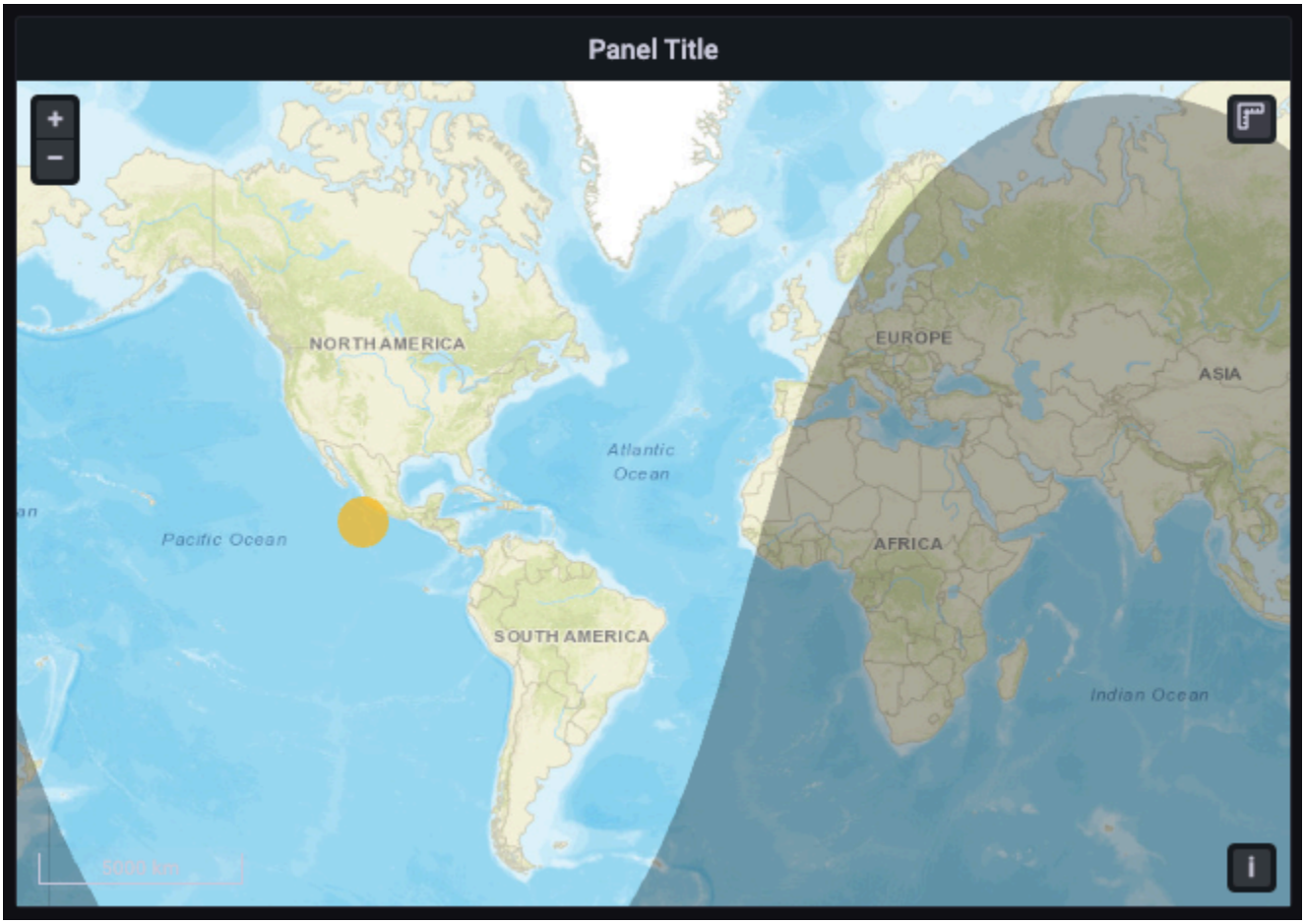
GeoJSON layer

The GeoJSON layer allows you to select and load a static GeoJSON file from the filesystem.

- **GeoJSON URL** provides a choice of GeoJSON files that ship with Grafana.
- **Default Style** controls which styles to apply when no rules above match.
  - **Color** configures the color of the default style
  - **Opacity** configures the default opacity
- **Style Rules** apply styles based on feature properties
  - **Rule** allows you to select a *feature*, *condition*, and *value* from the GeoJSON file in order to define a rule. The trash bin icon can be used to delete the current rule.
  - **Color** configures the color of the style for the current rule
  - **Opacity** configures the transparency level for the current rule.
- **Add style rule** creates additional style rules.
- **Display tooltip** lets you toggle tooltips for the layer.

Night / Day layer (Alpha)

The Night / Day layer displays night and day regions based on the current time range.



Options

- **Show** toggles the time source from panel time range.
- **Night region color** picks the color for the night region.
- **Display sun** toggles sun icon.
- **Opacity** from 0 (transparent) to 1 (opaque).
- **Display tooltip** lets you toggle tooltips for the layer.

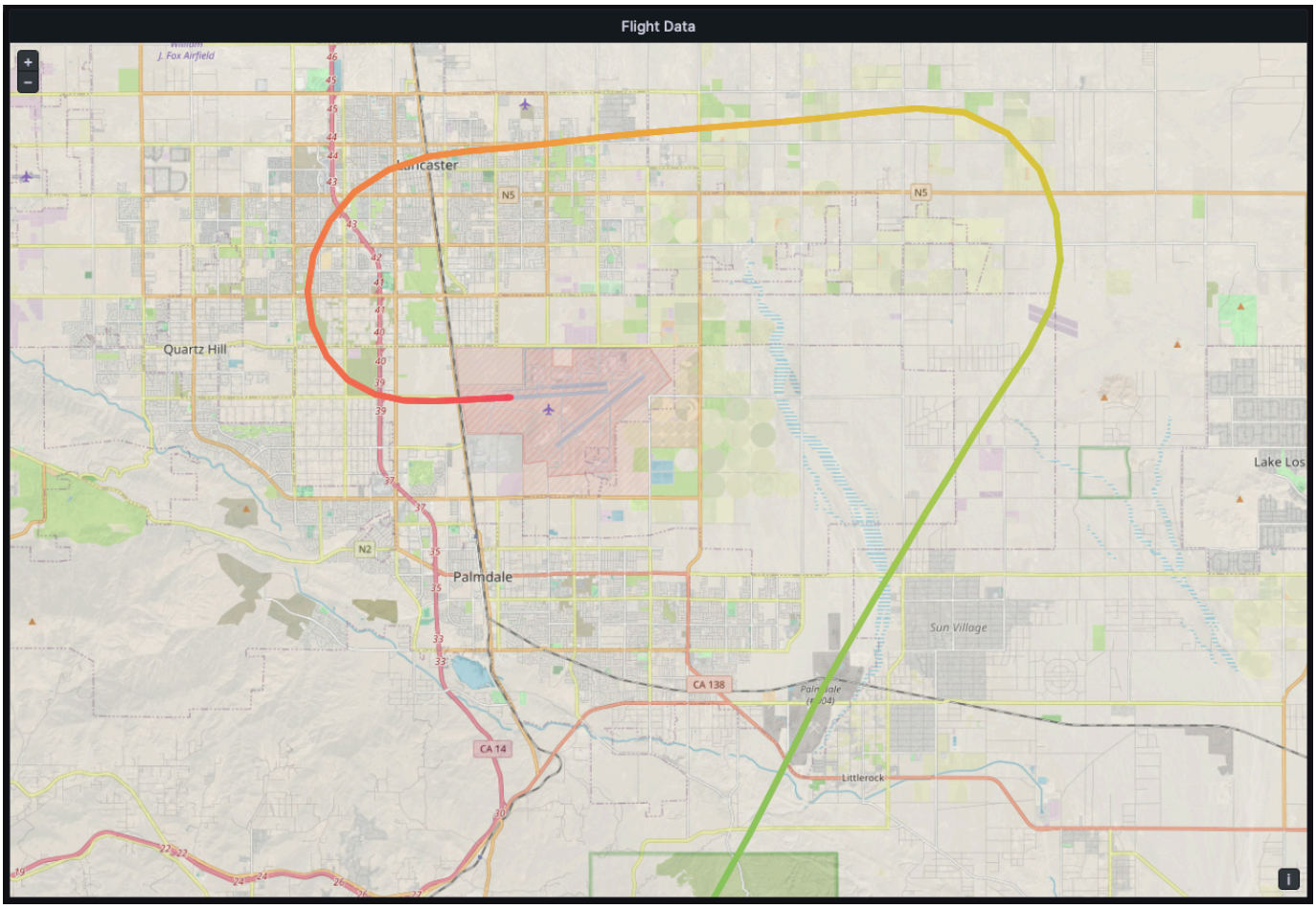
> **Note**
>
> For more information, see Extensions for OpenLayers - DayNight.

Route layer (preview)

The Route layer renders data points as a route.

Note

The Route layer is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.



Options

- **Size** sets the route thickness. Fixed value by default. When field data is selected, you can set the Min and Max range in which field data can scale.
- **Color** sets the route color. Set to fixed color by default, you can also tie the color to field data.
- **Fill opacity** configures the opacity of the route.
- **Text label** configures a text label for each route.
- **Arrow** configures the arrow styling to display along the route, in order of data.
  - **None**

- **Forward**
- **Reverse**

- **Display tooltip** lets you toggle tooltips for the layer.

> **Note**
>
> For more information, see Extensions for OpenLayers - Flow Line Style.

Photos layer (preview)

The Photos layer renders a photo at each data point.

> **Note**
>
> The Photos layer is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

Options

- **Image Source Field** lets you select a string field containing image data as a Base64 encoded image binary (`...`).
- **Kind** sets the frame style around the images. Choose from:
  - **Square**
  - **Circle**
  - **Anchored**
  - **Folio**
- **Crop** toggles whether the images are cropped to fit.
- **Shadow** toggles a box shadow behind the images.
- **Border** sets the border size around images.
- **Border color** sets the border color around images.
- **Radius** sets the overall size of images in pixels.
- **Display tooltip** lets you toggle tooltips for the layer.

> **Note**
>
> For more information, see Extensions for OpenLayers - Image Photo Style.

Network layer (preview)

The Network layer renders a network graph. This layers supports the same data format supported by the node graph visualization, with the addition of geospatial data included in the nodes data. The geospatial data is used to locate and render the nodes on the map.

> **Note**
>
> The Network layer is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.



> **Note**
>
> The Network layer is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

Options

- **Arrow** sets the arrow direction to display for each edge, with forward meaning source to target. Choose from:
  - **None**
  - **Forward**
  - **Reverse**
  - **Both**
- **Show legend** lets you toggle the legend for the layer. The legend only supports node data.
- **Display tooltip** lets you toggle tooltips for the layer.

Node styles

- **Size** configures the size of the nodes. The default is **Fixed size**, which makes all node sizes the same regardless of the data; however, there is also an option to size the nodes based on data corresponding to a selected field. **Min** and **Max** node sizes have to be set such that the nodes can scale within this range.
- **Color** configures the color of the nodes. The default is **Fixed color**, which sets all nodes to a specific color. There is also an option to have conditional colors depending on the selected field data point values and the color scheme set in the **Standard options** section.
- **Symbol** lets you choose the symbol, icon, or graphic to aid in providing additional visual context to your data. Choose from assets that are included with Grafana, such as simple symbols or the Unicon library. You can also specify a URL containing an image asset. The image must be a scalable vector graphic (SVG).
- **Fill opacity** configures the transparency of each node.
- **Rotation angle** configures the rotation angle of each node. The default is **Fixed value**, which makes all nodes rotate to the same angle regardless of the data; however, there is also an option to set the rotation of the nodes based on data corresponding to a selected field.
- **Text label** configures a text label for each node.

Edge styles

- **Size** configures the line width of the edges. The default is **Fixed size**, which makes all edge line widths the same regardless of the data; however, there is also an option to size the edges based

on data corresponding to a selected field. **Min** and **Max** edge sizes have to be set such that the edges can scale within this range.

- **Color** configures the color of the edges. The default is **Fixed color**, which sets all edges to a specific color. There is also an option to have conditional colors depending on the selected field data point values and the color scheme set in the **Standard options** section.
- **Fill opacity** configures the transparency of each edge.
- **Text label** configures a text label for each edge.

CARTO layer

CARTO layers are not supported in Amazon Managed Grafana.

XYZ tile layer

XYZ tile layers are not supported in Amazon Managed Grafana.

Open Street Map layer

Open Street Map layers (other than the default basedmap) are not supported in Amazon Managed Grafana.

ArcGIS layer

ArcGIS layers are not supported in Amazon Managed Grafana.

Map Controls

The map controls section contains various options for map information and tool overlays.

Zoom

Options for zoom controls.

Show zoom control

Displays zoom controls in the upper left corner.

Mouse wheel zoom

Turns on or off using the mouse wheel for zooming in or out.

Show attribution

Displays attribution for basemap layers on the map.

Show scale

Displays scale information in the bottom left corner.

> **Note**
>
> Displays units in [m]/[km].

Show measure tools

Displays measure tools in the upper right corner. Measurements appear only when this control is open.

- **Click** to start measuring
- **Continue clicking** to continue measurement
- **Double-click** to end measurement

> **Note**
>
> When you change measurement type or units, the previous measurement is removed from the map.
> If the control is closed and then re-opened, the most recent measurement is displayed.
> A measurement can be modified by clicking and dragging on it.

Length

Get the spherical length of a geometry. This length is the sum of the great circle distances between coordinates. For multi-part geometries, the length is the sum of the length of each part. Geometries are assumed to be in 'EPSG:3857'.

You can choose the following units for length measurements:

- **Metric (m/km)**
- **Feet (ft)**
- **Miles (mi)**

- **Nautical miles (nmi)**

Area

Get the spherical area of a geometry. This area is calculated assuming that polygon edges are segments of great circles on a sphere. Geometries are assumed to be in 'EPSG:3857'.

You can choose the following units for area measurements:

- **Square Meters (m^2)**
- **Square Kilometers (km^2)**
- **Square Feet (ft^2)**
- **Square Miles (mi^2)**
- **Acres (acre)**
- **Hectare (ha)**

Show debug

Displays debug information in the upper right corner of the map. This can be useful for debugging or validating a data source.

- **Zoom** displays the current zoom level of the map.
- **Center** displays the current **longitude**, and **latitude** of the map center.

Tooltip

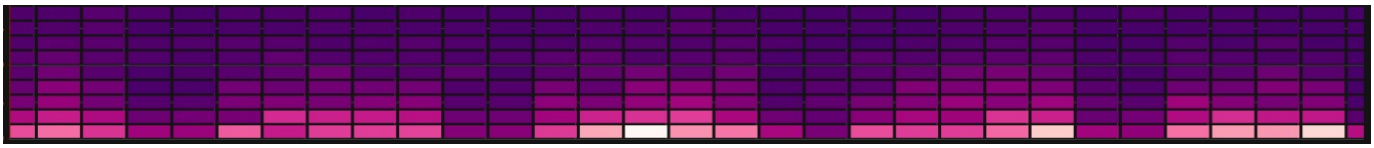
- **None** displays tooltips only when a data point is clicked.
- **Details** displays tooltips when a pointer hovers over a data point.

Heatmap

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The Heatmap panel visualization allows you to view histograms over time. For more information about histograms, refer to Introduction to histograms and heatmaps.



Calculate from data

This setting determines if the data is already a calculated heatmap (from the data source/transformer), or one that should be calculated in the panel.

X Bucket

This setting determines how the X-axis is split into buckets. You can specify a time interval in the **Size** input. For example, a time range of 1h makes the cells 1-hour wide on the X-axis.

Y Bucket

This setting determines how the Y-axis is split into buckets.

Y Bucket scale

Select one of the following Y-axis value scales:

- **linear** – Linear scale.
- **log (base 2)** – Logarithmic scale with base 2.
- **log (base 10)** – Logarithmic scale with base 10.
- **symlog** – Symlog scale.

Y Axes

Defines how the Y axis is displayed

Placement

- **Left** – On the left

- **Right** – On the right
- **Hidden** – Hidden

Unit

Unit configuration

Decimals

This setting determines decimal configuration.

Min/Max value

This setting configures the axis range.

Reverse

When selected, the axis appears in reverse order.

Display multiple y-axes

In some cases, you may want to display multiple y-axes. For example, if you have a dataset showing both temperature and humidity over time, you may want to show two y-axes with different units for these two series.

You can do this by adding field overrides. Follow the steps as many times as required to add as many y-axes as you need.

Colors

The color spectrum controls the mapping between value count (in each bucket) and the color assigned to each bucket. The leftmost color on the spectrum represents the minimum count and the color on the right most side represents the maximum count. Some color schemes are automatically inverted when using the light theme.

You can also change the color mode to Opacity. In this case, the color will not change but the amount of opacity will change with the bucket count

- **Mode**
  - **Scheme** – Bucket value represented by cell color.
    - **Scheme** – If the mode is **Scheme**, then select a color scheme.

- **opacity** – Bucket value represented by cell opacity. Opaque cell means maximum value.
  - **Color** – Cell base color.
  - **Scale** – Scale for mapping bucket values to the opacity.
    - **linear** – Linear scale. Bucket value maps linearly to the opacity.
    - **sqrt** – Power scale. Cell opacity calculated as `value ^ k`, where `k` is a configured **Exponent** value. If exponent is less than 1, you will get a logarithmic scale. If exponent is greater than 1, you will get an exponential scale. In case of 1, scale will be the same as linear.
  - **Exponent** – value of the exponent, greater than `0`.

Start/end color from value

By default, Grafana calculates cell colors based on minimum and maximum bucket values. With Min and Max you can overwrite those values. Consider a bucket value as a Z-axis and Min and Max as Z-Min and Z-Max, respectively.

- **Start** – Minimum value using for cell color calculation. If the bucket value is less than Min, then it is mapped to the “minimum” color. The series min value is the default value.
- **End** – Maximum value using for cell color calculation. If the bucket value is greater than Max, then it is mapped to the “maximum” color. The series max value is the default value.

Cell display

Use these settings to refine your visualization.

Additional display options

Tooltip

- **Show tooltip** – Show heatmap tooltip.
- **Show Histogram** – Show a Y-axis histogram on the tooltip. A histogram represents the distribution of the bucket values for a specific timestamp.
- **Show color scale** – Show a color scale on the tooltip. The color scale represents the mapping between bucket value and color.

Legend

Choose whether you want to display the heatmap legend on the visualization.

Exemplars

Set the color used to show exemplar data.

Histogram

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The histogram visualization calculates the distribution of values and presents them as a bar chart. The Y-axis and the height of each bar represent the count of values that fall into each bracket while the X-axis represents the value range.



Supported formats

Histogram visualization supports time series and any table results with one or more numerical fields.

Display options

Use these options to refine your visualizations.

Bucket size

The size of the buckets. Leave this empty for automatic bucket sizing (~10% of the full range).

Bucket offset

If the first bucket should not start at zero. A non-zero offset shifts the aggregation window. For example, 5-sized buckets that are 0–5, 5–10, 10–15 with a default 0 offset would become 2–7, 7–12, 12–17 with an offset of 2; offsets of 0, 5, or 10, in this case, would effectively do nothing. Typically, this option would be used with an explicitly defined bucket size rather than automatic. For this setting to affect, the offset amount should be greater than 0 and less than the bucket size; values outside this range will have the same effect as values within this range.

Combine series

This will merge all series and fields into a combined histogram.

Line width

Controls line width of the bars.

Fill opacity

Controls the fill opacity of the bars.

Gradient mode

Set the mode of the gradient fill. Fill gradient is based on the line color. To change the color, use the standard color scheme field option. Gradient appearance is influenced by the **Fill opacity** setting.

- **None** – No gradient fill, this is the default setting.
- **Opacity** – Transparency of the gradient is calculated based on the values on the Y-axis. Opacity of the fill is increasing with the values on the Y-axis.
- **Hue** – Gradient color is generated based on the hue of the line color.

Legend options

When the legend option is enabled, it can show either the value mappings or the threshold brackets. To show the value mappings in the legend, it's important that the Color scheme option under standard options is set to Single color or Classic palette. To see the threshold brackets in the legend, set the Color scheme to From thresholds. For more information about the legend, see Configure a legend.

Legend mode

Use these settings to define how the legend appears in your visualization.

- **List** – Displays the legend as a list. This is a default display mode of the legend.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement

Choose where to place the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend Values

Choose which of the standard calculations to show in the legend. You can have more than one. For more information, see Calculation types.

Legend calculations

Choose which calculations to show in the legend. You can select more than one.

Logs

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The logs panel visualization shows log lines from data sources that support logs, such as Elastic, Influx, and Loki. Typically, you would use this panel next to a graph panel to display the log output of a related process.

The logs panel shows the result of queries that were entered on the **Query** tab. The results of multiple queries are merged and sorted by time. You can scroll inside the panel if the data source returns more lines than can be displayed.

To limit the number of lines rendered, you can use the **Max data points** setting in the **Query options**. If it is not set, the data source will usually enforce a default limit.

Log level

For logs where a **level** label is specified, we use the value of the label to determine the log level and update color accordingly. If the log doesn't have a level label specified, we try to find out if its content matches any of the supported expressions (see below for more information). The log level is always determined by the first match. In case Grafana is not able to determine a log level, it will be visualized with **unknown** log level. For more information, see Log level.

Log details

Each log row has an extendable area with its labels and detected fields, for more robust interaction. Each field or label has a stats icon to display statistics in relation to all displayed logs.

Data links

By using data links, you can turn any part of a log message into an internal or external link. The created link is visible as a button in the **Links** section inside the **Log details** view.

Display options

Use the following settings to refine your visualization:

- **Time** – Show or hide the time column. This is the timestamp associated with the log line as reported from the data source.
- **Unique labels** – Show or hide the unique labels column, which shows only non-common labels.
- **Common labels** – Show or hide the common labels
- **Wrap lines** – Toggle line wrapping.

- **Prettify JSON** – Set this to `true` to pretty print all JSON logs. This setting does not affect logs in any format other than JSON.
- **Enable log details** – Toggle option to see the log details view for each log row. The default setting is `true`.
- **Order** – Display results in descending or ascending time order. The default is **Descending**, showing the newest logs first. Set to **Ascending** to show the oldest log lines first.

News

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The news visualization displays an RSS feed. By default, it displays articles from the Grafana Labs blog, and users can change this by entering a different RSS feed URL.

Enter the URL of an RSS in the **Display** section. This visualization type does not accept any other queries, and users should not expect to be able to ilter or query the RSS feed data in any way using this visualization.

Note

RSS feeds are loaded by the Grafana front end without a proxy. As a result, only RSS feeds that are configured with the appropriate CORS headers will load. If the RSS feed you're trying to display fails to load, consider re-hosting the RSS feed or creating your own proxy.

Node graph

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Node graphs can visualize directed graphs or networks. They use a directed force layout to effectively position the nodes, so they can help with displaying complex infrastructure maps, hierarchies, or run diagrams.

Data requirements

A node graph requires a specific shape of the data to be able to display its nodes and edges. Not every data source or query can be visualized with this graph. If you want to use this as a data source developer, see the section about data API.

A node graph consists of *nodes* and *edges*.

- A *node* is displayed as a circle. A node might represent an application, a service, or anything else that is relevant from an application perspective.
- An *edge* is displayed as a line that connects two nodes. The connection might be a request, an operation, or some other relationship between the two nodes.

Both nodes and edges can have associated metadata or statistics. The data source defines what information and values is shown, so different data sources can show different type of values or not show some values.

Nodes

Usually, nodes show two statistical values inside the node and two identifiers just below the node, usually name and type. Nodes can also show another set of values as a color circle around the node, with sections of different color representing different values that should add up to 1. For example, you can have the percentage of errors represented by red portion of the circle.

Additional details can be displayed in a context menu, which is displayed when you choose the node. There also can be additional links in the context menu that can target either other parts of the Grafana workspace or any external link.

Note

Node graph can show only 1,500 nodes. If this limit is crossed, a warning is visible in the upper right corner, and some nodes will be hidden. You can expand hidden parts of the graph by clicking on the **Hidden nodes** markers in the graph.

Edges

Edges can also show statistics when you hover over the edge. Similar to nodes, you can open a context menu with additional details and links by choosing the edge.

The first data source supporting this visualization is the AWS X-Ray data source for its service map feature. For more information, see Connect to an AWS X-Ray data source.

Navigating the node graph

Pan

You can pan within the node graph by choosing outside of any node or edge and dragging the pointer.

Zoom in or out

You can zoom by using the buttons on the upper left corner of the node graph, or use a mouse wheel or other scroll input with the `Ctrl` (or `Cmd`) key.

Explore hidden nodes

The number of nodes shown at a given time is limited to maintain a reasonable performance. Nodes that are outside this limit are hidden behind selectable markers that show an approximate number of hidden nodes that are connected to that edge. You can choose the marker to expand the graph around that node.

Grid view

You can switch to the grid view to have a better overview of the most interesting nodes in the graph. Grid view shows nodes in a grid without edges and can be sorted by stats shown inside the node or by stats represented by the a colored border of the nodes.

To sort the nodes, choose the stats inside the legend. The marker next to the stat name (either # or #) shows which stat is currently used for sorting and sorting direction.

Choose a node and then select the **Show in Graph layout** option to switch back to graph layout with focus on the selected node, to show it in context of the full graph.

Data API

This visualization needs a specific shape of the data to be returned from the data source in order to correctly display it.

Node Graph at minimum requires a data frame describing the edges of the graph. By default, node graph will compute the nodes and any stats based on this data frame. Optionally a second data frame describing the nodes can be sent in case there is need to show more node specific metadata. You have to set `frame.meta.preferredVisualisationType = 'nodeGraph'` on both data frames or name them `nodes` and `edges` respectively for the node graph to render.

Edges data from structure

Required fields:

Field name	Type	Description
id	string	Unique identifier of the edge.
source	string	Id of the source node.
target	string	Id of the target.

Optional fields:

Field name	Type	Description
mainstat	string/number	First stat shown in the overlay when hovering over the edge. It can be a string showing the value as is or it can be a number. If it is a number, any unit associated with that field is also shown.

Field name	Type	Description
secondarystat	string/number	Same as mainStat, but shown right under it.
detail__*	string/number	Any field prefixed with `detail__` will be shown in the header of context menu when clicked on the edge. Use `config.displayName` for a more human readable label.

Nodes data from structure

Required fields:

Field name	Type	Description
id	string	Unique identifier of the node. This ID is referenced by edge in its source and target field.

Optional fields:

Field name	Type	Description
title	string	Name of the node visible just under the node.
subtitle	string	Additional, name, type or other identifier shown under the title.
mainstat	string/number	First stat shown inside the node itself. It can either be a string showing the value as is or a number. If it is a number,

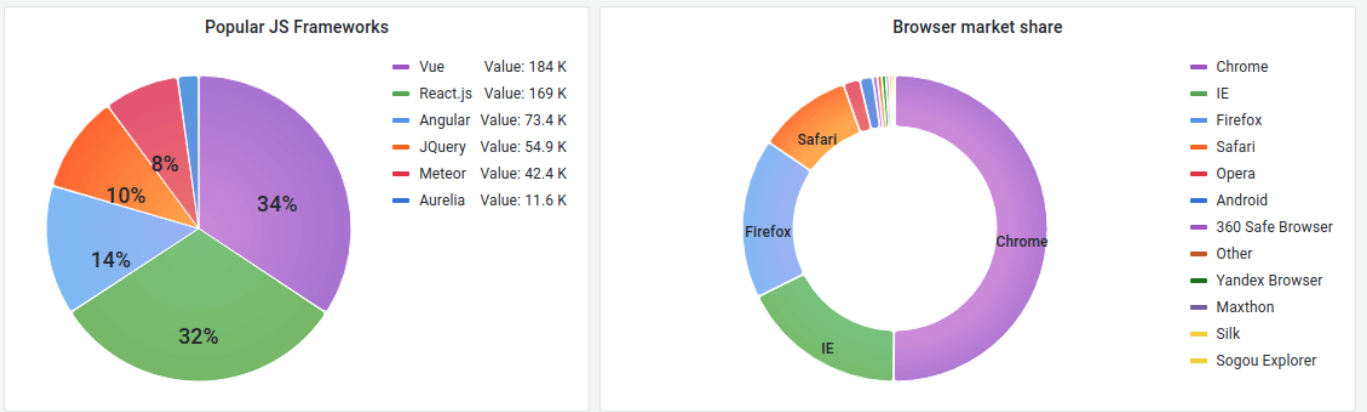
Field name	Type	Description
		any unit associated with that field is also shown.
secondarystat	string/number	Same as mainStat, but shown under it inside the node.
arc__*	number	Any field prefixed with `arc__` will be used to create the color circle around the node. All values in these fields should add up to 1. You can specify color using `config.color.fixedColor` .
detail__*	string/number	Any field prefixed with `detail__` will be shown in the header of context menu when clicked on the node. Use `config.displayName` for more human readable label.
color	string/number	Can be used to specify a single color instead of using the `arc__` fields to specify color sections. It can either be a string (it must be an acceptable HTML color string), or it can be a number, in which case the behavior depends on the `field.config.color.mode` setting. This can be used, for example, to create gradient colors controlled by a field value.

Field name	Type	Description
icon	string	Name of the icon to show inside the node instead of the default stats. Only Grafana built in icons are allowed (see the available icons here).
nodeRadius	number	Radius value in pixels. Used to manage node size.
highlighted	Boolean	Sets whether the node should be highlighted. Use, for example, to represent a specific path in the graph by highlighting several nodes and edges. Defaults to `false`.

Pie chart



This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.



The pie chart displays reduced series, or values in a series, from one or more queries, as they relate to each other, in the form of slices of a pie. The arc length, area and central angle of a slice are all proportional to the slices value, as it relates to the sum of all values. This type of chart is best used when you want a quick comparison of a small set of values in an aesthetically pleasing form.

Value options

Use the following options to refine the value in your visualization.

Show

Choose how much information to show.

- **Calculate** – Reduces each value to a single value per series.
- **All values** – Displays every value from a single series.

Calculation

Select a calculation to reduce each series when **Calculate** has been selected. For information about available calculations, refer to Calculation types.

Limit

When displaying every value from a single series, this limits the number of values displayed.

Fields

Select at least one field to display in the visualization. Each field name is available on the list, or you can select one of the following options:

- **Numeric fields** – All fields with numerical values.

- **All fields** – All fields that are not removed by transformations.
- **Time** – All fields with time values.

Pie chart options

Use these options to refine how your visualization looks.

Pie chart type

Select the pie chart display style. Can be either:

- **Pie** – A standard pie chart
- **Donut** – A pie chart with a hole in the middle

Labels

Select labels to display on the pie chart. You can select more than one.

- **Name** – The series or field name.
- **Percent** – The percentage of the whole.
- **Value** – The raw numerical value.

Labels are displayed in white over the body of the chart by default. You can select darker chart colors to make them more visible. Long names or numbers might be clipped.

Tooltip mode

When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.

- **Single** – The hover tooltip shows only a single series, the one that you are hovering over on the visualization.
- **All** – The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden** – Do not display the tooltip when you interact with the visualization.

Use an override to hide individual series from the tooltip.

Legend options

Use these settings to define how the legend appears in your visualization. For more information about the legend, refer to Configure a legend.

Legend visibility

Use the **Visibility** toggle to show or hide the legend.

Legend mode

Set the display mode of the legend.

- **List** – Displays the legend as a list. This is the default display mode of the legend.
- **Table** – Displays the legend as a table.

Legend placement

Choose where to display the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend values

Select values to display in the legend. You can select more than one.

- **Percent** – The percentage of the whole.
- **Value** – The raw numerical value.

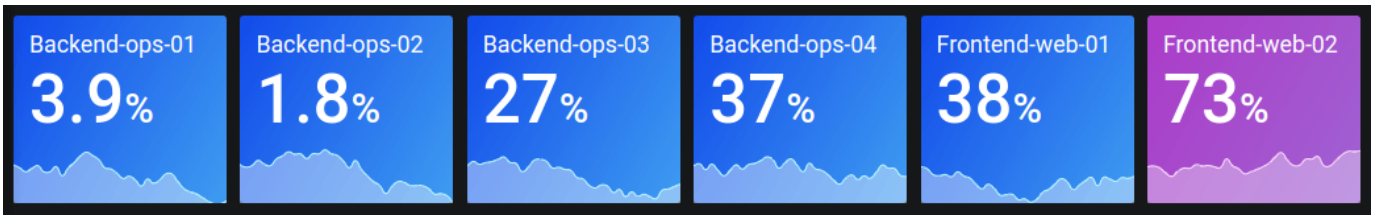
For more information about the legend, refer to Configure a legend.

Stat

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Stats show one large stat value with an optional graph sparkline. You can control the background or value color using thresholds or overrides. Stat panels also display percent change with configurable color mode options, making it easier to see trends at a glance.



By default, a Stat displays one of the following:

- Just the value for a single series or field.
- Both the value and name for multiple series or fields.

You can use the **Text mode** to control whether the text is displayed or not.

Automatic layout adjustment

The panel automatically adjusts the layout depending on available width and height in the dashboard. It automatically hides the graph (sparkline) if the panel becomes too small.

Value options

Use the following options to refine how your visualization displays the value:

Show

Choose how Grafana displays your data.

- **Calculate**

  Show a calculated value based on all rows.

  - **Calculation** – Select a reducer function that Grafana will use to reduce many fields to a single value. For a list of available calculations, see standard calculations.
  - **Fields** – Select the fields display in the visualization.

- **All values**

  Show a separate stat for every row. If you select this option, then you can also limit the number of rows to display.

  - **Limit** – The maximum number of rows to display. Default is 5,000.
  - **Fields** – Select the fields display in the visualization.

Stat styles

Style your visualization.

Orientation

Choose a stacking direction.

- **Auto** – Grafana selects what it thinks is the best orientation.
- **Horizontal** – Bars stretch horizontally, left to right.
- **Vertical** – Bars stretch vertically, top to bottom.

Text mode

You can use the Text mode option to control what text the visualization renders. If the value is not important, only the name and color is, then change the **Text mode** to **Name**. The value will still be used to determine color and is displayed in a tooltip.

- **Auto** – If the data contains multiple series or fields, show both name and value.
- **Value** – Show only value, never name. Name is displayed in the hover tooltip instead.
- **Value and name** – Always show value and name.
- **Name** – Show name instead of value. Value is displayed in the hover tooltip.
- **None** – Show nothing (empty). Name and value are displayed in the hover tooltip.

Wide layout

Set whether wide layout is enabled or not. Wide layout is enabled by default.

- **On** – Wide layout is turned on.
- **Off** – Wide layout is turned off.

> **Note**
>
> This option is only applicable when **Text mode** is set to **Value and name**. When wide layout is turned on, the value and name are displayed side-by-side with the value on the right, if the panel is wide enough. When wide layout is turned off, the value is always rendered underneath the name.

Color mode

Select a color mode.

- **None** – No color is applied to the value.
- **Value** – Applies colors to the value and graph area.
- **Background Gradient** – Applies color to the value, graph area, and background, with a slight background gradient.
- **Background Solid** – Applies color to the value, graph area, and background, with a solid background color.

Graph mode

Select a graph and sparkline mode.

- **None** – Hides the graph and only shows the value.
- **Area** – Shows the area graph below the value. This requires that your query returns a time column.

Text alignment

Choose an alignment mode.

- **Auto** – If only a single value is shown (no repeat), then the value is centered. If multiple series or rows are shown, then the value is left-aligned.
- **Center** – Stat value is centered.

Show percent change

Set whether percent change is displayed or not. By default it is not shown.

> **Note**
>
> This option is not applicable when the **Show** setting, under **Value options**, is set to **All values**.

Text size

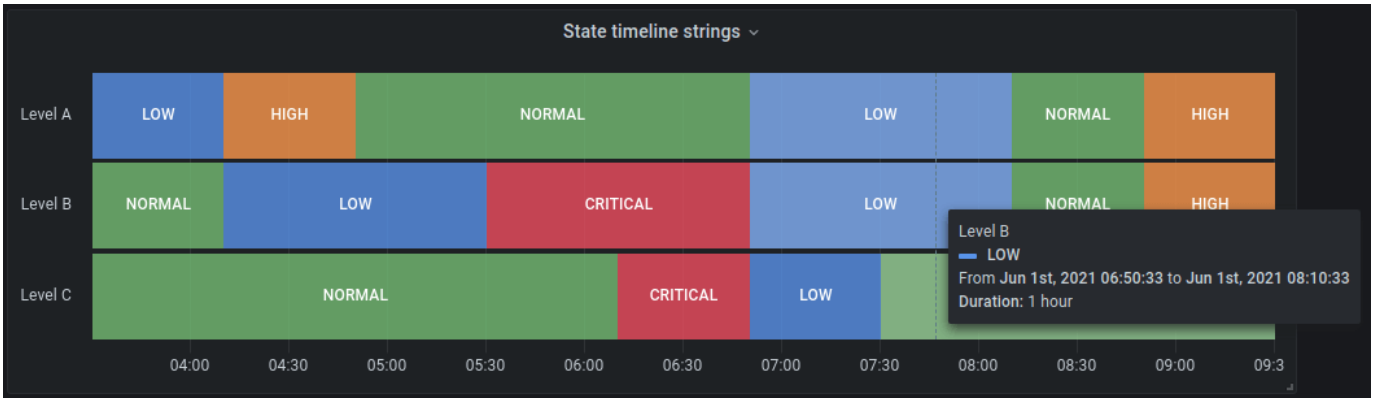
Adjust the sizes of the gauge text.

- **Title** – Enter a numeric value for the gauge title size.
- **Value** – Enter a numeric value for the gauge value size.

State timeline

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

State timelines show discrete state changes over time. Each field or series is rendered as its unique horizontal band. State regions can either be rendered with or without values. This panel works well with string or boolean states but can also be used with time series. When used with time series, the thresholds are used to turn the numerical values into discrete state regions. State timeline visualizations support pagination, making it easier to navigate large datasets.



State timeline options

Use these options to refine your visualizations:

Merge equal consecutive values

Controls whether Grafana merges identical values if they are next to each other.

Show values

Controls whether values are rendered inside the state regions. **Auto** will render values if there is sufficient space.

Align values

Controls value alignment inside state regions.

Row height

Controls space between rows. 1 = no space = 0.5 = 50% space.

Line width

Controls the line width of state regions.

Fill opacity

Controls the opacity of state regions.

Connect null values

Choose how null values, which are gaps in the data, appear on the graph. Null values can be connected to form a continuous line or set to a threshold above which gaps in the data are no longer connected.

- **Never** – Time series data points with gaps in the data are never connected.
- **Always** – Time series data points with gaps in the data are always connected.
- **Threshold** – Specify a threshold above which gaps in the data are no longer connected. This can be useful when the connected gaps in the data are of a known size or within a known range, and gaps outside this range should no longer be connected.

Disconnect values

Choose whether to set a threshold above which values in the data should be disconnected.

- **Never** – Time series data points in the data are never disconnected.
- **Threshold** – Specify a threshold above which values in the data are disconnected. This can be useful when desired values in the data are of a known size or within a known range, and values outside this range should no longer be connected.

Value mappings

To assign colors to boolean or string values, use Configure value mappings.

Time series data with thresholds

The visualization can be used with time series data as well. In this case, the thresholds are used to turn the time series into discrete colored state regions.

Legend options

When the legend option is enabled, it can show either the value mappings or the threshold brackets. To show the value mappings in the legend, it's important that the **Color scheme** option under Standard options is set to **Single color** or **Classic palette**. To see the threshold brackets in the legend, set the **Color scheme** to **From thresholds**.

Legend mode

Use these settings to define how the legend appears in your visualization. For more information about the legend, refer to Configure a legend.

- **List** – Displays the legend as a list. This is the default mode.
- **Table** – Displays the legend as a table.

- **Hidden** – Hides the legend.

Legend placement

Choose where to display the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend values

Choose which of the standard calculations to show in the legend. You can have more than one.

Status history

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Status histories show periodic states over time. Each field or series is rendered as a horizontal row. Boxes are rendered and centered around each value.

Supported data

A status history works with string, boolean, and numerical fields or time series. A time field is required. You can use value mappings to color strings or assign text values to numerical ranges.

Display options

Use these options to refine the visualization.

Show values

Controls whether values are rendered inside the value boxes. **Auto** will render values if there is sufficient space.

Column width

Controls the width of boxes. 1 = maximum space and 0 = minimum space.

Line width

Controls line width of state regions.

Fill opacity

Controls the fill opacity of state regions.

Value mappings

To assign colors to boolean or string values, use Configure value mappings.

Time series data with thresholds

The panel can be used with time series data as well. In this case, the thresholds are used to color the boxes. You can also use gradient color schemes to color values.

Legend options

When the legend option is enabled, it can show either the value mappings or the threshold brackets. To show the value mappings in the legend, it's important that the **Color scheme** option under Standard options is set to **Single color** or **Classic palette**. To see the threshold brackets in the legend, set the **Color scheme** to **From thresholds**.

Legend mode

Use these settings to define how the legend appears in your visualization. For more information about the legend, refer to Configure a legend.

- **List** – Displays the legend as a list. This is the default mode.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement

Choose where to display the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

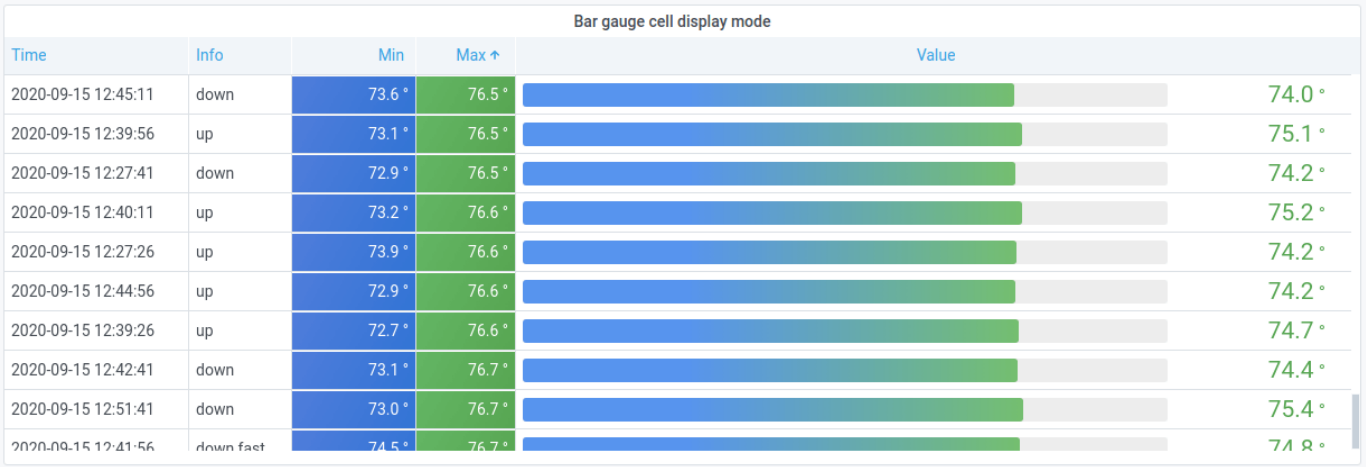
Legend values

Choose which of the standard calculations to show in the legend. You can have more than one.

Table

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Tables are very flexible, supporting multiple modes for time series and for tables, annotation, and raw JSON data. This visualization also provides date formatting, value formatting, and coloring options, including CSS cell styling, tooltip generation from table fields, a new Actions cell type for interactive buttons, auto-formatted cell values in Cell Inspect, and row-level conditional formatting for coloring entire rows based on data values.



> **Note**
>
> Annotations and alerts are not supported in tables.

Sort column

Choose a column title to change the sort order from default to descending to ascending. Each time you select the column, the sort order changes to the next option in the cycle. You can sort on multiple columns by holding the `shift` key when selecting additional columns.

Table options

Show header

Show or hide column names imported from your data source.

Column width

By default, Grafana automatically calculates the column width based on the table size and the minimum column width. This field option can override the setting and define the width for all columns in pixels.

For example, if you enter `100`, all the columns will be set to 100 pixels wide (the change takes place when you exit the field).

Minimum column width

By default, the minimum width of the table column is 150 pixels. This field option can override that default and will define the new minimum column width for the table panel in pixels.

For example, if you set the minimum to 75, all the columns will scale to no smaller than 75 pixels wide.

For small-screen devices, such as smartphones or tablets, you can reduce the default `150` pixel value to `50` to allow table based panels to render correctly in dashboards.

Column alignment

Choose how Grafana should align cell contents.

- Auto (default)
- Left
- Center
- Right

Cell type

By default, Grafana automatically chooses display settings. You can override the settings by choosing one of the following options to set the default for all fields. Additional configuration is available for some cell types.

> **Note**
>
> If you set these in the **Field** tab, then the type will apply to all fields, including the time field. You can set them in the **Override** tab to apply the change to one or more fields.

Color text

If thresholds are set, then the field text is displayed in the appropriate threshold color.

Color background (gradient or solid)

If thresholds are set, then the field background is displayed in the appropriate threshold color.

Gauge

Cells can be displayed as a graphical gauge, with several different presentation types.

- *Basic* – The basic mode will show a simple gauge with the threshold levels defining the color of gauge.
- *Gradient* – The threshold levels define a gradient.
- *LCD* – The gauge is split up in small cells that are lit or unlit.

Additionally, labels displayed alongside the gauges can be set to be colored by value, match the theme text color, or be hidden.

- **Value color**
- **Text color**
- **Hidden**

JSON view

Shows value formatted as code. If a value is an object the JSON view allowing browsing the JSON object will appear on hover.

Sparkline

Shows values rendered as a sparkline. Requires time series to table data transform.

Cell value inspect

Enables value inspection from table cell. The raw value is presented in a modal window.

> **Note**
>
> Cell value inspection is only available when cell display mode is set to Auto, Color text, Color background, or JSON View.

Column filter

You can temporarily change how column data is displayed. For example, you can order values from highest to lowest or hide specific values. For more information, see Filter table columns.

Pagination

Use this option to enable or disable pagination. It is a front-end option that does not affect queries. When enabled, the page size automatically adjusts to the height of the table.

Filter table columns

If you turn on the **Column filter**, then you can filter table options.

To turn on column filtering

1. In Grafana, navigate to the dashboard with the table with the columns that you want to filter.
2. On the table panel you want to filter, open the panel editor.
3. Choose the **Field** tab.
4. In **Table** options, turn on the **Column filter** option.

A filter (funnel) icon appears next to each column title.

Filter column values

To filter column values, choose the filter (funnel) icon next to a column title. Grafana displays the filter options for that column.

Choose the check box next to the values that you want to display. Enter text in the search field at the top to show those values in the display so that you can select them rather than scroll to find them.

Select from several operators to display column values:

- **Contains** – Matches a regex pattern (operator by default).
- **Expression** – Evaluates a Boolean expression. The character $ represents the column value in the expression (for example, `$ >= 10 && $ <= 12`).
- **Comparison operators** – You can use the typical comparison operators: =, !=, <, <=, >, >=.

Choose the checkbox above the **Ok** and **Cancel** buttons to add or remove all displayed values from the filter.

Clear column filters

Columns with filters applied have a blue funnel displayed next to the title.

To remove the filter, choose the blue funnel icon and then select **Clear filter**.

Table footer

You can use the table footer to show calculations on fields.

After you enable the table footer, you can select the **Calculation**, and then the **Fields** that you want to calculate.

The system applies the calculation to all numeric fields if you do not select a field.

Count rows

If you want to show the number of rows in the dataset instead of the number of values in the selected fields, select the **Count** calculation and enable **Count rows**.

Text

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Text visualizations enable you to directly include text or HTML in your dashboards. This can be used to add contextual information and descriptions or embed complex HTML.

Mode

Mode determines how embedded content appears. It has the following options:

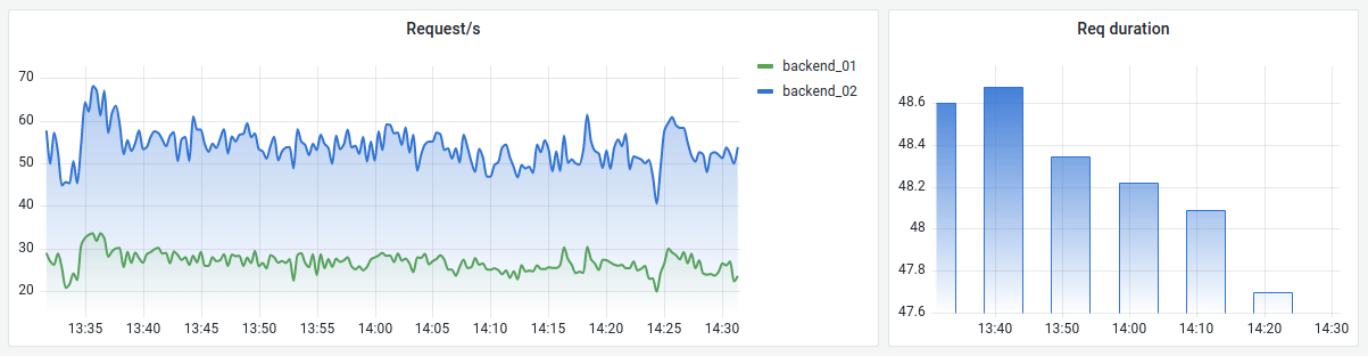
- **Markdown** – This option formats the content as markdown.
- **HTML** – This setting renders the content as sanitized HTML.
- **Code** – This setting renders content inside a read-only code editor. Select an appropriate language to apply syntax highlighting to the embedded text.

Variables

Variables in the content will be expanded for display.

Time series

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.



Time series visualizations are the default and primary way to visualize time series data as a graph. They can render series as lines, points, or bars. They're versatile enough to display almost any time-series data.

> **Note**
>
> You can migrate Graph panel visualizations to Time series visualizations. To migrate, on the **Panel** tab, choose **Time series visualization**. Grafana transfers all applicable settings.

Topics

- Tooltip options
- Legend options
- Graph styles
- Axis options
- Color options

Tooltip options

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Tooltip options control the information overlay that appears when you hover over data points in the graph.

Tooltip mode

- **Single** – The hover tooltip shows only a single series, the one that you are hovering over.
- **All** – The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden** – Do not display the tooltip when you interact with the visualization.

Legend options

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Legend options control the series names and statistics that appear under or to the right of the graph.

Legend mode

Use these settings to define how the legend appears in your visualization. For more information about the legend, see Configure a legend.

- **List** – Displays the legend as a list. This is the default display mode of a legend.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement

Choose where to display the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend values

Choose which of the standards calculations to show in the legend. You can have more than one.

Graph styles

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use this option to define how to display your time series data. You can use overrides to combine multiple styles in the same graph.

- **Lines** – Display the time series as a line on a graph.
- **Bars** – Display the time series as a series of bars on a graph, one for each data point.
- **Points** – Display the time series as dots on a graph, one for each data point.

Bar alignment

Set the position of the bar relative to a data point.where the point would be drawn on the graph. Because a bar has a width, it can be placed before, after, or centered on the point. The choices for this option are:

- 
  Before – The bar is drawn before the point. The point is placed on the trailing corner of the bar.
- 
  Center – The bar is drawn around the point. The point is placed in the center of the bar. This is the default.
- 
  After – The bar is drawn after the point. the point is placed on the leading corner of the bar.

Line width

Line width is a slider that controls the thickness for series lines or the outline for bars.

Fill opacity

Sets the opacity of a fill color. Fills are used, for example, to show the area under the line in a line graph, or as the color of bars in a bar graph.

Gradient mode

Gradient mode specifies the gradient fill, which is based on the series color. To change the color, use the standard color scheme field option. For more information, see Color scheme.

The gradient mode options are:

- **None** – No gradient fill. This is the default setting.
- **Opacity** – An opacity gradient where the opacity of the fill increases as the Y-axis values increase.
- **Hue** – A subtle gradient that is based on the hue of the series color.
- **Scheme** – A color gradient defined by your color scheme. This setting is used for the fill area and the line. For more information, see Color options.

The gradient appearance is influenced by the **Fill opacity** setting.

Show points

You can configure your visualization to add points to line or bars, with the following options.

- **Auto** – Grafana determines whether to show points based on the density of the data. Points are shown when the density is low.
- **Always** – Points are shown, regardless of the data density.
- **Never** – Do not show points.

Point size

Sets the size of the points, from 1 to 40 pixels in diameter.

Line interpolation

Choose how Grafana interpolates the series line.



The options are:

- **Linear** – Points are joined by straight lines.

- **Smooth** – Points are joined by curved lines that smooths transitions between points.
- **Step before** – The line is displayed as steps between points. Points are rendered at the end of the step.
- **Step after** – The line is displayed as steps between points. Points are rendered at the beginning of the step.

Line style

Set the style of the line. To change the color, use the standard color scheme field option.

The choices for line style are:

- **Solid** – Display a solid line. This is the default setting.
- **Dash** – Display a dashed line. When you choose this option, a list appears for you to select the length and gap (length, gap) for the line dashes. Dash spacing is set to 10, 10 by default.
- **Dots** – Display dotted lines. When you choose this option, a list appears for you to select the gap length for the dot spacing. Dot spacing is set to 10 by default.

Connect null values

Choose how null values, which are gaps in the data, appear on the graph. Null values can be connected to form a continuous line, or set to a threshold above which gaps in the data are no longer connected.

The choices for how to connect null values are:

- **Never** – Time series data points with gaps in the data are never connected.
- **Always** – Time series data points with gaps in the data are always connected.
- **Threshold** – Specify a threshold above which gaps in the data are no longer connected. This can be useful when the connected gaps in the data are of a known size or within a known range, and gaps outside the range should no longer be connected.

Disconnect values

Choose whether to add a gap between values in the data that have times between them above a specified threshold.

The choices for disconnect values are are:

- **Never** – Time series data points are never disconnected.
- **Threshold** – Specify a threshold above which values in the data are disconnected. This can be useful when desired values in the data are of a known size or within a known range, and values outside this range should no longer be connected.

Stack series

Stacking allows Grafana to display series on top of each other. Be cautious when using stacking in the visualization as it can easily create misleading graphs. To read more about why stacking might not be the best approach, refer to The Issue with Stacking.

The choices for stacking are:

- **Off** – Turns off series stacking.
- **Normal** – Stacks series on top of each other.
- **100%** – Stack by percentage, where all series together add up to 100%.

Stacking series in groups

You can override the stacking behavior to stack series in groups. For more information about creating an override, see Configure field overrides.

To stack series in groups

1. Edit the panel and choose **Overrides**.
2. Create a field override for the **Stack series** option.
3. In stacking mode, select **Normal**.
4. Name the stacking group in which you want the series to appear.

   The stacking group name option is only available when you create an override.

Fill below to

The **Fill below to** option fills the area between two series. This options is only available as a series or field override.

Using this option you can fill the area between two series, rather than from the series line down to 0. For example, if you had two series called *Max* and *Min*, you could select the *Max* series and override it to **Fill below to** the *Min* series. This would fill only the area between the two series lines.

Axis options

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Options under the axis category change how the x- and y-axes are rendered. Some options do not take effect until you click outside of the field option box you are editing. You can also press `Enter`.

Time zone

Set the desired time zones to display along the x-axis.

Placement

Select the placement of the Y-axis. The options are:

- **Auto** – Automatically assigns the y-axis to the series. When there are two or more series with different units, Grafana assigns the left axis to the first unit, and the right axis to the units that follow.
- **Left** – Display all y-axes on the left side.
- **Right** – Display all y-axes on the right side.
- **Hidden** – Hide all axes.

To selectively hide axes, add a field override that targets specific fields.

Label

Set a y-axis text label. If you have more than one y-axis, then you can assign different labels using an override.

Width

Set a fixed width of the axis. By default, Grafana dynamically calculates the width of an axis.

By setting the width of the axis, data with different axes types can share the same display proportions. This setting makes it easier for you to compare more than one graph's worth of data because the axes are not shifted or stretched within visual proximity to each other.

Show grid lines

Set the axis grid line visibility.

- **Auto** – Automatically show grid lines based on the density of the data.
- **On** – Always show grid lines.
- **Off** – Never show grid lines.

Color

Set the color of the axis.

- **Text** – Set the color based on theme text color.
- **Series** – Set the color based on the series color.

Show border

Set the axis border visibility.

Scale

Set how the y-axis values scale.

- **Linear** – Divides the scale into equal parts.
- **Logarithmic** – Use a logarithmic scale. When you select this option, a list appears for you to choose a binary (base 2) or common (base 10) logarithmic scale.
- **Symlog** – Use a symmetrical logarithmic scale. When you select this option, a list appears for you to choose a binary (base 2) or common (base 10) logarithmic scale. The linear threshold option allows you to set the threshold at which the scale changes from linear to logarithmic.

Centered zero

Sets the y-axis to be centered on zero.

Soft min and soft max

Set a **Soft min** or **soft max** option for better control of y-axis limits. By default, Grafana sets the range for the y-axis automatically based on the dataset.

Soft min and soft max settings can prevent small variations in the data from being magnified when it's mostly flat. In contrast, hard min and max values help prevent obscuring useful detail in the data by clipping intermittent spikes past a specific point.

To define hard limits of the y-axis, set standard min/max options. For more information, refer to Configure standard options.

Transform

Use this option to transform the series values without affecting the values shown in the tooltip, context menus, or legend. You have two transform options:

- **Negative Y transform** – Flip the results to negative values on the Y axis.
- **Constant** – Show the first value as a constant line.

> **Note**
>
> The transform option is only available as an override.

Display multiple y-axes

There are some cases where you want to display multiple y-axes. For example, if you have a dataset showing both temperature and humidity over time, you could show two y-axes with different units for these two series.

To display multiple y-axes, add a field override. Follow the steps as many times as required to add as many y-axes as you need.

Color options

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

By default, the graph uses the standard Color scheme option to assign series colors. You can also use the legend to open the color picker by selecting the legend series color icon. Setting color this way automatically creates an override rule that set's a specific color for a specific series.

The following are additional options that you can use to override series color defaults.

Classic palette

The most common setup is to use the **Classic palette** for graphs. This scheme automatically assigns a color for each field or series based on its order. If the order of a field changes in your query, the color also changes. You can manually configure a color for a specific field using an override rule.

Single color

Use this mode to specify a color. You can also select the colored line icon next to each series in the Legend to open the color picker. This automatically creates a new override that sets the color scheme to single color and the selected color.

By value color schemes

If you select a by value color scheme like **From thresholds (by value)** or **Green-Yellow-Red (by value)**, the **Color series by** option appears. This option controls which value (Last, Min, Max) to use to assign the series its color.

Scheme gradient mode

The **Gradient mode** option located under the **Graph styles** has a mode named **Scheme**. When you enable **Scheme**, the line or bar receives a gradient color defined from the selected **Color scheme**.

From thresholds

If the **Color scheme** is set to **From thresholds (by value)** and **Gradient mode** is set to **Scheme**, then the line or bar color changes as they cross the defined thresholds. You will see only the exact colors chosen in the scheme.

Gradient color schemes

Using a gradient color scheme *without* setting the **Gradient mode** to **Scheme**, means that the colors chosen will form a gradient between the colors chosen, as the values in the series move between the thresholds set.

Traces

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Traces visualizations let you follow a request as it traverses the services in your infrastructure. The traces visualization displays traces data in a diagram that allows you to easily interpret it.

For more information about traces and how to use them, refer to the following documentation:

- Tracing in Explore
- Tempo data source
- Getting started with Tempo in the *Grafana Labs Tempo Documentation*.

Adding a panel with tracing visualizations

Once you have tracing data available in your Grafana stack, you can add tracing panels to your Grafana dashboards.

Using a dashboard variable, `traceID`, lets you create a query to show specific traces for a given trace ID. For more information about dashboard variables, refer to the Variables documentation.

Prerequisites

Before you begin, you need:

- An Amazon Managed Grafana workspace.
- A Tempo data source connected to your workspace.

To view and analyze traces data in a dashboard, you need to add the traces visualization to your dashboard and define a query using the panel editor. The query determines the data that is displayed in the visualization. For more information on the panel editor, refer to the Panel editor documentation.

This procedure uses dashboard variables and templates to allow you to enter trace IDs which can then be visualized. You'll use a variable called `traceId` and add it as a template query.

To add a traces visualization query

1. In your workspace, create a new dashboard or go to an existing dashboard where you'd like to add traces visualizations.
2. Choose **Add visualization** from a new dashboard or choose **Add Panel** on an existing dashboard.
3. Select the appropriate tracing data source.
4. In the top-right of the panel editor, choose the **Visualizations** tab, and select **Traces**.
5. Under the **Panel options**, enter a **Title** for your trace panel. For more information on the panel editor, see Configure panel options.
6. In the query editor, select the **TraceQL** query type tab.
7. Enter `${traceId}` in the TraceQL query field to create a dashboard variable. This variable is used as the template query.
8. Choose **Apply** in the panel editor to add the panel to the dashboard.
9. Go to the dashboard **Settings** and add a new variable called `traceId`, of variable type **Custom**, giving it a label, if required. Choose **Apply** to add the variable to the dashboard.
10. Verify that the panel works by using a valid trace ID for the data source used for the trace panel and editing the ID in the dashboard variable.

Adding TraceQL with table visualizations

While you can add a trace visualization to a dashboard, having to manually add trace IDs as a dashboard variable is cumbersome. It's more useful to instead be able to use TraceQL queries to search for specific types of traces and then select appropriate traces from matching results.

Prerequisites

This procedure assumes you have completed the previous procedure.

To add TraceQL with table visualizations

1. In the same dashboard where you added the trace visualization, choose **Add panel** to add a new visualization panel.
2. Select the same trace data source you used in the previous section.
3. In the top-right of the panel editor, select the **Visualizations** tab, then choose **Table**.
4. In the query editor, choose the **TraceQL** tab.
5. Under the **Panel options**, enter a **Title** for your trace panel.
6. Add an appropriate TraceQL query to search for traces that you would like to visualize in the dashboard. For example, here is a simple, static query from a server called *`my-server`*.

   ```
   { .service.name = "my-server" && .http.status_code=500 }
   ```

   You can write the TraceQL query as a template query to take advantage of other dashboard variables, if they exist. This lets you create dynamic queries based on these variables.

When results are returned from a query, the results are rendered in the panel's table.

The results in the traces visualization include links to the **Explore** page that renders the trace. You can add other links to traces in the table that fill in the `traceId` dashboard variable when selected, so that the trace is visualized in the same dashboard.

To create a set of data links in the panel, use the following procedure.

To use a variable to add other links to traces

1. In the right-side menu, under **Data links**, choose **Add link**.
2. Add a **Title** for the data link.

3. Find the path to the dashboard by looking in your browser's address bar when the full dashboard is being rendered. Because this is a link to a dashboard in the same Grafana stack, only the path of the dashboard is required.

   For example, if your path is:

   ```
   https://g-example.grafana-workspace.us-east-1.amazonaws.com/d/1234abcd5/my-dashboard?orgId=1
   ```

   Then the path to the dashboard is:

   ```
   /d/1234abcd5/my-dashboard?orgId=1
   ```

4. In the **URL** field, make a self-reference to the dashboard that contains both of the panels. This self-reference uses the value of the selected trace in the table to fill in the dashboard variable. Use the path for the dashboard from the previous step and then fill in the value of `traceId` using the selected results from the TraceQL table. The trace ID is exposed using the `traceID` data field in the returned results, so use that as the value for the dashboard variable. For example:

   ```
   /d/1234abcd5/my-dashboard?orgId=1&var-traceId=${__data.fields["traceID"]}
   ```

5. Choose **Save** to save the data link.
6. Choose **Apply** from the panel editor to apply the panel to the dashboard.
7. Save the dashboard.

You should now see a list of matching traces in the table visualization. While selecting the **TraceID** or **SpanID** fields will give you the option to either open the **Explore** page to visualize the trace or following the data link, selecting any other field (such as `Start time`, `Name`, or `Duration`) automatically follows the data link, filling in the `traceId` dashboard variable, and then shows the relevant trace in the trace panel.

Trend

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Trend visualizations should be used for datasets that have a sequential, numeric X that is not time. Some examples are function graphs, rpm/torque curves, supply/demand relationships, and elevation or heart rate plots along a race course (with x as distance or duration from start).

Trend visualizations support all visual styles and options available in the time series visualization with the following exceptions:

- No annotations or time regions
- No shared cursor (or crosshair)
- No multi-timezone x axis
- No ability to change the dashboard time range via drag-selection

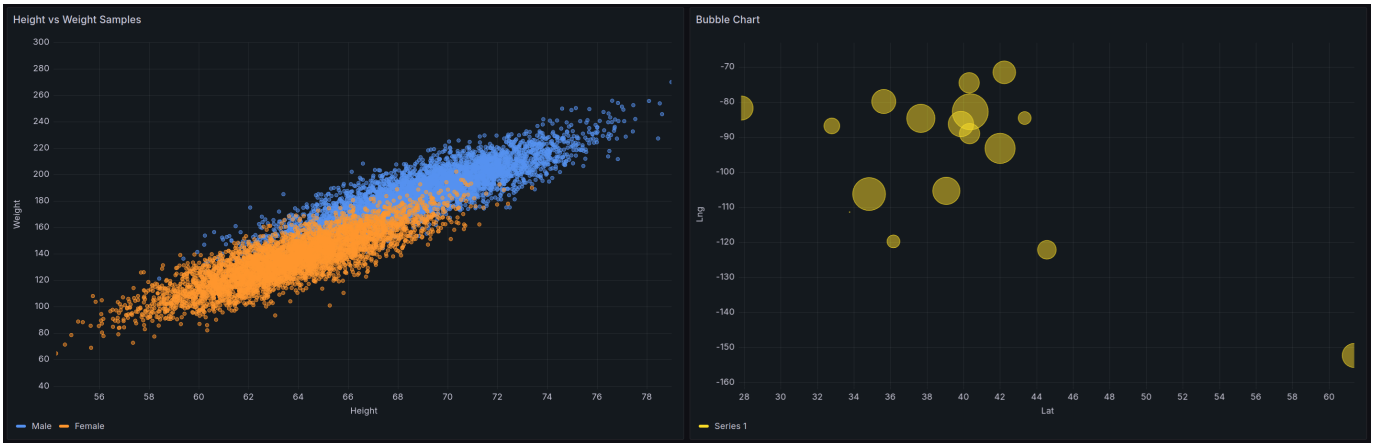
X Field selection

Use this option to select a field that contains increasing numeric values.

For example, you could represent engine power and torque versus speed where speed is plotted on the x axis and power and torque are plotted on the y axes.

XY Chart

XY charts provide a way to visualize arbitrary x and y values in a graph so that you can easily show the relationship between two variables. XY charts are typically used to create scatter plots. you can also use them to create bubble charts, where field values determine the size of each bubble.



Supported data formats

You can use any type of tabular data with at least two numeric fields in an XY chart. This type of visualization doesn't require time data.

Panel options

In the **Panel options** section of the panel editor pane, you set basic options like the panel title and description. You can also configure repeating panels in this section. For more information, see Configure panel options.

XY chart options

Series mapping

Set how series data is mapped in the chart.

- **Auto** – Automatically generates series from all available data frames (or datasets). You can filter to select only one frame.
- **Manual** – Explicitly define the series by selecting from available data frames.

Depending on your series mapping selection, the **Frame**, **X-field**, and **Y-field** options differ. The Auto and Manual series mapping sections describe these different options.

Auto series mapping options

When you select **Auto** as your series mapping mode, the following options are preconfigured, but you can also define them yourself.

- **Frame** – By default an XY chart displays all data frames. You can filter to select only one frame.

- **X-field** – Select which field X represents. By default,t his is the first number field in each data frame.
- **Y-field** – After the X-field is set, by default, all the remaining number fields in the data frame are designated as the Y-fields. You can use this option to explicitly choose which fields to use for Y.

  The series of the chart are generated from the Y-fields. To make changes to a series in an XY chart, make overrides to the Y-field. Any field you use in the Size field or Color field doesn't generate a series.

  You can also use overrides to exclude Y-fields individually. To do so, add an override with the following properties for each Y-field you want removed:
  - Override type: **Fields with name**
  - Override property: **Series > Hide in area**
  - Area: **Viz**

Manual series mapping options

When you select **Manual** as your series mode, you can add, edit, and delete series. To manage a series, select the **Series** field. To rename the series, select the series name.

In **Manual** mode, you must set the following options:

- **Frame** – Select your data frame or dataset. You can add as many frames as you want.
- **X-field** – Select which field X represents.
- **Y-field** – Select which field Y represents.

Size field

Use this option to set which field's values control the size of point in the chart. This value is relative to the min and max of all the values in the data frame.

When you select this option, you can then set the min and max point size options.

Color field

Use this option to set which field's values control the color of the points in the chart. To use the color value options under the **Standard options**, you must set this field.

Typically, this option is used when you only have one series displayed in the chart.

Show

Set how values are represented in the visualization.

- **Points** – Display values as points. When you select this option, the point size option is also displayed.
- **Lines** – Add a line between values. When you select this option, the line style and line width options are also displayed.
- **Both** – Display both points and lines.

Point size

Sets the size of all point in the chart, from one to 100 pixels in diameter. The default size is five pixels. You can set an override to set the pixel size by series (Y-field).

Min/Max point size

Use these optinos to control the minimum or maximum point size when you've set the **Size field** option. You can override these options for specific series.

Line style

Set the style of the line. To change the color, use the standard color scheme field option.

- **Solid** – Display a solid line. This is the default setting.
- **Dash** – Display a dashed line. When you choose this option, a drop-down list is displayed where you can select the length and gap setting for the line dashes. By default, the length and gap are set to `10, 10`.
- **Dots** – Display dotted lines. When you choose this option, a drop-down list is displayed where you can select dot spacing. By default, the dot spacing is set to `0, 10` (the first number represents dot length, and is always zero).

Line width

Sets the width of the lines, in pixels.

Tooltip options

Tooltip options control the information overlay that appears when you hover over data points in the graph.

Tooltip mode

- **Single** – The hover tooltip shows only a single series, the one that you are hovering over.
- **Hidden** – Do not display the tooltip when you interact with the visualization.

Use an override to hide individual series from the tooltip.

Max height

Set the maximum height of the tooltip box. The defautl is 600 pixels.

Legend options

Legend options control the series names and statistics that appear under or to the right of the graph. For more information about the legend, see Configure a legend.

Visibility

Toggle the switch to turn the legend on or off.

Mode

Use these settings to define how the legend appears in your visualization.

- **List** – Displays the legend as a list. This is the default display mode of a legend.
- **Table** – Displays the legend as a table.

Placement

Choose where to display the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Values

Choose which of the standards calculations to show in the legend. You can have more than one.

Width

Control how wide the legend is when place to the right side of the visualization. This option is only displayed if you set the legend placement to **Right**.

Axis options

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Options under the axis category change how the x- and y-axes are rendered. Some options don't take effect until you click outside of the field option box you are editing. You can also press `Enter`.

Placement (y-axis)

Select the placement of the Y-axis. The options are:

- **Auto** – Automatically assigns the y-axis to the series. When there are two or more series with different units, Grafana assigns the left axis to the first unit, and the right axis to the units that follow.
- **Left** – Display all y-axes on the left side.
- **Right** – Display all y-axes on the right side.
- **Hidden** – Hide all axes.

To selectively hide axes, add a field override that targets specific fields.

Label

Set a y-axis text label. If you have more than one y-axis, then you can assign different labels using an override.

Width

Set a fixed width of the axis. By default, Grafana dynamically calculates the width of an axis.

By setting the width of the axis, data with different axes types can share the same display proportions. This setting makes it easier for you to compare more than one graph's worth of data because the axes are not shifted or stretched within visual proximity to each other.

Show grid lines

Set the axis grid line visibility.

- **Auto** – Automatically show grid lines based on the density of the data.
- **On** – Always show grid lines.
- **Off** – Never show grid lines.

Color

Set the color of the axis.

- **Text** – Set the color based on theme text color.
- **Series** – Set the color based on the series color.

Show border

Set the axis border visibility.

Scale

Set how the y-axis values scale.

- **Linear** – Divides the scale into equal parts.
- **Logarithmic** – Use a logarithmic scale. When you select this option, a list appears for you to choose a binary (base 2) or common (base 10) logarithmic scale.
- **Symlog** – Use a symmetrical logarithmic scale. When you select this option, a list appears for you to choose a binary (base 2) or common (base 10) logarithmic scale. The linear threshold option allows you to set the threshold at which the scale changes from linear to logarithmic.

Centered zero

Sets the y-axis to be centered on zero.

Soft min and soft max

Set a **Soft min** or **soft max** option for better control of y-axis limits. By default, Grafana sets the range for the y-axis automatically based on the dataset.

Soft min and soft max settings can prevent small variations in the data from being magnified when it's mostly flat. In contrast, hard min and max values help prevent obscuring useful detail in the data by clipping intermittent spikes past a specific point.

To define hard limits of the y-axis, set standard min/max options. For more information, refer to Configure standard options.

Transform

Use this option to transform the series values without affecting the values shown in the tooltip, context menus, or legend. You have two transform options:

- **Negative Y transform** – Flip the results to negative values on the Y axis.
- **Constant** – Show the first value as a constant line.

> **Note**
>
> The transform option is only available as an override.

Display multiple y-axes

There are some cases where you want to display multiple y-axes. For example, if you have a dataset showing both temperature and humidity over time, you could show two y-axes with different units for these two series.

To display multiple y-axes, add a field override. Follow the steps as many times as required to add as many y-axes as you need.

Standard options

Standard options in the panel editor let you change how field data is displayed in your visualization. When you set a standard option, the change is applied to all fields or series. For more granular control over the display of fields, refer to Configure field overrides.

You can cusotmize the following standard options:

- **Field min/max** – Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.
- **Color scheme** – Set single or multiple colors for your entire visualization.

For more information, see Configure standard options.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

For more information, see Configure data links.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, you can target a particular set of fields and define multiple options for how those fields are displayed.

Choose from one of the following override options:

- **Fields with name** – Select a field from the list of all available fields.
- **Fields with name matching regex** – Specify fields to override with a regular expression.
- **Fields with type** – Select fields by type, such as string, numeric, or time.
- **Fields returned by query** – Select all fields returned by a specific query.
- **Fields with values** – Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, or **Total**.

For more information, see Configure field overrides.

Explore in Grafana version 12

This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana's dashboard UI provides functionality to build dashboards for visualization. *Explore* strips away the dashboard and panel options so that you can focus on the query. It helps you iterate until you have a working query and then you can build a dashboard from the query.

Note

If you just want to explore your data and do not want to create a dashboard, then Explore makes this much easier. If your data source supports graph and table data, then Explore shows the results both as a graph and a table. This allows you to see trends in the data and more details at the same time.

This page will get you started exploring your data. The following topics give you more details on specific features and uses for Explore.

- Query management in Explore
- Logs in Explore
- Tracing in Explore
- Correlations editor in Explore
- Inspector in Explore

Start exploring

> **Note**
>
> In order to access Explore, you must have an editor or an administrator role.

To access Explore

1. In your Grafana workspace, choose the Explore menu item from the left menu bar.

   An empty Explore tab opens.

   Alternately, to start with an existing query in a panel, choose the Explore option from the Panel menu. This opens an Explore tab with the query from the panel and allows you to tweak or iterate in the query outside of your dashboard.
2. Choose your data source from the dropdown in the top left.

   You can also choose **Open advanced data source picker** to see more options, including adding a data source (for Admins only).
3. Write the querying using the query editor provided by the selected data source.

   For more details about queries, see Query and transform data.
4. Run the query using the button in the top right corner.

Split and compare

The split view provides an easy way to compare visualizations side-by-side or to look at related data together on one page.

Top open the split view

1. In the Explore view, choose the **Split** button to duplicate the current query and split the page into two side-by-side queries.

> **Note**
>
> It is possible to select another data source for the new query which, for example, allows you to compare the same query for two different servers or to compare the staging environment to the production environment.

In split view, timepickers for both panels can be linked (if you change one, the other gets changed as well) by selecting a time-sync button attached to one of the timepickers. Linking timepickers keeps the start and the end times of the split view queries in sync. It ensures that you're looking at the same time interval in both split panels.

2. To close the newly created query, choose the **Close Split** button.

Content outline

The content outline is a side navigation bar that keeps track of the queries and visualizations you created in Explore. It allows you to navigate between them quickly.

The content outline also works in a split view. When you are in split view, the content outline is generated for each pane.

To open the content outline

1. Select the **Outline** button in the top left corner of the **Explore** screen.
2. Select any panel icon in the content outline to navigate to that panel.

Share Explore URLs

When using Explore, the URL in the browser address bar updates as you make changes to the queries. You can share or bookmark this URL.

> **Note**
>
> Explore may generate relatively long URLs. You can also generate and share a shortened link if the URL is too long for your tools.

Generating Explore URLs from external tools

Because Explore URLs have a defined structure, you can build a URL from external tools and open it in Grafana. The url structure is:

```
http://<workspace_url>/explore?
panes=<panes>&schemaVersion=<schema_version>&orgId=<org_id>
```

where:

- `org_id` is the organization ID
- `schema_version` is the schema version (should be set to the latest version, which is 1.
- `panes` is a url-encoded JSON object of panes, where each key is the pane ID and each value is an object matching the following schema:

```
{
  datasource: string; // the pane's root datasource UID, or `-- Mixed --` for mixed
 datasources
  queries: {
    refId: string; // an alphanumeric identifier for this query, must be unique
 within the pane, i.e. "A", "B", "C", etc.
    datasource: {
      uid: string; // the query's datasource UID ie: "AD7864H6422"
      type: string; // the query's datasource type-id, i.e: "loki"
    }
    // ... any other datasource-specific query parameters
  }[]; // array of queries for this pane
  range: {
    from: string; // the start time, in milliseconds since epoch
    to: string; // the end time, in milliseconds since epoch
  }
}
```

> **Note**
>
> The `from` and `to` fields also accept relative ranges as described in the Setting dashboard time range topic.

Share shortened link

The Share shortened link capability allows you to create smaller and simpler URLs of the format /goto/:uid instead of using longer URLs with query parameters. To create a shortened link to the query results, select the **Share** option in the Explore toolbar. A shortened link that is never used will automatically get deleted after seven (7) days. If a link is used at least once, it won't get deleted.

Sharing shortened links with absolute time

Short links have two options: keeping relative time (for example, from two hours ago to the current time) or absolute time (for example, from 8am to 10am). Sharing a shortened link by default will copy the time range selected, relative or absolute. Choosing the dropdown button next to the share shortened link button and selecting one of the options under **Time-Sync URL Links** will allow you to create a short link with the absolute time, meaning anyone receiving the link will see the same data you are seeing, even if they open the link at another time. This option will not affect the time range selected in your Explore view.

Query management in Explore

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can manage the queries that you have created in Explore, including a history of queries that you have run, and queries that you have starred.

Query history

Query history is a list of queries that you used in Explore. The history is stored in the Grafana database and it is not shared with other users. The retention period for queries in history is two weeks. Queries older than two weeks are automatically deleted. To open and interact with your history, select the **Query history** button in Explore.

> **Note**
>
> Starred (favorited) queries are not subject to the two weeks retention period and they are not deleted.

View query history

Query history lets you view the history of your querying. For each individual query, you can:

- Run the query.
- Create and/or edit a comment.
- Copy a query to the clipboard.
- Copy a shortened link with the query to the clipboard.
- Star (favorite) a query.

Manage favorite queries

All queries that have been starred in the Query history tab are displayed in the Starred tab. This allows you to access your favorite queries faster and to reuse these queries without typing them from scratch.

Sorting query history

By default, query history shows you the most recent queries. You can sort your history by date or by data source name in ascending or descending order.

To sort your query history

1. Select the **Sort queries by** field.
2. Select one of the following options:
   - **Newest first**
   - **Oldest first**

Filtering query history

You can filter your query history in Query history and Starred tab to a specific data source.

To filter history to a data source

1. Select the **Filter queries for specific data source(s)** field.
2. Select the data source for which you would like to filter your history. You can select multiple data sources.

> **Note**
>
> Queries ran using the Mixed data source will appear only when filtering for Mixed, and not when filtering by their individual data sources.

In the **Query history** tab it is also possible to filter queries by date using the slider:

- Use the vertical slider to filter queries by date.
- Adjust the start date by dragging the bottom handle.
- Adjust the end date by dragging the top handle.

Searching in query history

You can search in your history across queries and your comments. Search is possible for queries in the Query history tab and Starred tab.

To search in query history

1. Select the **Search queries** field.
2. Enter the term you are searching for into search field.

Query history settings

You can customize the query history in the Settings tab. Options are described in the following table.

Setting	Default value
Change the default active tab	Query history tab

Note

Query history settings are global, and applied to both panels in split mode.

Logs in Explore

This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Explore allows you to investigate your logs in different data sources, including:

- OpenSearch
- Amazon CloudWatch
- InfluxDB
- Loki

During an infrastructure monitoring and incident response, you can dig deeper into the metrics and logs to find the cause. Explore also allows you to correlate logs with other telemetry signals such as metrics, traces, or profiles, by viewing them side-by-side. You can also add correlations to external URLs directly from Explore, enabling seamless navigation between data sources and external systems.

The logs visualization panel provides improved log rendering with a field selector component for customizing which fields are displayed. Loki queries in Explore support forward direction search, allowing you to search logs from oldest to newest.

The results of log queries are displayed as individual log lines and as a graph showing the logs volume for the selected time period.

Logs volume

When working with data sources that support a full range logs volume, Explore automatically displays a graph showing the log distribution for all the entered log queries. This feature is currently supported by OpenSearch and Loki data sources.

> **Note**
>
> In Loki, this full range log volume is rendered by a metric query which can be expensive depending on the time range queried. This query can be particularly challenging to process for smaller Loki installations. To mitigate this, you can use a proxy like nginx in front of Loki to set a custom timeout (for example, 10 seconds) for these queries. Log volume histogram queries can be identified by looking for queries with the HTTP header `X-Query-Tags` with value `Source=logvolhist`; these headers are added by Grafana to all log volume histogram queries.

If the data source does not support loading the full range logs volume, the logs model calculates a time series by counting log rows and organizing them into buckets based on an automatically calculated time interval. The timestamp of the first log row is used to anchor the start of the logs volume in the results. The end of the time series is anchored to the time picker's **To** range. This way, you can still analyze and visualize log data efficiently even when the data source doesn't offer full range support.

Logs

In the following sections, you will find detailed explanations of how to visualize and interact with individual logs in Explore.

Logs navigation

Logs navigation, at the right side of the log lines, can be used to easily request additional logs. You can do this by clicking the **Older logs** button at the bottom of the navigation. This is especially useful when you reach the line limit and you want to see more logs. Each request that is run from the navigation is then displayed in the navigation as separate page. Every page shows `from` and `to` timestamps of the incoming log lines. You can see previous results by clicking on each page. Explore caches the last five requests run from the logs navigation, so you're not re-running the same queries when clicking on the pages, saving time and resources.

Visualization options

You can customize how logs are displayed and select which columns are shown.

Option	Description
Time	Shows or hides the time column. This is the timestamp associated with the log line as reported from the data source.
Unique labels	Shows or hides the unique labels column that includes only non-common labels. All common labels are displayed above.
Wrap lines	Set this to `true` if you want the display to use line wrapping. If set to `false`, it will result in horizontal scrolling.
Prettify JSON	Set this to `true` to pretty print all JSON logs. This setting does not affect logs in any format other than JSON.
Deduplication	Log data can be very repetitive and Explore can help by hiding duplicate log lines. There are a few different deduplication algorithm s that you can use. **Exact** matches are done on the whole line except for date fields. **Numbers** matches are done on the line after stripping out numbers such as durations, IP addresses, and so on. **Signature** is the most aggressive deduplication as it strips all letters and numbers and matches on the remaining whitespace and punctuation.
Display results order	You can change the order of received logs from the default descending order (newest first) to ascending order (oldest first).

Download log lines

To download log results in either `txt` or `json` format, use the **Download** button. This feature allows you to save the log data for further analysis or to share it with others in a convenient and accessible format.

Log result meta information

Above the received log lines you can find essential meta information, including:

- **Number of received logs** – Indicates the total count of logs received for the current query or time range.
- **Error** – Displays possible error in your log results.
- **Common labels** – Shows common labels.
- **Total bytes processed** – Represents the cumulative size of the log data processed in bytes.

> **Note**
>
> The availability of certain meta information may depend on the data source, and as a result, you may only see some of these details for specific data sources.

Escaping newlines

Explore automatically detects some incorrectly escaped sequences in log lines, such as newlines (`\n`, `\r`) or tabs (`\t`). When it detects such sequences, Explore provides an **Escape newlines** option.

To automatically fix incorrectly escaped sequences that Explore has detected

1. Choose **Escape newlines** to replace the sequences.
2. Manually review the replacements to confirm their correctness.

Explore replaces these sequences. When it does so, the option will change from **Escape newlines** to **Remove escaping**. Evaluate the changes as the parsing may not be accurate based on the input received. You can revert the replacements by selecting **Remove escaping**.

Log level

For the logs where a `level` label is specified, we use the value of this label to determine the log level and update color of each log line accordingly. If the log doesn't have specified level label, we try to find out if its content matches any of the supported expressions (see the following table for more information). The log level is always determined by the first match. In the case where Grafana is not able to infer a log level field, it will be visualized with an unknown log level.

> **Note**
>
> If you use a Loki data source and the `level` is part of your log line, you can use parsers (JSON, logfmt, regex,..) to extract the level information into a level label that is used to determine the level value. This will allow the histogram to show the various log levels as separate bars.

Supported log levels and mapping of log level abbreviation and expressions:

Log level	Color	Supported expressions
critical	purple	emerg, fatal, alert, crit, critical
error	red	err, eror, error
warning	yellow	warn, warning
info	green	info, information, informati onal, notice
debug	blue	dbug, debug
trace	light blue	trace
unknown	grey	*

Highlight searched words

When your query includes specific words or expressions to search for, Explore will highlight these in the log lines for better visibility. This highlighting feature makes it easier to identify and focus on the relevant content in your logs.

> **Note**
>
> The ability to highlight search words may vary depending on the data source. For some data sources, the highlighting of search words may not be available.

Log details view

In Explore, each log line has an expandable section called *Log details* that can be opened by choosing the log line. The Log details view provides additional information and exploration options in the form of *Fields* and *Links* attached to the log lines, enabling a more robust interaction and analysis.

Fields

Within the Log details view, you can filter displayed fields in two ways: a positive filter (to focus on an specific field) and a negative filter (to exclude certain fields). These filters will update the corresponding query that produced the log line, adding equality and inequality expressions accordingly. If the data source has support, as is the case for Loki and OpenSearch, log details will check if the field is already present in the current query showing and active state (for positive filters only), allowing you to toggle it off the query, or changing the filter expression from positive to negative.

You can select a subset of fields to visualize in the logs list instead of the complete log line by clicking on the eye icon. Each field has a stats icon to display statistics in relation to all displayed logs.

Links

Grafana offers the functionality of data links or correlations, enabling you to convert any part of a log message into an internal or external link. These links can be used to navigate to related data or external resources, providing a seamless and convenient way to explore further information.

Log context

Log context displays additional lines of context surrounding a log entry that matches a particular search query. This can be helpful in understanding the log entry's context, and is similar to the `-C` parameter in the `grep` command.

You may encounter long lines of text that make it difficult to read and analyze the context around each log entry. This is where the **Wrap lines** toggle can come in handy. By enabling this toggle, Grafana will automatically wrap long lines of text so that they fit within the visible width of the viewer. This can make it easier to read and understand the log entries.

The **Open in split view** button allows you to execute the context query for a log entry in a split screen in the Explore view. Choosing this button will open a new Explore pane with the context query displayed alongside the log entry, making it easier to analyze and understand the surrounding context.

The log context query can also be opened in a new browser tab by pressing the `Ctrl` (or `Cmd`) key while choosing the button to open the context modal. When opened in a new tab, the previously selected filters are applied as well.

Copy log line

You can easily copy the content of a selected log line to your clipboard by choosing the **Copy log line** button.

Copy link to log line

Linking of log lines in Grafana allows you to quickly navigate to specific log entries for precise analysis. By choosing the **Copy shortlink** button for a log line, you can generate and copy a short URL that provides direct access to the exact log entry within an absolute time range. When you open the link, Grafana will automatically scroll to the corresponding log line and highlight it with a blue background, making it easy to identify and focus on the relevant information.

> **Note**
>
> This is only supported in Loki and other data sources that provide an `id` field.

Tracing in Explore

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can use Explore to visualize traces from tracing data sources. The legacy Tempo Search and Loki Search interfaces have been removed in favor of the TraceQL-powered search experience.

The following data sources are supported.

- Tempo (supported ingestion formats: OpenTelemetry, Jaeger, and Zipkin)
- Jaeger
- AWS X-Ray
- Zipkin
- OpenSearch

For information on how to configure queries for the data sources listed above, refer to the documentation for specific data source.

Query editor

You can query and search tracing data using a data source's query editor.

Each data source can have its own query editor. The query editor for the Tempo data source is slightly different than the query editor for the Jaeger data source.

For information on querying each data source, refer to their documentation.

- Tempo
- Jaeger
- AWS X-Ray
- Zipkin

- OpenSearch

Trace View explanation

This section explains the elements of the Trace View dashboard.

Header

The header of the trace view has the following elements:

- Header title – Shows the name of the root span and trace ID.
- Search – Highlights spans containing the searched text.
- Metadata – Various metadata about the trace.

Minimap

Shows condensed view or the trace timeline. Drag your pointer over the minimap to zoom into smaller time range. Zooming will also update the main timeline, so it is easy to see shorter spans. Hovering over the minimap, when zoomed, will show the **Reset Selection** button which resets the zoom.

Span filters

Using span filters, you can filter your spans in teh trace timeline viewer. The more filters you add, the more specific are the filtered spans.

You can add one or more of the following filters:

- Resource service name
- Span name
- Duration
- Tags (which include tags, process tags, and log fields)

To show only the spans you have matched, choose the **Show matches only** toggle.

Timeline

Shows list of spans within the trace. Each span row consists of these components:

- Expand children button – Expands or collapses all the children spans of the selected span.
- Service name – Name of the service that logged the span.
- Operation name – Name of the operation that this span represents.
- Span duration bar – Visual representation of the operation duration within the trace.

Span details

Choosing the span row shows span details, including the following.

- Operation name
- Span metadata
- Tags – Any tags associated with this span.
- Process metadata – Metadata about the process that logged this span.
- Logs – List of logs logged by this span and associated key values. In case of Zipkin logs section shows Zipkin annotations.

Trace to logs

You can navigate from a span in a trace view directly to logs relevant for that span. This is available for Tempo, Jaeger, and Zipkin data sources. Refer to their relevant documentation for instructions on how to configure each data source.

Choose the document icon to open a split view in Explore with the configured data source and query relevant logs for the span.

Trace to metrics

> **Note**
>
> This feature is currently in beta

You can navigate form a span in a trace view directly to metrics relevant for that span. This feature is available for Tempo, Jaeger, and Zipkin data sources. Refer to their relevant documentation for details on configuration.

Trace to profiles

Using Trace to profiles, you can use Grafana's ability to correlate different signales by adding the functionality to link between traces and profiles.

Node graph

You can optionally expand the node graph for the displayed trace. Depending on the data source, this can show spans of the trace as nodes in the graph, or add some additional context, including the service graph based on the current trace.

Service Graph view

The Service Graph view visualizes the span metrics (traces data for rates, error rates, and durations (RED)) and service graphs. Once the requirements are set up, this pre-configured view is immediately available.

For more information, see Tempo data source page. You can also see the service graph view page in the *Grafana Labs Tempo documentation*.

Data API

This visualization needs a specific shape of the data to be returned from the data source in order to correctly display it.

The data source needs to return data frame and set `frame.meta.preferredVisualisationType = 'trace'`.

Data frame structure

Required fields;

Field name	Type	Description
traceID	string	Identifier for the entire trace. There should be only one trace in the data frame.
spanID	string	Identifier for the current span. SpanIDs should be unique per trace.
parentSpanID	string	SpanID of the parent span to create child parent relations

Field name	Type	Description
		hip in the trace view. Can be `undefined` for root span without a parent.
serviceName	string	Name of the service this span is part of.
serviceTags	TraceKeyValuePair[]	List of tags relevant for the service.
startTime	number	Start time of the span in millisecond epoch time.
duration	number	Duration of the span in milliseconds.

Optional fields:

Field name	Type	Description
logs	TraceLog[]	List of logs associated with the current span.
tags	TraceKeyValuePair[]	List of tags associated with the current span.
warnings	string[]	List of warnings associated with the current span.
stackTraces	string[]	List of stack traces associated with the current span.
errorIconColor	string	Color of the error icon in case span is tagged with `error: true`.

For details about the types see TraceSpanRow, TraceKeyValuePair and TraceLog on GitHub.

Correlations editor in Explore

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Correlations allow users to build a link between any two data sources. For more information, including an overview of correlations, see Correlations in Grafana version 12.

Creating a correlation

You can create correlations from the Explore page.

To create a correlation

1. In your Amazon Managed Grafana workspace, navigate to the Explore page.
2. Select a data source that you would like to be the source for a new correlation.
3. Run a query producing data in a supported visualization.

   Note

   Supported visualizations are Logs and Table.

4. Choose **+ Add** in the top toolbar and select **Add correlation** (you can also select **Correlations Editor** from the Command Palette.

   Explore is now in Correlations Editor mode indicated by a blue border and top bar. You can exit Correlations Editor by choosing **Exit** in the top bar.
5. You can now create the following new correlations for the visualization with links that are attached to the data that you can use to build a new query:

   - *Logs* – links are displayed next to field values inside log details for each log row.

- *Table* – every table cell is a link.

6. Choose a link to add a new correlation. Links are associated with a field that is used as a result field of a correlation. For more details, see Correlation configuration.
7. In the split view that opens, use the right pane to set up the target query source of the correlation. For more details, see Target query.
8. Build a target query using variables syntax with variables from the list provided at the top of the pane. The list contains sample values from the selected data row.
9. Provide a label and description (optional). A label will be used as the name of the link inside the visualization and can contain variables.
10. Provide transformations (optional; see below for details).
11. Choose **Save** in the top toolbar to save the correlation and exit Correlations Editor mode. The link used to create the correlation is replaced with a data link in each row. When the link is selected, the query you defined will run in another pane, with the variables replaced dynamically with the values from the selected row.

Transformations

Transformations allow you to extract values that exist in a field with other data. For example, using a transformation, you can extract one portion of a log line to use in a correlation. For more details on transformations in correlations, see Correlation Transformations.

After choosing one of the generated links in the editor mode, you can add transformations by selecting **Add transformation** in the **Transformations** dropdown menu.

To use a transformation in a correlation

1. Select a field to apply the transformation to. Select the portion of the field that you want to use for the transformation. For example, a log line. Once selected, the value of this field will be used to assist you in building the transformation.
2. Select the type of the transformation. See Correlation Transformations for the options and relevant settings.
3. Based on your selection, you might see one or more variables populate, or you might need to provide more specifications in options that are displayed.
4. Select **Add transformation to correlation** to add the specified variables to the list of available variables.

> **Note**
>
> For regular expressions in this dialog box, the `mapValue` referred to in other documentation is called `Variable Name` here. Grafana highlights any text that matches the expression in the field value. Use regular expression capture groups to select what portion of the match should be extracted. When a valid regular expression is provided, the variable and the value of that variable appear below the `Variable Name` field.

Correlations examples

The following examples show how to create correlations using the Correlations Editor in Explore. If you'd like to follow these examples, make sure to set up a test data source.

Creating a text to graph correlation

This example shows how to create a correlation using Correlations Editor in Explore.

Correlations allow you to use results of one query to run a new query in any data source. In this example, you will run a query that renders tabular data. The data will be used to run a different query that yields a graph result.

To follow this example, make sure you have set up a test data source.

To create a text to graph correlation

1. In Grafana, navigate to **Explore**.
2. Select the **test data source** from the dropdown menu at the top left of the page.
3. Choose **+ Add** in the dropdown menu to the right and select **Add correlation**.
4. Explore is now in Correlations Editor mode, indicated by a blue border.
5. Select the following scenario from the scenario dropdown menu: **CSV File**.
6. Select the file, **population_by_state.csv**. Each cell is a link that you can click on to begin creating a new correlation.
7. Click on any cell in the `State` column to create a new correlation that attaches a data link to that entry. For example, select `California`.
8. In the split view, select the same data source you selected in the left pane. The helper above the query editor contains all available variables you can use the target query. Variables contain all data fields (table columns) from the selected row.

9. In the **Scenario** menu, select **CSV Metric Values**. The `String Input` field in the Query editor provides variables with population values for each year: `${1980},${2000},${2020}`. This will generate a graph using variable values.
10. In the Query Editor **Alias** field, enter `${State}`.

    Run a query to see that it produces a graph using sample values from the variables.
11. Choose **Save** to save the correlation and exit the Correlations Editor.

    After the correlation is saved, Explore will rerun the query in the left pane. By clicking a state name, the query on the right is rerun with values from the row being inserted into the CSV, thus changing the graph. The query is rerun with updated values every time you click on a state name.

You can apply the same steps to any data source. Correlations allow you to create links in visualizations to run dynamic queries based on selected data. In this example we used data returned by a query to build a new query generating different visualization using the same data source. However, you can create correlations between any data sources to create custom exploration flows.

Creating a logs to table correlation

In this example, you will create a correlation to demonstrate how to use transformations to extract values from the log line and another field.

To follow this example, make sure you have set up a test data source.

To create a logs to table correlation

1. In Grafana, navigate to **Explore**.
2. Select the **test data source** from the dropdown menu at the top left of the page.
3. Choose **+ Add** in the dropdown menu to the right and select **Add correlation**.
4. Explore is now in Correlations Editor mode, indicated by a blue border.
5. In the **Scenario** menu, select **Logs**.
6. Expand a log line to see the correlation links. Select `Correlate with hostname`.
7. Explore opens in split view. Select the same data source you selected in the left pane. The helper above the query editor contains all available variables you can use the target query.
8. Expand the transformations section, and click **Add transformation**.

9. In the **Field** dropdown menu, select **message**. The log line shows up as example data.
10. Under **Type**, select **Logfmt**. This populates the list of variables.
11. Choose **Add transformation to correlation**.
12. Choose **Add transformation** again and under **Field**, select **hostname**.
13. Under **Type**, select **Regular expression**.
14. Under **Expression**, enter the following: `-([0-9]\*)`. This selects any numbers to the right of the dash.
15. Under **Variable Name**, enter the following: `hostNumber`. This populates the list of variables.
16. Choose **Add transformation to correlation** to add it to the other variables.
17. In the data source editor, open the **Scenario** dropdown menu and select **CSV Content**.
18. In the text box below, provide the following and save the correlation:

```
time,msg,hostNumber,status
${time},${msg},${hostNumber},${status}
```

This closes the split view and reruns the left query. Expand any log line to see the correlation button. Chooseing the correlation button opens the split view with the `time` (a field), `msg` (extracted with *logfmt* from the log line), `host number` (extracted with *regex* from the `hostname`) and the `status` (extracted with *logfmt* from the log line).

Inspector in Explore

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The inspector helps you understand and troubleshoot your queries. You can inspect the raw data, export that data to a comma-separated values (CSV) file, export log results in TXT format, and view query requests.

Inspector UI

The inspector has the following tabs:

- **Stats tab** – Shows how long your query takes and how much it returns.
- **Query tab** – Shows you the requests to the server sent when Grafana queries the data source.
- **JSON tab** – Allows you to view and copy the data JSON and data frame structure JSON.
- **Data tab** – Shows the raw data returned by the query.
- **Error tab** – Shows the error. Only visible when query returns error.

Inspector tasks

You can perform a variety of tasks in the Explore inspector.

Open the Inspector

After you run the query you would like to inspect, select the **Inspector** button.

The inspector pane opens on the bottom of the screen.

Inspect raw query results

You can view raw query results, that is the data returned by the query in a table.

In the **Inspector** tab, click the **Data** tab.

For multiple queries or for queries multiple nodes, there are additional options.

- **Show data frame:** Select the result set data you want to view.
- **Series joined by time:** View the raw data from all of your queries at once, one result set per column. You can click a column heading to sort the data.

Download raw query results as CSV

Grafana generates a CSV file in your default browser download location. You can open it in the viewer of your choice.

1. In the **Inspector** tab, get raw query results by following the instructions above.
2. Refine the query settings until you can see the raw data that you want to export.
3. Choose **Download CSV**.

In order to download a CSV file specifically formatted for Excel, expand **Data options** and then turn on the **Download for Excel** toggle before you select the **Download CSV** option.

Download log results as TXT

You can generate a TXT file of the logs you are currently viewing, by selecting **Download logs** in the **Inspector**tab.

Download trace results

Based on the data source type, Grafana generates a JSON file for the trace results in one of the supported formats: Jaeger, Zipkin, or OTLP formats.

1. Open the Inspector.
2. Inspect the log query results. Refine the results until you see the raw logs that you want to export.
3. Choose **Download logs**.

Inspect query performance

The **Stats** tab displays statistics that tell you how long your query takes, how many queries you send, and the number of rows returned. This information can help you troubleshoot your queries, especially if any of the numbers are unexpectedly high or low.

Statistics are displayed in read-only format.

View JSON model

You can explore and export data as well as data frame JSON models.

To view the JSON model

1. In the Inspector panel, click the **JSON** tab.
2. From the **Select source** dropdown, choose one of the following options:
   - **Data** – Displays a JSON object representing the data that was returned to Explore.
   - **DataFrame structure** – Displays the raw result set.
3. You can expand or collapse portions of the JSON to view separate sections. You can also select the **Copy to clipboard** option to copy JSON body and paste it into another application.

View raw request and response to data source

As you are working with Explore and the Inspector tab, you can view the raw request and response data that you are generating with a query. In the Inspector, select the **Query** tab and choose **Refresh** to see the raw data.

Grafana sends the query to the server and displays the result. You can drill down on specific portions of the query, expand or collapse all of it, or copy the data to the clipboard to use in other applications.

Drilldown apps in Grafana version 12

The Grafana Drilldown apps provide a queryless experience for exploring telemetry signals. Instead of writing complex queries, you can use intuitive point-and-click interactions to explore your metrics, logs, and traces data.

> **Note**
>
> Drilldown apps are introduced in Amazon Managed Grafana version 12. The Drilldown apps were previously known as the Explore apps. For example, Explore Logs is now Logs Drilldown, and Explore Metrics is now Metrics Drilldown.

The following Drilldown apps are available in Amazon Managed Grafana version 12:

Topics

- Metrics Drilldown
- Logs Drilldown
- Traces Drilldown
- Profiles Drilldown

Metrics Drilldown

The Metrics Drilldown app provides a queryless, point-and-click experience for exploring Prometheus metric data. You can quickly find related metrics without needing to write PromQL queries.

Key capabilities include:

- Browse and discover metrics without writing queries.
- OpenTelemetry filtering support that automates label joins for non-promoted resource attributes.
- Native histogram support for higher-resolution visualizations.
- Pivot from metrics to related logs to correlate signals.
- Entry point directly from alerting rules.

To access Metrics Drilldown, choose **Drilldown** and then **Metrics** from the Grafana navigation menu.

Logs Drilldown

The Logs Drilldown app provides a queryless experience for browsing Loki logs. You can discover or narrow down your search by using volume and text patterns without needing to compose LogQL queries.

Key capabilities include:

- Visualize log volumes to detect anomalies or significant changes over time.
- Customizable top-level filtering by any indexed label.
- Regex filtering for pattern matching in raw log lines.
- Log patterns for identifying common log structures.
- JSON viewer for structured log lines.
- Infinite scrolling to browse large log volumes without pagination.
- Shareable links to specific log lines or result sets.

To access Logs Drilldown, choose **Drilldown** and then **Logs** from the Grafana navigation menu.

Traces Drilldown

The Traces Drilldown app provides a queryless experience for exploring distributed Tempo traces. You can use Rate, Errors, and Duration (RED) metrics derived from traces to investigate and understand errors and latency issues within complex distributed systems.

Key capabilities include:

- Navigate from high-level RED signals to detailed span views.
- Use exemplars to jump from metrics to individual traces.
- Built-in filtering, comparisons, and root cause correlation.
- Transition between traces, logs, metrics, and profiles for a unified observability workflow.

To access Traces Drilldown, choose **Drilldown** and then **Traces** from the Grafana navigation menu.

Profiles Drilldown

The Profiles Drilldown app integrates with Pyroscope, the open-source continuous profiling platform, providing a queryless experience for browsing and analyzing profiling data.

Key capabilities include:

- View and analyze high-level service performance.
- Identify problem processes for optimization.
- Diagnose issues to determine root causes using flame graphs.

> **Note**
>
> The Profiles Drilldown app requires a Pyroscope data source to be configured in your workspace.

To access Profiles Drilldown, choose **Drilldown** and then **Profiles** from the Grafana navigation menu.

Correlations in Grafana version 12

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can create interactive links for Explore visualizations to run queries related to presented data by setting up Correlations.

A correlation defines how data in one data source is used to query data in another data source. Some examples:

- An application name returned in a logs data source can be used to query metrics related to that application in a metrics data source.
- A user name returned by an SQL data source can be used to query logs related to that particular user in a logs data source.

Explore takes user-defined correlations to display links inside the visualizations. You can click on a link to run the related query and see results in Explore Split View.

Explore visualizations that currently support showing links based on correlations:

- Logs
- Table

You can configure correlations using the **Administration > Plugins and data > Correlations** page in Grafana or directly in Explore. You can also add correlations to external URLs directly from Explore, enabling seamless navigation between data sources and external systems.

Topics

- Correlation configuration
- Create a new correlation

Correlation configuration

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Each correlation is configured with the following options:

Label

Link label, shown in the visualization.

Description

Optional description.

Source data source

The source of results that have links displayed.

Results field

Defines where the link is shown in a visualization.

Target query

The target query run when a link is clicked.

Transformations

Optional manipulations applied to the source data before it is passed to the target query.

For details about creating a correlation, see Create a correlation.

Source data source and result field

Links are shown in Explore visualizations for the results from the correlation's source data source. A link is assigned to one of the fields from the result provided in the correlation configuration (the results field). Each visualization displays fields with links in a different way.

Target query

The target query is run when a link is clicked in the visualization. You can use the query editor of the selected target data source to specify the target query. Source data results can be accessed inside the target query with variables.

Correlation Variables

You can use variables inside the target query to access the source data related to the query. Correlations use Grafana variable syntax. Variables are filled with values from the source results when the link is chosen. There are two types of variables you can use:

- Field variables (to access field values and labels).
- Correlation variables (to access field values and transformations).

Example: If source results contain a field called `employee`, the value of the field can be accessed with:

- A field variable `${__data.fields.employee}`.
- A correlation variable that maps the field value above to `${employee}`.

In addition to mapping field values to shorter variable names, more correlation variables can be created by applying transformations to existing fields.

Correlation creates a data link only if all variables have values in the selected data row. Global variables are the exception to this rule and are not required to be filled in from the returned data. These variables are interpolated automatically by data sources.

Correlation Transformations

Transformations provide a way to extract more variables out of field values. The output of transformations is a set of new variables that can be accessed as any other variable.

There are two types of transformations: logfmt and regular expression.

Each transformation uses a selected field value as the input. The output of a transformation is a set of new variables based on the type and options of the transformation.

Logfmt transformation

The logfmt transformation deconstructs a field value containing text formatted with logfmt key/value pairs. Each pair becomes a variable with the key being the name of the variable.

The logfmt transformation only requires specifying the input field name if you would like the transformation to apply to a different field than the results field. Example output variables for `field = "host=srv001 endpoint=/test app=foo"`:

name	value
host	srv001

name	value
endpoint	/test
app	foo

Regular expression transformation

The regular expression transformation deconstructs a field value based on the provided regular expression.

Regular expression transformation options:

field

Input field name

expression

Regular expression. Named capture groups are mapped to variables matching the group name. If non-named matching groups are used a variable is created out of the first match. The value overrides the variable matching the input field or a new variable is created if mapValue is provided (see examples in the following table).

mapValue

Used with simple regex groups without named matching groups. By default, the first match overrides the variable with the name of the field that is used as the input. To change that default behavior you can specify the mapValue property. The provided name is used to create a new variable. This can be useful if your target query requires both the exact value and a part of the value extracted with the transformation.

Example: Assuming the selected field name is `employee` and the field value is `John Doe`.

Various output variables based on expression and mapValue options:

expression	mapValue	output variables	comment
`/\w+ (\w+)/`	-	`employee=Doe`	No mapValue provided. The first

expression	mapValue	output variables	comment
			matching is mapped to the existing field name variable (`employee`).
`/(\w+) (\w+)/`	name	`name=John`	The first matching is mapped to a new variable called `name`.
`/(?\w+) (?\w+)/`	-	`firstName=John , lastName=Doe`	When named groups are used they are the names of the output variables and mapValue is ignored.
`/(?\w+) (?\w+)/`	name	`firstName=John , lastName=Doe`	Same as above

Create a new correlation

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can create correlations in the Explore correlations editor, or using the Grafana **Administration** page in your Amazon Managed Grafana workspace.

Prerequisites

You must have permission to add new correlations. Only users with write permissions to data sources can define new correlations.

Creating a correlation in Explore's correlations editor

You can create a correlation in the Explore correlation editor. For more details, see Creating a correlation.

Creating a correlation in the Administration page

You can use the Grafana console **Administration** page to create a correlation.

To create a correlation in the Administration page

1. Go to the **Administration** section in Grafana.
2. Under **Plugins and data**, open the **Correlations** page.
3. Choose the **Add** button in the top right corner.
4. Provide a **label** for the correlation.
5. (Optional) Provide an **description**.
6. Go to the next page.
7. Provide **target data source**.
8. Provide **target query** using variables.
9. Go to the next page.
10. Provide **source data source**.
11. Provide **results field**.
12. Add transformations if you need variables that are not fields in the source data source.
13. Choose **Add** to add a new transformation.
14. Select the type of a transformation.
15. Configure transformation depending on the selected type.
16. Save the correlation.

You can edit a correlation in the same way, but when editing, you can't change the selected data sources.

Alerts in Grafana version 12

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana Alerting allows you to learn about problems in your systems moments after they occur.

Monitor your incoming metrics data or log entries and set up your Alerting system to watch for specific events or circumstances and then send notifications when those things are found.

In this way, you eliminate the need for manual monitoring and provide a first line of defense against system outages or changes that could turn into major incidents.

Using Grafana Alerting, you create queries and expressions from multiple data sources — no matter where your data is stored — giving you the flexibility to combine your data and alert on your metrics and logs in new and unique ways. You can then create, manage, and take action on your alerts from a single, consolidated view, and improve your team's ability to identify and resolve issues quickly.

With Mimir and Loki alert rules you can run alert expressions closer to your data and at massive scale, all managed by the Grafana UI you are already familiar with.

Note

The `$value` variable in alert notification templates now returns the query value when querying a single data source. Review alert templates that use `$value` and update formatting if needed.

Key features and benefits

One page for all alerts

A single Grafana Alerting page consolidates both Grafana-managed alerts and alerts that reside in your Prometheus-compatible data source in one single place.

Multi-dimensional alerts

Alert rules can create multiple individual alert instances per alert rule, known as multi-dimensional alerts, giving you the power and flexibility to gain visibility into your entire system with just a

single alert rule. You do this by adding labels to your query to specify which component is being monitored and generate multiple alert instances for a single alert rule. For example, if you want to monitor each server in a cluster, a multi-dimensional alert will alert on each CPU, whereas a standard alert will alert on the overall server.

Route alerts

Route each alert instance to a specific contact point based on labels you define. Notification policies are the set of rules for where, when, and how the alerts are routed to contact points.

Silence alerts

Silences stop notifications from getting created and last for only a specified window of time. Silences allow you to stop receiving persistent notifications from one or more alert rules. You can also partially pause an alert based on certain criteria. You can create silences that apply to specific alert rules with granular permissions, providing more targeted alert suppression. Silences have their own dedicated section for better organization and visibility, so that you can scan your paused alert rules without cluttering the main alerting view.

Mute timings

A mute timing is a recurring interval of time when no new notifications for a policy are generated or sent. Use them to prevent alerts from firing a specific and reoccurring period, for example, a regular maintenance period.

Similar to silences, mute timings do not prevent alert rules from being evaluated, nor do they stop alert instances from being shown in the user interface. They only prevent notifications from being created.

Design your Alerting system

Monitoring complex IT systems and understanding whether everything is up and running correctly is a difficult task. Setting up an effective alert management system is therefore essential to inform you when things are going wrong before they start to impact your business outcomes.

Designing and configuring an alert management set up that works takes time.

Here are some tips on how to create an effective alert management set up for your business:

Which are the key metrics for your business that you want to monitor and alert on?

- Find events that are important to know about and not so trivial or frequent that recipients ignore them.
- Alerts should only be created for big events that require immediate attention or intervention.
- Consider quality over quantity.

Which type of Alerting do you want to use?

- Choose between Grafana-managed Alerting or Grafana Mimir or Loki-managed Alerting; or both.

How do you want to organize your alerts and notifications?

- Be selective about who you set to receive alerts. Consider sending them to whomever is on call or a specific Slack channel.
- Automate as far as possible using the Alerting API or alerts as code (Terraform).

How can you reduce alert fatigue?

- Avoid noisy, unnecessary alerts by using silences, mute timings, or pausing alert rule evaluation.
- Continually tune your alert rules to review effectiveness. Remove alert rules to avoid duplication or ineffective alerts.
- Think carefully about priority and severity levels.
- Continually review your thresholds and evaluation rules.

Grafana alerting limitations

- When aggregating rules from other systems, the Grafana alerting system can retrieve rules from all available Amazon Managed Service for Prometheus, Prometheus, Loki, and Alertmanager data sources. It might not be able to fetch rules from other supported data sources.

> **Important**
>
> Amazon Managed Grafana has an alert evaluation timeout of 30 seconds. Queries made by alerts have a maximum duration of 30 seconds due to the high volume of queries the alert

engine can generate. This timeout is not configurable. For more information, see Amazon Managed Grafana service quotas in the *AWS General Reference*.

Topics

- Overview
- Set up Alerting
- Configure alerting
- Manage your alerts

Overview

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Whether you're just starting out or you're a more experienced user of Grafana Alerting, learn more about the fundamentals and available features that help you create, manage, and respond to alerts; and improve your team's ability to resolve issues quickly.

Principles

In Prometheus-based alerting systems, you have an alert generator that creates alerts and an alert receiver that receives alerts. For example, Prometheus is an alert generator and is responsible for evaluating alert rules, while Alertmanager is an alert receiver and is responsible for grouping, inhibiting, silencing, and sending notifications about firing and resolved alerts.

Grafana Alerting is built on the Prometheus model of designing alerting systems. It has an internal alert generator responsible for scheduling and evaluating alert rules, as well as an internal alert receiver responsible for grouping, inhibiting, silencing, and sending notifications. Grafana doesn't use Prometheus as its alert generator because Grafana Alerting needs to work with many other data sources in addition to Prometheus. However, it does use Alertmanager as its alert receiver.

Alerts are sent to the alert receiver where they are routed, grouped, inhibited, silenced and notified. In Grafana Alerting, the default alert receiver is the Alertmanager embedded inside Grafana, and is referred to as the Grafana Alertmanager. However, you can use other Alertmanagers too, and these are referred to as External Alertmanagers.

Fundamentals

The following provides an overview of the different parts of Grafana alerting.

Alert rules

An alert rule is a set of criteria that determine when an alert should fire. It consists of one or more queries and expressions, a condition which needs to be met, an interval which determines how often the alert rule is evaluated, and a duration over which the condition must be met for an alert to fire.

Alert rules are evaluated over their interval, and each alert rule can have zero, one, or any number of alerts firing at a time. The state of the alert rule is determined by its most `severe` alert, which can be one of Normal, Pending, or Firing. For example, if at least one of an alert rule's alerts are firing then the alert rule is also firing. The health of an alert rule is determined by the status of its most recent evaluation. These can be OK, Error, and NoData.

Alert rules can be configured to keep their last state when a query returns no data or an error, preventing unnecessary alert state changes. Alert rules also maintain a version history, allowing you to track changes and revert to previous configurations. Alert rules support a Recovering state that indicates when an alert condition is no longer firing but has not yet returned to normal.

A very important feature of alert rules is that they support custom annotations and labels. These allow you to instrument alerts with additional metadata such as summaries and descriptions, and add additional labels to route alerts to specific notification policies.

Alerts

Alerts are uniquely identified by sets of key/value pairs called Labels. Each key is a label name and each value is a label value. For example, one alert might have the labels `foo=bar` and another alert might have the labels `foo=baz`. An alert can have many labels such as `foo=bar,bar=baz` but it cannot have the same label twice such as `foo=bar,foo=baz`. Two alerts cannot have the same labels either, and if two alerts have the same labels such as `foo=bar,bar=baz` and `foo=bar,bar=baz` then one of the alerts will be discarded. Alerts are resolved when the condition in the alert rule is no longer met, or the alert rule is deleted.

In Grafana Managed Alerts, alerts can be in Normal, Pending, Alerting, No Data or Error states. In Data source Managed Alerts, such as Mimir and Loki, alerts can be in Normal, Pending and Alerting, but not NoData or Error.

Contact points

Contact points determine where notifications are sent. For example, you might have a contact point that sends notifications to an email address, to Slack, to an incident management system (IRM) such as Grafana OnCall or Pagerduty, or to a webhook.

The notifications that are sent from contact points can be customized using notification templates. You can use notification templates to change the title, message, and structure of the notification. Notification templates are not specific to individual integrations or contact points.

Notification policies

Notification policies group alerts and then route them to contact points. They determine when notifications are sent, and how often notifications should be repeated.

Alerts are matched to notification policies using label matchers. These are human-readable expressions that assert if the alert's labels exactly match, do not exactly match, contain, or do not contain some expected text. For example, the matcher `foo=bar` matches alerts with the label `foo=bar` while the matcher `foo=~[a-zA-Z]+` matches alerts with any label called foo with a value that matches the regular expression `[a-zA-Z]+`.

By default, an alert can only match one notification policy. However, with the `continue` feature alerts can be made to match any number of notification policies at the same time. For more information about notification policies, see Notification Policies.

Silences and mute timings

Silences and mute timings allow you to pause notifications for specific alerts or even entire notification policies. Use a silence to pause notifications on an ad-hoc basis, such as while working on a fix for an alert; and use mute timings to pause notifications at regular intervals, such as during regularly scheduled maintenance windows.

Topics

- Data sources and Grafana alerting
- Alerting on numeric data

- Labels and annotations
- About alert rules
- Alertmanager
- Contact points
- Notifications
- Alerting high availability

Data sources and Grafana alerting

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

There are a number of data sources that are compatible with Grafana Alerting. Each data source is supported by a plugin. Grafana alerting requires that data source plugins be *backend* plugins, in order to evaluate rules using the data source, because the evaluation engine runs on the backend. Plugins must also specify that they are compatible with Grafana alerting.

Data sources are added and updated over time. The following data sources are known to be compatible with Grafana alerting.

- Connect to an Amazon CloudWatch data source
- Connect to an Azure Monitor data source
- Connect to an Amazon OpenSearch Service data source
- Connect to a Google Cloud Monitoring data source
- Connect to a Graphite data source
- Connect to an InfluxDB data source
- Connect to a Loki data source
- Connect to a Microsoft SQL Server data source

- Connect to a MySQL data source
- Connect to an OpenTSDB data source
- Connect to a PostgreSQL data source
- Connect to Amazon Managed Service for Prometheus and open-source Prometheus data sources
- Connect to a Jaeger data source
- Connect to a Zipkin data source
- Connect to a Tempo data source
- Configure a TestData data source for testing

For more detailed information about data sources and data source plugins in Amazon Managed Grafana, see Connect to data sources.

Alerting on numeric data

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This topic describes how Grafana handles alerting on numeric rather than time series data.

Among certain data sources, numeric data that is not time series can be directly alerted on, or passed into Server Side Expressions (SSE). This allows for more processing and resulting efficiency within the data source, and it can also simplify alert rules. When alerting on numeric data instead of time series data, there is no need to reduce each labeled time series into a single number. Instead labeled numbers are returned to Grafana instead.

Tabular data

This feature is supported with backend data sources that query tabular data:

- SQL data sources such as MySQL, Postgres, MSSQL, and Oracle.

- The Azure Kusto based services: Azure Monitor (Logs), Azure Monitor (Azure Resource Graph), and Azure Data Explorer.

A query with Grafana managed alerts or SSE is considered numeric with these data sources, if:

- The “Format AS” option is set to “Table” in the data source query.
- The table response returned to Grafana from the query includes only one numeric (e.g. int, double, float) column, and optionally additional string columns.

If there are string columns, then those columns become labels. The name of a column becomes the label name, and the value for each row becomes the value of the corresponding label. If multiple rows are returned, then each row should be uniquely identified their labels.

Example

For a MySQL table called “DiskSpace”:

Time	Host	Disk	PercentFree
2021-June-7	web1	/etc	3
2021-June-7	web2	/var	4
2021-June-7	web3	/var	8
...	...	...	...

You can query the data filtering on time, but without returning the time series to Grafana. For example, an alert that would trigger per Host, Disk when there is less than 5% free space:

```
SELECT Host , Disk , CASE WHEN PercentFree  < 5.0 THEN PercentFree  ELSE 0 END FROM (
       SELECT
          Host,
          Disk,
          Avg(PercentFree)
       FROM DiskSpace
       Group By
          Host,
          Disk
```

```
        Where __timeFilter(Time)
```

This query returns the following Table response to Grafana:

Host	Disk	PercentFree
web1	/etc	3
web2	/var	4
web3	/var	0

When this query is used as the **condition** in an alert rule, then the non-zero will be alerting. As a result, three alert instances are produced:

Labels	Status
{Host=web1,disk=/etc}	Alerting
{Host=web2,disk=/var}	Alerting
{Host=web3,disk=/var}	Normal

Labels and annotations

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Labels and annotations contain information about an alert. Both labels and annotations have the same structure: a set of named values; however their intended uses are different. An example of label, or the equivalent annotation, might be `alertname="test"`.

The main difference between a label and an annotation is that labels are used to differentiate an alert from all other alerts, while annotations are used to add additional information to an existing alert.

For example, consider two high CPU alerts: one for `server1` and another for `server2`. In such an example, we might have a label called `server` where the first alert has the label `server="server1"` and the second alert has the label `server="server2"`. However, we might also want to add a description to each alert such as `"The CPU usage for server1 is above 75%."`, where `server1` and `75%` are replaced with the name and CPU usage of the server (please refer to the documentation on Templating labels and annotations for how to do this). This kind of description would be more suitable as an annotation.

Labels

Labels contain information that identifies an alert. An example of a label might be `server=server1`. Each alert can have more than one label, and the complete set of labels for an alert is called its label set. It is this label set that identifies the alert.

For example, an alert might have the label set `{alertname="High CPU usage",server="server1"}` while another alert might have the label set `{alertname="High CPU usage",server="server2"}`. These are two separate alerts because although their `alertname` labels are the same, their `server` labels are different.

The label set for an alert is a combination of the labels from the datasource, custom labels from the alert rule, and a number of reserved labels such as `alertname`.

Custom Labels

Custom labels are additional labels from the alert rule. Like annotations, custom labels must have a name, and their value can contain a combination of text and template code that is evaluated when an alert is fired. Documentation on how to template custom labels can be found here.

When using custom labels with templates, it is important to make sure that the label value does not change between consecutive evaluations of the alert rule as this will end up creating large numbers of distinct alerts. However, it is OK for the template to produce different label values for different alerts. For example, do not put the value of the query in a custom label as this will end up creating a new set of alerts each time the value changes. Instead use annotations.

It is also important to make sure that the label set for an alert does not have two or more labels with the same name. If a custom label has the same name as a label from the datasource then it

will replace that label. However, should a custom label have the same name as a reserved label then the custom label will be omitted from the alert.

Annotations

Annotations are named pairs that add additional information to existing alerts. There are a number of suggested annotations in Grafana such as `description`, `summary`, `runbook_url`, `dashboardUId` and `panelId`. Like custom labels, annotations must have a name, and their value can contain a combination of text and template code that is evaluated when an alert is fired. If an annotation contains template code, the template is evaluated once when the alert is fired. It is not re-evaluated, even when the alert is resolved. Documentation on how to template annotations can be found here.

Topics

- How label matching works
- Labels in Grafana Alerting
- Templating labels and annotations

How label matching works

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use labels and label matchers to link alert rules to notification policies and silences. This allows for a very flexible way to manage your alert instances, specify which policy should handle them, and which alerts to silence.

A label matcher consists of 3 distinct parts, the **label**, the **value** and the **operator**.

- The **Label** field is the name of the label to match. It must exactly match the label name.
- The **Value** field matches against the corresponding value for the specified **Label** name. How it matches depends on the **Operator** value.

- The **Operator** field is the operator to match against the label value. The available operators are:

Operator	Description
=	Select labels that are exactly equal to the value.
!=	Select labels that are not equal to the value.
=~	Select labels that regex-match the value.
!~	Select labels that do not regex-match the value.

If you are using multiple label matchers, they are combined using the AND logical operator. This means that all matchers must match in order to link a rule to a policy.

Example

If you define the following set of labels for your alert:

```
{ foo=bar, baz=qux, id=12 }
```

then:

- A label matcher defined as `foo=bar` matches this alert rule.
- A label matcher defined as `foo!=bar` does *not* match this alert rule.
- A label matcher defined as `id=~[0-9]+` matches this alert rule.
- A label matcher defined as `baz!~[0-9]+` matches this alert rule.
- Two label matchers defined as `foo=bar` and `id=~[0-9]+` match this alert rule.

Exclude labels

You can also write label matchers to exclude labels.

Here is an example that shows how to exclude the label `team`. You can choose between any of these values to exclude the label.

- `team=""`
- `team!~.+`
- `team=~^$`

Labels in Grafana Alerting

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This topic explains why labels are a fundamental component of alerting.

- The complete set of labels for an alert is what uniquely identifies an alert within Grafana alerts.
- The Alertmanager uses labels to match alerts for silences and alert groups in notification policies.
- The alerting UI shows labels for every alert instance generated during evaluation of that rule.
- Contact points can access labels to dynamically generate notifications that contain information specific to the alert that is resulting in a notification.
- You can add labels to an alerting rule. Labels are manually configurable, use template functions, and can reference other labels. Labels added to an alerting rule take precedence in the event of a collision between labels (except in the case of Grafana reserved labels, see below for more information).

External Alertmanager compatibility

Grafana's built-in Alertmanager supports both Unicode label keys and values. If you are using an external Prometheus Alertmanager, label keys must be compatible with their data model. This means that label keys must only contain **ASCII letters**, **numbers**, as well as **underscores** and match the regex `[a-zA-Z_][a-zA-Z0-9_]*`. Any invalid characters will be removed or replaced by the Grafana alerting engine before being sent to the external Alertmanager according to the following rules:

- `Whitespace` will be removed.
- `ASCII characters` will be replaced with _.
- `All other characters` will be replaced with their lower-case hex representation. If this is the first character it will be prefixed with _.

> **Note**
>
> If multiple label keys are sanitized to the same value, the duplicates will have a short hash of the original label appended as a suffix.

Grafana reserved labels

> **Note**
>
> Labels prefixed with `grafana_` are reserved by Grafana for special use. If a manually configured label is added beginning with `grafana_` it will be overwritten in case of collision.

Grafana reserved labels can be used in the same way as manually configured labels. The current list of available reserved labels are:

Label	Description
grafana_folder	Title of the folder containing the alert.

Templating labels and annotations

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can use templates to include data from queries and expressions in labels and annotations. For example, you might want to set the severity label for an alert based on the value of the query, or use the instance label from the query in a summary annotation so you know which server is experiencing high CPU usage.

All templates should be written in text/template. Regardless of whether you are templating a label or an annotation, you should write each template inline inside the label or annotation that you are templating. This means you cannot share templates between labels and annotations, and instead you will need to copy templates wherever you want to use them.

Each template is evaluated whenever the alert rule is evaluated, and is evaluated for every alert separately. For example, if your alert rule has a templated summary annotation, and the alert rule has 10 firing alerts, then the template will be executed 10 times, once for each alert. You should try to avoid doing expensive computations in your templates as much as possible.

Examples

Rather than write a complete tutorial on text/template, the following examples attempt to show the most common use-cases we have seen for templates. You can use these examples verbatim, or adapt them as necessary for your use case. For more information about how to write text/template see the text/template documentation.

Print all labels, comma separated

To print all labels, comma separated, print the `$labels` variable:

```
{{ $labels }}
```

For example, given an alert with the labels `alertname=High CPU usage`, `grafana_folder=CPU alerts` and `instance=server1`, this would print:

```
alertname=High CPU usage, grafana_folder=CPU alerts, instance=server1
```

> **Note**
>
> If you are using classic conditions then `$labels` will not contain any labels from the query. Refer to the $labels variable for more information.

Print all labels, one per line

To print all labels, one per line, use a `range` to iterate over each key/value pair and print them individually. Here `$k` refers to the name and `$v` refers to the value of the current label:

```
{{ range $k, $v := $labels -}}
{{ $k }}={{ $v }}
{{ end }}
```

For example, given an alert with the labels `alertname=High CPU usage`, `grafana_folder=CPU alerts` and `instance=server1`, this would print:

```
alertname=High CPU usage
grafana_folder=CPU alerts
instance=server1
```

> **Note**
>
> If you are using classic conditions then `$labels` will not contain any labels from the query. Refer to the $labels variable for more information.

Print an individual label

To print an individual label use the `index` function with the `$labels` variable:

```
The host {{ index $labels "instance" }} has exceeded 80% CPU usage for the last 5
 minutes
```

For example, given an alert with the label `instance=server1`, this would print:

```
The host server1 has exceeded 80% CPU usage for the last 5 minutes
```

> **Note**
>
> If you are using classic conditions then `$labels` will not contain any labels from the query. Refer to the $labels variable for more information.

Print the value of a query

To print the value of an instant query you can print its Ref ID using the `index` function and the `$values` variable:

```
{{ index $values "A" }}
```

For example, given an instant query that returns the value 81.2345, this will print:

```
81.2345
```

To print the value of a range query you must first reduce it from a time series to an instant vector with a reduce expression. You can then print the result of the reduce expression by using its Ref ID instead. For example, if the reduce expression takes the average of A and has the Ref ID B you would write:

```
{{ index $values "B" }}
```

Print the humanized value of a query

To print the humanized value of an instant query use the `humanize` function:

```
{{ humanize (index $values "A").Value }}
```

For example, given an instant query that returns the value 81.2345, this will print:

```
81.234
```

To print the humanized value of a range query you must first reduce it from a time series to an instant vector with a reduce expression. You can then print the result of the reduce expression by using its Ref ID instead. For example, if the reduce expression takes the average of A and has the Ref ID B you would write:

```
{{ humanize (index $values "B").Value }}
```

Print the value of a query as a percentage

To print the value of an instant query as a percentage use the `humanizePercentage` function:

```
{{ humanizePercentage (index $values "A").Value }}
```

This function expects the value to be a decimal number between 0 and 1. If the value is instead a decimal number between 0 and 100 you can divide it by 100 either in your query or using a math expression. If the query is a range query you must first reduce it from a time series to an instant vector with a reduce expression.

Set a severity from the value of a query

To set a severity label from the value of a query use an if statement and the greater than comparison function. Make sure to use decimals (`80.0`, `50.0`, `0.0`, etc) when doing comparisons against `$values` as text/template does not support type coercion. You can find a list of all the supported comparison functions here.

```
{{ if (gt $values.A.Value 80.0) -}}
high
{{ else if (gt $values.A.Value 50.0) -}}
medium
{{ else -}}
low
{{- end }}
```

Print all labels from a classic condition

You cannot use `$labels` to print labels from the query if you are using classic conditions, and must use `$values` instead. The reason for this is classic conditions discard these labels to enforce uni-dimensional behavior (at most one alert per alert rule). If classic conditions didn't discard these labels, then queries that returned many time series would cause alerts to flap between firing and resolved constantly as the labels would change every time the alert rule was evaluated.

Instead, the `$values` variable contains the reduced values of all time series for all conditions that are firing. For example, if you have an alert rule with a query A that returns two time series, and a classic condition B with two conditions, then `$values` would contain `B0`, `B1`, `B2` and `B3`. If the classic condition B had just one condition, then `$values` would contain just `B0` and `B1`.

To print all labels of all firing time series use the following template (make sure to replace `B` in the regular expression with the Ref ID of the classic condition if it's different):

```
{{ range $k, $v := $values -}}
{{ if (match "B[0-9]+" $k) -}}
{{ $k }}: {{ $v.Labels }}{{ end }}
{{ end }}
```

For example, a classic condition for two time series exceeding a single condition would print:

```
B0: instance=server1
B1: instance=server2
```

If the classic condition has two or more conditions, and a time series exceeds multiple conditions at the same time, then its labels will be duplicated for each condition that is exceeded:

```
B0: instance=server1
B1: instance=server2
B2: instance=server1
B3: instance=server2
```

If you need to print unique labels you should consider changing your alert rules from uni-dimensional to multi-dimensional instead. You can do this by replacing your classic condition with reduce and math expressions.

Print all values from a classic condition

To print all values from a classic condition take the previous example and replace `$v.Labels` with `$v.Value`:

```
{{ range $k, $v := $values -}}
{{ if (match "B[0-9]+" $k) -}}
{{ $k }}: {{ $v.Value }}{{ end }}
{{ end }}
```

For example, a classic condition for two time series exceeding a single condition would print:

```
B0: 81.2345
B1: 84.5678
```

If the classic condition has two or more conditions, and a time series exceeds multiple conditions at the same time, then `$values` will contain the values of all conditions:

```
B0: 81.2345
B1: 92.3456
B2: 84.5678
B3: 95.6789
```

Variables

The following variables are available to you when templating labels and annotations:

The labels variable

The `$labels` variable contains all labels from the query. For example, suppose you have a query that returns CPU usage for all of your servers, and you have an alert rule that fires when any of your servers have exceeded 80% CPU usage for the last 5 minutes. You want to add a summary annotation to the alert that tells you which server is experiencing high CPU usage. With the `$labels` variable you can write a template that prints a human-readable sentence such as:

```
CPU usage for {{ index $labels "instance" }} has exceeded 80% for the last 5 minutes
```

> **Note**
>
> If you are using a classic condition then `$labels` will not contain any labels from the query. Classic conditions discard these labels in order to enforce uni-dimensional behavior (at most one alert per alert rule). If you want to use labels from the query in your template then follow the previous *Print all labels from a classic condition* example.

The value variable

The `$value` variable is a string containing the labels and values of all instant queries; threshold, reduce and math expressions, and classic conditions in the alert rule. It does not contain the results of range queries, as these can return anywhere from 10s to 10,000s of rows or metrics. If it did, for especially large queries a single alert could use 10s of MBs of memory and Grafana would run out of memory very quickly.

To print the `$value` variable in the summary you would write something like this:

```
CPU usage for {{ index $labels "instance" }} has exceeded 80% for the last 5 minutes:
 {{ $value }}
```

And it would look something like this:

```
CPU usage for instance1 has exceeded 80% for the last 5 minutes: [ var='A'
 labels={instance=instance1} value=81.234 ]
```

Here `var='A'` refers to the instant query with Ref ID A, `labels={instance=instance1}` refers to the labels, and `value=81.234` refers to the average CPU usage over the last 5 minutes.

If you want to print just some of the string instead of the full string then use the `$values` variable. It contains the same information as `$value`, but in a structured table, and is much easier to use then writing a regular expression to match just the text you want.

The values variable

The `$values` variable is a table containing the labels and floating point values of all instant queries and expressions, indexed by their Ref IDs.

To print the value of the instant query with Ref ID A:

```
CPU usage for {{ index $labels "instance" }} has exceeded 80% for the last 5 minutes:
 {{ index $values "A" }}
```

For example, given an alert with the labels `instance=server1` and an instant query with the value `81.2345`, this would print:

```
CPU usage for instance1 has exceeded 80% for the last 5 minutes: 81.2345
```

If the query in Ref ID A is a range query rather than an instant query then add a reduce expression with Ref ID B and replace `(index $values "A")` with `(index $values "B")`:

```
CPU usage for {{ index $labels "instance" }} has exceeded 80% for the last 5 minutes:
 {{ index $values "B" }}
```

Functions

The following functions are available to you when templating labels and annotations:

args

The `args` function translates a list of objects to a map with keys arg0, arg1 etc. This is intended to allow multiple arguments to be passed to templates.

```
{{define "x"}}{{.arg0}} {{.arg1}}{{end}}{{template "x" (args 1 "2")}}
```

```
1 2
```

externalURL

The `externalURL` function returns the external URL of the Grafana server.

```
{{ externalURL }}
```

```
https://example.com/grafana
```

graphLink

The `graphLink` function returns the path to the graphical view in Explore in Grafana version 12 for the given expression and data source.

```
{{ graphLink "{\"expr\": \"up\", \"datasource\": \"gdev-prometheus\"}" }}
```

```
/explore?left=["now-1h","now","gdev-prometheus",{"datasource":"gdev-
prometheus","expr":"up","instant":false,"range":true}]
```

humanize

The `humanize` function humanizes decimal numbers.

```
{{ humanize 1000.0 }}
```

```
1k
```

humanize1024

The `humanize1024` works similar to `humanize` but uses 1024 as the base rather than 1000.

```
{{ humanize1024 1024.0 }}
```

```
1ki
```

humanizeDuration

The `humanizeDuration` function humanizes a duration in seconds.

```
{{ humanizeDuration 60.0 }}
```

```
1m 0s
```

humanizePercentage

The `humanizePercentage` function humanizes a ratio value to a percentage.

```
{{ humanizePercentage 0.2 }}
```

```
20%
```

humanizeTimestamp

The `humanizeTimestamp` function humanizes a Unix timestamp.

```
{{ humanizeTimestamp 1577836800.0 }}
```

```
2020-01-01 00:00:00 +0000 UTC
```

match

The `match` function matches the text against a regular expression pattern.

```
{{ match "a.*" "abc" }}
```

```
true
```

pathPrefix

The `pathPrefix` function returns the path of the Grafana server.

```
{{ pathPrefix }}
```

```
/grafana
```

tableLink

The `tableLink` function returns the path to the tabular view in Explore in Grafana version 12 for the given expression and data source.

```
{{ tableLink "{\"expr\": \"up\", \"datasource\": \"gdev-prometheus\"}" }}
```

```
/explore?left=["now-1h","now","gdev-prometheus",{"datasource":"gdev-
prometheus","expr":"up","instant":true,"range":false}]
```

title

The `title` function capitalizes the first character of each word.

```
{{ title "hello, world!" }}
```

```
Hello, World!
```

toLower

The `toLower` function returns all text in lowercase.

```
{{ toLower "Hello, world!" }}
```

```
hello, world!
```

toUpper

The `toUpper` function returns all text in uppercase.

```
{{ toUpper "Hello, world!" }}
```

```
HELLO, WORLD!
```

reReplaceAll

The `reReplaceAll` function replaces text matching the regular expression.

```
{{ reReplaceAll "localhost:(.*)" "example.com:$1" "localhost:8080" }}
```

```
example.com:8080
```

About alert rules

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

An alerting rule is a set of evaluation criteria that determines whether an alert instance will fire. The rule consists of one or more queries and expressions, a condition, the frequency of evaluation, and the duration over which the condition needs to be met to start firing.

While queries and expressions select the data set to evaluate, a *condition* sets the threshold that the data must meet or exceed to create an alert.

An *interval* specifies how frequently an alerting rule is evaluated. *Duration*, when configured, indicates how long a condition must be met. The alert rules can also define alerting behavior in the absence of data.

Topics

- Alert rule types
- Recording rules
- Queries and conditions
- Alert instances
- Namespaces, folders and groups

- Alert rule evaluation
- State and health of alerting rules
- Notification templating

Alert rule types

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana supports several alert rule types. Learn more about each of the alert rule types, how they work, and decide which one is best for your use case.

Grafana managed rules

Grafana managed rules are the most flexible alert rule type. They allow you to create alerts that can act on data from any of your existing data sources.

In addition to supporting multiple data sources, you can add expressions to transform your data and express alert conditions.

In Grafana managed alerting:

- Alert rules are created within Grafana, based on one or more data sources.
- Alert rules are evaluated by the alert rule evaluation engine from within Grafana.
- Alerts are delivered using the internal Grafana Alertmanager.

Note

You can also configure alerts to be delivered using an external Alertmanager, or use both internal and external Alertmanagers. For more information, see Add an external alertmanager.

Data source managed rules

To create data source managed alert rules you must have a compatible Prometheus or Loki data source. You can check if your data source supports rule creation via Grafana by testing the data source and observing if the Ruler API is supported.

In data source managed alerting:

- Alert rules are created and stored within the data source itself.
- Alert rules can only be created based on Prometheus data.
- Alert rule evaluation and delivery is distributed across multiple nodes for high-availability and fault tolerance.

Choose an alert rule type

When choosing which alert rule type to use, consider the following comparison between Grafana managed alert rules and data source managed alert rules.

Feature	Grafana-managed alert rule	Loki/Mimir-managed alert rule
Create alert rules based on data from any of our supported data sources	Yes	No: You can only create alert rules that are based on Prometheus data. The data source must have the Ruler API enabled.
Mix and match data sources	Yes	No
Includes support for recording rules	No	Yes
Add expressions to transform your data and set alert conditions	Yes	No
Use images in alert notificat ions	Yes	No

Feature	Grafana-managed alert rule	Loki/Mimir-managed alert rule
Scaling	More resource intensive, depend on the database, and are likely to suffer from transient errors. They only scale vertically.	Store alert rules within the data source itself and allow for “infinite” scaling. Generate and send alert notifications from the location of your data.
Alert rule evaluation and delivery	Alert rule evaluation and delivery is done from within Grafana, using an external Alertmanager; or both.	Alert rule evaluation and alert delivery is distributed, meaning there is no single point of failure.

Recording rules

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Recording rules are available for compatible Prometheus or Loki data sources, and for Grafana-managed alerts. Recording rules allow you to precompute frequently used queries for better performance.

A recording rule allows you to pre-compute frequently needed or computationally expensive expressions and save their result as a new set of time series. This is useful if you want to run alerts on aggregated data or if you have dashboards that query computationally expensive expressions repeatedly.

Querying this new time series is faster, especially for dashboards since they query the same expression every time the dashboards refresh.

Read more about recording rules in Prometheus.

Queries and conditions

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

In Grafana, queries play a vital role in fetching and transforming data from supported data sources, which include databases like MySQL and PostgreSQL, time series databases like Prometheus, InfluxDB and Graphite, and services like OpenSearch, Amazon CloudWatch, Azure Monitor and Google Cloud Monitoring.

For more information on supported data sources, see Data sources and Grafana alerting.

The process of executing a query involves defining the data source, specifying the desired data to retrieve, and applying relevant filters or transformations. Query languages or syntaxes specific to the chosen data source are utilized for constructing these queries.

In Alerting, you define a query to get the data you want to measure and a condition that needs to be met before an alert rule fires.

An alert rule consists of one or more queries and expressions that select the data you want to measure.

For more information on queries and expressions, see Query and transform data.

Data source queries

Queries in Grafana can be applied in various ways, depending on the data source and query language being used. Each data source's query editor provides a customized user interface that helps you write queries that take advantage of its unique capabilities.

Because of the differences between query languages, each data source query editor looks and functions differently. Depending on your data source, the query editor might provide auto-completion features, metric names, variable suggestions, or a visual query-building interface.

Some common types of query components include:

Metrics or data fields – Specify the specific metrics or data fields you want to retrieve, such as CPU usage, network traffic, or sensor readings.

Time range – Define the time range for which you want to fetch data, such as the last hour, a specific day, or a custom time range.

Filters – Apply filters to narrow down the data based on specific criteria, such as filtering data by a specific tag, host, or application.

Aggregations – Perform aggregations on the data to calculate metrics like averages, sums, or counts over a given time period.

Grouping – Group the data by specific dimensions or tags to create aggregated views or breakdowns.

> **Note**
>
> Grafana does not support alert queries with template variables. More information is available here in the Grafana Labs forums.

Expression queries

In Grafana, an expression is used to perform calculations, transformations, or aggregations on the data source queried data. It allows you to create custom metrics or modify existing metrics based on mathematical operations, functions, or logical expressions.

By leveraging expression queries, users can perform tasks such as calculating the percentage change between two values, applying functions like logarithmic or trigonometric functions, aggregating data over specific time ranges or dimensions, and implementing conditional logic to handle different scenarios.

In Alerting, you can only use expressions for Grafana-managed alert rules. For each expression, you can choose from the math, reduce, and resample expressions. These are called multi-dimensional rules, because they generate a separate alert for each series.

You can also use a classic condition, which creates an alert rule that triggers a single alert when its condition is met. As a result, Grafana sends only a single alert even when alert conditions are met for multiple series.

> **Note**
>
> Classic conditions exist mainly for compatibility reasons and should be avoided if possible.

Reduce

Aggregates time series values in the selected time range into a single value.

Math

Performs free-form math functions/operations on time series and number data. Can be used to preprocess time series data or to define an alert condition for number data.

Resample

Realigns a time range to a new set of timestamps, this is useful when comparing time series data from different data sources where the timestamps would otherwise not align.

Threshold

Checks if any time series data matches the threshold condition.

The threshold expression allows you to compare two single values. It returns `0` when the condition is false and 1 if the condition is true. The following threshold functions are available:

- Is above (x > y)
- Is below (x < y)
- Is within range (x > y1 AND x < y2)
- Is outside range (x < y1 AND x > y2)

Classic condition

Checks if any time series data matches the alert condition.

> **Note**
>
> Classic condition expression queries always produce one alert instance only, no matter how many time series meet the condition. Classic conditions exist mainly for compatibility reasons and should be avoided if possible.

Aggregations

Grafana Alerting provides the following aggregation functions to enable you to further refine your query.

These functions are available for **Reduce** and **Classic condition** expressions only.

Function	Expression	What it does
avg	Reduce / Classic	Displays the average of the values
min	Reduce / Classic	Displays the lowest value
max	Reduce / Classic	Displays the highest value
sum	Reduce / Classic	Displays the sum of all values
count	Reduce / Classic	Counts the number of values in the result
last	Reduce / Classic	Displays the last value
median	Reduce / Classic	Displays the median value
diff	Classic	Displays the difference between the newest and oldest value
diff_abs	Classic	Displays the absolute value of diff
percent_diff	Classic	Displays the percentage value of the difference between newest and oldest value
percent_diff_abs	Classic	Displays the absolute value of percent_diff
count_non_null	Classic	Displays a count of values in the result set that aren't `null`

Alert condition

An alert condition is the query or expression that determines whether the alert will fire or not depending on the value it yields. There can be only one condition which will determine the triggering of the alert.

After you have defined your queries and/or expressions, choose one of them as the alert rule condition.

When the queried data satisfies the defined condition, Grafana triggers the associated alert, which can be configured to send notifications through various channels like email, Slack, or PagerDuty. The notifications inform you about the condition being met, allowing you to take appropriate actions or investigate the underlying issue.

By default, the last expression added is used as the alert condition.

Recovery threshold

To reduce the noise of flapping alerts, you can set a recovery threshold different to the alert threshold.

Flapping alerts occur when a metric hovers around the alert threshold condition and may lead to frequent state changes, resulting in too many notifications being generated.

Grafana-managed alert rules are evaluated for a specific interval of time. During each evaluation, the result of the query is checked against the threshold set in the alert rule. If the value of a metric is above the threshold, an alert rule fires and a notification is sent. When the value goes below the threshold and there is an active alert for this metric, the alert is resolved, and another notification is sent.

It can be tricky to create an alert rule for a noisy metric. That is, when the value of a metric continually goes above and below a threshold. This is called flapping and results in a series of firing - resolved - firing notifications and a noisy alert state history.

For example, if you have an alert for latency with a threshold of 1000ms and the number fluctuates around 1000 (say 980 ->1010 -> 990 -> 1020, and so on) then each of those will trigger a notification.

To solve this problem, you can set a (custom) recovery threshold, which basically means having two thresholds instead of one. An alert is triggered when the first threshold is crossed and is resolved only when the second threshold is crossed.

For example, you could set a threshold of 1000ms and a recovery threshold of 900ms. This way, an alert rule will only stop firing when it goes under 900ms and flapping is reduced.

Alert instances

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana managed alerts support multi-dimensional alerting. Each alert rule can create multiple alert instances. This is powerful if you are observing multiple series in a single expression.

Consider the following PromQL expression:

```
sum by(cpu) (
  rate(node_cpu_seconds_total{mode!="idle"}[1m])
)
```

A rule using this expression will create as many alert instances as the amount of CPUs we are observing after the first evaluation, allowing a single rule to report the status of each CPU.

Namespaces, folders and groups

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Alerts can be organized using folders for Grafana managed rules and namespaces for Mimir, Loki, or Prometheus rules and group names.

Namespaces and folders

When creating Grafana-managed rules, the folder can be used to perform access control and grant or deny access to all rules within a specific folder.

A namespace contains one or more groups. The rules within a group are run sequentially at a regular interval. The default interval is one minute. You can rename Grafana Mimi or Loki rule namespaces and groups, and edit group evaluation intervals.

Groups

The rules within a group are run sequentially at a regular interval, meaning no rules will be evaluated at the same time, and in order of appearance. The default interval is one minute. You can rename Grafana Mimir or Loki rule namespaces or Loki rule namespaces and groups, and edit group evaluation intervals.

Tip

If you want rules to be evaluated concurrently and with different intervals, consider storing them in different groups.

Note

Grafana managed alert rules are evaluated concurrently instead of sequentially.

Alert rule evaluation

This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use alert rule evaluation to determine how frequently an alert rule should be evaluated and how quickly it should change its state.

To do this, you need to make sure that your alert rule is in the right evaluation group and set a pending period time that works best for your use case.

Evaluation group

Every alert rule is part of an evaluation group. Each evaluation group contains an evaluation interval that determines how frequently the alert rule is checked.

Data-source managed alert rules within the same group are evaluated one after the other, while alert rules in different groups can be evaluated simultaneously. This feature is especially useful when you want to ensure that recording rules are evaluated before any alert rules.

Grafana-managed alert rules are evaluated at the same time, regardless of alert rule group. The default evaluation interval is set at 10 seconds, which means that Grafana-managed alert rules are evaluated every 10 seconds to the closest 10-second window on the clock, for example, 10:00:00, 10:00:10, 10:00:20, and so on. You can also configure your own evaluation interval, if required.

> **Note**
>
> Evaluation groups and alerts grouping in notification policies are two separate things. Grouping in notification policies allows multiple alerts sharing the same labels to be sent in the same time message.

Pending period

By setting a pending period, you can avoid unnecessary alerts for temporary problems.

In the pending period, you select the period in which an alert rule can be in breach of the condition until it fires.

Example

Imagine you have an alert rule evaluation interval set at every 30 seconds and the pending period to 90 seconds.

Evaluation will occur as follows:

[00:30] First evaluation - condition not met.

[01:00] Second evaluation - condition breached. Pending counter starts.**Alert starts pending.**

[01:30] Third evaluation - condition breached. Pending counter = 30s. **Pending state.**

[02:00] Fourth evaluation - condition breached. Pending counter = 60s **Pending state.**

[02:30] Fifth evaluation - condition breached. Pending counter = 90s. **Alert starts firing**

If the alert rule has a condition that needs to be in breach for a certain amount of time before it takes action, then its state changes as follows:

- When the condition is first breached, the rule goes into a "pending" state.
- The rule stays in the "pending" state until the condition has been broken for the required amount of time - pending period.
- Once the required time has passed, the rule goes into a "firing" state.
- If the condition is no longer broken during the pending period, the rule goes back to its normal state.

> **Note**
>
> If you want to skip the pending state, you can simply set the pending period to 0. This effectively skips the pending period and your alert rule will start firing as soon as the condition is breached.

When an alert rule fires, alert instances are produced, which are then sent to the Alertmanager.

State and health of alerting rules

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The state and health of alerting rules help you understand several key status indicators about your alerts.

There are three key components: *alert rule state*, *alert instance state*, and *alert rule health*. Although related, each component conveys subtly different information.

> **Important**
>
> **Alert rule evaluation result limit** – Beginning in Grafana version 12, the number of query evaluation results per alert rule is limited to 500. If the condition query of an alert rule produces more results than this limit, the evaluation results in an error. To receive alerts when this error occurs, configure `Alert state if execution error or timeout` to `Error` under the alert rule's settings.

Alert rule state

An alert rule can be in one of the following states:

State	Description
Normal	None of the time series returned by the evaluation engine is in a `Pending` or `Firing` state.
Pending	At least one time series returned by the evaluation engine is `Pending`.
Firing	At least one time series returned by the evaluation engine is `Firing`.
Recovering	The alert condition is no longer firing but has not yet returned to normal.

> **Note**
>
> Alerts will transition first to `pending` and then `firing`, thus it will take at least two evaluation cycles before an alert is fired.

Alert instance state

An alert instance can be in one of the following states:

State	Description
Normal	The state of an alert that is neither firing nor pending, everything is working correctly.
Pending	The state of an alert that has been active for less than the configured threshold duration.
Alerting	The state of an alert that has been active for longer than the configured threshold duration.
Recovering	The state of an alert that was previously firing but the alert condition is no longer met. The alert has not yet returned to normal.
NoData	No data has been received for the configured time window.
Error	The error that occurred when attempting to evaluate an alerting rule.

Keep last state

An alert rule can be configured to keep the last state when `NoData` or `Error` state is encountered. This will both prevent alerts from firing, and from resolving and re-firing. Just like normal evaluation, the alert rule will transition from `Pending` to `Firing` after the pending period has elapsed.

Alert rule health

An alert rule can have one the following health statuses:

State	Description
Ok	No error when evaluating an alerting rule.

State	Description
Error	An error occurred when evaluating an alerting rule.
NoData	The absence of data in at least one time series returned during a rule evaluation.

Special alerts for `NoData` and `Error`

When evaluation of an alerting rule produces state `NoData` or `Error`, Grafana Alerting will generate alert instances that have the following additional labels:

Label	Description
alertname	Either `DatasourceNoData` or `DatasourceError` depending on the state.
datasource_uid	The UID of the data source that caused the state.

You can handle these alerts the same way as regular alerts by adding a silence, route to a contact point, and so on.

Notification templating

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Notifications sent via contact points are built using notification templates. Grafana's default templates are based on the Go templating system where some fields are evaluated as text, while others are evaluated as HTML (which can affect escaping).

The default template default_template.go is a useful reference for custom templates.

Since most of the contact point fields can be templated, you can create reusable custom templates and use them in multiple contact points. To learn about custom notifications using templates, see Customize notifications.

Nested templates

You can embed templates within other templates.

For example, you can define a template fragment using the `define` keyword.

```
{{ define "mytemplate" }}
  {{ len .Alerts.Firing }} firing. {{ len .Alerts.Resolved }} resolved.
{{ end }}
```

You can then embed custom templates within this fragment using the `template` keyword. For example:

```
Alert summary:
{{ template "mytemplate" . }}
```

You can use any of the following built-in template options to embed custom templates.

Name	Notes
`default.title`	Displays high-level status information.
`default.message`	Provides a formatted summary of firing and resolved alerts.
`teams.default.message`	Similar to `default.messsage` , formatted for Microsoft Teams.

HTML in notification templates

HTML in alerting notification templates is escaped. We do not support rendering of HTML in the resulting notification.

Some notifiers support alternative methods of changing the look and feel of the resulting notification. For example, Grafana installs the base template for alerting emails to `<grafana-install-dir>/public/emails/ng_alert_notification.html`. You can edit this file to change the appearance of all alerting emails.

Alertmanager

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Alertmanager enables you to quickly and efficiently manage and respond to alerts. It receives alerts, handles mutings, inhibition, grouping, and routing by sending notifications out via your channel of choice, for example, email or Slack.

In Grafana, you can use the Grafana Alertmanager or an external Alertmanager. You can also run multiple alertmanagers; your decision depends on your set up and where your alerts are being generated.

Grafana Alertmanager

Grafana Alertmanager is an internal Alertmanager that is pre-configured and available for selection by default.

The Grafana Alertmanager can receive alerts from Grafana, but it cannot receive alerts from outside Grafana, for example, from Mimir or Loki.

> **Note**
> Inhibition rules are not supported in the Grafana Alertmanager.

External Alertmanager

If you want to use a single alertmanager to receive all your Grafana, Loki, Mimir, and Prometheus alerts, you can set up Grafana to use an external Alertmanager. This external Alertmanager can be configured and administered from within Grafana itself.

Here are two examples of when you might want to configure your own external alertmanager and send your alerts there instead of the Grafana Alertmanager:

1. You already have alertmanagers on-premise in your own Cloud infrastructure that you have set up and still want to use, because you have other alert generators, such as Prometheus.
2. You want to use both Prometheus on-premise and hosted Grafana to send alerts to the same alertmanager that runs in your Cloud infrastructure.

Alertmanagers are visible from the dropdown menu on the Alerting Contact Points, and Notification Policies pages.

If you are provisioning your data source, set the flag `handleGrafanaManagedAlerts` in the `jsonData` field to `true` to send Grafana-managed alerts to this Alertmanager.

Contact points



This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Contact points contain the configuration for sending notifications. A contact point is a list of integrations, each of which sends a notification to a particular email address, service or URL. Contact points can have multiple integrations of the same kind, or a combination of integrations of different kinds. For example, a contact point could contain a Pagerduty integration; an Amazon SNS and Slack integration; or a Pagerduty integration, a Slack integration, and two Amazon SNS integrations. You can also configure a contact point with no integrations; in which case no notifications are sent.

A contact point cannot send notifications until it has been added to a notification policy. A notification policy can only send alerts to one contact point, but a contact point can be added to a number of notification policies at the same time. When an alert matches a notification policy, the alert is sent to the contact point in that notification policy, which then sends a notification to each integration in its configuration.

Contact points can be configured for the Grafana Alertmanager as well as external alertmanagers.

You can also use notification templating to customize notification messages for contact point types.

Supported contact point types

The following table lists the contact point types supported by Grafana.

Name	Type
Amazon SNS	`sns`
OpsGenie	`opsgenie`
Pager Duty	`pagerduty`
Slack	`slack`
VictorOps	`victorops`

For more information about contact points, see Configure contact points and Customize notifications.

Notifications

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Choosing how, when, and where to send your alert notifications is an important part of setting up your alerting system. These decisions will have a direct impact on your ability to resolve issues quickly and not miss anything important.

As a first step, define your contact points, which define where to send your alert notifications. A contact point is a set of one or more integrations that are used to deliver notifications. Add notification templates to contact points for reuse and consistent messaging in your notifications.

Next, create a notification policy which is a set of rules for where, when and how your alerts are routed to contact points. In a notification policy, you define where to send your alert notifications by choosing one of the contact points you created.

Alertmanagers

Grafana uses Alertmanagers to send notifications for firing and resolved alerts. Grafana has its own Alertmanager, referred to as **Grafana** in the user interface, but also supports sending notifications from other Alertmanagers too, such as the Prometheus Alertmanager. The Grafana Alertmanager uses notification policies and contact points to configure how and where a notification is sent; how often a notification should be sent; and whether alerts should all be sent in the same notification, sent in grouped notifications based on a set of labels, or as separate notifications.

Notification policies

Notification policies control when and where notifications are sent. A notification policy can choose to send all alerts together in the same notification, send alerts in grouped notifications based on a set of labels, or send alerts as separate notifications. You can configure each notification policy to control how often notifications should be sent as well as having one or more mute timings to inhibit notifications at certain times of the day and on certain days of the week.

Notification policies are organized in a tree structure where at the root of the tree there is a notification policy called the default policy. There can be only one default policy and the default policy cannot be deleted.

Specific routing policies are children of the root policy and can be used to match either all alerts or a subset of alerts based on a set of matching labels. A notification policy matches an alert when its matching labels match the labels in the alert.

A nested policy can have its own nested policies, which allow for additional matching of alerts. An example of a nested policy could be sending infrastructure alerts to the Ops team; while a child policy might send high priority alerts to Pagerduty and low priority alerts to Slack.

All alerts, irrespective of their labels, match the default policy. However, when the default policy receives an alert it looks at each nested policy and sends the alert to the first nested policy that matches the alert. If the nested policy has further nested policies, then it can attempt to match the alert against one of its nested policies. If no nested policies match the alert then the policy itself is the matching policy. If there are no nested policies, or no nested policies match the alert, then the default policy is the matching policy.

For more detailed information about notification policies, see Notification policies.

Notification templates

You can customize notifications with templates. For example, templates can be used to change the title and message of notifications sent to Slack.

Templates are not limited to an individual integration or contact point, but instead can be used in a number of integrations in the same contact point and even integrations across different contact points. For example, a Grafana user can create a template called `custom_subject_or_title` and use it for both templating subjects in Pager Duty and titles of Slack messages without having to create two separate templates.

All notifications templates are written in Go's templating language, and are in the Contact points tab on the Alerting page.

For more detailed information about customizing notifications, see Customize notifications.

Silences

You can use silences to mute notifications from one or more firing rules. Silences do not stop alerts from firing or being resolved, or hide firing alerts in the user interface. A silence lasts as long as its duration which can be configured in minutes, hours, days, months or years.

For more detailed information about using silences, see Silencing alert notifications.

Notification policies

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Notification policies provide you with a flexible way of routing alerts to various different receivers. Using label matchers, you can modify alert notification delivery without having to update every individual alert rule.

In this section, you will learn more about how notification policies work and are structured, so that you can make the most out of setting up your notification policies.

Policy tree

Notification policies are *not* a list, but rather are structured according to a tree structure. This means that each policy can have child policies, and so on. The root of the notification policy tree is called the **Default notification policy**.

Each policy consists of a set of label matchers (0 or more) that specify which labels they are or aren't interested in handling.

For more information about label matching, see How label matching works.

Note

If you haven't configured any label matchers for your notification policy, your notification policy will match *all* alert instances. This may prevent child policies from being evaluated unless you have enabled **Continue matching siblings** on the notification policy.

Routing

To determine which notification policy will handle which alert instances, you have to start by looking at the existing set of notification policies, starting with the default notification policy.

If no policies other than the default policy are configured, the default policy will handle the alert instance.

If policies other than the default policy are defined, it will evaluate those notification policies in the order they are displayed.

If a notification policy has label matchers that match the labels of the alert instance, it will descend in to its child policies and, if there are any, will continue to look for any child policies that might have label matchers that further narrow down the set of labels, and so forth until no more child policies have been found.

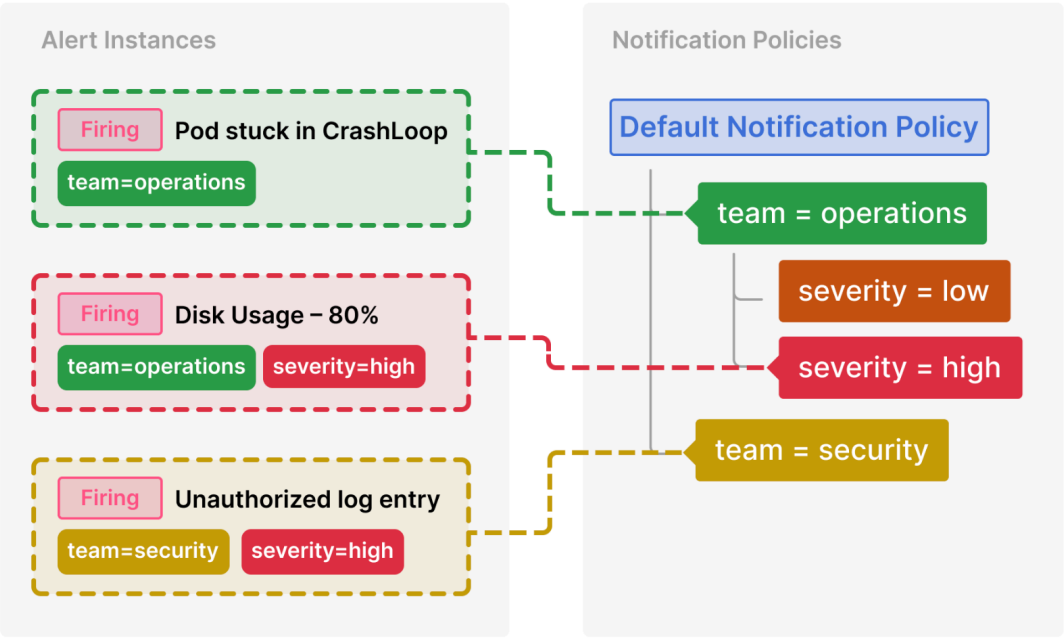
If no child policies are defined in a notification policy or if none of the child policies have any label matchers that match the alert instance's labels, the parent notification policy is used.

As soon as a matching policy is found, the system does not continue to look for other matching policies. If you want to continue to look for other policies that may match, enable **Continue matching siblings** on that particular policy.

Lastly, if none of the notification policies are selected the default notification policy is used.

Routing example

Here is an example of a relatively simple notification policy tree and some alert instances.



Here's a breakdown of how these policies are selected:

Pod stuck in CrashLoop does not have a `severity` label, so none of its child policies are matched. It does have a `team=operations` label, so the first policy is matched.

The `team=security` policy is not evaluated since we already found a match and **Continue matching siblings** was not configured for that policy.

Disk Usage – 80% has both a `team` and `severity` label, and matches a child policy of the operations team.

Unauthorized log entry has a `team` label but does not match the first policy (`team=operations`) since the values are not the same, so it will continue searching and match the `team=security` policy. It does not have any child policies, so the additional `severity=high` label is ignored.

Inheritance

In addition to child policies being a useful concept for routing alert instances, they also inherit properties from their parent policy. This also applies to any policies that are child policies of the default notification policy.

The following properties are inherited by child policies:

- Contact point
- Grouping options
- Timing options
- Mute timings

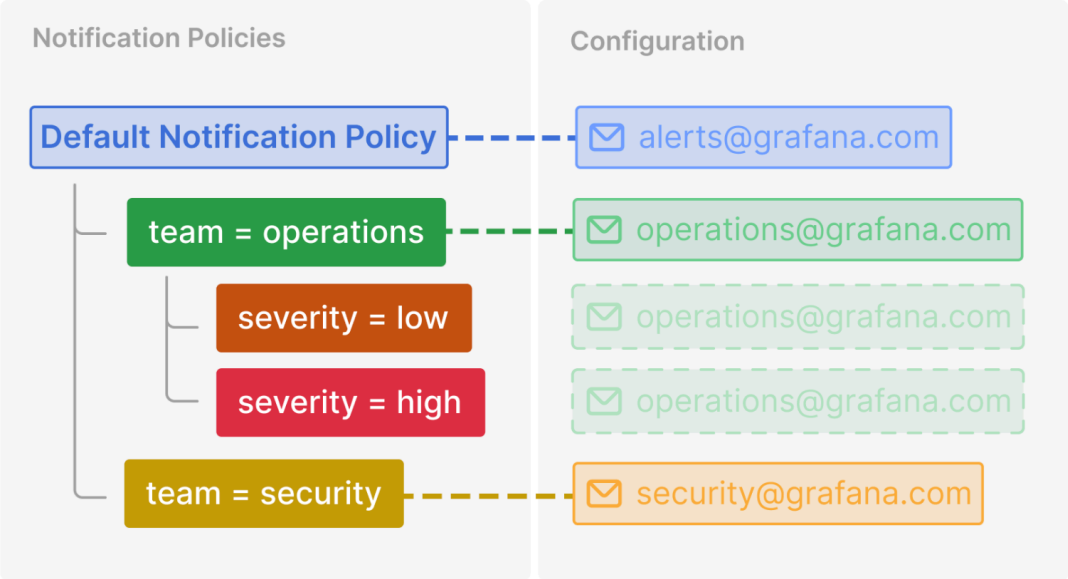
Each of these properties can be overwritten by an individual policy should you wish to override the inherited properties.

To inherit a contact point from the parent policy, leave it blank. To override the inherited grouping options, enable **Override grouping**. To override the inherited timing options, enable **Override general timings**.

Inheritance example

The example below shows how the notification policy tree from our previous example allows the child policies of the `team=operations` to inherit its contact point.

In this way, we can avoid having to specify the same contact point multiple times for each child policy.



Additional configuration options

Grouping

Grouping is an important feature of Grafana Alerting as it allows you to batch relevant alerts together into a smaller number of notifications. This is particularly important if notifications are delivered to first-responders, such as engineers on-call, where receiving lots of notifications in a short period of time can be overwhelming and in some cases can negatively impact a first-responders ability to respond to an incident. For example, consider a large outage where many of your systems are down. In this case, grouping can be the difference between receiving 1 phone call and 100 phone calls.

You choose how alerts are grouped together using the Group by option in a notification policy. By default, notification policies in Grafana group alerts together by alert rule using the `alertname` and `grafana_folder` labels (since alert names are not unique across multiple folders). Should you wish to group alerts by something other than the alert rule, change the grouping to any other combination of labels.

Disable grouping

Should you wish to receive every alert as a separate notification, you can do so by grouping by a special label called `...`. This is useful when your alerts are being delivered to an automated system instead of a first-responder.

A single group for all alerts

Should you wish to receive all alerts together in a single notification, you can do so by leaving Group by empty.

Timing options

The timing options decide how often notifications are sent for each group of alerts. There are three timers that you need to know about: Group wait, Group interval, and Repeat interval.

Group wait

Group wait is the amount of time Grafana waits before sending the first notification for a new group of alerts. The longer Group wait is the more time you have for other alerts to arrive. The shorter Group wait is the earlier the first notification will be sent, but at the risk of sending incomplete notifications. You should always choose a Group wait that makes the most sense for your use case.

Default 30 seconds

Group interval

Once the first notification has been sent for a new group of alerts, Grafana starts the Group interval timer. This is the amount of time Grafana waits before sending notifications about changes to the group. For example, another firing alert might have just been added to the group while an existing alert might have resolved. If an alert was too late to be included in the first notification due to Group wait, it will be included in subsequent notifications after Group interval. Once Group interval has elapsed, Grafana resets the Group interval timer. This repeats until there are no more alerts in the group after which the group is deleted.

Default 5 minutes

Repeat interval

Repeat interval decides how often notifications are repeated if the group has not changed since the last notification. You can think of these as reminders that some alerts are still firing. Repeat interval is closely related to Group interval, which means your Repeat interval must not only be greater than or equal to Group interval, but also must be a multiple of Group interval. If Repeat interval is not a multiple of Group interval it will be coerced into one. For example, if your Group interval is 5 minutes, and your Repeat interval is 9 minutes, the Repeat interval will be rounded up to the nearest multiple of 5 which is 10 minutes.

Default 4 hours

Alerting high availability

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Amazon Managed Grafana is configured for high availability, including running multiple instances across multiple availability zones for each workspace that you create.

Grafana Alerting uses the Prometheus model of separating the evaluation of alert rules from the delivering of notifications. In this model the evaluation of alert rules is done in the alert generator and the delivering of notifications is done in the alert receiver. In Grafana Alerting, the alert generator is the Scheduler and the receiver is the Alertmanager.

With high availability configurations, all alert rules are evaluated on all instances. You can think of the evaluation of alert rules as being duplicated. This is how Grafana Alerting makes sure that as long as at least one Grafana instance is working, alert rules will still be evaluated and notifications for alerts will still be sent. You will see this duplication in state history, and is a good way to tell if you are using high availability.

Set up Alerting

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Configure the features and integrations that you need to create and manage your alerts.

Prerequisites

Before you set up alerting, you must do the following.

- Configure your data sources.
- Ensure that the data source you choose are compatible with and supported by Grafana alerting.

To set up alerting

1. Configure alert rules.
   - Create Grafana-managed or data-source managed alert rules and recording rules.
2. Configure contact points.
   - Check the default contact point, and update the contact for your system.
   - Optionally, add new contact points and integrations.
3. Configure notification policies
   - Check the default notification policy, and update for your system.
   - Optionally, add additional nested policies.
   - Optionally, add labels and label matchers to control alert routing.

The following topics give you more information about additional configuration options, including configuring external alert managers and routing Grafana-managed alerts outside of Grafana.

Topics

- Adding an external Alertmanager
- Provisioning Grafana Alerting resources

Adding an external Alertmanager

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Set up Grafana to use an external Alertmanager as a single Alertmanager to receive all of your alerts. This external Alertmanager can then be configured and administered from within Grafana itself.

Note

You can't use Amazon Managed Service for Prometheus as an external Alertmanager.

Once you have added the alertmanager, you can use the Grafana Alerting UI to manage silences, contact points, and notification policies. A dropdown option in these pages allows you to switch between alertmanagers.

External alertmanagers are configured as data sources using Grafana Configuration from the main Grafana navigation menu. This enables you to manage the contact points and notification policies of external alertmanagers from within Grafana and also encrypts HTTP basic authentication credentials that were previously visible when configuring external alertmanagers by URL.

Note

Starting with Grafana 9.2, the URL configuration of external alertmanagers from the Admin tab on the Alerting page is deprecated. It will be removed in a future release.

To add an external Alertmanager

1. Choose **Connections** from the main left menu.
2. Search for `Alertmanager`.
3. Choose the **Create a new data source** button.
4. Fill out the fields on the page, as required.

   If you are provisioning your data source, set the flag `handleGrafanaManagedAlerts` in the `jsonData` field to `true` to send Grafana-managed alerts to this Alertmanager.

> **Note**
>
> Prometheus, Grafana Mimir, and Cortex implementations of Alertmanager are supported. For Prometheus, contact points and notification policies are read-only in the Grafana Alerting UI.

5. Choose **Save & test**.

Provisioning Grafana Alerting resources

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Alerting infrastructure is often complex, with many pieces of the pipeline that often live in different places. Scaling this across multiple teams and organizations is an especially challenging task. Grafana Alerting provisioning makes this process easier by enabling you to create, manage, and maintain your alerting data in a way that best suits your organization.

There are two options to choose from:

1. Provision your alerting resources using the Alerting Provisioning HTTP API.

> **Note**
>
> Typically, you cannot edit API-provisioned alert rules from the Grafana UI.
> In order to enable editing, add the x-disable-provenance header to the following requests when creating or editing your alert rules in the API:
>
> ```
> POST /api/v1/provisioning/alert-rules
> PUT /api/v1/provisioning/alert-rules/{UID}
> ```

2. Provision your alerting resources using Terraform.

> **Note**
>
> Currently, provisioning for Grafana Alerting supports alert rules, contact points, mute timings, and templates. Provisioned alerting resources using file provisioning or Terraform can only be edited in the source that created them and not from within Grafana or any other source. For example, if you provision your alerting resources using files from disk, you cannot edit the data in Terraform or from within Grafana.

Topics

- Create and manage alerting resources using Terraform
- Viewing provisioned alerting resources in Grafana

Create and manage alerting resources using Terraform

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use Terraform's Grafana Provider to manage your alerting resources and provision them into your Grafana system. Terraform provider support for Grafana Alerting makes it easy to create, manage, and maintain your entire Grafana Alerting stack as code.

For more information on managing your alerting resources using Terraform, refer to the Grafana Provider documentation in the Terraform documentation.

Complete the following tasks to create and manage your alerting resources using Terraform.

1. Create an API key for provisioning.
2. Configure the Terraform provider.

3. Define your alerting resources in Terraform.
4. Run `terraform apply` to provision your alerting resources.

Prerequisites

- Ensure you have the grafana/grafana Terraform provider 1.27.0 or higher.
- Ensure you are using Grafana 9.1 or higher. If you created your Amazon Managed Grafana instance with Grafana version 9, this will be true.

Create an API key for provisioning

You can create a normal Grafana API key to authenticate Terraform with Grafana. Most existing tooling using API keys should automatically work with the new Grafana Alerting support. For information specifically about creating keys for use with Terraform, see Using Terraform for Amazon Managed Grafana automation.

To create an API key for provisioning

1. Create a new service account for your CI pipeline.
2. Assign the role “Access the alert rules Provisioning API.”
3. Create a new service account token.
4. Name and save the token for use in Terraform.

Alternatively, you can use basic authentication. To view all the supported authentication formats, see Grafana authentication in the Terraform documentation.

Configure the Terraform provider

Grafana Alerting support is included as part of the Grafana Terraform provider.

The following is an example you can use to configure the Terraform provider.

```
terraform {
    required_providers {
        grafana = {
            source = "grafana/grafana"
            version = ">= 1.28.2"
        }
    }
```

```
}

provider "grafana" {
    url = <YOUR_GRAFANA_URL>
    auth = <YOUR_GRAFANA_API_KEY>
}
```

Provision contact points and templates

Contact points connect an alerting stack to the outside world. They tell Grafana how to connect to your external systems and where to deliver notifications. There are over fifteen different integrations to choose from. This example uses a Slack contact point.

To provision contact points and templates

1. Copy this code block into a .tf file on your local machine. Replace *<slack-webhook-url>* with your Slack webhook URL (or other contact point details).

   This example creates a contact point that sends alert notifications to Slack.

   ```
   resource "grafana_contact_point" "my_slack_contact_point" {
       name = "Send to My Slack Channel"

       slack {
           url = <slack-webhook-url>
           text = <<EOT
   {{ len .Alerts.Firing }} alerts are firing!

   Alert summaries:
   {{ range .Alerts.Firing }}
   {{ template "Alert Instance Template" . }}
   {{ end }}
   EOT
       }
   }
   ```

2. Enter text for your notification in the text field.

   The `text` field supports Go-style templating. This enables you to manage your Grafana Alerting notification templates directly in Terraform.

3. Run the command `terraform apply`.
4. Go to the Grafana UI and check the details of your contact point.

You cannot edit resources provisioned via Terraform from the UI. This ensures that your alerting stack always stays in sync with your code.

5. Click **Test** to verify that the contact point works correctly.

> **Note**
>
> You can re-use the same templates across many contact points. In the example above, a shared template is embedded using the statement `{{ template "Alert Instance Template" . }}`
> This fragment can then be managed separately in Terraform:

```
resource "grafana_message_template" "my_alert_template" {
    name = "Alert Instance Template"

    template = <<EOT
{{ define "Alert Instance Template" }}
Firing: {{ .Labels.alertname }}
Silence: {{ .SilenceURL }}
{{ end }}
EOT
}
```

Provision notification policies and routing

Notification policies tell Grafana how to route alert instances, as opposed to where. They connect firing alerts to your previously defined contact points using a system of labels and matchers.

To provision notification policies and routing

1. Copy this code block into a .tf file on your local machine.

   In this example, the alerts are grouped by `alertname`, which means that any notifications coming from alerts which share the same name, are grouped into the same Slack message.

   If you want to route specific notifications differently, you can add sub-policies. Sub-policies allow you to apply routing to different alerts based on label matching. In this example, we apply a mute timing to all alerts with the label a=b.

```
resource "grafana_notification_policy" "my_policy" {
    group_by = ["alertname"]
    contact_point = grafana_contact_point.my_slack_contact_point.name

    group_wait = "45s"
    group_interval = "6m"
    repeat_interval = "3h"

    policy {
        matcher {
            label = "a"
            match = "="
            value = "b"
        }
        group_by = ["..."]
        contact_point = grafana_contact_point.a_different_contact_point.name
        mute_timings = [grafana_mute_timing.my_mute_timing.name]

        policy {
            matcher {
                label = "sublabel"
                match = "="
                value = "subvalue"
            }
            contact_point = grafana_contact_point.a_third_contact_point.name
            group_by = ["..."]
        }
    }
}
```

2. In the mute_timings field, link a mute timing to your notification policy.
3. Run the command `terraform apply`.
4. Go to the Grafana UI and check the details of your notification policy.

> **Note**
>
> You cannot edit resources provisioned from Terraform from the UI. This ensures that your alerting stack always stays in sync with your code.

5. Click **Test** to verify that the notification point is working correctly.

Provision mute timings

Mute timings provide the ability to mute alert notifications for defined time periods.

To provision mute timings

1. Copy this code block into a .tf file on your local machine.

   In this example, alert notifications are muted on weekends.

```
resource "grafana_mute_timing" "my_mute_timing" {
    name = "My Mute Timing"

    intervals {
        times {
          start = "04:56"
          end = "14:17"
        }
        weekdays = ["saturday", "sunday", "tuesday:thursday"]
        months = ["january:march", "12"]
        years = ["2025:2027"]
    }
}
```

2. Run the command `terraform apply`.
3. Go to the Grafana UI and check the details of your mute timing.
4. Reference your newly created mute timing in a notification policy using the `mute_timings` field. This will apply your mute timing to some or all of your notifications.

   Note

   You cannot edit resources provisioned from Terraform from the UI. This ensures that your alerting stack always stays in sync with your code.

5. Click **Test** to verify that the mute timing is working correctly.

Provision alert rules

Alert rules enable you to alert against any Grafana data source. This can be a data source that you already have configured, or you can define your data sources in Terraform alongside your alert rules.

To provision alert rules

1. Create a data source to query and a folder to store your rules in.

   In this example, the Configure a TestData data source for testing data source is used.

   Alerts can be defined against any backend datasource in Grafana.

```
resource "grafana_data_source" "testdata_datasource" {
    name = "TestData"
    type = "testdata"
}

resource "grafana_folder" "rule_folder" {
    title = "My Rule Folder"
}
```

2. Define an alert rule.

   For more information on alert rules, refer to how to create Grafana-managed alerts.

3. Create a rule group containing one or more rules.

   In this example, the `grafana_rule_group` resource group is used.

```
resource "grafana_rule_group" "my_rule_group" {
    name = "My Alert Rules"
    folder_uid = grafana_folder.rule_folder.uid
    interval_seconds = 60
    org_id = 1

    rule {
        name = "My Random Walk Alert"
        condition = "C"
        for = "0s"

        // Query the datasource.
        data {
            ref_id = "A"
            relative_time_range {
                from = 600
                to = 0
            }
            datasource_uid = grafana_data_source.testdata_datasource.uid
```

```
            // `model` is a JSON blob that sends datasource-specific data.
            // It's different for every datasource. The alert's query is defined
 here.
            model = jsonencode({
                intervalMs = 1000
                maxDataPoints = 43200
                refId = "A"
            })
        }

        // The query was configured to obtain data from the last 60 seconds. Let's
 alert on the average value of that series using a Reduce stage.
        data {
            datasource_uid = "__expr__"
            // You can also create a rule in the UI, then GET that rule to obtain
 the JSON.
            // This can be helpful when using more complex reduce expressions.
            model = <<EOT
{"conditions":[{"evaluator":{"params":[0,0],"type":"gt"},"operator":
{"type":"and"},"query":{"params":["A"]},"reducer":{"params":
[],"type":"last"},"type":"avg"}],"datasource":
{"name":"Expression","type":"__expr__","uid":"__expr__"},"expression":"A","hide":false,"int
EOT
            ref_id = "B"
            relative_time_range {
                from = 0
                to = 0
            }
        }

        // Now, let's use a math expression as our threshold.
        // We want to alert when the value of stage "B" above exceeds 70.
        data {
            datasource_uid = "__expr__"
            ref_id = "C"
            relative_time_range {
                from = 0
                to = 0
            }
            model = jsonencode({
                expression = "$B > 70"
                type = "math"
                refId = "C"
            })
```

```
        }
    }
}
```

4. Go to the Grafana UI and check your alert rule.

   You can see whether the alert rule is firing. You can also see a visualization of each of the alert rule's query stages.

   When the alert fires, Grafana routes a notification through the policy you defined.

   For example, if you chose Slack as a contact point, Grafana's embedded Alertmanager automatically posts a message to Slack.

Viewing provisioned alerting resources in Grafana

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can verify that your alerting resources were created in Grafana.

To view your provisioned resources in Grafana

1. Open your Grafana instance.
2. Navigate to Alerting.
3. Click an alerting resource folder, for example, Alert rules.

   Provisioned resources are labeled **Provisioned**, so that it is clear that they were not created manually.

Note

You cannot edit provisioned resources from Grafana. You can only change the resource properties by changing the provisioning file and restarting Grafana or carrying out a hot reload. This prevents changes being made to the resource that would be overwritten if a file is provisioned again or a hot reload is carried out.

Configure alerting

This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Configure the features and integrations that you need to create and manage your alerts.

Topics

- Configure Grafana managed alert rules
- Configure data source managed alert rules
- Configure recording rules
- Configure contact points
- Configure notification policies

Configure Grafana managed alert rules

This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana-managed rules are the most flexible alert rule type. They allow you to create alerts that can act on data from any of our supported data sources. In addition to supporting multiple data sources, you can also add expressions to transform your data and set alert conditions. Using images in alert notifications is also supported. This is the only type of rule that allows alerting from multiple data sources in a single rule definition.

Multiple alert instances can be created as a result of one alert rule (also known as multi-dimensional alerting).

Grafana managed alert rules can only be edited or deleted by users with Edit permissions for the folder storing the rules.

If you delete an alerting resource created in the UI, you can no longer retrieve it. To make a backup of your configuration and to be able to restore deleted alerting resources, create your alerting resources using Terraform, or the Alerting API.

In the following procedures, we'll go through the process of creating your Grafana-managed alert rules.

To create a Grafana-managed alert rule, use the in-workspace alert creation flow and follow these steps to help you.

Set alert rule name

1. Choose **Alerting** -> **Alert rules** -> **+ New alert rule**.
2. Enter a name to identify your alert rule.

   This name is displayed in the alert rule list. It is also the `alertname` label for every alert instance that is created from this rule.

Next, define a query to get the data you want to measure and a condition that needs to be met before an alert rule fires.

To define the query and condition

1. Select a data source.

2. From the **Options** dropdown, specify a time range.

   > **Note**
   >
   > Grafana Alerting only supports fixed relative time ranges, for example, `now-24hr: now`.
   > It does not support absolute time ranges: `2021-12-02 00:00:00 to 2021-12-05 23:59:592` or semi-relative time ranges: `now/d to: now`.

3. Add a query.

   To add multiple queries, choose **Add query**.

   All alert rules are managed by Grafana by default. If you want to switch to a data source-managed alert rule, click **Switch to data source-managed alert rule**.

4. Add one or more expressions.

   1. For each expression, select either **Classic condition** to create a single alert rule, or choose from the **Math**, **Reduce**, and **Resample** options to generate a separate alert for each series.

      > **Note**
      >
      > When using Prometheus, you can use an instant vector and built-in functions, so you don't need to add additional expressions.

   2. Choose **Preview** to verify that the expression is successful.

5. [Optional] To add a recovery threshold, turn the **Custom recovery threshold** toggle on and fill in a value for when your alert rule should stop firing.

   You can only add one recovery threshold in a query and it must be the alert condition.

6. Choose **Set as alert condition** on the query or expression you want to set as your alert condition.

Use alert rule evaluation to determine how frequently an alert rule should be evaluated and how quickly it should change its state.

To do this, you need to make sure that your alert rule is in the right evaluation group and set a pending period time that works best for your use case.

To set alert evaluation behavior

1. Select a folder or choose **+ New folder**.
2. Select an evaluation group or click **+ New evaluation group**.

   If you are creating a new evaluation group, specify the interval for the group.

   All rules within the same group are evaluated concurrently over the same time interval.
3. Enter a pending period.

   The pending period is the period in which an alert rule can be in breach of the condition until it fires.

   Once a condition is met, the alert goes into the **Pending** state. If the condition remains active for the duration specified, the alert transitions to the **Firing** state, else it reverts to the **Normal** state.
4. Turn on pause alert notifications, if required.

   > **Note**
   >
   > Pause alert rule evaluation to prevent noisy alerting while tuning your alerts. Pausing stops alert rule evaluation and does not create any alert instances. This is different to mute timings, which stop notifications from being delivered, but still allow for alert rule evaluation and the creation of alert instances.
   > You can pause alert rule evaluation to prevent noisy alerting while tuning your alerts. Pausing stops alert rule evaluation and does not create any alert instances. This is different to mute timings, which stop notifications from being delivered, but still allow for alert rule evaluation and the creation of alert instances.
5. In **Configure no data and error handling**, configure alerting behavior in the absence of data.

   Use the guidelines later in this section.

Add labels to your alert rules to set which notification policy should handle your firing alert instances.

All alert rules and instances, irrespective of their labels, match the default notification policy. If there are no nested policies, or no nested policies match the labels in the alert rule or alert instance, then the default notification policy is the matching policy.

To configure notifications

1. Add labels if you want to change the way your notifications are routed.

   Add custom labels by selecting existing key-value pairs from the drop down, or add new labels by entering the new key or value.
2. Preview your alert instance routing set up.

   Based on the labels added, alert instances are routed to the notification policies displayed.

   Expand each notification policy to view more details.
3. Choose **See details** to view alert routing details and a preview.

Add annotations to provide more context on the alert in your alert notification message.

Annotations add metadata to provide more information on the alert in your alert notification message. For example, add a **Summary** annotation to tell you which value caused the alert to fire or which server it happened on.

To add annotations

1. [Optional] Add a summary.

   Short summary of what happened and why.
2. [Optional] Add a description.

   Description of what the alert rule does.
3. [Optional] Add a Runbook URL.

   Webpage where you keep your runbook for the alert
4. [Optional] Add a custom annotation
5. [Optional] Add a dashboard and panel link.

   Links alerts to panels in a dashboard.
6. Choose **Save rule**.

Single and multi-dimensional rule

For Grafana managed alerts, you can create a rule with a classic condition or you can create a multi-dimensional rule.

- **Rule with classic condition**

  Use the classic condition expression to create a rule that triggers a single alert when its condition is met. For a query that returns multiple series, Grafana does not track the alert state of each series. As a result, Grafana sends only a single alert even when alert conditions are met for multiple series.

- **Multi-dimensional rule**

  To generate a separate alert for each series, create a multi-dimensional rule. Use `Math`, `Reduce`, or `Resample` expressions to create a multi-dimensional rule. For example:

  - Add a `Reduce` expression for each query to aggregate values in the selected time range into a single value (not needed for rules using numeric data).
  - Add a `Math` expression with the condition for the rule. Not needed in case a query or a reduce expression already returns `0` if rule should not fire, or a positive number if it should fire. Some examples: `$B > 70` if it should fire in case value of B query/expression is more than 70. `$B < $C * 100` in case it should fire if value of B is less than value of C multiplied by 100. If queries being compared have multiple series in their results, series from different queries are matched if they have the same labels or one is a subset of the other.

> **Note**
>
> Grafana does not support alert queries with template variables. More information is available at https://community.grafana.com/t/template-variables-are-not-supported-in-alert-queries-while-setting-up-alert/2514.

Configure no data and error handling

Configure alerting behavior when your alert rule evaluation returns no data or an error.

Note

Alert rules that are configured to fire when an evaluation returns no data or error only fire when the entire duration of the evaluation period has finished. This means that rather than immediately firing when the alert rule condition is breached, the alert rule waits until the time set as the **For** field has finished and then fires, reducing alert noise and allowing for temporary data availability issues.

If your alert rule evaluation returns no data, you can set the state on your alert rule to appear as follows:

No Data	Description
No Data	Creates a new alert `DatasourceNoData` with the name and UID of the alert rule, and UID of the datasource that returned no data as labels.
Alerting	Sets alert rule state to `Alerting`. The alert rule waits until the time set in the **For** field has finished before firing.
Ok	Sets alert rule state to `Normal`.

If your evaluation returns an error, you can set the state on your alert rule to appear as follows:

Error	Description
Error	Creates an alert instance `DatasourceError` with the name and UID of the alert rule, and UID of the datasource that returned no data as labels.
Alerting	Sets alert rule state to `Alerting`. The alert rule waits until the time set in the **For** field has finished before firing.

Error	Description
Ok	Sets alert rule state to `Normal`.

Resolve stale alert instances

An alert instance is considered stale if its dimension or series has disappeared from the query results entirely for two evaluation intervals.

Stale alert instances that are in the `Alerting/NoData/Error` states are automatically marked as `Resolved` and the `grafana_state_reason` annotation is added to the alert instance with the reason `MissingSeries`.

Create alerts from panels

Create alerts from any panel type. This means you can reuse the queries in the panel and create alerts based on them.

1. Navigate to a dashboard in the **Dashboards** section.
2. In the top right corner of the panel, choose the three dots (ellipses).
3. From the dropdown menu, select **More...** and then choose **New alert rule**.

This will open the alert rule form, allowing you to configure and create your alert based on the current panel's query.

Configure data source managed alert rules

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Create alert rules for an external Grafana Mimir or Loki instance that has ruler API enabled; these are called data source managed alert rules.

> **Note**
>
> Alert rules for an external Grafana Mimir or Loki instance can be edited or deleted by users with Editor or Admin roles.
> If you delete an alerting resource created in the UI, you can no longer retrieve it. To make a backup of your configuration and to be able to restore deleted alerting resources, create your alerting resources using Terraform, or the Alerting API.

Prerequisites

- Verify that you have write permission to the Prometheus or Loki data source. Otherwise, you will not be able to create or update Grafana Mimir managed alert rules.
- For Grafana Mimir and Loki data sources, enable the Ruler API by configuring their respective services.
  - **Loki** - The `local` rule storage type, default for the Loki data source, supports only viewing of rules. To edit rules, configure one of the other rule storage types.
  - **Grafana Mimir** - use the `/prometheus` prefix. The Prometheus data source supports both Grafana Mimir and Prometheus, and Grafana expects that both the Query API and Ruler API are under the same URL. You cannot provide a separate URL for the Ruler API.

> **Note**
>
> If you do not want to manage alert rules for a particular Loki or Prometheus data source, go to its settings and clear the **Manage alerts via alerting UI** checkbox.

In the following procedures, we'll guide you through the process of creating your data source managed alert rules.

To create a data source-managed alert rule, use the in-workspace alert creation flow and follow these steps to help you.

To set the alert rule name

1. Choose **Alerting** -> **Alert rules** -> **+ New alert rule**.
2. Enter a name to identify your alert rule.

This name is displayed in the alert rule list. It is also the `alertname` label for every alert instance that is created from this rule.

Define a query to get the data you want to measure and a condition that needs to be met before an alert rule fires.

To define query and condition

1. All alert rules are managed by Grafana by default. To switch to a data source managed alert rule, choose **Switch to data source-managed alert rule**.
2. Select a data source from the drop-down list.

   You can also choose **Open advanced data source picker** to see more options, including adding a data source (Admins only).
3. Enter a PromQL or LogQL query.
4. Choose **Preview alerts**.

Use alert rule evaluation to determine how frequently an alert rule should be evaluated and how quickly it should change its state.

To set alert evaluation behavior

1. Select a namespace or choose **+ New namespace**.
2. Select an evaluation group or choose **+ New evaluation group**.

   If you are creating a new evaluation group, specify the interval for the group.

   All rules within the same group are evaluated sequentially over the same time interval.
3. Enter a pending period.

   The pending period is the period in which an alert rule can be in breach of the condition until it fires.

   Once a condition is met, the alert goes into the `Pending` state. If the condition remains active for the duration specified, the alert transitions to the `Firing` state, else it reverts to the `Normal` state.

Add labels to your alert rules to set which notification policy should handle your firing alert instances.

All alert rules and instances, irrespective of their labels, match the default notification policy. If there are no nested policies, or no nested policies match the labels in the alert rule or alert instance, then the default notification policy is the matching policy.

Configure notifications

- Add labels if you want to change the way your notifications are routed.

  Add custom labels by selecting existing key-value pairs from the drop down, or add new labels by entering the new key or value.

Add annotations to provide more context on the alert in your alert notifications.

Annotations add metadata to provide more information on the alert in your alert notifications. For example, add a `Summary` annotation to tell you which value caused the alert to fire or which server it happened on.

To add annotations

1. [Optional] Add a summary.

   Short summary of what happened and why.
2. [Optional] Add a description.

   Description of what the alert rule does.
3. [Optional] Add a Runbook URL.

   Webpage where you keep your runbook for the alert
4. [Optional] Add a custom annotation
5. [Optional] Add a dashboard and panel link.

   Links alerts to panels in a dashboard.
6. Choose **Save rule**.

Configure recording rules

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can create and manage recording rules for an external Grafana Mimir or Loki instance. Recording rules calculate frequently needed expressions or computationally expensive expressions in advance and save the result as a new set of time series. Querying this new time series is faster, especially for dashboards since they query the same expression every time the dashboards refresh.

Note

Recording rules are run as instance rules, and run every 10 seconds.

Prerequisites

- Verify that you have write permissions to the Prometheus or Loki data source. You will be creating or updating alerting rules in your data source.
- For Grafana Mimir and Loki data sources, enable the ruler API by configuring their respective services.
  - **Loki** – The `local` rule storage type, default for the Loki data source, supports only viewing of rules. To edit rules, configure one of the other storage types.
  - **Grafana Mimir** – Use the `/prometheus` prefix. The Prometheus data source supports both Grafana Mimir and Prometheus, and Grafana expects that both the Query API and Ruler API are under the same URL. You cannot provide a separate URL for the Ruler API.

> **Note**
>
> If you do not want to manage alerting rules for a particular Loki or Prometheus data source, go to its settings and clear the **Manage alerts via Alerting UI** check box.

To create recording rules

1. From your Grafana console, in the Grafana menu, choose **Alerting**, **Alert rules**.
2. Choose **New recording rule**.
3. Set rule name.

   The recording rule name must be a Prometheus metric name and contain no whitespace.
4. Define query
   - Select your Loki or Prometheus data source.
   - Enter a query.
5. Add namespace and group.
   - From the **Namespace** dropdown, select an existing rule namespace or add a new one. Namespaces can contain one or more rule groups and only have an organizational purpose.
   - From the **Group** dropdown, select an existing group within the selected namespace or add a new one. Newly created rules are appended to the end of the group. Rules within a group are run sequentially at a regular interval, with the same evaluation time.
6. Add labels.
   - Add custom labels selecting existing key-value pairs from the dropdown, or add new labels by entering the new key or value.
7. Choose **Save rule** to save the rule, or **Save rule and exit** to save the rule and go back to the Alerting page.

Configure contact points

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use contact points to define how your contacts are notified when an alert rule fires.

Note

You can create and edit contact points for Grafana managed alerts. Contact points for data source managed alerts are read-only.

Working with contact points

The following procedures show how to add, edit, delete, and test a contact point.

To add a contact point

1. In the left-side menu, choose **Alerting**.
2. Choose **Contact points**.
3. From the **Choose Alertmanager** dropdown, select an Alertmanager. The Grafana Alertmanager is selected by default.
4. On the **Contact Points** tab, choose **+ Add contact point**.
5. Enter a **Name** for the contact point.
6. From **Integration**, choose a type, and fill out the mandatory fields based on that type. For example, if you choose Slack, enter the Slack channels and users who should be contacted.
7. If available for the contact point you selected, choose any desired **Optional settings** to specify additional settings.
8. Under **Notification settings**, optionally select **Disable resolved message** if you do not want to be notified when an alert resolves.
9. To add another contact point integration, choose **Add contact point integration** and repeat the steps for each contact point type needed.
10. Save your changes.

To edit a contact point

1. In the left-side menu, choose **Alerting**.
2. Choose **Contact points** to see a list of existing contact points.
3. Select the contact point to edit, then choose **Edit**.
4. Update the contact point, and then save your changes.

You can delete contact points that are not in use by a notification policy.

To delete a contact point

1. In the left-side menu, choose **Alerting**.
2. Choose **Contact points** to open the list of existing contact points.
3. On the **Contact points**, select the contact point to delete, then choose **More**, **Delete**.
4. In the confirmation dialog box, choose **Yes, delete**.

> **Note**
>
> If the contact point is in use by a notification policy, you must delete the notification policy or edit it to use a different contact point before deleting the contact point.

After your contact point is created, you can send a test notification to verify that it is configured properly.

To send a test notification

1. In the left-side menu, choose **Alerting**.
2. Choose **Contact points** to open the list of existing contact points.
3. On the **Contact points**, select the contact point to test, then choose **Edit**. You can also create a new contact point if needed.
4. Choose **Test** to open the contact point testing dialog.
5. Choose whether to send a predefined test notification or choose **Custom** to add your own custom annotations and labels in the test notification.
6. Choose **Send test notification** to test the alert with the given contact points.

Configure contact point integrations

Configure contact point integrations in Grafana to select your preferred communication channel for receiving notifications when your alert rules are firing. Each integration has its own configuration options and setup process. In most cases, this involves providing an API key or a Webhook URL.

Once configured, you can use integrations as part of your contact points to receive notifications whenever your alert changes its state. In this section, we'll cover the basic steps to configure an integration, using PagerDuty as an example, so you can start receiving real-time alerts and stay on top of your monitoring data.

List of supported integrations

The following table lists the contact point types supported by Grafana.

Name	Type
Amazon SNS	`sns`
OpsGenie	`opsgenie`
Pager Duty	`pagerduty`
Slack	`slack`
VictorOps	`victorops`

Configuring PagerDuty for alerting

To set up PagerDuty, you must provide an integration key. Provide the following details.

Setting	Description		
Integration Key	Integration key for PagerDuty		
Severity	Level for dynamic notifications. Default is `critical`.		

Setting	Description		
Custom Details	Additional details about the event		

The `CustomDetails` field is an object containing arbitrary key-value pairs. The user-defined details are merged with the ones used by default.

The default values for `CustomDetails` are:

```
{
 "firing":       `{{ template "__text_alert_list" .Alerts.Firing }}`,
 "resolved":     `{{ template "__text_alert_list" .Alerts.Resolved }}`,
 "num_firing":   `{{ .Alerts.Firing | len }}`,
 "num_resolved": `{{ .Alerts.Resolved | len }}`,
}
```

In case of duplicate keys, the user-defined details overwrite the default ones.

Configure notification policies

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Notification policies determine how alerts are routed to contact points.

Policies have a tree structure, where each policy can have one or more nested policies. Each policy, except for the default policy, can also match specific alert labels.

Each alert is evaluated by the default policy and subsequently by each nested policy.

If you enable the `Continue matching subsequent sibling nodes` option for a nested policy, then evaluation continues even after one or more matches. A parent policy's configuration

settings and contact point information govern the behavior of an alert that does not match any of the child policies. A default policy governs any alert that does not match a nested policy.

For more information on notification policies, see Notifications.

The following procedures show you how to create and manage notification policies.

To edit the default notification policy

1. In the left-side menu, choose **Alerting**.
2. Choose **Notification policies**.
3. From the **Choose Alertmanager** dropdown, select the Alertmanager you want to edit.
4. In the **Default policy** section, choose **...**, **Edit**.
5. In **Default contact point**, update the contact point where notifications should be sent for rules when alert rules do not match any specific policy.
6. In **Group by**, choose the labels to group alerts by. If multiple alerts are matched for this policy, then they are grouped by these labels. A notification is sent per group. If the field is empty (the default), then all notifications are sent in a single group. Use a special label, `...` to group alerts by all labels (which effectively disables grouping).
7. In **Timing options**, select from the following options.
   - **Group wait** – Time to wait to buffer alerts of the same group before sending an initial notification. The default is 30 seconds.
   - **Group interval** – Minimum time interval between two notifications for a group. The default is 5 minutes.
   - **Repeat interval** – Minimum time interval before resending a notification if no new alerts were added to the group. The default is 4 hours.
8. Choose **Save** to save your changes.

To create a new notification policy, you need to follow its tree structure. New policies created on the trunk of the tree (the default policy), are the tree branches. Each branch can have their own nested policies. This is why you will always be adding a new **nested** policy, either under the default policy, or under an already nested policy.

To add a new nested policy

1. In the left-side menu, choose **Alerting**.

2. Choose **Notification policies**.
3. From the **Choose Alertmanager** dropdown, select the Alertmanager you want to edit.
4. To add a top level specific policy, go to the Specific routing section (eitehr to the default policy, or to antoher existing policy in which you would like to add a new nested policy) and choose **+ New nested policy**.
5. In the matching labels section, add one or more rules for matching alert labels.
6. In the **Contact point** dropdown, select the contact point to send notifications to if an alert matches only this specific policy and not any of the nested policies.
7. Optionally, enable **Continue matching subsequent sibling nodes** to continue matching sibling policies even after the alert matched the current policy. When this option is enabled, you can get more than one notification for one alert.
8. Optionally, enable **Override grouping** to specify the same grouping as the default policy. If the option is not enabled, the default policy grouping is used.
9. Optionally, enable **Override general timings** to override the timing options configured in the group notification policy.
10. Choose **Save policy** to save your changes.

To edit a nested policy

1. In the left-side menu, choose **Alerting**.
2. Choose **Notification policies**.
3. Select the policy that you want to edit, then choose **...**, **Edit**.
4. Make any changes (as when adding a nested policy).
5. Save your changes.

Searching for policies

You can search within the tree of policies by *Label matchers* or *contact points*.

- To search by contact point, enter a partial or full name of a contact point in the **Search by contact point** field. The policies that use that contact point will be highlighted in the user interface.

- To search by label, enter a valid label matcher in the **Search by matchers** input field. Multiple matchers can be entered, separated by a comma. For example, a valid matcher input could be `severity=high, region=~EMEA|NA`.

> **Note**
>
> When searching by label, all matched policies will be exact matches. Partial matches and regex-style matches are not supported.

Manage your alerts

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Once you have set up your alert rules, contact points, and notification policies, you can use Grafana alerting to manage your alerts in practice.

Topics

- Customize notifications
- Manage contact points
- Silencing alert notifications
- View and filter alert rules
- Mute timings
- View the state and health of alert rules
- View and filter by alert groups
- View notification errors

Customize notifications

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Customize your notifications with notifications templates.

You can use notification templates to change the title, message, and format of the message in your notifications.

Notification templates are not tied to specific contact point integrations, such as Amazon SNS or Slack. However, you can choose to create separate notification templates for different contact point integrations.

You can use notification templates to:

- Add, remove, or re-order information in the notification including the summary, description, labels and annotations, values, and links
- Format text in bold and italic, and add or remove line breaks

You cannot use notification templates to:

- Change the design of notifications in instant messaging services such as Slack and Microsoft Teams

Topics

- Using Go's templating language
- Create notification templates
- Using notification templates
- Template reference

Using Go's templating language

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You write notification templates in Go's templating language, text/template.

This section provides an overview of Go's templating language and writing templates in text/template.

Dot

In text/template there is a special cursor called dot, and is written as `.`. You can think of this cursor as a variable whose value changes depending where in the template it is used. For example, at the start of a notification template `.` refers to the `ExtendedData` object, which contains a number of fields including `Alerts`, `Status`, `GroupLabels`, `CommonLabels`, `CommonAnnotations` and `ExternalURL`. However, dot might refer to something else when used in a `range` over a list, when used inside a `with`, or when writing feature templates to be used in other templates. You can see examples of this in Create notification templates, and all data and functions in the Template reference.

Opening and closing tags

In text/template, templates start with `{{` and end with `}}` irrespective of whether the template prints a variable or runs control structures such as if statements. This is different from other templating languages such as Jinja where printing a variable uses `{{` and `}}` and control structures use `{%` and `%}`.

Print

To print the value of something use `{{` and `}}`. You can print the value of dot, a field of dot, the result of a function, and the value of a variable. For example, to print the `Alerts` field where dot refers to `ExtendedData` you would write the following:

```
{{ .Alerts }}
```

Iterate over alerts

To print just the labels of each alert, rather than all information about the alert, you can use a `range` to iterate the alerts in `ExtendedData`:

```
{{ range .Alerts }}
{{ .Labels }}
{{ end }}
```

Inside the range dot no longer refers to `ExtendedData`, but to an `Alert`. You can use `{{ .Labels }}` to print the labels of each alert. This works because `{{ range .Alerts }}` changes dot to refer to the current alert in the list of alerts. When the range is finished dot is reset to the value it had before the start of the range, which in this example is `ExtendedData`:

```
{{ range .Alerts }}
{{ .Labels }}
{{ end }}
{{/* does not work, .Labels does not exist here */}}
{{ .Labels }}
{{/* works, cursor was reset */}}
{{ .Status }}
```

Iterate over annotations and labels

Let's write a template to print the labels of each alert in the format `The name of the label is $name, and the value is $value`, where `$name` and `$value` contain the name and value of each label.

Like in the previous example, use a range to iterate over the alerts in `.Alerts` such that dot refers to the current alert in the list of alerts, and then use a second range on the sorted labels so dot is updated a second time to refer to the current label. Inside the second range use `.Name` and `.Value` to print the name and value of each label:

```
{{ range .Alerts }}
{{ range .Labels.SortedPairs }}
The name of the label is {{ .Name }}, and the value is {{ .Value }}
{{ end }}
{{ range .Annotations.SortedPairs }}
```

```
The name of the annotation is {{ .Name }}, and the value is {{ .Value }}
{{ end }}
{{ end }}
```

The index functions

To print a specific annotation or label use the `index` function.

```
{{ range .Alerts }}
The name of the alert is {{ index .Labels "alertname" }}
{{ end }}
```

If statements

You can use if statements in templates. For example, to print `There are no alerts` if there are no alerts in `.Alerts` you would write the following:

```
{{ if .Alerts }}
There are alerts
{{ else }}
There are no alerts
{{ end }}
```

With

With is similar to if statements, however unlike if statements, `with` updates dot to refer to the value of the with:

```
{{ with .Alerts }}
There are {{ len . }} alert(s)
{{ else }}
There are no alerts
{{ end }}
```

Variables

Variables in text/template must be created within the template. For example, to create a variable called `$variable` with the current value of dot you would write the following:

```
{{ $variable := . }}
```

You can use `$variable` inside a range or `with` and it will refer to the value of dot at the time the variable was defined, not the current value of dot.

For example, you cannot write a template that use `{{ .Labels }}` in the second range because here dot refers to the current label, not the current alert:

```
{{ range .Alerts }}
{{ range .Labels.SortedPairs }}
{{ .Name }} = {{ .Value }}
{{/* does not work because in the second range . is a label not an alert */}}
There are {{ len .Labels }}
{{ end }}
{{ end }}
```

You can fix this by defining a variable called `$alert` in the first range and before the second range:

```
{{ range .Alerts }}
{{ $alert := . }}
{{ range .Labels.SortedPairs }}
{{ .Name }} = {{ .Value }}
{{/* works because $alert refers to the value of dot inside the first range */}}
There are {{ len $alert.Labels }}
{{ end }}
{{ end }}
```

Range with index

You can get the index of each alert within a range by defining index and value variables at the start of the range:

```
{{ $num_alerts := len .Alerts }}
{{ range $index, $alert := .Alerts }}
This is alert {{ $index }} out of {{ $num_alerts }}
{{ end }}
```

Define templates

You can define templates that can be used within other templates, using `define` and the name of the template in double quotes. You should not define templates with the same name as other templates, including default templates such as `__subject`, `__text_values_list`,

`__text_alert_list`, `default.title` and `default.message`. Where a template has been created with the same name as a default template, or a template in another notification template, Grafana might use either template. Grafana does not prevent, or show an error message, when there are two or more templates with the same name.

```
{{ define "print_labels" }}
{{ end }}
```

Execute templates

You can execute defined template within your template using `template`, the name of the template in double quotes, and the cursor that should be passed to the template:

```
{{ template "print_labels" . }}
```

Pass data to templates

Within a template dot refers to the value that is passed to the template.

For example, if a template is passed a list of firing alerts then dot refers to that list of firing alerts:

```
{{ template "print_alerts" .Alerts }}
```

If the template is passed the sorted labels for an alert then dot refers to the list of sorted labels:

```
{{ template "print_labels" .SortedLabels }}
```

This is useful when writing reusable templates. For example, to print all alerts you might write the following:

```
{{ template "print_alerts" .Alerts }}
```

Then to print just the firing alerts you could write this:

```
{{ template "print_alerts" .Alerts.Firing }}
```

This works because both `.Alerts` and `.Alerts.Firing` are lists of alerts.

```
{{ define "print_alerts" }}
{{ range . }}
{{ template "print_labels" .SortedLabels }}
{{ end }}
{{ end }}
```

Comments

You can add comments with {{/* and */}}:

```
{{/* This is a comment */}}
```

To prevent comments from adding line breaks use:

```
{{- /* This is a comment with no leading or trailing line breaks */ -}}
```

Indentation

You can use indentation, both tabs and spaces, and line breaks, to make templates more readable:

```
{{ range .Alerts }}
  {{ range .Labels.SortedPairs }}
    {{ .Name }} = {{ .Value }}
  {{ end }}
{{ end }}
```

However, indentation in the template will also be present in the text. Next we will see how to remove it.

Remove spaces and line breaks

In text/template use {{- and -}} to remove leading and trailing spaces and line breaks.

For example, when using indentation and line breaks to make a template more readable:

```
{{ range .Alerts }}
  {{ range .Labels.SortedPairs }}
    {{ .Name }} = {{ .Value }}
  {{ end }}
{{ end }}
```

The indentation and line breaks will also be present in the text:

```
    alertname = "Test"

    grafana_folder = "Test alerts"
```

You can remove the indentation and line breaks from the text changing `}}` to `-}}` at the start of each range:

```
{{ range .Alerts -}}
  {{ range .Labels.SortedPairs -}}
    {{ .Name }} = {{ .Value }}
  {{ end }}
{{ end }}
```

The indentation and line breaks in the template are now absent from the text:

```
alertname = "Test"
grafana_folder = "Test alerts"
```

Create notification templates

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Create reusable notification templates to send to your contact points.

You can add one or more templates to your notification template.

Your notification template name must be unique. You cannot have two templates with the same name in the same notification template or in different notification templates. Avoid defining templates with the same name as default templates, such as: `__subject`, `__text_values_list`, `__text_alert_list`, `default.title` and `default.message`.

In the Contact points tab, you can see a list of your notification templates.

Creating notification templates

To create a notification template

1. Choose **Alerting**, **Contact points**.
2. Choose the **Notification Templates** tab, and then **+ Add notification template**.
3. Choose a name for the notification template, such as `email.subject`.
4. Write the content of the template in the content field.

   For example:

   ```
   {{ if .Alerts.Firing -}}
      {{ len .Alerts.Firing }} firing alerts
      {{ end }}
      {{ if .Alerts.Resolved -}}
      {{ len .Alerts.Resolved }} resolved alerts
      {{ end }}
   ```

5. Save your changes.

   `{{ define "email.subject" }}` (where `email.subject` is the name of your template) and `{{ end }}` is automatically added to the start and end of the content.

To create a notification template that contains more than one template

1. Choose **Alerting**, **Contact points**.
2. Choose the **Notification Templates** tab, and then **+ Add notification template**.
3. Enter a name for the overall notification template. For example, `email`.
4. Write each template in the Content field, including `{{ define "`*`name-of-template`*`" }}` and `{{ end }}` at the start and end of each template. You can use descriptive names for each of the templates in the notification template, for example, `email.subject` or `email.message`. In this case, do not reuse the name of the notification template you entered above.

   Later sections show detailed examples for templates you might create.

5. Click Save.

Preview notification templates

Preview how your notification templates will look before using them in your contact points, helping you understand the result of the template you are creating as well as giving you a chance to fix any errors before saving the template.

> **Note**
>
> Notification previews are only available for Grafana Alertmanager.

To preview your notification templates

1. Choose **Alerting**, **Contact points**.
2. Choose the **Notification Templates** tab, and then **+ Add notification template**, or edit an existing template.
3. Add or update your template content.

   Default data is provided and you can add or edit alert data to it as well as alert instances. You can add alert data directly in the Payload data window itself, or click **Select alert instances** or **Add custom alerts**.
4. [Optional] To add alert data from existing alert instances:

   1. Choose **Select alert instances**.
   2. Hover over the alert instances to view more information about each alert instance/
   3. Choose **Confirm** to add the alert instance to the payload.
5. [Optional] To add alert data using the Alert data editor, choose **Add custom data**:

   1. Add annotations, custom labels, or set a dashboard or panel.
   2. Toggle Firing or resolved, depending on whether you want to add firing or resolved alerts to your notification.
   3. Choose **Add alert data**.
   4. Choose **Refresh preview** to see what your template content will look like and the corresponding payload data.

   If there are any errors in your template, they are displayed in the Preview and you can correct them before saving.

6. Save your changes.

Creating a template for the subject of message

Create a template for the subject of an email that contains the number of firing and resolved alerts, as in this example:

```
1 firing alerts, 0 resolved alerts
```

To create a template for the subject of an email

1. Create a template called `email.subject` with the following content:

   ```
   {{ define "email.subject" }}
   {{ len .Alerts.Firing }} firing alerts, {{ len .Alerts.Resolved }} resolved alerts
   {{ end }}
   ```

2. Use the template when creating your contact point integration by putting it into the **Subject** field with the `template` keyword.

   ```
   {{ template "email.subject" . }}
   ```

Creating a template for the message of an email

Create a template for the message of an email that contains a summary of all firing and resolved alerts, as in this example:

```
There are 2 firing alerts, and 1 resolved alerts

Firing alerts:

- alertname=Test 1 grafana_folder=GrafanaCloud has value(s) B=1
- alertname=Test 2 grafana_folder=GrafanaCloud has value(s) B=2

Resolved alerts:

- alertname=Test 3 grafana_folder=GrafanaCloud has value(s) B=0
```

To create a template for the message of an email

1. Create a notification template called `email` with two templates in the content: `email.message_alert` and `email.message`.

   The `email.message_alert` template is used to print the labels and values for each firing and resolved alert while the `email.message` template contains the structure of the email.

```
{{- define "email.message_alert" -}}
{{- range .Labels.SortedPairs }}{{ .Name }}={{ .Value }} {{ end }} has value(s)
{{- range $k, $v := .Values }} {{ $k }}={{ $v }}{{ end }}
{{- end -}}

{{ define "email.message" }}
There are {{ len .Alerts.Firing }} firing alerts, and {{ len .Alerts.Resolved }}
 resolved alerts

{{ if .Alerts.Firing -}}
Firing alerts:
{{- range .Alerts.Firing }}
- {{ template "email.message_alert" . }}
{{- end }}
{{- end }}

{{ if .Alerts.Resolved -}}
Resolved alerts:
{{- range .Alerts.Resolved }}
- {{ template "email.message_alert" . }}
{{- end }}
{{- end }}

{{ end }}
```

2. Use the template when creating your contact point integration by putting it into the **Text Body** field with the `template` keyword.

```
{{ template "email.message" . }}
```

Creating a template for the title of a Slack message

Create a template for the title of a Slack message that contains the number of firing and resolved alerts, as in the following example:

```
1 firing alerts, 0 resolved alerts
```

To create a template for the title of a Slack message

1. Create a template called `slack.title` with the following content:

   ```
   {{ define "slack.title" }}
   {{ len .Alerts.Firing }} firing alerts, {{ len .Alerts.Resolved }} resolved alerts
   {{ end }}
   ```

2. Execute the template from the title field in your contact point integration.

   ```
   {{ template "slack.title" . }}
   ```

Creating a template for the content of a Slack message

Create a template for the content of a Slack message that contains a description of all firing and resolved alerts, including their labels, annotations, and Dashboard URL.

> **Note**
>
> This template is for Grafana managed alerts only. To use the template for data source managed alerts, delete the references to DashboardURL and SilenceURL. For more information about configuring Prometheus notifications, see the Prometheus documentation on notifications.

```
1 firing alerts:

[firing] Test1
Labels:
- alertname: Test1
- grafana_folder: GrafanaCloud
Annotations:
- description: This is a test alert
```

```
Go to dashboard: https://example.com/d/dlhdLqF4z?orgId=1

1 resolved alerts:

[firing] Test2
Labels:
- alertname: Test2
- grafana_folder: GrafanaCloud
Annotations:
- description: This is another test alert
Go to dashboard: https://example.com/d/dlhdLqF4z?orgId=1
```

To create a template for the content of a Slack message

1. Create a template called `slack` with two templates in the content: `slack.print_alert` and `slack.message`.

   The `slack.print_alert` template is used to print the labels, annotations, and DashboardURL while the `slack.message` template contains the structure of the notification.

   ```
   {{ define "slack.print_alert" -}}
   [{{.Status}}] {{ .Labels.alertname }}
   Labels:
   {{ range .Labels.SortedPairs -}}
   - {{ .Name }}: {{ .Value }}
   {{ end -}}
   {{ if .Annotations -}}
   Annotations:
   {{ range .Annotations.SortedPairs -}}
   - {{ .Name }}: {{ .Value }}
   {{ end -}}
   {{ end -}}
   {{ if .DashboardURL -}}
     Go to dashboard: {{ .DashboardURL }}
   {{- end }}
   {{- end }}

   {{ define "slack.message" -}}
   {{ if .Alerts.Firing -}}
   {{ len .Alerts.Firing }} firing alerts:
   {{ range .Alerts.Firing }}
   {{ template "slack.print_alert" . }}
   {{ end -}}
   ```

```
{{ end }}
{{ if .Alerts.Resolved -}}
{{ len .Alerts.Resolved }} resolved alerts:
{{ range .Alerts.Resolved }}
{{ template "slack.print_alert" .}}
{{ end -}}
{{ end }}
{{- end }}
```

2. Execute the template from the text body field in your contact point integration:

```
{{ template "slack.message" . }}
```

Template both email and Slack with shared templates

Instead of creating separate notification templates for each contact point, such as email and Slack, you can share the same template.

For example, if you want to send an email with this subject and Slack message with this title `1 firing alerts, 0 resolved alerts`, you can create a shared template.

To create a shared template

1. Create a template called `common.subject_title` with the following content:

```
{{ define "common.subject_title" }}
{{ len .Alerts.Firing }} firing alerts, {{ len .Alerts.Resolved }} resolved alerts
{{ end }}
```

2. For email, run the template from the subject field in your email contact point integration:

```
{{ template "common.subject_title" . }}
```

3. For Slack, run the template from the title field in your Slack contact point integration:

```
{{ template "common.subject_title" . }}
```

Using notification templates

Use templates in contact points to customize your notifications.

To use a template when creating a contact point

1. From the **Alerting** menu, choose the **Contact points** tab to see a list of existing contact points.
2. Choose **New**. Alternately, you can edit an existing contact point by choosing the **Edit** icon.
3. Enter the templates you wish to use in a field, such as **Message** or **Subject**. To enter a template, use the form `{{ template "template_name" . }}`, replacing *template_name* with the name of the template you want to use.
4. Choose **Save contact point**.

Template reference

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This section provides reference information for creating your templates.

Alert (type)

The alert type contains the following data.

Name	Kind	Description	Example
Status	string	`firing` or `resolved`.	`{{ .Status }}`
Labels	KeyValue	A set of labels attached to the alert.	`{{ .Labels }}`
Annotations	KeyValue	A set of annotations attached to the alert.	`{{ .Annotations }}`

Name	Kind	Description	Example
Values	KeyValue	The values of all expressions, including Classic Conditions	`{{ .Values }}`
StartsAt	time.Time	Time the alert started firing.	`{{ .StartsAt }}`
EndsAt	time.Time	Only set if the end time of an alert is known. Otherwise set to a configurable timeout period from the time since the last alert was received.	`{{ .EndsAt }}`
GeneratorURL	string	A back link to Grafana or external Alertmanager.	`{{ .GeneratorURL }}`
SilenceURL	string	A link to silence the alert (with labels for this alert pre-filled). Only for Grafana managed alerts.	`{{ .SilenceURL}}`
DashboardURL	string	Link to grafana dashboard, if alert rule belongs to one. Only for Grafana managed alerts.	`{{ .DashboardURL }}`

Name	Kind	Description	Example
PanelURL	string	Link to grafana dashboard panel, if alert rule belongs to one. Only for Grafana managed alerts.	`{{ .PanelURL }}`
Fingerprint	string	Fingerprint that can be used to identify the alert.	`{{ .Fingerprint }}`
ValueString	string	A string that contains the labels and value of each reduced expression in the alert.	`{{ .ValueString }}`

ExtendedData

The ExtendedData object contains the following properties.

Name	Kind	Description	Example
Receiver	`string`	The name of the contact point sending the notification.	`{{ .Receiver }}`
Status	`string`	The status is `firing` if at least one alert is firing, otherwise `resolved`.	`{{ .Status }}`
Alerts	`[]Alert`	List of all firing and resolved alerts in this notification.	`There are {{ len .Alerts }} alerts`

Name	Kind	Description	Example
Firing alerts	`[]Alert`	List of all firing alerts in this notification.	`There are {{ len .Alerts.Firing }} firing alerts`
Resolved alerts	`[]Alert`	List of all resolved alerts in this notification.	`There are {{ len .Alerts.Resolved }} resolved alerts`
GroupLabels	`KeyValue`	The labels that group these alerts int his notification.	`{{ .GroupLabels }}`
CommonLabels	`KeyValue`	The labels common to all alerts in this notification.	`{{ .CommonLabels }}`
CommonAnnotations	`KeyValue`	The annotations common to all alerts in this notification.	`{{ .CommonAnnotations }}`
ExternalURL	`string`	A link to the Grafana workspace or Alertmanager that sent this notification.	`{{ .ExternalURL }}`

KeyValue type

The `KeyValue` type is a set of key/value string pairs that represent labels and annotations.

In addition to direct access of the data stored as a `KeyValue`, there are also methods for sorting, removing, and transforming the data.

Name	Arguments	Returns	Notes	Example
SortedPairs		Sorted list of key and value string pairs		`{{ .Annotations.SortedPairs }}`
Remove	[]string	KeyValue	Returns a copy of the Key/Value map without the given keys.	`{{ .Annotations.Remove "summary" }}`
Names		[]string	List of names	`{{ .Names }}`
Values		[]string	List of values	`{{ .Values }}`

Time

Time is from the Go `time` package. You can print a time in a number of different formats. For example, to print the time that an alert fired in the format `Monday, 1st January 2022 at 10:00AM`, you write the following template:

```
{{ .StartsAt.Format "Monday, 2 January 2006 at 3:04PM" }}
```

You can find a reference for Go's time format here.

Manage contact points

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The **Contact points** list view lists all existing contact points and notification templates.

On the **Contact points** tab, you can:

- Search for names and types of contact points and integrations.
- View all existing contact points and integrations.
- View how many notification policies each contact point is being used for, and navigate directly to the linked notification policies.
- View the status of notification deliveries.
- Export individual contact points or all contact points in JSON, YAML, or Terraform format.
- Delete contact points that are not in use by a notification policy.

On the **Notification templates** tab, you can:

- View, edit, copy, or delete existing notification templates.

Silencing alert notifications

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can suppress alert notifications with a *silence*. A silence only stops notifications from being created: Silences do not prevent alert rules from being evaluated, and they do not stop alerting instances from being shown in the user interface. When you silence an alert, you specify a window of time for it to be suppressed.

Note

To suppress alert notifications at regular time intervals, for example, during regular maintenance periods, use Mute timings rather than silences.

To add a silence

1. From your Grafana console, in the Grafana menu, choose **Alerting**.

2. Choose **Silences**.
3. Choose an Alertmanager from the **Alertmanager** dropdown.
4. Choose **Create Silence**.
5. Select the start and end date in **Silence start and end** to indicate when the silence should go into effect and when it should end.
6. As an alternative to setting an end time, in **Duration**, specify how long the silence is enforced. This automatically updates the end time in the **Silence start and end** field.
7. In the **Label** and **Value** fields, enter one or more *Matching Labels*. Matchers determine which rules the silence applies to. Any matching alerts (in firing state), will show in the **Affected alerts instances** field.
8. Optionally, add a **Comment** describing the silence.
9. Choose **Submit**.

To edit a silence

1. From your Grafana console, in the Grafana menu, choose **Alerting**.
2. Choose **Silences** to view the list of existing silences.
3. Find the silence you want to edit, then choose **Edit** (pen icon).
4. Make any desired changes, then choose **Submit** to save your changes.

You can edit an existing silence by choosing the **Edit** icon (pen).

To create a URL link to a silence form

When linking to a silence form, provide the default matching labels and comment via `matcher` and `comment` query parameters. The `matcher` parameter should be in the following format `[label][operator][value]` where the `operator` parameter can be one of the following: = (equals, not regex), `!=` (not equals, not regex), =~ (equals, regex), `!~` (not equals, regex). The URL can contain many query parameters with the key `matcher`. For example, to link to silence form with matching labels `severity=critical` & `cluster!~europe-.*` and comment `Silence critical EU alerts`, create a URL `https://mygrafana/alerting/silence/new?matcher=severity%3Dcritical&matcher=cluster!~europe-*&comment=Silence%20critical%20EU%20alert`.

To link to a new silence page for an external Alertmanager, add an `alertmanager` query parameter

To remove a silence

1. From your Grafana console, in the Grafana menu, choose **Alerting**.
2. Choose **Silences** to view the list of existing silences.
3. Select the silence that you want to end, and choose **Unsilence**. This ends the alert suppression.

> **Note**
>
> Unsilencing ends the alert suppression, as if the end time was set for the current time. Silences that have ended (automatically or manually) are retained and listed for five days. You cannot remove a silence from the list manually.

View and filter alert rules

> This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The **Alerting** page lists alerting rules. By default, rules are grouped by types of data sources. The **Grafana** section lists rules managed by Grafana. Alert rules for Prometheus compatible data sources are also listed here. You can view alerting rules for Prometheus compatible data sources but you cannot edit them.

The Mimir/Cortex/Loki rules section lists all rules for Mimir, Cortex, or Loki data sources. Cloud alert rules are also listed in this section.

When managing large volumes of alerts, you can use extended alert rule search capabilities to filter on folders, evaluation groups, and rules. Additionally, you can filter alert rules by their properties like labels, state, type, and health.

View alert rules

Using Grafana alerts, you can view all of your alerts in one page.

To view alerting details

1. From your Grafana console, in the Grafana menu, choose **Alerting**, **Alert rules**. By default, the list view is displayed.
2. In **View as**, you can toggle between the Grouped, List, and State views by choosing the option you prefer.
3. Expand the rule row to view the rule labels, annotations, data sources, the rule queries, and a list of alert instances resulting from the rule.

From this page, you can also make copies of an alert rule to help you reuse existing rules.

Export alert rules

You can export rules to YAML or JSON in the Grafana workspace.

- Choose the **Export rule group** icon next to each alert rule group to export to YAML, JSON, or Terraform.
- Choose **Export rules** to export all Grafana managed alert rules to YAML, JSON, or Terraform.
- Choose **More**, **Modify export** next to each individual alert rule within a group to edit provisioned alert rules and export a modified version.

View query definitions for provisioned alerts

View read-only query definitions for provisioned alerts. Check quickly if your alert rule queries are correct, without diving into your "as-code" repository for rule definitions.

Grouped view

Grouped view shows Grafana alert rules grouped by folder and Loki or Prometheus alert rules grouped by `namespace + group`. This is the default rule list view, intended for managing rules. You can expand each group to view a list of rules in this group. Expand a rule further to view its details. You can also expand action buttons and alerts resulting from the rule to view their details.

State view

State view shows alert rules grouped by state. Use this view to get an overview of which rules are in what state. Each rule can be expanded to view its details. Action buttons and any alerts generated by this rule, and each alert can be further expanded to view its details.

Filtering alert rules

You can filter the alerting rules that appear on the **Alerting** page in several ways.

To filter alert rules

1. From **Select data sources**, select a data source.. You can see alert rules that query the selected data source.
2. In **Search by label**, enter search criteria using label selectors. For example, `environment=production;region=~US|EU,severity!=warning`.
3. From **Filter alerts by state**, select an alerting state you want to see. You can see alerting rules that match that state. Rules matching other states are hidden.

Mute timings

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A mute timing is a recurring interval of time when no new notifications for a policy are generated or sent. Use them to prevent alerts from firing a specific and reoccurring period, for example, a regular maintenance period.

Similar to silences, mute timings do not prevent alert rules from being evaluated, nor do they stop alert instances from being shown in the user interface. They only prevent notifications from being created.

You can configure Grafana managed mute timings as well as mute timings for an external Alertmanager data source.

Mute timings vs Silences

The following table highlights the differences between mute timings and silences.

Mute timing	Silence
Uses time interval definitions that can reoccur.	Has a fixed start and end time.
Is created and then added to notification policies.	Uses labels to match against an alert to determine whether to silence or not.

Adding a mute timing

You can create mute timings in your Grafana workspace.

To add a mute timing

1. From your Grafana console, in the Grafana menu, choose **Alerting**.
2. Choose **Notification policies**, and then select the **Mute Timings** tab.
3. From the **Alertmanager** dropdown, select the Alertmanager you want to edit.
4. Choose the **+ Add mute timing** button.
5. Fill out the form to create a time interval to match against for your mute timing.
6. Save your mute timing.

Adding a mute timing to a notification policy

Once you have a mute timing, you use it by adding it to notification policy that you want to mute at regular intervals.

To add a mute timing to a notification policy

1. From your Grafana console, in the Grafana menu, choose **Alerting**.
2. Choose **Notification policies**, and then select the **Notification Policies** tab.
3. Select the notification policy you would like to add the mute timing to, and choose **...**, **Edit**.
4. From the **Mute timings** dropdown, select the mute timings you would like to add to the policy.
5. Save your changes.

Time intervals

A time interval is a specific duration during which alerts are suppressed. The duration typically consists of a specific time range and the days of the week, month, or year.

Support time interval options are:

- **Time range** – The time inclusive of the start and exclusive of the end time (in UTC, if no location has been selected, otherwise local time.
- **Location** – Sets the location for the timing—the time range is displayed in local time for the location.
- **Days of the week** – The day or range of days of the week. For example, `monday:thursday`.
- **Days of the month** – The dates within a month. Values can range from `1-31`. Negative values specify days of the month in reverse order, so `-1` represents the last day of the month.
- **Months** – The months of the year in either numerical of full calendar month name. For example, `1, may:august`.
- **Years** – The year or years for the interval. For example, `2023:2024`.

Each of these elements can be a list, and at least one item in the element must be satisfied to be a match. Fields also support ranges, using `:`. For example, `monday:thursday`.

If a field is left blank, any moment of time will match the field. For an instant of to match a complete time intervale, all fields must match. A mute timing can contain multiple time intervals.

If you want to specify an exact duration, specify all the options needed for that duration. For example, if you want to create a time interval for the first Monday of the month, for March, June, September, and December, between the hours of 12:00 and 24:00 UTC, your time interval specification could be:

- Time range:
  - Start time: `12:00`
  - End time: `24:00`
- Days of the week: `monday`
- Months: `3, 6, 9, 12`
- Days of the month: `1:7`

View the state and health of alert rules

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The state and health of alert rules gives you several key status indicators about your alerts.

There are three components:

- Alert rule state
- Alert instance state
- Alert rule health

Although related, each component conveys subtly different information.

To view the state and health of your alert rules

1. From your Grafana console, in the Grafana menu, choose **Alerting**.
2. Choose **Alert rules** to view the list of existing alerts.
3. Choose an alert rule to view its state and health.

Alert rule state

An alert rule can be in any of the following states:

State	Description		
Normal	None of the time series returned by the evaluation engine		

State	Description
	is in a `pending` or `firing` state.
Pending	At least one time series returned by the evaluation engine is `pending`.
Firing	At least one time series returned by the evaluation engine is `firing`.

> **Note**
>
> Alerts transition first to `pending` and then `firing`, thus it takes at least two evaluation cycles before an alert is fired.

Alert instance state

An alert instance can be in any of the following states:

State	Description
Normal	The state of an alert that is neither `pending` nor `firing`. Everything is working as expected.
Pending	The state of an alert that has been active for less than the

State	Description		
	configured threshold duration.		
Alerting	The state of an alert that has been active for longer than the configured threshold duration.		
No data	No data has been received for the configured time window.		
Alerting	An error occurred when attempting to evaluate an alerting rule.		

Keep last state

An alert rule can be configured to keep the last state when a `NoData` or `Error` state is encountered. This will both prevent alerts from firing, and from resolving and re-firing. Just like normal evaluation, the alert rule will transition from `pending` to `firing` after the pending period has elapsed.

Alert rule health

An alert rule can have one of the following health statuses.

State	Description		
Ok	No errors when evaluating the alert rule.		

State	Description
Error	An error occurred when evaluating the alert rule.
NoData	The absence of data in at least one time series returned during a rule evaluation.
{status}, KeepLast	The rule would have received another status, but was configured to keep the last state of the alert rule.

Special alerts for NoData and Error

When evaluation of an alert rule produces the state `NoData` or `Error`, Grafana alerting will generate alert instances that have the following additional labels.

Label	Description
alertname	Either `DatasourceNoData` or `DatasourceError`, depending on the state.
datasource_uid	The UID of the data source that caused the state.

Note

You will need to set the no data or error handling to `NoData` or `Error` in the alert rule, as described in the Configure Grafana managed alert rules topic, to generate the additional labels.

You can handle these alerts the same way as regular alerts, including adding silences, routing to a contact point, and so on.

View and filter by alert groups

This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Alert groups show grouped alerts from an Alertmanager instance. By default, alert rules are grouped by the label keys for the default policy in notification policies. Grouping common alert rules into a single alert group prevents duplicate alert rules from being fired.

You can view alert groups and also filter for alert rules that match specific criteria.

To view alert groups

1. From your Grafana console, in the Grafana menu, choose **Alerting**.
2. Choose **Groups** to view existing groups.
3. From the **Alertmanager** dropdown, select an external Alertmanager as your data source.
4. From the **Custom group by** dropdown, select a combination of labels to view a grouping other than the default. This is useful for debugging and verifying your grouping of notification policies.

If an alert does not contain labels specified either in the grouping of the root policy or the custom grouping, then the alert is added to a catch all group with a header of `No grouping`.

You can filter alerts by label or state of the alerts.

To filter by label

- In **Search**, enter an existing label to view alerts matching the label.

  For example, `environment=production,region=~US|EU,severity!=warning`.

To filter by state

- In **States**, select from Active, Suppressed, or Unprocessed states to view alerts matching your selected state. All other alerts are hidden.

View notification errors

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 12.x**.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

View notification errors and understand why they failed to be sent or were not received.

Note

This feature is only supported for Grafana Alertmanager.

To view notification errors

1. From the left menu, choose **Alerting** then **Contact points**.

   If any contact points are failing, a message at the right-hand corner of the workspace tells you that there are errors, and how many.

2. Select a contact point to view the details of errors for that contact point.

Error details are displayed if you hover over the Error icon.

If a contact point has more than one integration, you see all errors for each of the integrations listed.

3. In the Health column, check the status of the notification.

   This can be either OK, No attempts, or Error.

For a complete list of new features, breaking changes, and version history, see Grafana version 12.

Working in Grafana version 10

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

When you create your Grafana workspace, you have the option of which version of Grafana to use. The following topics describe using a Grafana workspace that uses version 10 of Grafana.

Topics

- Dashboards in Grafana version 10
- Panels and visualizations in Grafana version 10
- Explore in Grafana version 10
- Correlations in Grafana version 10
- Alerts in Grafana version 10

Dashboards in Grafana version 10

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A dashboard is a set of one or more panels organized and arranged into one or more rows. Grafana ships with a variety of panels making it easy to construct the right queries, and customize the visualization so that you can create the perfect dashboard for your need. Each panel can interact with data from any configured Connect to data sources.

Dashboard snapshots are static. Queries and expressions cannot be re-executed from snapshots. As a result, if you update any variables in your query or expression, it will not change your dashboard data.

Topics

- Using dashboards
- Building dashboards
- Managing dashboards
- Managing playlists
- Sharing dashboards and panels
- Variables
- Assessing dashboard usage
- Troubleshoot dashboards
- Searching Dashboards in Grafana version 10

Using dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This topic provides an overview of dashboard features and shortcuts, and describes how to use dashboard search.

Features

You can use dashboards to customize the presentation of your data. The following image shows the dashboard interface in the Amazon Managed Grafana workspace.



Feature	Description
1. Home	Select the Grafana home icon to be redirecte d to the home page configured in the Grafana instance.
2. Title	When you select the dashboard title, you can search for dashboards contained in the current folder.

Feature	Description
3. Sharing a dashboard	Use this option to share the current dashboard by link or snapshot. You can also export the dashboard definition from the share modal.
4. Adding a new panel	Use this option to add a panel, dashboard row, or library panel to the current dashboard.
5. Save dashboard	Choose the Save icon to save changes to your dashboard.
6. Dashboard insights	Choose to view analytics about your dashboard, including information about users, activity, and query counts. For more informati on, see Assessing dashboard usage.
7. Dashboard settings	Use this option to change the dashboard name, folder, or tags and manage variables and annotation queries. For more informati on about dashboard settings, see Modifying dashboard settings.
8. Time picker dropdown	Use to select relative time range options and set custom absolute time ranges. You can change the **Timezone** and **fiscal year** settings from the time range controls by clicking the **Change time settings** button. Time settings are saved on a per-dashboard basis.
9. Zoom out time range	Use to zoom out the time range. For more information about how to use time range controls, see Setting dashboard time range.
10. Refresh dashboard	Select to immediately trigger queries and refresh dashboard data.

Feature	Description
11. Refresh dashboard time interval	Select a dashboard auto refresh time interval.
12. View mode	Select to display the dashboard on a large screen such as a TV or a kiosk. View mode hides irrelevant information such as navigatio n menus.
13. Dashboard panel	The primary building block of a dashboard is the panel. To add a new panel, dashboard row, or library panel, select **Add panel**. • Library panels can be shared among many dashboards. • To move a panel, drag the panel header to another location. • To resize a panel, select and drag the lower right corner of the panel.
14. Graph legend	Change series colors, y-axis, and series visibilit y directly from the legend.
15. Dashboard row	A dashboard row is a logical divider within a dashboard that groups panels together. • Rows can be collapsed or expanded to hide parts of the dashboard. • Panels inside a collapsed row do not issue queries. • Use repeating rows to create rows dynamically based on a template variable. For more information about repeating rows, see Creating dashboards.

Keyboard shortcuts

Grafana has a number of keyboard shortcuts available. To display all keyboard shortcuts available to you, press **?** or **h** on your keyboard.

- `Ctrl+S` saves the current dashboard.
- `f` opens the dashboard finder/search.
- `d+k` toggles kiosk mode (hides the menu).
- `d+e` expands all rows.
- `d+s` opens dashboard settings.
- `Ctrl+K` opens the command palette.
- `Esc` exits panel when in fullscreen view or edit mode. Also returns you to the dashboard from the dashboard settings.

Focused panel

To use shortcuts targeting a specific panel, hover over a panel with your pointer.

- `e` toggles panel edit view
- `v` toggles panel fullscreen view
- `ps` opens panel share feature
- `pd` duplicates panel
- `pr` removes panel
- `pl` toggles panel legend

Setting dashboard time range

Grafana provides several ways to manage the time ranges of the data being visualized, for dashboard, panels and also for alerting.

This section describes supported time units and relative ranges, the common time controls, dashboard-wide time settings, and panel-specific time settings.

Time units and relative ranges

Grafana supports the following time units: `s (seconds)`, `m (minutes)`, `h (hours)`, `d (days)`, `w (weeks)`, `M (months)`, `Q (quarters)`, and `y (years)`.

The minus operator enables you to step back in time, relative to the current date and time, or now. If you want to display the full period of the unit (day, week, or month), append `/<time unit>` to the end. To view fiscal periods, use `fQ (fiscal quarter)` and `fy (fiscal year)` time units.

The plus operator enables you to step forward in time, relative to now. For example, you can use this feature to look at predicted data in the future.

The following table provides example relative ranges.

Example relative range	From	To
Last 5 minutes	`now-5m`	now
The day so far	`now/d`	now
This week	`now/w`	`now/w`
This week so far	`now/w`	now
This month	`now/M`	`now/M`
This month so far	`now/M`	now
Previous Month	`now-1M/M`	`now-1M/M`
This year so far	`now/Y`	now
This Year	`now/Y`	`now/Y`
Previous fiscal year	`now-1y/fy`	`now-1y/fy`

Note

Grafana Alerting does not support the following syntaxes:

- `now+n` for future timestamps.
- `now-1n/n` for *start of n until end of n*, because this is an absolute timestamp.

Common time range controls

The dashboard and panel time controls have a common user interface. The following describes common time range controls.

- Current time range, also called the *time picker*, shows the time range currently displayed in the dashboard or panel you are viewing. Hover your cursor over the field to see the exact time stamps in the range and their source (such as the local browser time). Click the *current time range* to change it. You can change the current time using a *relative time range*, such as the last 15 minutes, or an absolute time range, such as `2020-05-14 00:00:00` to `2020-05-15 23:59:59`.
- The **relative time range** can be selected from the **Relative time ranges** list. You can filter the list using the input field at the top. Some examples of time ranges include *Last 30 minutes*, *Last 12 hours*, *Last 7 days*, *Last 2 years*, *Yesterday*, *Day before yesterday*, *This day last week*, *Today so far*, *This week so far*, and *This month so far*.
- **Absolute time range** can be set in two ways: Typing exact time values or relative time values into the **From** and **To** fields and clicking **Apply time range**, or clicking a date or date range from the calendar displayed when you click the **From** or **To** field. To apply your selections, click **Apply time range**. You can also choose from a list of recently used absolute time ranges.
- **Semi-relative time range** can be selected in the absolute time range settings. For example, to show activity since a specific date, you can choose an absolute time for the start time, and a relative time (such as now) for the end time.

  Using a semi-relative time range, as time progresses, your dashboard will automatically and progressively zoom out to show more history and fewer details. At the same rate, as high data resolution decreases, historical trends over the entire time period will become more clear.

  > **Note**
  >
  > Alerting does not support semi-relative time ranges.

- **Zoom out** by selecting the zoom out icon (or by using Cmd+Z or Ctrl+Z as a keyboard shortcut). This increases the view, showing a larger time range in the dashboard or panel visualization.
- **Zoom in** by selecting a time range you want to view on the graph in the visualization.

  > **Note**
  >
  > Zooming in is only applicable to graph visualizations.

Refresh dashboards

Click the **Refresh dashboard** icon to immediately run every query on the dashboard and refresh the visualizations. Grafana cancels any pending requests when you trigger a refresh.

By default, Grafana does not automatically refresh the dashboard. Queries run on their own schedule according to the panel settings. However, if you want to regularly refresh the dashboard, then click the down arrow next to the **Refresh dashboard** icon and then select a refresh interval.

Control the time range using a URL

You can control the time range of a dashboard by providing the following query parameters in the dashboard URL.

- `from` defines the lower limit of the time range, specified in ms epoch, or relative time.
- `to` defines the upper limit of the time range, specified in ms epoch, or relative time.
- `time` and `time.window` defines a time range from `time-time.window/2` to `time+time.window/2`. Both parameters should be specified in ms. For example ? `time=1500000000000&time.window=10000` results in 10s time range from 1499999995000 to 1500000005000.

Building dashboards

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

After you create a Grafana workspace and sign in, you can create dashboards and modify settings to suit your needs. A dashboard is made up of panels with visualizations. Each panel has a query associated with it, to pull data from one of your Connect to data sources.

You can create more interactive and dynamic dashboards by adding and using variables. Instead of hard-coding the server, application, or other names in your metric queries, you can use variables in their place.

Topics

- Creating dashboards
- Importing dashboards
- Exporting dashboards
- Modifying dashboard settings
- Dashboard URL variables
- Managing library panels
- Managing dashboard version history
- Managing dashboard links
- Annotate visualizations
- Dashboard JSON model
- Best practices for dashboards

Creating dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Creating a dashboard

Dashboards and panels allow you to show your data in visual form using Grafana. Each panel needs at least one query to display a visualization. Before you get started, complete the following prerequisites.

- Ensure that you have the proper permissions. For more information about permissions, see Users, teams, and permissions.
- Identify the dashboard to which you want to add the panel.
- Understand the query language of the target data source.

- Ensure that data source for which you are writing a query has been added. For more information, see Connect to data sources.

To create a dashboard:

1. Sign into Grafana, and select **Dashboards** from the left menu.
2. Select **New**, then **New dashboard**.
3. On the empty dashboard, select **+ Add visualization**. This opens the new visualization dialog box.
4. Select a data source. You can choose an existing data source, one of Grafana's built in data sources for testing, or choose **Configure a new data source** to set up a new one (only users with Admin permissions can configure new data sources).

   The **Edit panel** view opens, with your data source selected. You can change the data source for the panel later, using the **Query** tab of the panel editor, if needed.
5. Write or construct a query in the query language of your data source. Choose the refresh dashboard icon to perform a query on the data source, seeing the results as you go.
6. In the **Visualization** list, select a visualization type. Grafana displays a preview of your query results with the visualization applied. For more information, see Visualizations options.
7. Under **Panel options**, you can enter a title and description for your panel.
8. Most visualizations need some adjustment before they display the exact information that you need. You can adjust panel settings in the following ways.
   - Configure value mappings
   - Visualization-specific options
   - Override field values
   - Configure thresholds
   - Configure standard options
9. When you've finished configuring your panel, choose **Save** to save the dashboard.

   Alternatively, select **Apply** to see changes without leaving the panel editor.
10. Add a note to describe the visualization (or describe your changes) and then click **Save** in the upper-right corner of the page.

> **Note**
>
> Notes are helpful if you need to revert the dashboard to a previous version.

11. Choose **Save**.
12. Optionally, you can add more panels to the dashboard by choosing **Add** in the dashboard header, and selecting **Visualization** from the drop-down.

Copying an existing dashboard

You can quickly copy an existing dashboard, to jumpstart creating a new one.

To copy an existing dashboard

1. Select **Dashboards** from the left menu.
2. Choose the dashboard you want to copy, to open it.
3. Select **Settings** (gear icon) in the top right of the dashboard.
4. Select **Save as**in the top right corner of the dashboard.
5. (Optional) Specify the name, folder, description, and whether or not to copy the original dashboard tags for the copied dashboard.
6. Select **Save**.

Configuring repeating rows

You can configure Grafana to dynamically add panels or rows to a dashboard based on the value of a variable. Variables dynamically change your queries across all rows in a dashboard. For more information about repeating panels, see Configure repeating panels.

You can also repeat rows if you have variables set with `Multi-value` or `Include all values` selected.

Before you get started, ensure that the query includes a multi-value variable, then you should complete the following steps.

To configure repeating rows

1. Select **Dashboards** from the left menu, then choose the dashboard you want to modify.

2. At the top of the dashboard, select **Add**, and then select **Row** from the drop down.

   If the dashboard is empty, you can alternately select the **+ Add row** button in the middle of the dashboard.

3. Hover over the row title and select the **Settings** (gear) icon that appears.
4. On the **Row Options** dialog box, add a title and select the variable for which you want to add repeating rows.

   > **Note**
   >
   > To provide context to dashboard users, add the variable to the row title.

5. Select **Update**.

Repeating rows and the Dashboard special data source

If a row includes panels using the special Dashboard data source—the data source that uses a result set from another panel in the smae dashboard—then corresponding panels in repeated rows will reference the panel in the original row, not the ones in the repeated rows.

For example, in a dashboard:

- `Row 1` includes `Panel 1A` and `Panel 1B`.
- `Panel 1B` uses the results from `Panel 1A` by using the `Dashboard` data source.
- Repeating `Row 2` includes `Panel 2A` and `Panel 2B`.
- `Panel 2B` references `Panel 1A`, not `Panel 2A`.

To move a panel

1. Open the dashboard.
2. Select the panel title and drag the panel to the new location. You can place a panel on a dashboard in any location.

To resize a panel

1. Open the dashboard.

2. To adjust the size of the panel, drag the lower-right corner of the panel. You can size a dashboard panel to suits your needs.

Importing dashboards

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can import preconfigured dashboards into your Amazon Managed Grafana workspace.

To import a dashboard

1. Sign into your Amazon Managed Grafana workspace.
2. Select **Dashboards** from the left menu.
3. Select **New** and choose **Import** in the drop down menu.
4. Next you need to choose the dashboard JSON definition to import. You have three choices for how to import JSON:
   - Upload a file containing dashboard JSON.
   - Directly copy JSON text into the text area.
   - Paste a Grafana Labs dashboard URL or ID into the field. For more information on grafana.com dashboard URLs, see the next section.
   - (Optional) Change any dashboard details that you wish to change.
   - Select a data source, if required.
   - Choose **Import**.
   - Save the dashboard.

Finding dashboards on grafana.com

The Dashboards page on grafana.com provides you with dashboards for common server applications. Browse a library of official and community-built dashboards and import them to quickly get up and running.

> **Note**
>
> To import dashboards from grafana.com, your Amazon Managed Grafana workspace must have access to the internet.

Exporting dashboards

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can use the Grafana UI or the HTTP API to export dashboards.

The dashboard export action creates a Grafana JSON file that contains everything you need, including layout, variables, styles, data sources, queries, and so on, so that you can later import the dashboard.

Making a dashboard portable

If you want to export a dashboard for others to use, you can add template variables for things like a metric prefix (use a constant variable) and server name.

A template variable of the type `Constant` will automatically be hidden in the dashboard, and will also be added as a required input when the dashboard is imported.

To export a dashboard

1. Open the dashboard that you want to export.

2. Select the share icon.
3. Choose **Export**.
4. Choose **Save to file.**

> **Note**
>
> Grafana downloads a JSON file to your local machine.

Modifying dashboard settings

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The dashboard settings page enables you to:

- Edit general dashboard properties, including time settings.
- Add annotation queries.
- Add dashboard variables.
- Add links.
- View the dashboard JSON model

To access the dashboard setting page

1. Open a dashboard in edit mode.
2. Click **Dashboard settings** (gear icon) located at the top of the page.

Modifying dashboard time settings

Adjust dashboard time settings when you want to change the dashboard timezone, the local browser time, and specify auto-refresh time intervals.

To modify dashboard time settings

1. On the **Dashboard** settings page, select **General**.
2. Navigate to the **Time Options** section.
3. Specify time settings according to the following descriptions.
4.
   - **Timezone** – Specify the local time zone of the service or system that you are monitoring. This can be helpful when monitoring a system or service that operates across several time zones.
     - **Default** – Grafana uses the default selected time zone for the user profile, team, or organization. If no time zone is specified for the user profile, a team the user is a member of, or the organization, then Grafana uses the local browser time.
     - **Local browser time** – The time zone configured for the viewing user browser is used. This is usually the same time zone as set on the computer.
     - Use standard ISO 8601 time zones, including UTC.
   - **Auto-refresh** – Customize the options displayed for relative time and the auto-refresh options. Entries are comma separated and accept any valid time unit.
   - **Now delay** – Override the now time by entering a time delay. Use this option to accommodate known delays in data aggregation to avoid null values.
   - **Hide time picker** – Selecting this option if you do not want the dashboard to display the time picker.

> **Note**
>
> To have time controls, your data must include a time column. See the documentation for your specific data source for more information about including a time column.

Adding an annotation query

An annotation query is a query that queries for events. These events can be visualized in graphs across the dashboard as vertical lines along with a small icon you can hover over to see the event information.

To add an annotation query

1. On the **Dashboard settings** page, select **Annotations**.
2. Select **Add annotation query**.
3. Enter a name and select a data source.
4. Complete the rest of the form to build a query and annotation.

The query editor UI changes based on the data source that you select. see the Data source documentation for details on how to construct a query. Or, for data source plugins that you install from the Find plugins with the plugin catalog, you can use the documentation on the Grafana Labs website.

Adding a variable

Variables enable you to create more interactive and dynamic dashboards. Instead of hard-coding things like server, application, and sensor names in your metric queries, you can use variables in their place. Variables are displayed as dropdown lists at the top of the dashboard. These dropdowns make it easy to change the data being displayed in your dashboard.

For more information about variables, see Variables.

To add a variable

1. On the **Dashboard settings** page, click **Variable** in the left side section menu and then the **Add variable** button.
2. In the **General** section, add the name of the variable. This is the name that you will later use in queries.
3. Select a variable **Type**.

> **Note**
>
> The variable type that you select impacts which fields that you populate on the page.

4. Define the variable and click **Update**.

Adding a link

Dashboard links enable you to place links to other dashboards and websites directly below the dashboard header. Links provide for easy navigation to other, related dashboards and content.

To add a link

1. On the **Dashboard settings** page, choose **Links** from the left side section menu and then the **Add link** button.
2. Enter a title and in the **Type** field, select **Dashboard** or **Link**.
3. To add a dashboard link, add an optional tag, select any of the dashboard link Options, and click **Apply**.

   > **Note**
   >
   > Using tags creates a dynamic dropdown of dashboards that all have a specific tag.

4. To add a web link, add a URL and tooltip text that appears when the user hovers over the link, select an icon that appears next to the link, and select any of the dashboard link options.

View dashboard JSON model

A dashboard in Grafana is represented by a JSON object, which stores the metadata of a dashboard. Dashboard metadata includes dashboard properties, metadata from panels, template variables, panel queries, and so on. The JSON metadata defines the dashboard.

To view a dashboard JSON model, on the **Dashboard settings** page, click **JSON**.

For more information about the JSON fields, see JSON fields.

Dashboard URL variables

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana can apply variable values passed as query parameters in dashboard URLs. For more information, see Manage dashboard links and Templates and variables.

Passing variables as query parameters

Grafana interprets query string parameters prefixed with `var-` as variables in the given dashboard.

For example, in this URL:

```
https://${your-domain}/path/to/your/dashboard?var-example=value
```

The query parameter `var-example=value` represents the dashboard variable example with a value of `value`.

Passing multiple values for a variable

To pass multiple values, repeat the variable parameter once for each value.

```
https://${your-domain}/path/to/your/dashboard?var-example=value1&var-example=value2
```

Grafana interprets `var-example=value1&var-example=value2` as the dashboard variable example with two values: `value1` and `value2`.

Adding variables to dashboard links

Grafana can add variables to dashboard links when you generate them from a dashboard's settings. For more information and steps to add variables, see Manage dashboard links.

Passing ad hoc filters

Ad hoc filters apply key or value filters to all metric queries that use a specified data source. For more information, see Ad hoc filters.

To pass an ad hoc filter as a query parameter, use the variable syntax to pass the ad hoc filter variable, and also provide the key, the operator as the value, and the value as a pipe-separated list.

For example, in this URL:

`https://${your-domain}/path/to/your/dashboard?var-adhoc=example_key|=|example_value`

The query parameter `var-adhoc=key|=|value` applies the ad hoc filter configured as the adhoc dashboard variable using the `example_key` key, the = operator, and the `example_value` value.

> **Note**
>
> When sharing URLs with ad hoc filters, remember to encode the URL. In the above example, replace the pipes (|) with `%7C` and the equality operator (=) with `%3D`.

Controlling time range using the URL

To set a dashboard's time range, use the `from`, `to`, `time`, and `time.window` query parameters. Because these are not variables, they do not require the `var-` prefix.

Managing library panels

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A library panel is a reusable panel that you can use in any dashboard. When you change a library panel, the change propagates to all instances of where the panel is used. Library panels streamline reuse of panels across multiple dashboards.

You can save a library panel in a folder alongside saved dashboards.

Creating a library panel

When you create a library panel, the panel on the source dashboard is converted to a library panel as well. You need to save the original dashboard after a panel is converted.

To create a library panel

1. Open the panel you want to convert to a library panel in edit mode.
2. In the panel display options, click the down arrow option to start changes to the visualization.

3. Select **Library panels**, and then **+ Create library panel**. This opens the create dialog.
4. In **Library panel name**, enter the name you want for the panel.
5. In **Save in folder**, select the folder to save the library panel.
6. Select **Create library panel** to save your changes to the library.
7. Save the dashboard.

After a library panel is created, you can modify the panel using any dashboard on which it appears. After you save the changes, all instances of the library panel reflect these modifications.

You can also create a library panel directly from the edit menu of any panel, by selecting **More...** then **Create library panel**.

Adding a library panel to a dashboard

Add a Grafana library panel to a dashboard when you want to provide visualizations to other dashboard users.

To add a library panel to a dashboard

1. Select **Dashboards** on the left menu.
2. Select **New**, and then choose **New dashboard** from the drop down.
3. On the empty dashboard, select **+ Import library panel**. You will see a list of your library panels.
4. Filter the list or search to find the panel you want to add.
5. Click a panel to add it to the dashboard.

Unlinking a library panel

Unlink a library panel when you want to make a change to the panel and not affect other instances of the library panel.

To unlink a library panel

1. Select **Dashboards** on the left menu.
2. Select **Library panels**.
3. Select a library panel that is being used in different dashboards.
4. Select the panel that you want to unlink.

5. Select the panel title (or hover the pointer anywhere over the panel), to display the actions menu on the top right corner of the panel.
6. Select **Edit**. The panel will open in edit mode.
7. Select **Unlink** on the top right corner of the page.
8. Choose **Yes, unlink**.

Viewing a list of library panels

You can view a list of available library panels and search for a library panel.

To view a list of library panels

1. Select **Dashboards** on the left menu.
2. Select **Library panels**. You can see a list of previously defined library panels.
3. Search for a specific library panel if you know its name. You can also filter the panels by folder or type.

Deleting a library panel

Delete a library panel when you no longer need it.

To delete a library panel

1. Select **Dashboards** on the left menu.
2. Select **Library panels**.
3. Select the delete icon next to the library panel name for the panel you wish to delete.

Managing dashboard version history

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Whenever you save a version of your dashboard, a copy of that version is saved so that previous versions of your dashboard are not lost. A list of these versions is available by entering the dashboard settings and then selecting **Versions** in the left side menu.

> **Note**
>
> The most recent 20 versions of a dashboard are saved.

The dashboard version history feature lets you compare and restore to previously saved dashboard versions.

Comparing two dashboard versions

To compare two dashboard versions, select the two versions from the list that you wish to compare. Click **Compare versions** to view the diff between the two versions. This brings up the version diff view. By default, you'll see a textual summary of the changes.

If you want to view the diff of the raw JSON that represents your dashboard, you can do that by clicking the **View JSON Diff** button at the bottom.

Restoring to a previously saved dashboard version

If you need to restore to a previously saved dashboard version, you can either select the **Restore** button on the right of a row in the dashboard version list, or select the **Restore to version *<x>*** button in the diff view. Selecting either of these will prompt you to confirm the restoration.

After restoring to a previous version, a new version will be created containing the same exact data as the previous version, only with a different version number. This is indicated in the **Notes column** for the row in the new dashboard version. This ensures your previous dashboard versions are not affected by the change.

Managing dashboard links

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can use links to navigate between commonly used dashboards or to connect others to your visualizations. Links let you create shortcuts to other dashboards, panels, and even external websites.

Grafana supports dashboard links, panel links, and data links. Dashboard links are displayed at the top of the dashboard. Panel links are accessible by clicking an icon on the top left corner of the panel.

Choosing which link to use

Start by figuring out how you're currently navigating between dashboards. If you're often jumping between a set of dashboards and struggling to find the same context in each, links can help optimize your workflow.

The next step is to figure out which link type is right for your workflow. Even though all the link types in Grafana are used to create shortcuts to other dashboards or external websites, they work in different contexts.

- If the link relates to most if not all of the panels in the dashboard, use dashboard links.
- If you want to drill down into specific panels, use panel links.
- If you want to link to an external site, you can use either a dashboard link or a panel link.
- If you want to drill down into a specific series, or even a single measurement, use data links.

Controlling the time range using a URL

To control the time range of a panel or dashboard, you can provide query parameters in the dashboard URL:

- `from` defines lower limit of the time range, specified in ms epoch.
- `to` defines upper limit of the time range, specified in ms epoch.
- `time` and `time.window` defines a time range from `time-time.window/2` to `time+time.window/2`. Both params should be specified in ms. For example ? `time=1500000000000&time.window=10000` will result in 10s time range from 1499999995000 to 1500000005000.

Dashboard links

When you create a dashboard link, you can include the time range and current template variables to directly jump to the same context in another dashboard. This way, you don't have to worry whether the person you send the link to is looking at the right data. For other types of links, see Data link variables.

Dashboard links can also be used as shortcuts to external systems, such as submitting a GitHub issue with the current dashboard name.

After adding a dashboard link, it will show up in the upper-right corner of your dashboard.

Adding links to dashboards

Add links to other dashboards at the top of your current dashboard.

To add a link to a dashboard

1. While viewing the dashboard you want to link, click the gear at the top of the screen to open **Dashboard settings**.
2. Select **Links** and then **Add Dashboard Link** or **New**.
3. In **Type**, select **dashboards**.
4. Select link options from the following.

   - **With tags** – Enter tags to limit the linked dashboards to only the ones with the tags you enter. Otherwise, Grafana includes links to all other dashboards.
   - **As dropdown** – If you are linking to many dashboards, By default, Grafana displays them all side-by-side across the top of your dashboard. Selecting this option and adding an optional title, will display the links in a dropdown.
   - **Time range** – Select this option to include the dashboard time range in the link. When the user clicks the link, the linked dashboard will open with the indicated time range already set.
   - **Variable values** – Select this option to include template variables currently used as query parameters in the link. When the user clicks the link, any matching templates in the linked dashboard are set to the values from the link. For more information, see Dashboard URL variables.
   - **Open in new tab** – Select this option if you want the dashboard link to open in a new tab or window.
5. Click **Add**.

Adding a URL link to a dashboard

Add a link to a URL at the top of your current dashboard. You can link to any available URL, including dashboards, panels, or external sites. You can even control the time range to ensure the user is zoomed in on the right data in Grafana.

To add a URL link to a dashboard

1. While viewing the dashboard you want to link, select the gear at the top of the screen to open **Dashboard settings**.
2. Select **Links** and then **Add Dashboard Link** or **New**.
3. In **Type**, select **Link**.
4. Select link options from the following.

   - **URL** – Enter the URL you want to link to. Depending on the target, you might want to include field values.
   - **Title** – Enter the title you want the link to display.
   - **Tooltip** – Enter the tooltip you want the link to display.
   - **Icon** – Choose the icon that you want displayed with the link.
   - **Time range** – Select this option to include the dashboard time range in the link. When the user clicks the link, the linked dashboard will open with the indicated time range set.
     - `from` – Defines lower limit of the time range, specified in ms epoch.
     - `to` – Defines upper limit of the time range, specified in ms epoch.
     - `time` and `time.window` – Define a time range from `time-time.window/2` to `time+time.window/2`. Both params should be specified in ms. For example ? `time=1500000000000&time.window=10000` will result in 10s time range from 1499999995000 to 1500000005000.
   - **Variable values** – Select this option to include template variables currently used as query parameters in the link. When the user clicks the link, any matching templates in the linked dashboard are set to the values from the link.

     The variable format is as follows:

     ```
     https://${you-domain}/path/to/your/dashboard?var-${template-varable1}=value1&var-{template-variable2}=value2
     ```

- **Open in a new tab** – Select this option if you want the dashboard link to open in a new tab or window

5. Select **Add**.

Updating a dashboard link

To change or update an existing dashboard link, follow this procedure.

To update a dashboard link

1. In **Dashboard settings,** on the **Links** tab, select the existing link that you want to edit.
2. Change the settings and then choose **Update**.

Duplicating a dashboard link

To duplicate an existing dashboard link, select the duplicate icon next to the existing link that you want to duplicate.

Deleting a dashboard link

To delete an existing dashboard link, select the trash icon next to the link that you want to delete.

Panel links

Each panel can have its own set of links that are shown in the upper left corner of the panel. You can link to any available URL, including dashboards, panels, or external sites. You can even control the time range to ensure the user is zoomed in on the right data in Grafana.

To see available panel links, select the icon to the right of the panel title.

- **Adding a panel link**: Each panel can have its own set of links that are shown in the upper left corner of the panel. You can link to any available URL, including dashboards, panels, or external sites. You can even control the time range to ensure the user is zoomed in on the right data in Grafana. Click the icon on the top left corner of a panel to see available panel links.

  1. Hover your cursor over the panel to which you want to add a link.
  2. Select the menu, and choose **Edit**, or you can use the keyboard shortcut, e.
  3. Expand the **Panel options** section, and scroll down to **Panel links**.
  4. Select **Add link**.

5. Enter a **Title**. This is a human-readable label for the link that will be displayed in the UI.
6. Enter the **URL** you want to link to. Press `Ctrl+Space` (or `Cmd+Space`) and select the URL field to see available variables. By adding template variables to your panel link, the link sends the user to the right context, with the relevant variables already set.

   You can also use time variables.

   - `from` defines the lower limit of the time range, specified in ms epoch.
   - `to` defines the upper limit of the time range, specified in ms epoch.
   - `time` and `time.window` defines a time range from `time-time.window/2` to `time+time.window/2`. Both parameters should be specified in ms. For example `?time=1500000000000&time.window=10000` results in 10s time range from 1499999995000 to 1500000005000.

- **Updating a panel link**

  1. Select a panel (or place the cursor over the panel) to display an actions menu at the top right of the panel.
  2. From the menu, select the **Edit**.

     You can also use the keyboard shortcut, e.
  3. Expand the **Panel options** section, and scroll down to **Panel links**.
  4. Find the link that you want to change, and select the **Edit** (pencil) icon next to it.
  5. Make any necessary changes.
  6. Select **Save** to save changes and close the window.
  7. Save changes to your dashboard by selecting **Save** in the upper right.

- **Deleting a panel link**

  1. Select a panel (or place the cursor over the panel) to display an actions menu at the top right of the panel.
  2. From the menu, select the **Edit**.

     You can also use the keyboard shortcut, e.
  3. Expand the **Panel options** section, and scroll down to **Panel links**.
  4. Find the link that you want to delete, and select the **X** icon next to it.
  5. Select **Save** in the upper right to save your changes to the dashboard.

Annotate visualizations

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Annotations provide a way to mark points on a visualization with rich events. They are visualized as vertical lines and icons on all graph panels. When you hover over an annotation, you can get event description and event tags. The text field can include links to other systems with more detail.

You can annotate visualizations in three ways:

- Directly in the panel, using the built-in annotations query.
- Using the Grafana HTTP API.
- Configuring annotation queries in the dashboard settings.

In the first two cases, you're creating new annotations, while in the last you're querying existing annotations from data sources. The built-in annotation query also supports this.

This section explains the first and third options; for information about using the Grafana HTTP API, refer to Annotations API.

Annotations are supported for the following visualization types:

- Time series
- State timeline
- Candlestick

Create annotations in panels

Grafana comes with the ability to add annotation events directly from a panel using the built-in annotations query that exists on all dashboards. Annotations that you create this way are stored in Grafana.

To add annotations directly in the panel:

- The dashboard must already be saved.
- The built-in query must be enabled.

To add an annotation

1. In the dashboard select the panel to which you're adding the annotation. A context menu will appear.
2. In the context menu, select **Add annotation**.
3. (Optional) Add an annotation description and tags.
4. Select **Save**.

Alternatively, to add an annotation, press `Ctrl` (or `Cmd`) while selecting the panel, and the **Add annotation** popover will appear.

Region annotations

You can also create annotations that cover a region, or period of time in a visualization.

To add a region annotation

1. In the dashboard press `Ctrl` (or `Cmd`) while selecting an area of the panel.
2. Add an annotation description and tags (optional).
3. Click **Save**.

To edit an annotation

1. In the dashboard, hover over an annotation indicator on a panel.
2. Select the **Edit** (pencil) icon in the annotation tooltip.
3. Modify the description and/or tags.
4. Select **Save**.

To delete an annotation

1. In the dashboard, hover over an annotation indicator on a panel.

2. Select the **Delete** (trash) icon in the annotation tooltip.

Fetch annotations through dashboard settings

In the dashboard settings, under **Annotations**, you can add new queries to fetch annotations using any data source, including the built-in data annotation data source. Annotation queries return events that can be visualized as event markers in graphs across the dashboard.

To add a new annotation query

1. Select the **Settings** (gear) icon in the dashboard header to open the settings menu.
2. Select **Annotations**.
3. Click **Add annotation query**.
4. Enter a name for the annotation query.

   This name is given to the toggle (checkbox) that will allow you to enable showing annotation events from this query.
5. Select the data source for the annotations.

   You can also choose **Open advanced data source picker** to see more options, including adding a new data source (available for Admins only).
6. If you don't want to use the annotation query right away, clear the **Enabled** checkbox.
7. If you don't want the annotation query toggle to be displayed in the dashboard, select the **Hidden** checkbox.
8. Select a color for the event markers.
9. In the **Show in** drop-down, choose one of the following options:

   - **All panels** – The annotations are displayed on all panels that support annotations.
   - **Selected panels** – The annotations are displayed on all the panels you select.
   - **All panels except** – The annotations are displayed on all panels except the ones you select.
10. Configure the query.

    The annotation query options are different for each data source. For information about annotations in a specific data source, see Connect to data sources.

Built-in query

After you add an annotation, they will still be visible. This is due to the built-in annotation query that exists on all dashboards. This annotation query will fetch all annotation events that originate from the current dashboard, which are stored in Grafana, and show them on the panel where they were created. This includes alert state history annotations.

By default, the built-in annotation query uses the `Grafana` special data source, and manual annotations are only supported using this data source. You can use another data source in the built-in annotation query, but you'll only be able to create automated annotations using the query editor for that data source.

To add annotations directly to the dashboard, this query must be enabled.

To confirm the built-in query is enabled

1. Select the dashboard **settings** (gear) icon in the dashboard header to open the dashboard settings menu.
2. Select **Annotations**.
3. Find the **Annotations & Alerts (Built-in)** query.

   If it shows **Disabled** before the name of the query, then you'll need to select the query name to open it and update the setting.

To stop annotations from being fetched and drawn

1. Select the dashboard **settings** (gear) icon in the dashboard header to open the dashboard settings menu.
2. Select **Annotations**.
3. Select the **Annotations & Alerts (Built-in)** query.
4. Select the **Enabled** toggle to turn it off.

When you copy a dashboard using the **Save As** feature it will get a new dashboard id, so annotations created on the source dashboard will no longer be visible on the copy. You can still show them if you add a new **Annotation Query** and filter by tags. However, this only works if the annotations on the source dashboard had tags to filter by.

Filtering queries by tag

You can create new queries to fetch annotations from the built-in annotation query using the `Grafana` data source by setting **Filter by** to `Tags`.

For example, create an annotation query name `outages` and specify a tag `outage`. This query will show all annotations (from any dashboard or via API) with the `outage` tag. If multiple tags are defined in an annotation query, then Grafana will only show annotations matching all the tags. To modify the behavior, enable **Match any**, and Grafana will show annotations that contain any one of the tags you provided.

You can also use template variables in the tag query. This means if you have a dashboard showing stats for different services and a template variable that dictates which services to show, you can use the same template variable in your annotation query to only show annotations for those services.

Dashboard JSON model

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A dashboard in Grafana is represented by a JSON object, which stores metadata of its dashboard. Dashboard metadata includes dashboard properties, metadata from panels, template variables, and panel queries.

To view the JSON of a dashboard

1. Navigate to a dashboard.
2. In the top navigation menu, select the **Dashboard settings** (gear) icon.
3. Select **JSON Model**.

JSON fields

When a user creates a new dashboard, a new dashboard JSON object is initialized with the following fields.

Note

In the following JSON, id is shown as null, which is the default value assigned to it until a dashboard is saved. After a dashboard is saved, an integer value is assigned to the `id` field.

```
{
  "id": null,
  "uid": "cLV5GDCkz",
  "title": "New dashboard",
  "tags": [],
  "timezone": "browser",
  "editable": true,
  "graphTooltip": 1,
  "panels": [],
  "time": {
    "from": "now-6h",
    "to": "now"
  },
  "timepicker": {
    "time_options": [],
    "refresh_intervals": []
  },
  "templating": {
    "list": []
  },
  "annotations": {
    "list": []
  },
  "refresh": "5s",
  "schemaVersion": 17,
  "version": 0,
  "links": []
}
```

The following describes each field in the dashboard JSON.

Name	Usage
`id`	unique numeric identifier for the dashboard (generated by the db)

Name	Usage
`uid`	unique dashboard identifier that can be generated by anyone. string (8-40)
`title`	current title of dashboard
`tags`	tags associated with dashboard, an array of strings
`style`	theme of dashboard, such as `dark` or `light`
`timezone`	timezone of dashboard, such as `utc` or `browser`
`editable`	whether a dashboard is editable or not
`graphTooltip`	0 for no shared crosshair or tooltip (default), 1 for shared crosshair, 2 for shared crosshair and shared tooltip
`time`	time range for dashboard, such as `last 6 hours` or `last 7 days`
`timepicker`	timepicker metadata, see timepicker section for details
`templating`	templating metadata, see templating section for details
`annotations`	annotations metadata, see annotations for how to add them
`refresh`	auto-refresh interval
`schemaVersion`	version of the JSON schema (integer), incremented each time a Grafana update brings changes to this schema

Name	Usage
`version`	version of the dashboard (integer), incremented each time the dashboard is updated
`panels`	panels array (see the next section for detail)

Panels

Panels are the building blocks of a dashboard. It consists of data source queries, type of graphs, aliases, and more. Panel JSON consists of an array of JSON objects, each representing a different panel. Most of the fields are common for all panels but some fields depend on the panel type. The following is an example of panel JSON of a text panel.

```
"panels": [
  {
    "type": "text",
    "title": "Panel Title",
    "gridPos": {
      "x": 0,
      "y": 0,
      "w": 12,
      "h": 9
    },
    "id": 4,
    "mode": "markdown",
    "content": "# title"
  }
```

Panel size and position

The gridPos property describes the panel size and position in grid coordinates.

- `w` – 1 to 24 (the width of the dashboard is divided into 24 columns)
- `h` – In grid height units, each represents 30 pixels.
- `x` – The x position, in the same unit as `w`.
- `y` – The y position, in the same unit as `h`.

The grid has a negative gravity that moves up panels if there is empty space above a panel.

Timepicker

```
"timepicker": {
    "collapse": false,
    "enable": true,
    "notice": false,
    "now": true,
    "refresh_intervals": [
      "5s",
      "10s",
      "30s",
      "1m",
      "5m",
      "15m",
      "30m",
      "1h",
      "2h",
      "1d"
    ],
    "status": "Stable",
    "type": "timepicker"
  }
```

Templating

The `templating` field contains an array of template variables with their saved values along with some other metadata.

```
"templating": {
    "enable": true,
    "list": [
       {
        "allFormat": "wildcard",
        "current":  {
          "tags": [],
          "text": "prod",
          "value": "prod"
        },
        "datasource": null,
        "includeAll": true,
        "name": "env",
        "options": [
           {
```

```
            "selected": false,
            "text": "All",
            "value": "*"
          },
           {
            "selected": false,
            "text": "stage",
            "value": "stage"
          },
           {
            "selected": false,
            "text": "test",
            "value": "test"
          }
        ],
        "query": "tag_values(cpu.utilization.average,env)",
        "refresh": false,
        "type": "query"
      },
       {
        "allFormat": "wildcard",
        "current":  {
          "text": "apache",
          "value": "apache"
        },
        "datasource": null,
        "includeAll": false,
        "multi": false,
        "multiFormat": "glob",
        "name": "app",
        "options": [
           {
            "selected": true,
            "text": "tomcat",
            "value": "tomcat"
          },
           {
            "selected": false,
            "text": "cassandra",
            "value": "cassandra"
          }
        ],
        "query": "tag_values(cpu.utilization.average,app)",
        "refresh": false,
```

```
        "regex": "",
        "type": "query"
      }
    ]
  }
```

The following table describes the usage of the templating fields.

Name	Usage
`enable`	whether templating is enabled or not
`list`	an array of objects each representing one template variable
`allFormat`	format to use while fetching all values from data source, including `wildcard`, `glob`, `regex`, `pipe`.
`current`	shows current selected variable text/value on the dashboard
`datasource`	shows data source for the variables
`includeAll`	whether all value option is available or not
`multi`	whether multiple values can be selected or not from variable value list
`multiFormat`	format to use while fetching timeseries from data source
`name`	name of variable
`options`	array of variable text/value pairs available for selection on dashboard
`query`	data source query used to fetch values for a variable

Name	Usage
`refresh`	configures when to refresh a variable
`regex`	extracts part of a series name or metric node segment
`type`	type of variable, `custom`, `query`, or `interval`

Best practices for dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This section provides information about best practices for Grafana administrators and users about how to build and maintain Grafana dashboards.

For information about the different kinds of dashboards you can create, refer to the Grafana dashboards: A complete guide to all the different types you can build blog post on the Grafana Labs website.

Note

This section can help you create a strategy for your monitoring and dashboard maintenance. You know your systems best, and should use this section to guide your understanding. Ultimately, it is your responsibility to create the best strategy for your system.

Common observability strategies

When you have a lot to monitor, like a server farm, you need a strategy to decide what is important enough to monitor. This page describes several common methods for choosing what to monitor.

A logical strategy allows you to make uniform dashboards and scale your observability platform more easily.

Guidelines for strategies

- The USE method tells you how happy your machines are, the RED method tells you how happy your users are.
- USE reports on causes of issues.
- RED reports on user experience and is more likely to report symptoms of problems.
- Monitoring both is important to understanding your system. As a best practice, alert on symptoms rather than causes. Typically, alerting is configured on RED dashboards.

USE method

USE stands for:

- **Utilization** – Percent time the resource is busy, such as node CPU usage.
- **Saturation** – Amount of work a resource has to do, often queue length or node load.
- **Errors** – Count of error events.

This method is best for hardware resources in infrastructure, such as CPU, memory, and network devices. For more information, see Brendan Gregg's blog post The USE Method.

RED method

RED stands for:

- **Rate** – Requests per second
- **Errors** – Number of requests that are failing.
- **Duration** – Amount of time these requests take, distribution of latency measurements.

This method is most applicable to services, especially a microservices environment. For each of your services, instrument the code to expose these metrics for each component. RED dashboards are good for alerting and SLAs. A well-designed RED dashboard is a proxy for user experience.

For more information, see Tom Wilkie's blog post The RED method: How to instrument your services.

The Four Golden Signals

According to the Google SRE handbook, if you can only measure four metrics of your user-facing system, focus on these four.

This method is similar to the RED method, but it includes saturation.

- **Latency** – Time taken to serve a request.
- **Traffic** – How much demand is placed on your system.
- **Errors** – Rate of requests that are failing.
- **Saturation** – How "full" your system is,

Dashboard management maturity model

Dashboard management maturity refers to how well-designed and efficient your dashboard ecosystem is. We recommend periodically reviewing your dashboard setup to gauge where you are and how you can improve.

Broadly speaking, dashboard maturity can be defined as low, medium, or high.

Much of the content for this topic was taken from the KubeCon 2019 talk Fool-Proof Kubernetes Dashboards for Sleep-Deprived Oncalls.

Low – default state

At this stage, you have no coherent dashboard management strategy. Almost everyone starts here.

How can you tell you are here?

- Everyone can modify your dashboards.
- Lots of copied dashboards, little to no dashboard reuse.

- One-off dashboards that hang around forever.
- No version control (dashboard JSON in version control).
- Lots of browsing for dashboards, searching for the right dashboard. This means lots of wasted time trying to find the dashboard you need.
- Not having any alerts to direct you to the right dashboard.

Medium – methodical dashboards

At this stage, you are starting to manage your dashboard use with methodical dashboards. You might have laid out a strategy, but there are some things you could improve.

How can you tell you are here?

- Prevent sprawl by using template variables. For example, you don't need a separate dashboard for each node, you can use query variables. Even better, you can make the data source a template variable too, so you can reuse the same dashboard across different clusters and monitoring backends.

  Refer to the list of examples in Variables, for ideas.
- Methodical dashboards according to an observability strategy.
- Hierarchical dashboards with drill-downs to the next level.
- Dashboard design reflects service hierarchies. For example, you could use the RED method with one row per service. The row order could reflect the data flow, as you scroll down the dashboard.
- Compare like to like: split service dashboards when the magnitude differs. Make sure aggregated metrics don't drown out important information.
- Expressive charts with meaningful use of color and normalizing axes where you can.
  - Example of meaningful color: Blue means it's good, red means it's bad. Thresholds can help with that.
  - Example of normalizing axes: When comparing CPU usage, measure by percentage rather than raw number, because machines can have a different number of cores. Normalizing CPU usage by the number of cores reduces cognitive load because the viewer can trust that at 100% all cores are being used, without having to know the number of CPUs.
- Directed browsing cuts down on guessing.
  - Template variables make it harder to browse randomly or aimlessly.
  - Most dashboards should be linked to by alerts.

  - Browsing is directed with links. For more information, see Managing dashboard links.
- Version-controlled dashboard JSON.

High – optimized use

At this stage, you have optimized your dashboard management use with a consistent and thoughtful strategy. It requires maintenance, but the results are worth it.

- Actively reducing sprawl.
  - Regularly review existing dashboards to make sure they are still relevant.
  - Only approved dashboards added to master dashboard list.
  - Tracking dashboard use. You can take advantage of Usage insights.
- Consistency by design.
- Use scripting libraries to generate dashboards, ensure consistency in pattern and style.
  - grafonnet (Jsonnet)
  - grafanalib (Python)
- No editing in the browser. Dashboard viewers change views with variables.
- Browsing for dashboards is the exception, not the rule.
- Perform experimentation and testing in a separate Grafana instance dedicated to that purpose, not your production instance. When a dashboard in the test environment is proven useful, then add that dashboard to your main Grafana instance.

Best practices for creating dashboards

This section outlines some best practices to follow when creating Grafana dashboards.

Before you begin

Here are some principles to consider before you create a dashboard.

A dashboard should tell a story or answer a question

What story are you trying to tell with your dashboard? Try to create a logical progression of data, such as large to small or general to specific. What is the goal for this dashboard? (Hint: If the dashboard doesn't have a goal, then ask yourself if you really need the dashboard.)

Keep your graphs simple and focused on answering the question that you are asking. For example, if your question is "which servers are in trouble?", then maybe you don't need to show all the server data. Just show data for the ones in trouble.

Dashboards should reduce cognitive load, not add to it

Cognitive load is basically how hard you need to think about something in order to figure it out. Make your dashboard easy to interpret. Other users and future you (when you're trying to figure out what broke at 2AM) will appreciate it.

Ask yourself:

- Can I tell what exactly each graph represents? Is it obvious, or do I have to think about it?
- If I show this to someone else, how long will it take them to figure it out? Will they get lost?

Have a monitoring strategy

It's easy to make new dashboards. It's harder to optimize dashboard creation and adhere to a plan, but it's worth it. This strategy should govern both your overall dashboard scheme and enforce consistency in individual dashboard design.

Refer to Common observability strategies and Dashboard management maturity levels for more information.

Write it down

Once you have a strategy or design guidelines, write them down to help maintain consistency over time.

Best practices to follow

- When creating a new dashboard, make sure it has a meaningful name.
  - If you are creating a dashboard to play or experiment, then put the word `TEST` or `TMP` in the name.
  - Consider including your name or initials in the dashboard name or as a tag so that people know who owns the dashboard.
  - Remove temporary experiment dashboards when you are done with them.
- If you create many related dashboards, think about how to cross-reference them for easy navigation. For more information, see Best practices for managing dashboards, later in this section.

- Grafana retrieves data from a data source. A basic understanding of Connect to data sources in general, and your specific data sources is important.
- Avoid unnecessary dashboard refreshing to reduce the load on the network or backend. For example, if your data changes every hour, then you don't need to set the dashboard refresh rate to 30 seconds.
- Use the left and right Y-axes when displaying time series with different units or ranges.
- Add documentation to dashboards and panels.
  - To add documentation to a dashboard, add a Text panel visualization to the dashboard. Record things like the purpose of the dashboard, useful resource links, and any instructions users might need to interact with the dashboard.
  - To add documentation to a panel, edit the panel settings and add a description. Any text you add will appear if you hover your cursor over the small `i` in the top left corner of the panel.
- Reuse your dashboards and enforce consistency by using templates and variables.
- Be careful with stacking graph data. The visualizations can be misleading, and hide important data. We recommend turning it off in most cases.

Best practices for managing dashboards

This page outlines some best practices to follow when managing Grafana dashboards.

Before you begin

Here are some principles to consider before you start managing dashboards.

Strategic observability

There are several common observability strategies. You should research them and decide whether one of them works for you or if you want to come up with your own. Either way, have a plan, write it down, and stick to it.

Adapt your strategy to changing needs as necessary.

Maturity level

What is your dashboard maturity level? Analyze your current dashboard setup and compare it to the Dashboard management maturity model. Understanding where you are can help you decide how to get to where you want to be.

Best practices to follow

- Avoid dashboard sprawl, meaning the uncontrolled growth of dashboards. Dashboard sprawl negatively affects time to find the right dashboard. Duplicating dashboards and changing "one thing" (worse: keeping original tags) is the easiest kind of sprawl.
  - Periodically review the dashboards and remove unnecessary ones.
  - If you create a temporary dashboard, perhaps to test something, prefix the name with `TEST:`. Delete the dashboard when you are finished.
- Copying dashboards with no significant changes is not a good idea.
  - You miss out on updates to the original dashboard, such as documentation changes, bug fixes, or additions to metrics.
  - In many cases copies are being made to simply customize the view by setting template parameters. This should instead be done by maintaining a link to the master dashboard and customizing the view with URL parameters.
- When you must copy a dashboard, clearly rename it and *do not* copy the dashboard tags. Tags are important metadata for dashboards that are used during search. Copying tags can result in false matches.
- Maintain a dashboard of dashboards or cross-reference dashboards. This can be done in several ways:
  - Create dashboard links, panel, or data links. Links can go to other dashboards or to external systems. For more information, refer to Manage dashboard links.
  - Add a Dashboard list panel. You can then customize what you see by doing tag or folder searches.
  - Add a Text panel and use markdown to customize the display.

Managing dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

On the **Dashboards** page of your workspace (available by selecting **Dashboards** from the left menu), you can perform dashboard management tasks, including organizing your dashboards into folders.

For more information about creating dashboards, see Building dashboards.

Browse dashboards

On the **Dashboards** page, you can browse and manage folders and dashboards. This includes options to:

- Create folders and dashboards.
- Move dashboards between folders.
- Delete multiple dashboards and folders.
- Navigate to a folder.
- Manage folder permissions. For more information, see Dashboard and folder permissions.

Creating dashboard folders

Folders help you organize and group dashboards, which is useful when you have many dashboards or multiple teams using the same Grafana instance. Subfolders allow you to create a nested hierarchy in your dashboard organization.

Prerequisites

Ensure that you have Grafana Admin permissions. For more information about dashboard permissions, see Dashboard and folder permissions.

To create a dashboard folder

1. Sign in to Grafana.
2. On the left menu, select **Dashboards**.
3. On the **Dashboards** page, select **New** then choose **New folder** in the drop down.
4. Enter a unique name and click **Create**.

Note

When you save a dashboard, you can either select a folder for the dashboard to be saved in or create a new folder.

To edit the name of a folder

1. Select **Dashboards**in the left menu.
2. Select the folder to rename
3. Select the **Edit title** (pencil) icon in the header and update the name of the folder.

   The new folder name is automatically saved.

Folder permissions

You can assign permissions to a folder. Dashboard in the folder inherit any permissions that you've assigned to the folder. You can assign permissions to organization roles, teams, and users.

To modify permissions for a folder

1. Select **Dashboards** from the left menu.
2. Select the folder in the list.
3. On the folder's details page, select **Folder actions** and select **Manage permissions** in the drop down list.
4. Update the permissions as desired.

Changes are saved automatically.

Managing playlists

This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A *playlist* is a list of dashboards that are displayed in a sequence. You might use a playlist to build situational awareness or to present your metrics to your team or visitors. Grafana automatically scales dashboards to any resolution, which makes them perfect for large screens. You can access the playlist feature from Grafana's side menu in the **Dashboards** submenu.

Accessing, sharing, and controlling a playlist

Use the information in this section to access existing playlists. Start and control the display of a playlist using one of the five available modes.

To access a playlist

1. Select **Playlists** from the left menu.
2. Choose a playlist from the list of existing playlists.

Starting a playlist

You can start a playlist in five different view modes. View mode determines how the menus and navigation bar appear on the dashboards.

By default, each dashboard is displayed for the amount of time entered in the **Interval** field, which you set when you create or edit a playlist. After you start a playlist, you can control it with the navigation bar at the top of the page.

To start a playlist

1. Access the playlist page to see a list of existing playlists.
2. Find the playlist that you want to start, then click **Start playlist**.

   The start playlist dialog box will open.
3. Select one of the five playlist modes available based on the information in the following table.
4. Click **Start <playlist name>**.

The playlist displays each dashboard for the time specified in the `Interval` field, set when creating or editing a playlist. After a playlist starts, you can control it using the navigation bar at the top of your screen.

Mode	Description
Normal mode	• The side menu remains visible. • The navigation bar, row, and panel controls appear at the top of the screen.
TV mode	• The side menu and dashboard submenu (including variable dropdowns and dashboard links) are hidden or removed. • The navigation bar, row, and panel controls appear at the top of the screen. • Enabled automatically after one minute of user inactivity. • Enable it manually using the `d v` sequence shortcut, or by appending the parameter `?inactive` to the dashboard URL. • Disable it with any pointer movement or keyboard action.
TV mode (with auto fit panels)	• The navigation bar, row, and panel controls appear at the top of the screen. • Dashboard panels automatically adjust to optimize space on screen.
Kiosk mode	• The side menu, navigation bar, row and panel controls are completely hidden/re moved from view. • You can enable it manually using the `d v` sequence shortcut after the playlist has started. • You can disable it manually with the same shortcut.
Kiosk mode (with auto fit panels)	• The side menu, navigation bar, row, and panel controls are completely hidden/re moved from view.

Mode	Description
	• Dashboard panels automatically adjust to optimize space on screen.

Controlling a playlist

You can control a playlist in **Normal** or **TV** mode after it has started, using the navigation bar at the top of your screen. Press the `Esc` key in your keyboard to stop the playlist.

Button	Action
Next (double-right arrow)	Advances to the next dashboard.
Back (left arrow)	Returns to the previous dashboard.
Stop (square)	Ends the playlist, and exits to the current dashboard.
Cycle view mode (monitor icon)	Rotates the display of the dashboards in different view modes.
Time range	Displays data within a time range. It can be set to display the last 5 minutes up to 5 years ago, or a custom time range, using the down arrow.
Refresh (circle arrow)	Reloads the dashboard, to display the current data. It can be set to reload automatically every 5 seconds to 1 day, using the dropdown arrow.

Creating a playlist

You can create a playlist to present dashboards in a sequence with a set order and time interval between dashboards.

To create a playlist

1. Select **Dashboards** from the left menu.

2. Select **Playlists** on the playlist page.
3. Select **New playlist**.
4. Enter a descriptive name in the **Name** text box.
5. Enter a time interval in the **Interval** text box. The dashboards you add are listed in a sequential order.
6. In **Dashboards**, add existing dashboards to the playlist using the **Add by title** and **Add by tag** dropdown options.
7. Optionally:
   - Search for a dashboard by its name, a regular expression, or a tag.
   - Filter your results by starred status or tags.
   - Rearrange the order of the dashboards you have added using the up and down arrow icon.
   - Remove a dashboard from the playlist by clicking the **x** icon beside the dashboard.
8. Select **Save** to save your changes.

Saving a playlist

You can save a playlist and add it to your **Playlists** page, where you can start it.

> **Important**
>
> Ensure all the dashboards that you want to appear in your playlist are added when creating or editing the playlist before saving it.

To save a playlist

1. Select **Dashboards** in the left menu.
2. Select **Playlists** to view the playlists available to you.
3. Choose the playlist of your choice.
4. Edit the playlist.
5. Check that the playlist has a **Name**, **Interval**, and at least one **Dashboard** added to it.
6. Select **Save** to save your changes.

Editing or deleting a playlist

You can edit a playlist by updating its name, interval time, and by adding, removing, and rearranging the order of dashboards, or you can delete the playlist.

To edit a playlist

1. Select **Edit playlist** on the playlist page.
2. Update the name and time interval, then add or remove dashboards from the playlist using instructions in Create a playlist, above.
3. Select **Save** to save your changes.

To delete a playlist

1. Select **Playlists**.
2. Select **Remove** next to the playlist you want to delete.

To rearrange dashboard order in a playlist

1. Next to the dashboard you want to move, click the up or down arrow.
2. Select **Save** to save your changes.

To remove a dashboard

1. Select **Remove** to remove a dashboard from the playlist.
2. Select **Save** to save your changes.

Sharing a playlist in view mode

You can share a playlist by copying the link address on the view mode you prefer, and pasting the URL to your destination.

To share a playlist in view mode

1. From the **Dashboards** left side menu, choose **Playlists**.
2. Select **Start playlist** next to the playlist you want to share.
3. In the dropdown, right click the view mode you prefer.

4. Select **Copy Link Address** to copy the URL to your clipboard.
5. Paste the URL to your destination.

Sharing dashboards and panels

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana enables you to share dashboards and panels with other users within an organization and in certain situations, publicly on the Web. You can share using:

- A direct link
- A snapshot
- An export link (for dashboards only)

You must have an authorized viewer permission to see an image rendered by a direct link.

When you share a panel or dashboard as a snapshot, a snapshot (which is a panel or dashboard at the moment you take the snapshot) is publicly available on the web. Anyone with a link to it can access it. Because snapshots do not require any authorization to view, Grafana removes information related to the account it came from, as well as any sensitive data from the snapshot.

Sharing a dashboard

You can share a dashboard as a direct link or as a snapshot. You can also export a dashboard.

> **Note**
> If you change a dashboard, ensure that you save the changes before sharing.

To Share a dashboard

1. Select **Dashboards** from the left menu in your workspace.
2. Choose the dashboard you want to share.
3. Select the share icon at the top of the screen.

   The share dialog box opens and shows the **Link** tab.

Sharing a direct link

The **Link** tab shows the current time range, template variables, and the default theme. You can also share a shortened URL.

To share a directly link

1. Select **Copy**. This action copies the default or the shortened URL to the clipboard.
2. Send the copied URL to a Grafana user with authorization to view the link.

Publishing a snapshot

A dashboard snapshot shares an interactive dashboard publicly. Grafana strips sensitive data such as queries (metric, template, and annotation) and panel links, leaving only the visible metric data and series names embedded in the dashboard. Dashboard snapshots can be accessed by anyone with the link.

You can publish snapshots to your local instance.

To publish a snapshot

1. Select the **Snapshot** tab.
2. Select the **Local Snapshot**.
3. Grafana generates a link of the snapshot. Copy the snapshot link, and share it either within your organization or publicly on the web.

Exporting a dashboard

Grafana dashboards can easily be exported and imported. For more information, see the import and export sections in Building dashboards.

Sharing a panel

You can share a panel as a direct link, or as a snapshot. You can also create library panels using the **Share** option on any panel.

To share a panel

1. Select the panel title of the panel you want to share. The panel menu opens.
2. Select **Share**. The share dialog box will open and show the **Link** tab.

Using a direct link

The **Link** tab shows the current time range, template variables, and the default theme. You can optionally enable a shortened URL to share.

To use a direct link

1. Select **Copy** to copy the default or the shortened URL to the clipboard.
2. Send the copied URL to a Grafana user with authorization to view the link.

Publishing a snapshot of a panel

A panel snapshot is a share of an interactive panel publicly. Grafana strips sensitive data leaving only the visible metric data and series names embedded in the dashboard. Panel snapshots can be accessed by anyone with the link.

You can publish snapshots to your local instance.

To publish a snapshot of a panel

1. In the **Share Panel** dialog box, select the **Snapshot** tab.
2. Select **Local Snapshot**. Grafana generates the link of the snapshot.
3. Copy the snapshot link, and share it either within your organization or publicly on the web.

If you created a snapshot by mistake, click **Delete snapshot** to remove the snapshot from your Grafana instance.

Creating a library panel

To create a library panel from the **Share Panel** dialog box.

To create a library panel

1. Select **Library panel**.
2. In **Library panel name**, enter the name.
3. In **Save in folder**, select the folder in which to save the library panel. By default, the root folder is selected.
4. Select **Create library panel** to save your changes.
5. Save the dashboard.

Variables

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A variable is a placeholder for a value. You can use variables in metric queries and in panel titles. So when you change the value, using the dropdown at the top of the dashboard, your panel's metric queries will change to reflect the new value.

Variables allow you to create more interactive and dynamic dashboards. Instead of hard-coding things like server, application, and sensor names in your metric queries, you can use variables in their place. Variables are displayed as dropdown lists at the top of the dashboard. These dropdowns make it easy to change the data being displayed in your dashboard.

These can be especially useful for administrators who want to allow Grafana viewers to quickly adjust visualizations but do not want to give them full editing permissions. Grafana Viewers can use variables.

Variables and templates also allow you to single-source dashboards. If you have multiple identical data sources or servers, you can make one dashboard and use variables to change what you are viewing. This simplifies maintenance and upkeep enormously.

Templates

A template is any query that contains a variable. For example, if you were administering a dashboard to monitor several servers, you could make a dashboard for each server, or you could create one dashboard and use panels with template queries, such as the following.

```
wmi_system_threads{instance=~"$server"}
```

Variable values are always synced to the URL using the syntax var-<varname>=value.

Examples

Variables are listed in dropdown lists across the top of the screen. Select different variables to see how the visualizations change.

To see variable settings, navigate to **Dashboard Settings > Variables**. Click a variable in the list to see its settings.

Variables can be used in titles, descriptions, text panels, and queries. Queries with text that starts with $ are templates. Not all panels will have template queries.

Variable best practices

- Variable dropdown lists are displayed in the order they are listed in the variable list in **Dashboard settings**.
- Put the variables that you will change often at the top, so they will be shown first (far left on the dashboard).
- Variables preselect the topmost value in the dropdown list by default. If you want to choose an empty value instead, change the variable settings, as follows:
  1. Select the **Include All Option** checkbox.
  2. In the **Custom all value** field, enter the value +.

Topics

- Add and manage variables
- Inspect Variables
- Variable syntax

Add and manage variables

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The following table lists the types of variables in Grafana.

Variable type	Description
Query	Query-generated list of values such as metric names, server names, sensor IDs, data centers, and so on.
Custom	Define the variable options manually using a comma-separated list.
Text box	Display a free text input field with an optional default value.
Constant	Define a hidden constant.
Data source	Quickly change the data source for an entire dashboard.
Interval	Interval variables represent time spans.
Ad hoc filters	Key-value filters that are automatically added to all metric queries for a data source (Prometheus, Loki, InfluxDB, and Elasticsearch only).
Global variables	Built-in variables that can be used in expressio ns in the query editor.

Variable type	Description
Chained variables	Variable queries can contain other variables.

Topics

- Entering General options
- Adding a query variable
- Adding a custom variable
- Adding a text box variable
- Adding a constant variable
- Adding a data source variable
- Adding an interval variable
- Adding ad hoc filters
- Configure variable selection options
- Global variables
- Chained variables
- Manage variables
- Filter variables with regex

Entering General options

You must enter general options for any type of variable that you create.

To enter general options

1. Navigate to the dashboard you want to make a variable for and select the **Dashboard settings** (gear) icon at the top of the page.
2. On the **Variables** tab, select **New variable**.
3. Enter a **Name** for the variable.
4. In the **Type** list, select **Query**.
5. (Optional) In **Label**, enter the display name of the variable dropdown.

   If you don't enter a display name, then the dropdown label is the variable name.

6. Choose a **Hide** option:

   - **No selection (blank)** – The variable dropdown displays the variable **Name** or **Label** value.
   - **Label** – The variable dropdown only displays the selected variable value and a down arrow.
   - **Variable** – No variable dropdown is displayed on the dashboard.

Adding a query variable

Query variables enable you to write a data source query that can return a list of metric names, tag values, or keys. For example, a query variable might return a list of server names, sensor IDs, or data centers. The variable values change as they dynamically fetch options with a data source query.

Query variables are generally only supported for strings. If your query returns numbers or any other data type, you might need to convert them to strings in order to use them as variables. For the Azure data source, for example, you can use the tostring function for this purpose.

Query expressions can contain references to other variables and in effect create linked variables. Grafana detects this and automatically refreshes a variable when one of its linked variables change.

> **Note**
>
> Query expressions are different for each data source. For more information, refer to the documentation for your data source.

To add a query variable

1. Enter general options, as above.
2. In the **Data source** list, select the target data source for the query.
3. In the **Refresh** list, select when the variable should update options.

   - **On Dashboard Load** – Queries the data source every time the dashboard loads. This slows down dashboard loading, because the variable query needs to be completed before dashboard can be initialized.
   - **On Time Range Change** – Queries the data source when the dashboard time range changes. Only use this option if your variable options query contains a time range filter or is dependent on the dashboard time range.

4. In the **Query** field, enter a query.

   - The query field varies according to your data source. Some data sources have custom query editors.
   - The query must return values named `__text` and `__value`. For example, in SQL, you can use a query such as `SELECT hostname AS __text, id AS __value from MyTable`. Queries for other languages will vary depending on syntax.
   - If you need more room in a single input field query editor, then hover your cursor over the lines in the lower right corner of the field and drag downward to expand.
5. (Optional) In the **Regex** field, type a regex expression to filter or capture specific parts of the names returned by your data source query. To see examples, refer to Filter variables with regex.
6. In the **Sort** list, select the sort order for values to be displayed in the dropdown list. The default option, **Disabled**, means that the order of options returned by your data source query will be used.
7. (Optional) Enter Selection Options.
8. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
9. Select **Add** to add the variable to the dashboard.

Adding a custom variable

Use a *custom* variable for a value that does not change, such as a number or a string.

For example, if you have server names or Region names that never change, then you might want to create them as custom variables rather than query variables. Because they do not change, you might use them in chained variables rather than other query variables. That would reduce the number of queries Grafana must send when chained variables are updated.

To add a custom variable

1. Enter general options, as above.
2. In the **Values separated by comma** list, enter the values for this variable in a comma-separated list. You can include numbers, strings, or key-value pairs separated by a space and a colon. For example, `key1 : value1,key2 : value2`.
3. (Optional) Enter Selection Options.

4. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
5. Select **Add** to add the variable to the dashboard.

Adding a text box variable

Text box variables display a free text input field with an optional default value. This is the most flexible variable, because you can enter any value. Use this type of variable if you have metrics with high cardinality or if you want to update multiple panels in a dashboard at the same time.

To add a text box variable

1. Enter general options, as above.
2. (Optional) In the **Default value** field, select the default value for the variable. If you do not enter anything in this field, then Grafana displays an empty text box for users to type text into.
3. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
4. Select **Add** to add the variable to the dashboard.

Adding a constant variable

Constant variables enable you to define a hidden constant. This is useful for metric path prefixes for dashboards you want to share. When you export a dashboard, constant variables are converted to import options.

Constant variables are *not* flexible. Each constant variable only holds one value, and it cannot be updated unless you update the variable settings.

Constant variables are useful when you have complex values that you need to include in queries but don't want to retype in every query. For example, if you had a server path called `i-0b6a61efe2ab843gg`, then you could replace it with a variable called `$path_gg`.

To add a constant variable

1. Enter general options, as above.
2. In the **Value** field, enter the variable value. You can enter letters, numbers, and symbols. You can even use wildcards if you use raw format.

3. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
4. Select **Add** to add the variable to the dashboard.

Adding a data source variable

Data source variables enable you to quickly change the data source for an entire dashboard. They are useful if you have multiple instances of a data source, perhaps in different environments.

To add a data source variable

1. Enter general options, as above.
2. In the **Type** list, select the target data source for the variable.

   You can also choose **Open advanced data source picker** to see more options, including adding a data source (Admins only). For more information, see Connect to data sources.
3. (Optional) In **Instance name filter**, enter a regex filter for which data source instances to choose from in the variable value dropdown list. Leave this field empty to display all instances.
4. (Optional) Enter Selection Options.
5. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
6. Select **Add** to add the variable to the dashboard.

Adding an interval variable

Use an *interval* variable to represents time spans such as `1m`,`1h`, or `1d`. You can think of them as a dashboard-wide *group by time* command. Interval variables change how the data is grouped in the visualization. You can also use the Auto Option to return a set number of data points per time span.

You can use an interval variable as a parameter to group by time (for InfluxDB), date histogram interval (for Elasticsearch), or as a summarize function parameter (for Graphite).

To add an interval variable

1. Enter general options, as above.

2. In the **Values** field, enter the time range intervals that you want to appear in the variable dropdown list. The following time units are supported: `s (seconds)`, `m (minutes)`, `h (hours)`, `d (days)`, `w (weeks)`, `M (months)`, and `y (years)`. You can also accept or edit the default values: `1m,10m,30m,1h,6h,12h,1d,7d,14d,30d`.
3. (Optional) Turn on the **Auto Option** if you want to add the `auto` option to the list. This option allows you to specify how many times the current time range should be divided to calculate the current `auto` time span. If you turn it on, then two more options appear:
   - **Step count** – Select the number of times the current time range will be divided to calculate the value, similar to the **Max data points** query option. For example, if the current visible time range is 30 minutes, then the `auto` interval groups the data into 30 one-minute increments. The default value is 30 steps.
   - **Min Interval** – The minimum threshold below which the step count intervals will not divide the time. To continue the 30 minute example, if the minimum interval is set to 2m, then Grafana would group the data into 15 two-minute increments.
4. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
5. Select **Add** to add the variable to the dashboard.

Interval variable examples

The following example shows a template variable `myinterval` in a Graphite function:

```
summarize($myinterval, sum, false)
```

Adding ad hoc filters

Ad hoc filters enable you to add key-value filters that are automatically added to all metric queries that use the specified data source. Unlike other variables, you do not use ad hoc filters in queries. Instead, you use ad hoc filters to write filters for existing queries.

> **Note**
>
> Ad hoc filter variables only work with Prometheus, Loki, InfluxDB, and Elasticsearch data sources.

1. Enter general options, as above.
2. In the **Data source** list, select the target data source.

   You can also choose **Open advanced data source picker** to see more options, including adding a data source (Admins only). For more information, see Connect to data sources.
3. Select **Add** to add the variable to the dashboard.

Create ad hoc filters

Ad hoc filters are one of the most complex and flexible variable options available. Instead of a regular list of variable options, this variable allows you to build a dashboard-wide ad hoc query. Filters you apply in this manner are applied to all panels on the dashboard.

Configure variable selection options

Selection Options are a feature you can use to manage variable option selections. All selection options are optional, and they are off by default.

Multi-value variables

Interpolating a variable with multiple values selected is tricky as it is not straight forward how to format the multiple values into a string that is valid in the given context where the variable is used. Grafana tries to solve this by allowing each data source plugin to inform the templating interpolation engine what format to use for multiple values.

> **Note**
>
> The **Custom all value** option on the variable must be blank for Grafana to format all values into a single string. If it is left blank, then Grafana concatenates (adds together) all the values in the query. For example, `value1,value2,value3`. If a custom `all` value is used, then instead the value will be `*` or `all`.

Multi-value variables with a Graphite data source

Graphite uses glob expressions. A variable with multiple values is, in this case, be interpolated as `{host1,host2,host3}` if the current variable value iss *host1*, *host2*, and *host3*.

Multi-value variables with a Prometheus or InfluxDB datasource

InfluxDB and Prometheus use regex expressions, so the same variable is interpolated as `(host1|host2|host3)`. Every value is also regex escaped. If it were not, a value with a regex control character would break the regex expression.

Multi-value variables with an Elastic data source

Elasticsearch uses lucene query syntax, so the same variable is formatted as `("host1" OR "host2" OR "host3")`. In this case, every value is escaped so that the value only contains lucene control words and quotation marks.

Troubleshoot multi-value variables

Automatic escaping and formatting can cause problems and it can be tricky to grasp the logic behind it. Especially for InfluxDB and Prometheus where the use of regex syntax requires that the variable is used in regex operator context.

If you do not want Grafana to do this automatic regex escaping and formatting, then you must do one of the following:

- Turn off the **Multi-value** or **Include All option** options.
- Use the raw format.

Include All option

Grafana adds an `All` option to the variable dropdown list. If a user selects this option, then all variable options are selected.

Custom all value

This option is only visible if the **Include All option** is selected.

Enter regex, globs, or lucene syntax in the **Custom all value** field to define the value of the `All` option.

By default the `All` value includes all options in a combined expression. This can become very long and can have performance problems. Sometimes it can be better to specify a custom all value, like a wildcard regex.

To have custom regex, globs, or lucene syntax in the **Custom all value** option, it is never escaped so you will have to think about what is a valid value for your data source.

Global variables

Grafana has global built-in variables that can be used in expressions in the query editor. This topic lists them in alphabetical order and defines them. These variables are useful in queries, dashboard links, panel links, and data links.

$__dashboard

This variable is the name of the current dashboard.

$__from and $__to

Grafana has two built-in time range variables: `$__from` and `$__to`. They are currently always interpolated as epoch milliseconds by default, but you can control date formatting.

Syntax	Example result	Description
`${__from}`	1594671549254	Unix millisecond epoch
`${__from:date}`	2020-07-13T20:19:09.254Z	No args, defaults to ISO 8601/RFC 3339
`${__from:date:iso}`	2020-07-13T20:19:09.254Z	ISO 8601/RFC 3339
`${__from:date:seconds}`	1594671549	Unix seconds epoch
`${__from:date:YYYY-MM}`	2020-07	Any custom date format that does not include the : character

The syntax above also works with `${__to}`.

$__interval

You can use the `$__interval` variable as a parameter to group by time (for InfluxDB, MySQL, Postgres, MSSQL), Date histogram interval (for Elasticsearch), or as a *summarize* function parameter (for Graphite).

Grafana automatically calculates an interval that can be used to group by time in queries. When there are more data points than can be shown on a graph, the queries can be made more efficient

by grouping by a larger interval. For example, if you are looking at a graph of 3 months worth of data, you might not be able to see detail at the minute level. Grouping by the hour or day makes the query more efficient without affecting what the graph shows. The `$__interval` is calculated using the time range and the width of the graph (the number of pixels).

Approximate Calculation: `(to - from) / resolution`

For example, when the time range is 1 hour and the graph is full screen, then the interval might be calculated to `2m` - points are grouped in 2 minute intervals. If the time range is 6 months and the graph is full screen, then the interval might be `1d` (1 day) - points are grouped by day.

In the InfluxDB data source, the legacy variable `$interval` is the same variable. `$__interval` should be used instead.

The InfluxDB and Elasticsearch data sources have `Group by time interval` fields that are used to hard code the interval or to set the minimum limit for the `$__interval` variable (by using the > syntax, for example `>10m`).

$__interval_ms

This variable is the `$__interval` variable in milliseconds, not a time interval formatted string. For example, if the `$__interval` is `20m` then the `$__interval_ms` is `1200000`.

$__org

This variable is the ID of the current organization. `${__org.name}` is the name of the current organization.

$__user

`${__user.id}` is the ID of the current user. `${__user.login}` is the login handle of the current user. `${__user.email}` is the email for the current user.

$__range

Only supported for Prometheus and Loki data sources. This variable represents the range for the current dashboard. It is calculated by `to - from`. It has a millisecond and a second representation called `$__range_ms` and `$__range_s`.

$__rate_interval

Only supported for Prometheus data sources. The `$__rate_interval` variable is meant to be used in the rate function.

$timeFilter or $__timeFilter

The `$timeFilter` variable returns the currently selected time range as an expression. For example, the time range interval `Last 7 days` expression is `time > now() - 7d`.

This is used in several places, including:

- The WHERE clause for the InfluxDB data source. Grafana adds it automatically to InfluxDB queries when in Query Editor mode. You can add it manually in Text Editor mode: `WHERE $timeFilter`.
- Log Analytics queries in the Azure Monitor data source.
- SQL queries in MySQL, Postgres, and MSSQL.
- The `$__timeFilter` variable is used in the MySQL data source.

$__timezone

The `$__timezone` variable returns the currently selected time zone, either `utc` or an entry of the IANA time zone data base (for example, `America/New_York`).

If the currently selected time zone is *Browser Time*, Grafana will try to determine your browser time zone.

Chained variables

Chained variables, also called *linked variables* or *nested variables*, are query variables with one or more other variables in their variable query.

Chained variable queries are different for every data source, but the premise is the same for all. You can use chained variable queries in any data source that allows them.

Extremely complex linked templated dashboards are possible, 5 or 10 levels deep. Technically, there is no limit to how deep or complex you can go, but the more links you have, the greater the query load.

Best practices and tips

The following practices will make your dashboards and variables easier to use.

Creating new linked variables

- Chaining variables create parent/child dependencies. You can envision them as a ladder or a tree.
- The easiest way to create a new chained variable is to copy the variable that you want to base the new one on. In the variable list, click the **Duplicate variable** icon to the right of the variable entry to create a copy. You can then add on to the query for the parent variable.
- New variables created this way appear at the bottom of the list. You might need to drag it to a different position in the list to get it into a logical order.

Variable order

You can change the orders of variables in the dashboard variable list by clicking the up and down arrows on the right side of each entry. Grafana lists variable dropdowns left to right according to this list, with the variable at the top on the far left.

- List variables that do not have dependencies at the top, before their child variables.
- Each variable should follow the one it is dependent on.
- Remember there is no indication in the UI of which variables have dependency relationships. List the variables in a logical order to make it easy on other users (and yourself).

Complexity consideration

The more layers of dependency you have in variables, the longer it will take to update dashboards after you change variables.

For example, if you have a series of four linked variables (country, Region, server, metric) and you change a root variable value (country), then Grafana must run queries for all the dependent variables before it updates the visualizations in the dashboard.

Manage variables

The variables page lets you add variables and manage existing variables. It also allows you to inspect variables and identify whether a variable is being referenced (or used) in other variables or dashboard.

Move – You can move a variable up or down the list using drag and drop.

Clone – To clone a variable, click the clone icon from the set of icons on the right. This creates a copy of the variable with the name of the original variable prefixed with `copy_of_`.

Delete – To delete a variable, click the trash icon from the set of icons on the right.

Filter variables with regex

Using the Regex Query option, you filter the list of options returned by the variable query or modify the options returned.

This page shows how to use regex to filter/modify values in the variable dropdown.

Using the Regex Query Option, you filter the list of options returned by the Variable query or modify the options returned. For more information, refer to the Mozilla guide on Regular expressions.

The following examples show filtering on the following list of options

```
backend_01
backend_02
backend_03
backend_04
```

Filter so that only the options that end with `01` or `02` are returned

Regex:

```
/
(
01|02
)
$/
```

Result:

```
backend_01
backend_02
```

Filter and modify the options using a regex capture group to return part of the text

Regex:

```
/.*
(
01|02
```

```
)
/
```

Result:

```
01
02
```

Filter and modify - Prometheus Example

For this list of options:

```
up{instance="demo.robustperception.io:9090",job="prometheus"} 1 1521630638000
up{instance="demo.robustperception.io:9093",job="alertmanager"} 1 1521630638000
up{instance="demo.robustperception.io:9100",job="node"} 1 1521630638000
```

This regex:

```
/. *instance="
(
[^"]*
)
.*/
```

Returns these results:

```
demo.robustperception.io:9090
demo.robustperception.io:9093
demo.robustperception.io:9100
```

Filter and modify using named text and value capture groups

Using named capture groups, you can capture separate 'text' and 'value' parts from the options returned by the variable query. This allows the variable dropdown list to contain a friendly name for each value that can be selected.

For example, when querying the `node_hwmon_chip_names` Prometheus metric, the `chip_name` is a lot friendlier that the `chip` value. So the following variable query result:

```
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_0",chip_name="enp216s0f0np0"} 1
```

```
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_1",chip_name="enp216s0f0np1"} 1
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_2",chip_name="enp216s0f0np2"} 1
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_3",chip_name="enp216s0f0np3"} 1
```

Passed through the following Regex:

```
/chip_name="(?<text>[ ^ " ] + ) |chip=" (?<value >[ ^ " ] + )/g
```

Would produce the following dropdown list:

```
Display Name          Value
------------          -------------------------
enp216s0f0np0         0000:d7:00_0_0000:d8:00_0
enp216s0f0np1         0000:d7:00_0_0000:d8:00_1
enp216s0f0np2         0000:d7:00_0_0000:d8:00_2
enp216s0f0np3         0000:d7:00_0_0000:d8:00_3
```

Only `text` and `value` capture group names are supported.

Inspect Variables

The variables page lets you easily identify whether a variable is being referenced (or used) in other variables or dashboard.

Any variable that is referenced or used has a green check mark next to it, while unreferenced variables have an orange caution icon next to them. In addition, all referenced variables have a dependency icon next to the green check mark. You can select the icon to view the dependency map. The dependency map can be moved. You can zoom in or out with the mouse wheel or equivalent.

Variable syntax



This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Panel titles and metric queries can refer to variables using two different syntaxes.

- `$varname` – This syntax is easy to read, but it does not allow you to use a variable in the middle of a word.

  Example: `apps.frontend.$server.requests.count`
- `${var_name}` – Use this syntax when you want to use a variable in the middle of an expression.
- `${var_name:<format>}` – This format gives you more control over how Grafana interprets values. For more information, see *Advanced variable format options*, following this list.
- `[[varname]]` – Do not use. This syntax is old and has been deprecated. It will be removed in a future release.

Before queries are sent to your data source the query is *interpolated*, meaning the variable is replaced with its current value. During interpolation, the variable value might be *escaped* in order to conform to the syntax of the query language and where it is used. For example, a variable used in a regex expression in an InfluxDB or Prometheus query will be regex escaped.

Advanced variable format options

The formatting of the variable interpolation depends on the data source, but there are some situations where you might want to change the default formatting.

For example, the default for the MySQL data source is to join multiple values as comma-separated with quotes: `'server01','server02'`. In some cases, you might want to have a comma-separated string without quotes: `server01,server02`. You can make that happen with advanced variable formatting options listed below.

General syntax

Syntax: `${var_name:option}`

If any invalid formatting option is specified, then `glob` is the default/fallback option.

CSV

Formats variables with multiple values as a comma-separated string.

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:csv}'
```

```
Interpolation result:  'test1,test2'
```

Distributed - OpenTSDB

Formats variables with multiple values in custom format for OpenTSDB.

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:distributed}'
Interpolation result:  'test1,servers=test2'
```

Doublequote

Formats single- and multi-valued variables into a comma-separated string, escapes " in each value by \" and quotes each value with ".

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:doublequote}'
Interpolation result:  '"test1","test2"'
```

Glob - Graphite

Formats variables with multiple values into a glob (for Graphite queries).

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:glob}'
Interpolation result:  '{test1,test2}'
```

JSON

Formats variables with multiple values as a comma-separated string.

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:json}'
Interpolation result:  '["test1", "test2"]'
```

Lucene - Elasticsearch

Formats variables with multiple values in Lucene format for Elasticsearch.

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:lucene}'
```

```
Interpolation result:  '("test1" OR "test2")'
```

Percentencode

Formats single and multivalued variables for use in URL parameters.

```
servers = [ 'foo()bar BAZ',  'test2' ]
String to interpolate:  '${servers:percentencode}'
Interpolation result:  'foo%28%29bar%20BAZ%2Ctest2'
```

Pipe

Formats variables with multiple values into a pipe-separated string.

```
servers = [ 'test1.',  'test2' ]
String to interpolate:  '${servers:pipe}'
Interpolation result:  'test1.|test2'
```

Raw

Turns off data source-specific formatting, such as single quotes in an SQL query.

```
servers = [ 'test.1',  'test2' ]
String to interpolate:  '${var_name:raw}'
Interpolation result:  'test.1,test2'
```

Regex

Formats variables with multiple values into a regex string.

```
servers = [ 'test1.',  'test2' ]
String to interpolate:  '${servers:regex}'
Interpolation result:  '(test1\.|test2)'
```

Singlequote

Formats single- and multi-valued variables into a comma-separated string, escapes `'` in each value by `\'` and quotes each value with `'`.

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:singlequote}'
```

```
Interpolation result:  "'test1','test2'"
```

Sqlstring

Formats single- and multi-valued variables into a comma-separated string, escapes `'` in each value by `''` and quotes each value with `'`.

```
servers = [ "test'1",  "test2" ]
String to interpolate:  '${servers:sqlstring}'
Interpolation result:  "'test''1','test2'"
```

Text

Formats single- and multi-valued variables into their text representation. For a single variable, it will just return the text representation. For multi-valued variables, it will return the text representation combined with +.

```
servers = [ "test1",  "test2" ]
String to interpolate:  '${servers:text}'
Interpolation result:  "test1 + test2"
```

Query parameters

Formats single- and multi-valued variables into their query parameter representation. Example: `var-foo=value1&var-foo=value2`

```
servers = [ "test1",  "test2" ]
String to interpolate:  '${servers:queryparam}'
Interpolation result:  "var-servers=test1&var-servers=test2"
```

Assessing dashboard usage

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

For every dashboard and data source, you can access usage information.

Dashboard insights

To see dashboard usage information, select **Dashboard insights** in the top bar.

Dashboard insights show the following information.

- **Stats** – The number of daily queries and errors for the past 30 days.
- **Users & activity** – The daily view count for the last 30 days; last activities on the dashboard and recent users (with a limit of 20).

Data source insights

Data source insights provide information about how a data source has been used in the past 30 days, such as:

- Queries per day
- Errors per day
- Query load time per day (averaged in ms)

To find data source insights

1. Select **Connections** in the main navigation of your workspace.
2. Select **Data sources**.
3. Choose a data source.
4. Select the **Insights** tab.

Presence indicator

When you are signed in and look at a dashboard, you can know who is looking at the same dashboard as you are through a presence indicator, which displays avatars of users who have recently interacted with the dashboard. The default timeframe is 10 minutes. To see the user's name, hover over the user's avatar. The avatars come from Gravatar based on the user's email.

When there are more active users on a dashboard than can fit within the presence indicator, click the **+X** icon. Doing this will open dashboard insights, which contain more details about recent user activity.

Sorting dashboards by using insights data

In the search view, you can use insights data to help you find most-used, broken, and unused dashboards. You can sort dashboards by the following.

- Views
- Errors
- Views
- Created time
- Updated time

Troubleshoot dashboards

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use the following strategies to help you troubleshoot common dashboard problems.

Dashboard is slow

- Are you trying to render dozens (or hundreds or thousands) of time series on a graph? This can cause the browser to lag. Try using functions like `highestMax` (in Graphite) to reduce the number of returned series.
- Sometimes series names can be very large. This causes larger response sizes. Try using `alias` to reduce the size of the returned series names.
- Are you querying many time series or a long time range? Both of these conditions can cause Grafana or your data source to pull in a lot of data, which may slow the dashboard down. Try reducing one or both of these.
- There could be high load on your network infrastructure. If the slowness isn't consistent, this may be the problem.

Dashboard refresh rate issues

By default, Grafana queries your data source every 30 seconds. However, setting a low refresh rate on your dashboards puts unnecessary stress on the backend. In many cases, querying this frequently isn't necessary because the data source isn't sending data often enough for there to be changes every 30 seconds.

We recommend the following:

- Only enable auto-refreshing on dashboards, panels, or variables if necessary. Users can refresh their browser manually.
- If you require auto-refreshing, then set the refresh rate to a longer time period that makes sense, such as once a minute, every 10 minutes, or every hour.
- Check the time range of your dashboard. If your dashboard has a longer time range, such as a week, then you really don't need automated refreshing and you should disable it.

Handling or rendering null data is wrong or confusing

Some applications publish data intermittently; for example, they only post a metric when an event occurs. By default, Grafana graphs connect lines between the data points, but this can be deceptive.

Graphs that have the **Connect null values** option set to **Always**, will connect lines where there are missing values.

One way to fix this is to use bars instead of lines and have the **No value** option (under **Standard options**) set to `0`. In this case, the missing data will show up as areas of the graph with no data.

Searching Dashboards in Grafana version 10

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can search for dashboards by dashboard name and by panel title. When you search for dashboards, the system returns all dashboards available within the Grafana instance, even if you do not have permission to view the contents of the dashboard.

Search dashboards using dashboard name

Enter any part of the dashboard name in the search bar. The search returns results for any partial string match in real-time, as you type.

Dashboard search is:

- Real-time
- *Not* case sensitive
- Functional across stored and file based dashboards.

> **Tip**
>
> You can use your keyboard arrow keys to navigate the results and press `Enter` to open the selected dashboard.

Search dashboards using panel title

You can search for a dashboard by the title of a panel that appears in a dashboard. If a panel's title matches your search query, the dashboard appears in the search results.

Filter dashboard search results by tags

Tags are a great way to organize your dashboards, especially as the number of dashboards grow. You can add and manage tags in dashboard **Settings**.

When you select multiple tags, Grafana shows dashboards that include all selected tags.

To filter dashboard search result by a tag, complete one of the following steps:

- To filter dashboard search results by tag, choose a tag that appears in the right column of the search results.

  You can continue filtering by choosing additional tags.

- To see a list of all available tags, click the **Filter by tags** dropdown menu and select a tag.

  All tags will be shown, and when you select a tag, the dashboard search will be instantly filtered.

> **Tip**
>
> When using only a keyboard, press the `tab` key and navigate to the **Filter by tag** dropdown menu, press the down arrow key to activate the menu and locate a tag, and press `Enter` to select the tag.

Command palette

You can use the command palette to do the following:

- Search for and open dashboards and folders.
- Create dashboards and alert rules.
- Locate pages within Grafana.
- Change the theme to dark or light.

To open the command palette, enter `ctrl+k` (`cmd+k` in MacOS). You can also select the search input in the Grafana navigation bar.

> **Note**
>
> To go to the previous step, press `backspace` with the command palette empty.

Panels and visualizations in Grafana version 10

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The *panel* is the basic visualization building block in Grafana. Each panel has a query editor specific to the data source selected in the panel. The query editor allows you to build a query that returns the data you want to visualize.

There are a wide variety of styling and formatting options for each panel. Panels can be dragged, dropped, and resized to rearrange them on the dashboard.

Before you add a panel, ensure that you have configured a data source.

Additional panel types might be available by installing additional plugins to your workspace.

For details about using specific data sources, see Connect to data sources.

Topics

- Panel editor overview
- The panel inspect view
- Query and transform data
- Configure panel options
- Configure standard options
- Configure a legend
- Configure data links
- Configure value mappings
- Configure thresholds
- Configure field overrides
- Visualizations available in Grafana version 10

Panel editor overview

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

In the panel editor, you can update all the elements of a visualization, including the data source, queries, time range, and display options.

To add a panel to a new dashboard, select **+ Add visualization** in the middle of the dashboard. To add a panel to an existing dashboard, select **Add** in the dashboard header and select **Visualization** in the drop down. You can also copy and paste an existing panel from the same or a different dashboard.

Panel menu

To access the panel editor, hover over the top-right corner of any panel. Select the panel menu icon that appears and select **Edit**.

The panel menu also gives you access to the following actions:

- **View** – View the panel in full screen.
- **Edit** – Open the panel editor to edit panel and visualization options.
- **Share** – Share the panel as a link, or library panel.
- **Explore** – Open the panel in **Explore**, where you can focus on your query.
- **Inspect** – Open the **Inspect** drawer, where you can review the panel data, stats, metadata, JSON, and query.
  - **Data** – Open the **Inspect** drawer in the **Data** tab.
  - **Query** – Open the **Inspect** drawer in the **Query** tab.
  - **Panel JSON** – Open the **Inspect** drawer in the **JSON** tab.
- **Extensions** – Access other actions provided by installed applications, such as declaring an incident. This option only appears if you have app plugins installed which contribute an extension to the panel menu.
- **More** – Access other panel actions.
  - **Duplicate** – Make a copy of the panel. Duplicated panels query data separately from the original panel. If you want to use the same query results, you can use the `Dashboard` data source in the second panel.
  - **Copy** – Copy the panel to the clipboard.
  - **Create library panel** – Create a panel that can be imported into other dashboards.

  - **Create alert** – Open the alert rule configuration page in **Alerting**, where you can create a Grafana-managed alert based on the panel queries.
  - **Hide legend** – Hide the panel legend.
  - **Get help** – Send a snapshot or panel data to Grafana Labs Technical Support.
- **Remove** – Remove the panel from the dashboard.

Panel editor

This section describes the areas of the Grafana panel editor.

- Panel header – The header section lists the dashboard in which the panel appears and the following controls:
  - **Discard** – Discards changes you have made to the panel since you last saved the dashboard.
  - **Save** – Saves changes you made to the panel.
  - **Apply** – Applies changes you made and closes the panel editor, returning you to the dashboard. You will have to save the dashboard to persist the applied changes.
- Visualization preview – The visualization preview section contains the following options:
  - **Table view** – Convert any visualization to a table so you can see the data. Table views are helpful for troubleshooting. This view only contains the raw data. It does not include transformations you might have applied to the data or the formatting options available in the Table visualization.
  - **Fill** – The visualization preview fills the available space. If you change the width of the side pane or height of the bottom pane the visualization changes to fill the available space.
  - **Actual** – The visualization preview will have the exact size as the size on the dashboard. If not enough space is available, the visualization will scale down preserving the aspect ratio.
  - **Time range controls** – **Default** is either the browser local timezone or the timezone selected at a higher level.
- Data section – The data section contains tabs where you enter queries, transform your data, and create alert rules (if applicable).
  - **Query tab** – Select your data source and enter queries here. For more information, see Query and transform data. When you initially create a dashboard, you are prompted to select a data source. You can update the data source or the query in this tab.
  - **Transform tab** – Apply data transformations. For more information, see Query and transform data.

- **Alert tab** – Write alert rules. For more information, see Alerts in Grafana version 10.
- Panel display options – The display options section contains tabs where you configure almost every aspect of your data visualization. The details vary based on the visualization type selected.

Panel inspect drawer

The inspect drawer helps you understand and troubleshoot your panels. You can view the raw data for any panel, export that data to a comma-separated values (CSV) file, view query requests, and export panel and data JSON.

> **Note**
>
> Not all panel types include all tabs. For example, dashboard list panels do not have raw data to inspect, so they do not display the Stats, Data, or Query tabs.

The panel inspector consists of the following options:

- The panel inspect drawer displays as a drawer on the right side. Select the arrow in the upper right corner to expand or reduce the drawer pane.
- **Data tab** – Shows the raw data returned by the query with transformations applied. Field options such as overrides and value mappings are not applied by default.
- **Stats tab** – Shows how long your query takes and how much it returns.
- **JSON tab** – Allows you to view and copy the panel JSON, panel data JSON, and data frame structure JSON. This is useful if you are provisioning or administering Grafana.
- **Query tab** – Shows you the requests to the server sent when Grafana queries the data source.
- **Error tab** – Shows any errors returned by a query. The tab is only visible when a query returns an error.

The panel inspect view

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The panel inspect view, which you can open via the panel menu, helps you understand and troubleshoot your panels. You can inspect the raw data for any Amazon Managed Grafana panel, export that data to a comma-separated values (CSV) file, view query requests, and export panel and data JSON.

Note

Not all panel types include all tabs. For example, dashboard list panels do not have raw data to inspect, so they do not display the Stats, Data, or Query tabs.

The panel inspector consists of the following options:

1. The panel inspector displays **Inspect:** at the top of the pane. Select the arrow in the upper right corner to expand or reduce the pane.
2. **Data tab** – Shows the raw data returned by the query with transformations applied. Field options such as overrides and value mappings are not applied by default.
3. **Stats tab** – Shows how long your query takes and how much it returns.
4. **JSON tab** – Allows you to view and copy the panel JSON, panel data JSON, and data frame structure JSON. This is useful if you are provisioning or administering Grafana.
5. **Query tab** – Shows you the requests sent to the server when Grafana queries the data source.
6. **Error tab** – Shows any errors. Only visible when a query returns an error.

Downloading raw query results

Amazon Managed Grafana generates a CSV file that contains your data, including any transformations to that data. You can choose to view the data before or after the panel applies field options or field option overrides.

To download raw query results

1. Edit the panel that contains the query data you want to download.
2. In the query editor, select **Query Inspector**.

3. Select **Data**.

   If your panel contains multiple queries or queries multiple nodes, then you have additional options.

   - **Select result** – Choose which result set data you want to view.
   - **Transform data**
   - **Join by time** – View raw data from all your queries at once, one result set per column. Select a column heading to reorder the data.
4. To see data before the system applies field overrides, select the **Formatted data** toggle.
5. To download a CSV file specifically formatted for Excel, select the **Download for Excel** toggle.
6. Select **Download CSV**.

Inspecting query performance

The **Stats** tab displays statistics that tell you how long your query takes, how many queries you send, and the number of rows returned. This information can help you troubleshoot your queries, especially if any of the numbers are unexpectedly high or low.

To inspect query performance

1. Edit the panel that contains the query with performance you want to inspect.
2. In the query editor, select **Query Inspector**.
3. Select **Stats**.

Statistics are displayed in read-only format.

Inspecting query request and response

You can inspect query request and response data when you want to troubleshoot a query that returns unexpected results, or fails to return expected results.

1. Edit the panel that contains the query you want to export.
2. In the query editor, select **Query Inspector**.
3. Select **Refresh**.

   The panel populates with response data.

4. Make adjustments, as necessary and re-run the query.
5. To download the query request and response data, click the **Copy to clipboard** icon and paste the results into another application.

Query and transform data

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Amazon Managed Grafana supports many types of data sources. Data source *queries* return data that Grafana can *transform* and visualize. Each data source uses its own query language, and data source plugins each implement a query-building user interface called a query editor.

Topics

- About queries
- Navigate the query tab
- Adding a query
- Manage queries
- Query options
- Write expression queries
- Share query results with another panel
- Transform data
- Troubleshoot queries
- Calculation types

About queries

Grafana panels communicate with data sources via queries, which retrieve data for the visualization. A query is a question written in the query language used by the data source.

You can configure query frequency and data collection limits in the panel's data source options. Grafana supports up to 26 queries per panel.

You can find more information about each data source's query language in the Data sources section.

Query editors

Each data source's *query editor* provides a customized user interface that helps you write queries that take advantage of its unique capabilities.

Because of the differences between query languages, each data source query editor looks and functions differently. Depending on your data source, the query editor might provide auto-completion features, metric names, variable suggestions, or a visual query-building interface.

For details on a specific data source's unique query editor features, including information about queries and syntax, see the data source documentation:

- For data sources included with Amazon Managed Grafana, see Built-in data sources.
- For data sources included with Grafana Enterprise, see Connect to Enterprise data sources.
- For other data source plugins that you install via the Find plugins with the plugin catalog, the documentation is linked within the listing in the plugin catalog.

Query syntax

Data sources use different query languages to request data. For details on a specific data source's unique query language, refer to its documentation.

PostgreSQL example:

```
SELECT hostname FROM host WHERE region IN($region)
```

PromQL example:

```
query_result(max_over_time(<metric>[${__range_s}s]) != <state>)
```

Special data sources

Grafana also includes three special data sources: **Grafana**, **Mixed**, and **Dashboard**. For more information, see Connect to data sources.

Navigate the query tab

A panel's **Query** tab consists of the following elements:

- **Data source selector** – Selects the data source to query.
- **Query options** – Sets maximum data retrieval parameters and query run time intervals.
- **Query inspector button** – Opens the query inspector panel, where you can view and optimize your query.
- **Query editor list** – Lists the queries you've written.
- **Expressions** – Uses the expression builder to create alert expressions. For more information about expressions, see Write expression queries.

Adding a query

A query returns data that Grafana visualizes in dashboard panels. When you create a panel, Grafana automatically selects the default data source.

To add a query

1. Edit the panel to which you're adding a query.
2. Choose the **Query** tab.
3. Choose the **Data source** dropdown menu and select a data source.
4. Choose **Query options** to configure the maximum number of data points you need. For more information about query options, see Query options.
5. Write the query using the query editor.
6. Choose **Apply**.

Grafana queries the data source and visualizes the data.

Manage queries

Grafana organizes queries in collapsible query rows. Each query row contains a query editor and is identified with a letter (A, B, C, and so on).

To manage your queries, you can copy queries, hide queries, remove queries, reorder queries, and toggle help for the query editor.

Query options

Choose **Query options** next to the data source selector to see settings for the selected data source. Changes you make here affect only queries made in this panel.

Grafana sets defaults that are shown in dark gray text. Changes are displayed in white text. To return a field to the default setting, delete the white text from the field.

Panel data source query options include:

- **Max data points** – If the data source supports it, this sets the maximum number of data points for each series returned. If the query returns more data points than the max data points setting, then the data source reduces the number of points returned by aggregating them together by average, max, or another function.

  You can limit the number of points to improve query performance or smooth the visualized line. The default value is the width (or number of pixels) of the graph, because you can only visualize as many data points as the graph panel has room to display.

  With streaming data, Grafana uses the max data points value for the rolling buffer. Streaming is a continuous flow of data, and buffering divides the stream into chunks.
- **Min interval** – Sets a minimum limit for the automatically calculated interval, which is typically the minimum scrape interval. If a data point is saved every 15 seconds, you don't benefit from having an interval lower than that. You can also set this to a higher minimum than the scrape interval to retrieve queries that are more coarse-grained and well-functioning.
- **Interval** – Sets a time span that you can use when aggregating or grouping data points by time.

  Grafana automatically calculates an appropriate interval that you can use as a variable in templated queries. The variable is measured in either seconds (`$__interval`) or milliseconds (`$__interval_ms`).

  Intervals are typically used in aggregation functions like sum or average.
  For example, this is a Prometheus query that uses the interval variable:
  `rate(http_requests_total[$__interval])`.

  This automatic interval is calculated based on the width of the graph. As the user zooms out on a visualization, the interval grows, resulting in a more coarse-grained aggregation. Likewise, if the user zooms in, the interval decreases, resulting in a more fine-grained aggregation.

  For more information, see Global variables.

- **Relative time** – Overrides the relative time range for individual panels, which causes them to be different than what is selected in the dashboard time picker in the top-right corner of the dashboard. You can use this to show metrics from different time periods or days on the same dashboard.

> **Note**
>
> Panel time overrides have no effect when the dashboard's time range is absolute.

Example	Relative time field
Last 5 minutes	`now-5m`
The day so far	`now/d`
Last 5 days	`now-5d/d`
This week so far	`now/w`
Last 2 years	`now-2y/y`

- **Time shift** – Overrides the time range for individual panels by shifting its start and end relative to the time picker. For example, you can shift the time range for the panel to be two hours earlier than the dashboard time picker.

> **Note**
>
> Panel time overrides have no effect when the dashboard's time range is absolute.

Example	Time shift field
Last entire week	`1w/w`
Two entire weeks ago	`2w/w`
Last entire month	`1M/M`

Example	Time shift field
This entire year	1d/y
Last entire year	1y/y

- **Cache timeout** – *(Visible only if available in the data source)* Overrides the default cache timeout if your time series store has a query cache. Specify this value as a numeric value in seconds.

Write expression queries

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Server-side expressions enable you to manipulate data returned from queries with math and other operations. Expressions create new data and do not manipulate the data returned by data sources.

About expressions

Server-side expressions allow you to manipulate data returned from queries with math and other operations. Expressions create new data and do not manipulate the data returned by data sources, aside from some minor data restructuring to make the data acceptable input for expressions.

Using expressions

Expressions are most commonly used by Grafana alerting. The processing is done server-side, so expressions can operate without a browser session. However, expressions can also be used with backend data sources and visualization.

Note

Expressions do not work with legacy dashboard alerts.

Expressions are meant to augment data sources by enabling queries from different data sources to be combined or by providing operations unavailable in a data source.

> **Note**
>
> When possible, you should do data processing inside the data source. Copying data from storage to the Grafana server for processing is inefficient, so expressions are targeted at lightweight data processing.

Expressions work with data source queries that return time series or numeric data. They also operate on multiple-dimensional data. For example, a query that returns multiple series, where each series is identified by labels or tags.

An individual expression takes one or more queries or other expressions as input and adds data to the result. Each individual expression or query is represented by a variable that is a named identifier known as its RefID (e.g., the default letter `A` or `B`).

To reference the output of an individual expression or a data source query in another expression, this identifier is used as a variable.

Types of expressions

Expressions work with two types of data.

- A collection of time series.
- A collection of numbers, where each number is an item.

Each collection is returned from a single data source query or expression and represented by the RefID. Each collection is a set, where each item in the set is uniquely identified by its dimensions which are stored as labels or key-value pairs.

Data source queries

Server-side expressions only support data source queries for backend data sources. The data is generally assumed to be labeled time series data.

Data source queries, when used with expressions, are run by the expression engine. When it does this, it restructures data to be either one time series or one number per data frame. So for example

if using a data source that returns multiple series on one frame in the table view, you might notice it looks different when run with expressions.

Currently, the only non-time series format (number) is supported when using data frames are you have a table response that returns a data frame with no time, string columns, and one number column:

The following example table produces a number that works with expressions. The string columns become labels and the number column the corresponding value. For example `{"Loc": "MIA", "Host": "A"}` with a value of 1.

Loc	Host	Avg_CPU
MIA	A	1
NYC	B	2

Operations

You can use the following operations in expressions: math, reduce, and resample.

Math

Math is for free-form math formulas on time series or number data. Math operations take numbers and time series as input and changes them to different numbers and time series.

Data from other queries or expressions are referenced with the RefID prefixed with a dollar sign, for example `$A`. If the variable has spaces in the name, then you can use a brace syntax like `${my variable}`.

Numeric constants can be in decimal (`2.24`), octal (with a leading zero like `072`), or hex (with a leading 0x like `0x2A`). Exponentials and signs are also supported (for example, `-0.8e-2`).

Operators

The arithmetic (+, binary and unary -, *, /, %, exponent **), relational (<, >, ==, !=, >=, <=), and logical (&&, ||, and unary !) operators are supported.

How the operation behaves with data depends on if it is a number or time series data.

With binary operations, such as `$A + $B` or `$A || $B`, the operator is applied in the following ways depending on the type of data:

- If both `$A` and `$B` are a number, then the operation is performed between the two numbers.
- If one variable is a number, and the other variable is a time series, then the operation between the value of each point in the time series and the number is performed.
- If both `$A` and `$B` are time series data, then the operation between each value in the two series is performed for each time stamp that exists in both `$A` and `$B`. The `Resample` operation can be used to line up time stamps.

Summary:

- Number <Operation> number = number
- Number <Operation> series = series
- Series <Operation> series = series

Because expressions work with multiple series or numbers represented by a single variable, binary operations also perform a union (join) between the two variables. This is done based on the identifying labels associated with each individual series or number.

So if you have numbers with labels like `{host=web01}` in `$A` and another number in `$B` with the same labels then the operation is performed between those two items within each variable, and the result will share the same labels. The rules for the behavior of this union are as follows:

- An item with no labels will join to anything.
- If both `$A` and `$B` each contain only one item (one series, or one number), they will join.
- If labels are exact match they will join.
- If labels are a subset of the other, for example an item in `$A` is labeled `{host=A,dc=MIA}` and an item in `$B` is labeled `{host=A}` they will join.
- If within a variable such as `$A` there are different tag keys for each item, the join behavior is undefined.

The relational and logical operators return 0 for false 1 for true.

Math Functions

While most functions exist in the own expression operations, the math operation does have some functions that similar to math operators or symbols. When functions can take either numbers or series, than the same type as the argument will be returned. When it is a series, the operation of performed for the value of each point in the series.

abs

abs returns the absolute value of its argument which can be a number or a series. For example `abs(-1)` or `abs($A)`.

is_inf

is_inf takes a number or a series and returns `1` for `Inf` values (negative or positive) and `0` for other values. For example `is_inf($A)`.

> **Note**
>
> If you need to specifically check for negative infinity for example, you can do a comparison like `$A == infn()`.

is_nan

is_nan takes a number or a series and returns `1` for `NaN` values and `0` for other values. For example `is_nan($A)`. This function is required for this check because `NaN` is not equal to `NaN`.

is_null

is_null takes a number or a series and returns `1` for `null` values and `0` for other values. For example `is_null($A)`.

is_number

is_number takes a number or a series and returns `1` for all real number values and `0` for other values (which are `null`, `Inf+`, `Inf-`, and `NaN`). For example `is_number($A)`.

log

Log returns the natural logarithm of its argument which can be a number or a series. If the value is less than 0, `NaN` is returned. For example `log(-1)` or `log($A)`.

inf, infn, nan, and null

The inf, infn, nan, and null functions all return a single value of the name. They primarily exist for testing. Example: `null()`.

round

Round returns a rounded integer value. For example, `round(3.123)` or `round($A)`.

ceil

Ceil rounds the number up to the nearest integer value. For example, `ceil(3.123)` returns 4.

floor

Floor rounds the number down to the nearest integer value. For example, `floor(3.123)` returns 3.

Reduce

Reduce takes one or more time series returned from a query or an expression and turns each series into a single number. The labels of the time series are kept as labels on each outputted reduced number.

Fields:

- **Function** – The reduction function to use
- **Input** – The variable (refID (such as A)) to resample
- **Mode** – Allows control behavior of reduction function when a series contains non-numerical values (null, NaN, +-Inf)

Reduction Functions

Count

Count returns the number of points in each series.

Mean

Mean returns the total of all values in each series divided by the number of points in that series. In `strict` mode if any values in the series are null or nan, or if the series is empty, NaN is returned.

Min and Max

Min and Max return the smallest or largest value in the series respectively. In `strict` mode if any values in the series are null or nan, or if the series is empty, NaN is returned.

Sum

Sum returns the total of all values in the series. If series is of zero length, the sum will be 0. In `strict` mode if there are any NaN or Null values in the series, NaN is returned.

Last

Last returns the last number in the series. If the series has no values then returns NaN.

Reduction Modes

Strict

In Strict mode the input series is processed as is. If any values in the series are non-numeric (null, NaN or +-Inf), NaN is returned.

Drop Non-Numeric

In this mode all non-numeric values (null, NaN or +-Inf) in the input series are filtered out before running the reduction function.

Replace Non-Numeric

In this mode all non-numeric values are replaced by a pre-defined value.

Resample

Resample changes the time stamps in each time series to have a consistent time interval. The main use case is so you can resample time series that do not share the same timestamps so math can be performed between them. This can be done by resample each of the two series, and then in a Math operation referencing the resampled variables.

Fields:

- **Input** – The variable of time series data (refID (such as `A`)) to resample
- **Resample** to – The duration of time to resample to, for example `10s`. Units can be `s` for seconds, `m` for minutes, `h` for hours, `d` for days, `w` for weeks, and `y` for years.
- **Downsample** – The reduction function to use when there are more than one data point per window sample. See the reduction operation for behavior details.
- **Upsample** – The method to use to fill a window sample that has no data points.
  - **pad** fills with the last knowm value
  - **backfill** with next known value

- **fillna** to fill empty sample windows with NaNs

Write an expression

If your data source supports them, then Grafana displays the **Expression** button and shows any existing expressions in the query editor list.

To write an expression

1. Open the panel.
2. Below the query, choose **Expression**.
3. In the **Operation** field, select the type of expression you want to write.
4. Write the expression.
5. Choose **Apply**.

Special cases

When any queried data source returns no series or numbers, the expression engine returns `NoData`. For example, if a request contains two data source queries that are merged by an expression, if `NoData` is returned by at least one of the data source queries, then the returned result for the entire query is `NoData`. For more information about how Grafana Alerting processes `NoData` results, see Configure Grafana managed alert rules.

In the case of using an expression on multiple queries, the expression engine requires that all of the queries return an identical timestamp. For example, if using math to combine the result of multiple SQL queries which each use `SELECT NOW() AS "time"`, the expression will only work if all queries evaluate `NOW()` to an identical timestamp, which does not always happen. To resolve this, you can replace `NOW()` with an arbitrary time, such as `SELECT 1 AS "time"`, or any other valid UNIX timestamp.

Share query results with another panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana lets you use the query result from one panel for any other panel in the dashboard. Sharing query results across panels reduces the number of queries made to your data source, which can improve the performance of your dashboard.

The *Dashboard* data source lets you select a panel in your dashboard that contains the queries you want to share the results for. Instead of sending a separate query for each panel, Grafana sends one query and other panels use the query results to construct visualizations.

This strategy can drastically reduce the number of queries being made when you for example have several panels visualizing the same data.

To share query results

1. Create a dashboard.
2. Change the title to `Source panel`. You'll use this panel as a source for the other panels.
3. Define the query or queries that you want to share.

   If you don't have a data source available, use the **Grafana** data source, which returns a random time series that you can use for testing.
4. Add a second panel and select the **Dashboard** data source in the query editor.
5. In the **Use results from panel list**, select the first panel you created.

All queries defined in the source panel are now available to the new panel. Queries made in the source panel can be shared with multiple panels.

You can click on any of the queries to go to the panel where they are defined.

Transform data

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Transformations are a powerful way to manipulate data returned by a query before the system applies a visualization. Using transformations, you can:

- Rename fields
- Join time series data
- Perform mathematical operations across queries
- Use the output of one transformation as the input to another transformation

For users that rely on multiple views of the same dataset, transformations offer an efficient method of creating and maintaining numerous dashboards.

You can also use the output of one transformation as the input to another transformation, which results in a performance gain.

Note

Sometimes the system cannot graph transformed data. When that happens, click the **Table view** toggle above the visualization to switch to a table view of the data. This can help you understand the final result of your transformations.

Transformation types

Grafana provides a number of ways that you can transform data. There is a complete list of transformation functions below.

Order of transformations

When there are multiple transformations, Grafana applies them in the order they are listed. Each transformation creates a result set that then passes on to the next transformation in the processing pipeline.

The order in which Grafana applies transformations directly impacts the results. For example, if you use a Reduce transformation to condense all the results of one column into a single value, then you can only apply transformations to that single value.

Add a transformation function to data

The following steps guide you in adding a transformation to data. This documentation does not include steps for each type of transformation.

To add a transformation to a panel

1. Navigate to the panel where you want to add one or more transformations.
2. Hover over any part of the panel to display the actions menu on the top right corner.
3. From the actions menu choose **Edit**.
4. Select the **Transform** tab.
5. Select a transformation. A transformation row appears where you configure the transformation options.
6. To apply another transformation, Choose **Add transformation**. This transformation acts on the result set returned by the previous transformation.

Debug a transformation

To see the input and the output result sets of the transformation, choose the debug (bug) icon on the right side of the transformation row. This will display the input data, and the result of the transformation as output.

The input and output results sets can help you debug a transformation.

Disable a transformation

You can disable or hide a transformation by choosing the show (eye) icon on the top right of the transformation row. This disables the applied actions of that specific transformation and can help to identify issues when you change several transformations one after another.

Filter a transformation

If your transformation uses more than one query, you can filter these, and apply the selected transformation to only one of the queries. To do this, choose the filter icon on the top right of the transformation row. This opens a dropdown with a list of queries used on the panel. From here, you can select the query you want to transform.

You can also filter by annotations (which includes exemplars) to apply transformations to them. When you do so, the list of fields changes to reflect those in the annotation or exemplar tooltip.

The filter icon is always displayed if your panel has more than one query or source of data (that is panel or annotation data), but it may not work if previous transformations for merging the queries' outputs are applied. This is because one transformation takes the output of the previous one.

Delete a transformation

We recommend that you remove transformations that you don't need. When you delete a transformation, you remove the data from the visualization.

Prerequisites:

Identify all dashboards that rely on the transformation and inform impacted dashboard users.

To delete a transformation

1. Open a panel for editing.
2. Select the **Transform** tab.
3. Choose the trash icon next to the transformation you want to delete.

Transformation functions

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can perform the following transformations on your data.

Add field from calculation

Use this transformation to add a new field calculated from two other fields. Each transformation allows you to add one new field.

- **Mode** - Select a mode:
  - **Reduce row** – Apply selected calculation on each row of selected fields independently.

- **Binary operation** – Apply basic binary operations (for example, sum or multiply) on values in a single row from two selected fields.
- **Unary operation** – Apply basic unary operations on values in a single row from a selected field. The available operations are:
  - **Absolute value (abs)** – Returns the absolute value of a given expression. It represents the distance from zero as a positive number.
  - **Natural exponential (exp)** – Returns *e* raised to the power of a given expression.
  - **Natural logarithm (ln)** – Returns the natural logarithm of a given expression.
  - **Floor (floor)** – Returns the largest integer less than or equal to a given expression.
  - **Ceiling (ceil)** – returns the smallest integer greater than or equal to a given expression.
- **Cumulative functions** – Apply functions on teh current row and all preceding rows.
  - **Total** – Calculates the cumulative total up to and including the current row.
  - **Mean** – Calculates the mean up to and including the current row.
- **Window functions** – Apply window functions. The window can either be *trailing* or *centered*. With a trailing window the current row will be the last row in the window. With a centered window, the window will be centered on the current row. For even windows sizes, the window will be centered between the current row and the previous row.
  - **Mean** – Calculates the moving mean or running average.
  - – Calculates the moving standard deviation.
  - **Variance** – Calculate the moving variance.
- **Row index** – Insert a field with the row index.

- **Field name** – Select the names of fields you want to use in the calculation for the new field.
- **Calculation** – If you select **Reduce row** mode, then the **Calculation** field appears. Select the field to see a list of calculation choices you can use to create the new field. For information about available calculations, refer to Calculation types.
- **Operation** – If you select **Binary operation** or **Unary operation** mode, then the **Operation** fields appear. These fields allow you to do basic math operations on values in a single row from two selected fields. You can also use numerical values for binary operations.
- **As percentile** – If you select **Row index** mode, then the **As percentile** switch appears. This switch allows you to transform the row index as a percentage of the total number of rows.
- **Alias** – (Optional) Enter the name of your new field. If you leave this blank, then the field will be named to match the calculation.

- **Replace all fields** – (Optional) Select this option if you want to hide all other fields and display only your calculated field in the visualization.

> **Note**
>
> **Cumulative functions** and **Window functions** are current in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

Concatenate fields

Concatenate fields

Use this transformation to combines all fields from all frames into one result.

For example, if you have separate queries retrieving temperature and uptime data (query A) and air quality index and error information (query b), applying the concatenate transformation yields a consolidated data frame with all relevant information in one view.

Consider these two following.

Query A:

Temp	Uptime
15.4	1230233

Query B:

AQI	Errors
3.2	5

After you concatenate the fields, the data frame would be:

Temp	Uptime	AQI	Errors
15.4	1230233	3.2	5

This transformation simplifies the process of merging data from different sources, providing a comprehensive view for analysis and visualization.

Config from query results

Config from query results

Use this transformation to select one query and extract standard options like **Min**, **Max**, **Unit** and **Thresholds** and apply them to other query results. This enables dynamic visualization configuration, based on the data returned by a specific query.

Options

- **Config query** – Select the query that returns the data you want to use as configuration.
- **Apply to** – Select the fields or series to which the configuration should be applied.
- **Apply to options** – Specify a field type or use a field name regex, depending on your selection in **Apply to**.

Field mapping table

Below the configuration options, you'll find the field mapping table. This table lists all fields found in the data returned by the config query, along with the **Use as** and **Select** options. It provides control over mapping fields to config properties, and for multiple rows, it allows you to choose which value to select.

The following example shows an input query and a query used as field configuration.

Input query

Time	Value
1626178119127	10

Time	Value
1626178119129	30

Config query

Time	Value
1626178119127	100
1626178119129	100

Output query (same as input, but now with config on the value field)

Time	Value (config: Max=100)
1626178119127	10
1626178119129	30

Each field now has a maximum configuration option set. Options such as **Min**, **Max**, **Unit**, and **Thresholds** are part of teh field configuration. If set, they are used by the visualization instead of any options manually configured in the panel editor options pane.

Value mappings

You can also transform a query result into value mappings. With this option, every row in the configuration query result defines a single value mapping row. See the following example.

Config query result

Value	Text	Color
L	Low	blue
M	Medium	green

Value	Text	Color
H	High	red

In the field mapping, specify:

Field	Use as	Select
Value	Value mappings / Value	All values
Text	Value mappings / Text	All values
Color	Value mappings / Color	All values

Grafana builds value mappings from your query result and applies them to the real data query results. You should see values being mapped and colored according to the config query results.

Convert field type

Use this transformation to modify the field type of the specified field.

This transformation has the following options:

- **Field** – Select from available fields.
- **as** – Select the FieldType to convert to.
  - **Numeric** – attempts to make the values numbers.
  - **String** – will make the values strings.
  - **Time** – attempts to parse the values as time.
    - Will show an option to specify a DateFormat as input by a string, like `yyyy-mm-dd` or `DD MM YYYY hh:mm:ss`.
  - **Boolean** – will make the values Boolean.
  - **Enum** – will make the values enums.
    - Will show a table to manage the enums.
  - **Other** – attempts to parse the values as json.

For example consider the following query that could be modified by selecting the time field, as **Time**, and Date Format as YYYY.

Time	Mark	Value
2017-07-01	above	25
2018-08-02	below	22
2019-09-02	below	29
2020-10-04	above	22

The result:

Time	Mark	Value
2017-01-01 00:00:00	above	25
2018-01-01 00:00:00	below	22
2019-01-01 00:00:00	below	29
2020-01-01 00:00:00	above	22

This transformation allows you to flexibly adapt your data types, ensuring compatibility and consistency in your visualizations.

Extract Fields

Use this transformation to select a source of data and extract content from it in different formats. This transformation has the following fields:

- **Source** – Select the field for the source of data.
- **Format** – Choose one of the following:
  - **JSON** – Parse JSON content from the source.
  - **Key+value pairs** – Parse content in the format a=b or c : d from the source.
  - **Auto** – Discover fields automatically.

- **Replace All Fields** – (Optional) Select this option to hide all other fields and display only your calculated field in the visualization.
- **Keep Time** - (Optional) Available only if **Replace All Fields** is true. Keeps the time field in the output.

Consider the following dataset:

Dataset Example

Timestamp	json_data
1636678740000000000	{"value": 1}
1636678680000000000	{"value": 5}
1636678620000000000	{"value": 12}

You could prepare the data to be used by a Time series panel with this configuration:

- Source: json_data
- Format: JSON
  - Field: value
  - Alias: my_value
- Replace all fields: true
- Keep time: true

This will generate the following output:

Transformed Data

Timestamp	my_value
1636678740000000000	1
1636678680000000000	5

Timestamp	my_value
1636678620000000000	12

With this transformation, you can extract and format data in various ways. You can customize the extraction format based on your specific data needs.

Lookup fields from resource

Use this transformation to enrich a field value by looking up additional fields from an external source.

This transformation has the following fields:

- **Field** – Select a text field from your dataset.
- **Lookup** – Choose from **Countries**, **USA States**, and **Airports**.

> **Note**
>
> This transformation only supports spatial data.

For example, if you have this data:

Dataset Example

Location	Values
AL	0
AK	10
Arizona	5
Arkansas	1
Somewhere	5

With this configuration:

- Field: location
- Lookup: USA States

You'll get the following output:

Transformed Data

Location	ID	Name	Lng	Lat	Values
AL	AL	Alabama	-80.891064	12.448457	0
AK	AK	Arkansas	-100.891064	24.448457	10
Arizona					5
Arkansas					1
Somewhere					5

This transformation lets you augment your data by fetching additional information from external sources, providing a more comprehensive dataset for analysis and visualization.

Filter data by query refId

Use this transformation to hide one or more queries in panels that have multiple queries.

Grafana displays the query identification letters in dark gray text. Choose a query identifier to toggle filtering. If the query letter is white, then the results are displayed. If the query letter is dark, then the results are hidden.

> **Note**
>
> This transformation is not available for Graphite because this data source does not support correlating returned data with queries.

Filter data by values

Use this transformation to selectively filter data points directly within your visualization. This transformation provides options to include or exclude data based on one or more conditions applied to a selected field.

This transformation is very useful if your data source does not natively filter by values. You might also use this to narrow values to display if you are using a shared query.

The available conditions for all fields are:

- **Regex** – Match a regex expression.
- **Is Null** – Match if the value is null.
- **Is Not Null** – Match if the value is not null.
- **Equal** – Match if the value is equal to the specified value.
- **Different** – Match if the value is different than the specified value.

Additional available conditions for number fields are:

- **Greater** – Match if the value is greater than the specified value.
- **Lower** – Match if the value is lower than the specified value.
- **Greater or equal** – Match if the value is greater or equal.
- **Lower or equal** – Match if the value is lower or equal.
- **Range** – Match a range between a specified minimum and maximum, min and max included.

Consider the following dataset:

Time	Temperature	Altitude
2020-07-07 11:34:23	32	101
2020-07-07 11:34:22	28	125
2020-07-07 11:34:21	26	110
2020-07-07 11:34:20	23	98

Time	Temperature	Altitude
2020-07-07 10:32:24	31	95
2020-07-07 10:31:22	20	85
2020-07-07 09:30:57	19	101

If you **Include** the data points that have a temperature below 30°C, the configuration will look as follows:

- Filter Type: 'Include'
- Condition: Rows where 'Temperature' matches 'Lower Than' '30'

And you will get the following result, where only the temperatures below 30°C are included:

Transformed Data

Time	Temperature	Altitude
2020-07-07 11:34:22	28	125
2020-07-07 11:34:21	26	110
2020-07-07 11:34:20	23	98
2020-07-07 10:31:22	20	85
2020-07-07 09:30:57	19	101

You can add more than one condition to the filter. For example, you might want to include the data only if the altitude is greater than 100. To do so, add that condition to the following configuration:

- Filter type: 'Include' rows that 'Match All' conditions
- Condition 1: Rows where 'Temperature' matches 'Lower' than '30'
- Condition 2: Rows where 'Altitude' matches 'Greater' than '100'

When you have more than one condition, you can choose if you want the action (include/exclude) to be applied on rows that **Match all** conditions or **Match any** of the conditions you added.

In the example above, we chose **Match all** because we wanted to include the rows that have a temperature lower than 30°C *AND* an altitude higher than 100. If we wanted to include the rows that have a temperature lower than 30°C *OR* an altitude higher than 100 instead, then we would select **Match any**. This would include the first row in the original data, which has a temperature of 32°C (does not match the first condition) but an altitude of 101 (which matches the second condition), so it is included.

Conditions that are invalid or incompletely configured are ignored.

This versatile data filtering transformation lets you to selectively include or exclude data points based on specific conditions. Customize the criteria to tailor your data presentation to meet your unique analytical needs.

Filter fields by name

Use this transformation to remove parts of your query results. There are three ways to filter field names:

- Enter a regular expression.
- Manually select included fields.
- Use a dashboard variable.

Use a regular expression

When you filter using a regular expression, field names that match the regular expression are included. For example, using the regular expression `'prod.*'` would return only the fields that start with `prod`

The regular expression can include an interpolated dashboard variable using the `${variableName}` syntax.

Manually select included fields

Select or deselect field names to remove them from the result. If a regular expression is also included, fields that are matched by the expression are included, even if they're unchecked.

Use a dashboard variable

Select **From variable** to let you select a dashboard variable that's used to include fields. By setting up a dashboard variable with multiple choices, the same fields can be displayed across multiple visualizations.

This transformation provides flexibility in tailoring your query results to focus on the specific fields you need for effective analysis and visualization.

Format string

Use this transformation to customize the output of a string field. This transformation has the following fields:

- **Upper case** – Formats the entire string in uppercase characters.
- **Lower case** – Formats the entire string in lowercase characters.
- **Sentence case** – Formats the first character of the string in uppercase.
- **Title case** – Formats the first character of each word in the string in uppercase.
- **Pascal case** – Formats the first character of each word in the string in uppercase and doesn't include spaces between words.
- **Camel case** – Formats the first character of each word in the string in uppercase, except the first word, and doesn't include spaces between words.
- **Snake case** – Formats all characters in the string in lowercase and uses underscores instead of spaces between words.
- **Kebab case** – Formats all characters in the string in lowercase and uses dashes instead of spaces between words.
- **Trim** – Removes all leading and trailing spaces from the string.
- **Substring** – Returns a substring of the string, using the specified start and end positions.

This transformation provides a convenient way to standardize and tailor the presentation of string data for better visualization and analysis.

> **Note**
>
> This transformation is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

Format time

Use this transformation to customize the output of a time field. Output can be formatted using Moment.js format strings. For example, if you want to display only the year of a time field, the format string 'YYYY' can be used to show the calendar year (for example, 1999 or 2012).

Before Transformation:

Timestamp	Event
1636678740000000000	System Start
1636678680000000000	User Login
1636678620000000000	Data Updated

After applying 'YYYY-MM-DD HH:mm:ss':

Timestamp	Event
2021-11-12 14:25:40	System Start
2021-11-12 14:24:40	User Login
2021-11-12 14:23:40	Data Updated

This transformation lets you tailor the time representation in your visualizations, providing flexibility and precision in displaying temporal data.

> **Note**
>
> This transformation is available in workspaces compatbile with Grafana v10 as an alpha feature.

Group by

This transformation groups the data by a specified field (column) value and processes calculations on each group. Select to see a list of calculation choices.

Here's an example of original data.

Time	Server ID	CPU Temperature	Server Status
7/7/2020 11:34:20 AM	server 1	80	Shutdown
7/7/2020 11:34:20 AM	server 3	62	OK
7/7/2020 10:32:20 AM	server 2	90	Overload
7/7/2020 10:31:22 AM	server 3	55	OK
7/7/2020 9:30:57 AM	server 3	62	Rebooting
7/7/2020 9:30:05 AM	server 2	88	OK
7/7/2020 9:28:06 AM	server 1	80	OK
7/7/2020 9:25:05 AM	server 2	88	OK
7/7/2020 9:23:07 AM	server 1	86	OK

This transformation goes in two steps. First you specify one or multiple fields to group the data by. This will group all the same values of those fields together, as if you sorted them. For instance if we group by the Server ID field, then it would group the data this way:

Time	Server ID	CPU Temperature	Server Status
7/7/2020 11:34:20 AM	server 1	80	Shutdown
7/7/2020 9:28:06 AM	server 1	80	OK
7/7/2020 9:23:07 AM	server 1	86	OK

Time	Server ID	CPU Temperature	Server Status
7/7/2020 10:32:20 AM	server 2	90	Overload
7/7/2020 9:30:05 AM	server 2	88	OK
7/7/2020 9:25:05 AM	server 2	88	OK
7/7/2020 11:34:20 AM	server 3	62	OK
7/7/2020 10:31:22 AM	server 3	55	OK
7/7/2020 9:30:57 AM	server 3	62	Rebooting

All rows with the same value of Server ID are grouped together.

After choosing which field you want to group your data by, you can add various calculations on the other fields, and apply the calculation to each group of rows. For instance, we could want to calculate the average CPU temperature for each of those servers. So we can add the *mean* calculation applied on the CPU Temperature field to get the following:

Server ID	CPU Temperature (mean)
server 1	82
server 2	88.6
server 3	59.6

And we can add more than one calculation. For instance:

- For field Time, we can calculate the *Last* value, to know when the last data point was received for each server
- For field Server Status, we can calculate the *Last* value to know what is the last state value for each server

- For field Temperature, we can also calculate the *Last* value to know what is the latest monitored temperature for each server

We would then get:

Server ID	CPU Temperature (mean)	CPU Temperature (last)	Time (last)	Server Status (last)
server 1	82	80	7/7/2020 11:34:20 AM	Shutdown
server 2	88.6	90	7/7/2020 10:32:20 AM	Overload
server 3	59.6	62	7/7/2020 11:34:20 AM	OK

This transformation enables you to extract key information from your time series and display it in a convenient way.

Grouping to matrix

Use this transformation to combine three fields—which are used as input for the **Column**, **Row**, and **Cell value** fields from the query output—and generate a matrix. The matrix is calculated as follows:

Original data

Server ID	CPU Temperature	Server Status
server 1	82	OK
server 2	88.6	OK
server 3	59.6	Shutdown

We can generate a matrix using the values of `Server Status` as column names, the `Server ID` values as row names, and the `CPU Temperature` as content of each cell. The content of each cell

will appear for the existing column (`Server Status`) and row combination (`Server ID`). For the rest of the cells, you can select which value to display between: **Null**, **True**, **False**, or **Empty**.

Output

Server IDServer Status	OK	Shutdown
server 1	82	
server 2	88.6	
server 3		59.6

Use this transformation to construct a matrix by specifying fields from your query results. The matrix output reflects the relationships between the unique values in these fields. This helps you present complex relationships in a clear and structured matrix format.

Group to nested table

Use this transformation to group the data by a specified field (column) value and process calculation on each group. Records are generated that share the same grouped field value, to be displayed in a nested table.

To calculate a statistic for a field, select the box next to the field and choose the **Calculate** option. This will add another selection box with statistics to be selected.

The following table shows sample data.

Time	Server ID	CPU Temperature	Server Status
7/7/2020 11:34:20 AM	server 1	80	Shutdown
7/7/2020 11:34:20 AM	server 3	62	OK
7/7/2020 10:32:20 AM	server 2	90	Overload

Time	Server ID	CPU Temperature	Server Status
7/7/2020 10:31:22 AM	server 3	55	OK
7/7/2020 9:30:57 AM	server 3	62	Rebooting
7/7/2020 9:30:05 AM	server 2	88	OK
7/7/2020 9:28:06 AM	server 1	80	OK
7/7/2020 9:25:05 AM	server 2	88	OK
7/7/2020 9:23:07 AM	server 1	86	OK

This transformation has two steps. First, specify one or more fields by which to group the data. This groups all the same values of those fields together, as if you sorted them For instance, if you group by the `Server ID` field, Grafana groups the data this way:

Server ID	Data
server 1	<table><tr><th>Time</th><th>CPU Temperature</th><th>Server Status</th></tr><tr><td>7/7/2020 11:34:20 AM</td><td>80</td><td>Shutdown</td></tr><tr><td>7/7/2020 9:28:06 AM</td><td>80</td><td>OK</td></tr><tr><td>7/7/2020 9:23:07 AM</td><td>86</td><td>OK</td></tr></table>

<table>
<tr><th>Server ID</th><th>Data</th></tr>
<tr><td>server 2</td><td>
<table>
<tr><th>Time</th><th>CPU Temperature</th><th>Server Status</th></tr>
<tr><td>7/7/2020 10:32:20 AM</td><td>90</td><td>Overload</td></tr>
<tr><td>7/7/2020 9:30:05 AM</td><td>88</td><td>OK</td></tr>
<tr><td>7/7/2020 9:25:05 AM</td><td>88</td><td>OK</td></tr>
</table>
</td></tr>
<tr><td>server 3</td><td>
<table>
<tr><th>Time</th><th>CPU Temperature</th><th>Server Status</th></tr>
<tr><td>7/7/2020 11:34:20 AM</td><td>62</td><td>OK</td></tr>
<tr><td>7/7/2020 10:31:22 AM</td><td>55</td><td>OK</td></tr>
<tr><td>7/7/2020 9:30:57 AM</td><td>62</td><td>Rebooting</td></tr>
</table>
</td></tr>
</table>

After choosing the field by which you want to group your data, you can add various calculations on the other fields and apply the calculation to each group of rows. For instance, you might want to calculate the average CPU temperature for each of those servers. To do so, add the mean calculation applied on the CPU Temperature field to get the following result:

Server ID	CPU Temperatute (mean)	
server 1	82	**Time** / **Server Status** 7/7/2020 11:34:20 AM — Shutdown 7/7/2020 9:28:06 AM — OK 7/7/2020 9:23:07 AM — OK
server 2	88.6	**Time** / **Server Status** 7/7/2020 10:32:20 AM — Overload 7/7/2020 9:30:05 AM — OK 7/7/2020 9:25:05 AM — OK
server 3	59.6	**Time** / **Server Status** 7/7/2020 11:34:20 AM — OK

Server ID	CPU Temperatute (mean)		
		Time	Server Status
		7/7/2020 10:31:22 AM	OK
		7/7/2020 9:30:57 AM	Rebooting

Create heatmap

Use this transformation to prepare histogram data for visualizing trends over time. Similar to the heatmap visualization, this transformation converts histogram metrics into temporal buckets.

X Bucket

This setting determines how the x-axis is split into buckets.

- **Size** – Specify a time interval in the input field. For example, a time range of 1h creates cells one hour wide on the x-axis.
- **Count** – For non-time-related series, use this option to define the number of elements in a bucket.

Y Bucket

This setting determines how the y-axis is split into buckets.

- **Linear**
- **Logarithmic** – Choose between log base 2 or log base 10.
- **Symlog** – Uses a symmetrical logarithmic scale. Choose between log base 2 or log base 10, allowing for negative values.

Assume you have the following dataset:

Timestamp	Value
2023-01-01 12:00:00	5
2023-01-01 12:15:00	10
2023-01-01 12:30:00	15
2023-01-01 12:45:00	8

- With X Bucket set to `Size: 15m` and Y Bucket as `Linear`, the histogram organizes values into time intervals of 15 minutes on the x-axis and linearly on the y-axis.
- For X Bucket as `Count: 2` and Y Bucket as `Logarithmic (base 10)`, the histogram groups values into buckets of two on the x-axis and use a logarithmic scale on the y-axis.

Histogram

Use this transformation to generate a histogram based on input data, allowing you to visualize the distribution of values.

- **Bucket size** – The range between the lowest and highest items in a bucket (xMin to xMax).
- **Bucket offset** – The offset for non-zero-based buckets.
- **Combine series** – Create a unified histogram using all available series.

Original data

Series 1:

A	B	C
1	3	5
2	4	6
3	5	7
4	6	8

A	B	C
5	7	9

Series 2:

C
5
6
7
8
9

Output

xMin	xMax	A	B	C	C
1	2	1	0	0	0
2	3	1	0	0	0
3	4	1	1	0	0
4	5	1	1	0	0
5	6	1	1	1	1
6	7	0	1	1	1
7	8	0	1	1	1
8	9	0	0	1	1
9	10	0	0	1	1

Visualize the distribution of values using the generated histogram, providing insights into the data's spread and density.

Join by field

Use this transformation to merge multiple results into a single table, enabling the consolidation of data from different queries.

The is especially useful for converting multiple time series results into a single wide table with a shared time field.

Inner join

An inner join merges data from multiple tables where all tables share the same value from the selected field. This type of join excludes data where values do not match in every result.

Use this transformation to combine the results from multiple queries (combining on a passed join field or the first time column) into one result, and drop rows where a successful join cannot occur.

In the following example, two queries return table data. It is visualized as two separate tables before applying the inner join transformation.

Query A:

Time	Job	Uptime
7/7/2020 11:34:20 AM	node	25260122
7/7/2020 11:24:20 AM	postgre	123001233
7/7/2020 11:14:20 AM	postgre	345001233

Query B:

Time	Server	Errors
7/7/2020 11:34:20 AM	server 1	15
7/7/2020 11:24:20 AM	server 2	5
7/7/2020 11:04:20 AM	server 3	10

The result after applying the inner join transformation looks like the following:

Time	Job	Uptime	Server	Errors
7/7/2020 11:34:20 AM	node	25260122	server 1	15
7/7/2020 11:24:20 AM	postgre	123001233	server 2	5

Outer join

An outer join includes all data from an inner join and rows where values do not match in every input. While the inner join joins Query A and Query B on the time field, the outer join includes all rows that don't match on the time field.

In the following example, two queries return table data. It is visualized as two tables before applying the outer join transformation.

Query A:

Time	Job	Uptime
7/7/2020 11:34:20 AM	node	25260122
7/7/2020 11:24:20 AM	postgre	123001233
7/7/2020 11:14:20 AM	postgre	345001233

Query B:

Time	Server	Errors
7/7/2020 11:34:20 AM	server 1	15
7/7/2020 11:24:20 AM	server 2	5
7/7/2020 11:04:20 AM	server 3	10

The result after applying the outer join transformation looks like the following:

Time	Job	Uptime	Server	Errors
7/7/2020 11:04:20 AM			server 3	10
7/7/2020 11:14:20 AM	postgre	345001233		
7/7/2020 11:34:20 AM	node	25260122	server 1	15
7/7/2020 11:24:20 AM	postgre	123001233	server 2	5

Join by labels

Use this transformation to join multiple results into a single table.

This is especially useful for converting multiple time series results into a single wide table with a shared **Label** field.

- **Join** – Select the label to join by between the labels available or common across all time series.
- **Value** – The name for the output result.

Example

Input 1: `series1{what='Temp', cluster='A', job='J1'}`

Time	Value
1	10
2	200

Input 2: `series2{what='Temp', cluster='B', job='J1'}`

Time	Value
1	10
2	200

Input 3: `series3{what='Speed', cluster='B', job='J1'}`

Time	Value
22	22
28	77

Config:

```
value: 'what'
```

Output:

cluster	job	Temp	Speed
A	J1	10	
A	J1	200	
B	J1	10	22
B	J1	200	77

Combine and organize time series data effectively with this transformation for comprehensive insights.

Labels to fields

Use this transformation to convert time series results with labels or tags into a table, including each label's keys and values in the result. Display labels as either columns or row values for enhanced data visualization.

Given a query result of two time series:

- Series 1 – labels `Server=Server A`, `Datacenter=EU`
- Series 2 – labels `Server=Server B`, `Datacenter=EU`

In **Columns** mode, the result looks like this:

Time	Server	Datacenter	Value
7/7/2020 11:34:20 AM	Server A	EU	1
7/7/2020 11:34:20 AM	Server B	EU	2

In “Rows” mode, the result has a table for each series and show each label value like this:

label	value
Server	Server A
Datacenter	EU

label	value
Server	Server B
Datacenter	EU

Value field name

If you selected Server as the **Value field name**, then you would get one field for every value of the Server label.

Time	Datacenter	Server A	Server B
7/7/2020 11:34:20 AM	EU	1	2

Merging behavior

The labels to fields transformer is internally two separate transformations. The first acts on single series and extracts labels to fields. The second is the merge transformation that joins all the results into a single table. The merge transformation tries to join on all matching fields. This merge step is required and cannot be turned off.

To illustrate this, here is an example where you have two queries that return time series with no overlapping labels.

- Series 1 – labels `Server=ServerA`
- Series 2 – labels `Datacenter=EU`

This will first result in these two tables:

Time	Server	Value
7/7/2020 11:34:20 AM	ServerA	10

Time	Datacenter	Value
7/7/2020 11:34:20 AM	EU	20

After merge:

Time	Server	Value	Datacenter
7/7/2020 11:34:20 AM	ServerA	10	
7/7/2020 11:34:20 AM		20	EU

Limit

Use this transformation to restrict the number of rows displayed, providing a more focused view of your data. This is particularly useful when dealing with large datasets.

The following is an example illustrating the impact of the **Limit** transformation on a response from a data source:

Time	Metric	Value
7/7/2020 11:34:20 AM	Temperature	25
7/7/2020 11:34:20 AM	Humidity	22
7/7/2020 10:32:20 AM	Humidity	29
7/7/2020 10:31:22 AM	Temperature	22
7/7/2020 9:30:57 AM	Humidity	33
7/7/2020 9:30:05 AM	Temperature	19

Here is the result after adding a Limit transformation with a value of ‘3’:

Time	Metric	Value
7/7/2020 11:34:20 AM	Temperature	25
7/7/2020 11:34:20 AM	Humidity	22

Time	Metric	Value
7/7/2020 10:32:20 AM	Humidity	29

This transformation helps you tailor the visual presentation of your data to focus on the most relevant data.

Merge series/tables

Use this transformation to combine the result from multiple queries into a single result, which is particularly useful when using the table panel visualization. The transformation merges values into the same row if shared fields contain the same data.

Here's an exampleillustration the impact of the **Merge series/tables** transformation on two queries returning table data:

Query A:

Time	Job	Uptime
7/7/2020 11:34:20 AM	node	25260122
7/7/2020 11:24:20 AM	postgre	123001233

Query B:

Time	Job	Errors
7/7/2020 11:34:20 AM	node	15
7/7/2020 11:24:20 AM	postgre	5

Here is the result after applying the Merge transformation:

Time	Job	Errors	Uptime
7/7/2020 11:34:20 AM	node	15	25260122
7/7/2020 11:24:20 AM	postgre	5	123001233

This transformation combines values from Query A and Query B into a unified table, enhancingthe presentation for better insights.

Organize fields by name

Use this transformation to rename, reorder, or hide fields returned by a single query in your panel. This transformation only works in panels with a single query. If your panel has multiple queries, then you must either apply an *Outer join* transformation or remove the extra queries.

Transforming fields

Grafana displays a list of fields returned by the query. You can:

- **Change field order** – Drag a field to a new location in the list.
- **Hide or show a field** – Use the eye icon next to the field name to toggle the visibility of a field.
- **Rename fields** – Type a new name into the **Rename** box.

Example

Given this initial query result:

Time	Metric	Value
2020-07-07 11:34:20	Temperature	25
2020-07-07 11:34:20	Humidity	22
2020-07-07 10:32:20	Humidity	29

You could apply a rename field override to create:

Time	Sensor	Reading
2020-07-07 11:34:20	Temperature	25
2020-07-07 11:34:20	Humidity	22
2020-07-07 10:32:20	Humidity	29

This transformation lets you tailor the display of query results, ensuring a clear and insightful representation of your data in Grafana.

Partition by values

Use this transformation to streamline the process of graphing multiple series without the need for multiple queries with different `WHERE` clauses.

> **Note**
>
> This function is available in workspaces compatible with Grafana version 9 and above.

This is particularly useful when using a metrics SQL table, as in this example:

Time	Region	Value
10/20/2022 12:00:00 PM	US	1520
10/20/2022 12:00:00 PM	EU	2936
10/20/2022 1:00:00 AM	US	1327
10/20/2022 1:00:00 AM	EU	912

With the *Partition by values* transformer, you can issue a single query and split the results by unique values in one or more columns (fields) of your choosing. The following example uses `Region`.

```
SELECT Time, Region, Value FROM metrics WHERE Time > '2022-10-20'
```

Time	Region	Value
10/20/2022 12:00:00 PM	US	1520
10/20/2022 1:00:00 AM	US	1327

Time	Region	Value
10/20/2022 12:00:00 PM	EU	2936
10/20/2022 1:00:00 AM	EU	912

This transformation simplifies the process and enhances the flexibility of visualizing multiple series within the same time series visualization.

Prepare times series

Use this transformation to address issues when a data source returns time series data in a format that isn't compatible with the desired visualization. This transformation allows you to convert time series data from between wide and long formats.

Multi-frame time series

Use this option to transform the time series data frame from the wide format to the long format. This is particularly helpful when your data source delivers time series information in a format that needs to be reshaped for optimal compatibility with your visualization.

Example

This input:

Timestamp	Value1	Value2
2023-01-01 00:00:00	10	20
2023-01-01 01:00:00	15	25

Could be transformed to:

Timestamp	Variable	Value
2023-01-01 00:00:00	Value1	10
2023-01-01 00:00:00	Value2	20
2023-01-01 01:00:00	Value1	15
2023-01-01 01:00:00	Value2	25

Wide time series

Use this option to transform the time series data frame from the long format to the wide format. This is particularly helpful when your data source delivers time series data in a long format and your visualization requires a wide format.

Example

This input:

Timestamp	Variable	Value
2023-01-01 00:00:00	Value1	10
2023-01-01 00:00:00	Value2	20
2023-01-01 01:00:00	Value1	15
2023-01-01 01:00:00	Value2	25

Could be transformed to:

Timestamp	Value1	Value2
2023-01-01 00:00:00	10	20
2023-01-01 01:00:00	15	25

Reduce

Use this transformation applies a calculation to each field in the data frame and return a single value. This transformation is particularly useful for consolidating mulitiple time series data into a more compact, summarized format. Time fields are removed when applying this transformation.

Consider the input:

Query A:

Time	Temp	Uptime
2020-07-07 11:34:20	12.3	256122
2020-07-07 11:24:20	15.4	1230233

Query B:

Time	AQI	Errors
2020-07-07 11:34:20	6.5	15
2020-07-07 11:24:20	3.2	5

The reduce transformer has two modes:

- **Series to rows** – Creates a row for each field and a column for each calculation.
- **Reduce fields** – Keeps the existing frame structure, but collapses each field into a single value.

For example, if you used the **First** and **Last** calculation with a **Series to rows** transformation, then the result wouldbe:

Field	First	Last
Temp	12.3	15.4
Uptime	256122	1230233

Field	First	Last
AQI	6.5	3.2
Errors	15	5

The **Reduce fields** with the **Last** calculation, results in two frames, each with one row:

Query A:

Temp	Uptime
15.4	1230233

Query B:

AQI	Errors
3.2	5

Rename by regex

Use this transformation to rename parts of the query results using a regular expression and replacement pattern.

You can specify a regular expression, which is only applied to matches, along with a replacement pattern that support back references. For example, let's imagine you're visualizing CPU usage per host and you want to remove the domain name. You could set the regex to `([^\.]+)\..+` and the replacement pattern to `$1`, `web-01.example.com` would become `web-01`.

This transformation lets you tailor your data to meet your visualization needs, making your dashboards more informative and user-friendly.

Rows to fields

Use this transformation to convert rows into separate fields. This can be useful because fields can be styled and configured individually. It can also use additional fields as sources for dynamic field

configuration or map them to field labels. The additional labels can then be used to define better display names for the resulting fields.

This transformation includes a field table which lists all fields in the data returned by the configuration query. This table gives you control over what field should be mapped to each configuration property (the **Use as** option). You can also choose which value to select if there are multiple rows in the returned data.

This transformation requires:

- One field to use as the source of field names.

  By default, the transform uses the first string field as the source. You can override this default setting by selecting **Field name** in the **Use as** column for the field you want to use instead.
- One field to use as the source of values.

  By default, the transform uses the first number field as the source. But you can override this default setting by selecting **Field value** in the **Use as** column for the field you want to use instead.

Useful when visualizing data in:

- Gauge
- Stat
- Pie chart

Map extra fields to labels

If a field does not map to config property Grafana will automatically use it as source for a label on the output field.

Example:

Name	DataCenter	Value
ServerA	US	100
ServerB	EU	200

Output:

ServerA (labels: DataCenter: US)	ServerB (labels: DataCenter: EU)
100	200

The extra labels can now be used in the field display name to provide more complete field names.

If you want to extract config from one query and apply it to another you should use the *config from query results* transformation.

Example

Input:

Name	Value	Max
ServerA	10	100
ServerB	20	200
ServerC	30	300

Output:

ServerA (config: max=100)	ServerB (config: max=200)	ServerC (config: max=300)
10	20	30

As you can see each row in the source data becomes a separate field. Each field now also has a max config option set. Options like **Min**, **Max**, **Unit** and **Thresholds** are all part of field configuration and if set like this will be used by the visualization instead of any options manually configured in the panel editor options pane.

This transformation enables the conversion of rows into individual fields, facilitates dynamic field configuration, and maps additional fields to labels.

Series to rows

Use this transformation to combine the result from multiple time series data queries into one single result. This is helpful when using the table panel visualization.

The result from this transformation will contain three columns: Time, Metric, and Value. The Metric column is added so you easily can see from which query the metric originates from. Customize this value by defining Label on the source query.

In the example below, we have two queries returning time series data. It is visualized as two separate tables before applying the transformation.

Query A:

Time	Temperature
2020-07-07 11:34:20	25
2020-07-07 10:31:22	22
2020-07-07 09:30:05	19

Query B:

Time	Humidity
2020-07-07 11:34:20	24
2020-07-07 10:32:20	29
2020-07-07 09:30:57	33

Here is the result after applying the Series to rows transformation.

Time	Metric	Value
2020-07-07 11:34:20	Temperature	25
2020-07-07 11:34:20	Humidity	22

Time	Metric	Value
2020-07-07 10:32:20	Humidity	29
2020-07-07 10:31:22	Temperature	22
2020-07-07 09:30:57	Humidity	33
2020-07-07 09:30:05	Temperature	19

This transformation facilitates the consolidation of results from multiple time series queries, providing a streamlined and unified dataset for efficient analysis and visualization in a tabular format.

Sort by

Use this transformation to sort each frame within a query result based on a specified field, making your data easier to understand and analyze. By configuring the desired field for sorting, you can control the order in which the data is presented in the table or visualization.

Use the **Reverse** switch to inversely order the values within the specified field. This functionality is particularly useful when you want to quickly toggle between ascending and descending order to suit your analytical needs.

For example, in a scenario where time-series data is retrieved from a data source, the **Sort by** transformation can be applied to arrange the data frames based on the timestamp, either in ascending or descending order, depending on the analytical requirements. This capability ensures that you can easily navigate and interpret time-series data, gaining valuable insights from the organized and visually coherent presentation.

Spatial

Use this transformation to apply spatial operations to query results.

- **Action** – Select an action:
  - **Prepare spatial field** – Set a geometry field based on the results of other fields.
    - **Location mode** – Select a location mode (these options are shared by the **Calculate value** and **Transform** modes):
      - **Auto** – Automatically identify location data based on default field names.

    - **Coords** – Specify latitude and longitude fields.
    - **Geohash** – Specify a geohash field.
    - **Lookup** – Specify Gazetteer location fields.
- **Calculate value** – Use the geometry to define a new field (heading/distance/area).
  - **Function** – Choose a mathematical operation to apply to the geometry:
    - **Heading** – Calculate the heading (direction) between two points.
    - **Area** – Calculate the area enclosed by a polygon defined by the geometry.
    - **Distance** – Calculate the distance between two points.
- **Transform** – Apply spatial operations to the geometry.
  - **Operation** – Choose an operation to apply to the geometry:
    - **As line** – Create a single line feature with a vertex at each row.
    - **Line builder** – Create a line between two points.

This transformation allows you to manipulate and analyze geospatial data, enabling operations such as creating lines between points, calculating spatial properties, and more.

Time series to table transform

Use this transformation to convert time series results into a table, transforming a time series data frame into a **Trend** field. The **Trend** field can then be rendered using the sparkline cell type, generating an inline sparkline for each table row. If there are multiple time series queries, each will result in a separate table data frame. These can be joined using join or merge transforms to produce a single table with multiple sparklines per row.

For each generated **Trend** field value, a calculation function can be selected. The default is **Last non-null value**. This value is displayed next to the sparkline and used for sorting table rows.

Regression analysis

Use this transformation to create a new data frame containing values predicted by a statistical model. This is useful for finding a trend in chaotic data. It works by fitting a mathematical function to the data, using either linear or polynomial regression. The data frame can then be used in a visualization to display a trendline.

There are two different models:

- **Linear regression** – Fits a linear function to the data.

- **Polynomial regression** – Fits a polynomial function to the data.

> **Note**
>
> This transformation is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

Troubleshoot queries

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This page provides information to solve common dashboard problems.

I get different results when I rearrange my functions

Function order is very important. Just like in math, the order that you place your functions can affect the result.

Inspect your query request and response

The most common problems are related to the query and response from your data source. Even if it looks like a bug or visualization issue in Grafana, it is almost always a problem with the data source query or the data source response. Start by inspecting your panel query and response.

For more information, refer to Inspect request and response data.

My query is slow

How many data points is your query returning? A query that returns lots of data points will be slow. Try this:

- In **Query options**, limit the **Max data points** returned.

- In **Query options**, increase the **Min interval** time.
- In your query, use a `group by` function.

Calculation types

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The following table contains a list of calculations you can perform in Grafana. You can find these calculations in the **Transform** tab and in the bar gauge, gauge, and stat visualizations.

Calculation	Description
All nulls	True when all values are null
All values	Array with all values
All unique values	Array with all unique values
All zeros	True when all values are 0
Change count	Number of times the field's value changes
Count	Number of values in a field
Delta	Cumulative change in value, only counts increments
Difference	Difference between first and last value of a field
Difference percent	Percentage change between first and last value of a field

Calculation	Description
Distinct count	Number of unique values in a field
First	First value in a field
First* (not null)	First, not null value in a field (also excludes NaNs)
Last	Last value in a field
Last* (not null)	Last, not null value in a field (also excludes NaNs)
Max	Maximum value of a field
Mean	Mean value of all values in a field
Variance	Variance of all values in a field
StdDev	Standard deviation of all values in a field
Min	Minimum value of a field
Min (above zero)	Minimum, positive value of a field
Range	Difference between maximum and minimum values of a field
Step	Minimal interval between values of a field
Total	Sum of all values in a field

Configure panel options

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A Grafana panel is a visual representation of data that you can customize by defining a data source query, transforming and formatting data, and configuring visualization settings.

A panel editor includes a query builder and a series of options that you can use to transform data and add information to your panels.

This topic describes how to:

- Open a panel for editing
- Add a panel title and description
- View a panel JSON model
- Configure repeating rows and panels

Editing a panel

After you add a panel to a dashboard, you can open it at any time to change or update queries, add data transformation, and change visualization settings.

To edit a panel

1. Open the dashboard that contains the panel you want to edit.
2. Hover over any part of the panel to display the actions menu in the top right corner.
3. Choose the menu and select **Edit**.

   To use a keyboard shortcut to open the panel, hover over the panel and press e.

   The panel opens in edit mode.

Add a title and description to a panel

Add a title and description to a panel to share with users any important information about the visualization. For example, use the description to document the purpose of the visualization.

1. Edit a panel.

2. In the panel display options pane, locate the **Panel options** section.
3. Enter a **Title**.

   Text entered in this field appears in a tooltip in the panel editor and in the dashboard.
4. Write a description of the panel and the data you are displaying.

   Text entered in this field appears in a tooltip in the upper-left corner of the panel.

   You can use variables you have defined in the **Title** and **Description** field, but not global variables.

Viewing a panel JSON model

Explore and export panel, panel data, and data frame JSON models.

To view a panel JSON model

1. Open the dashboard that contains the panel.
2. Hover over any part of the panel to display the actions menu on the top right corner.
3. From the menu, select **Inspect > Panel JSON**.
4. In the **Select source** field, choose one of the following options:

   - **Panel JSON** – Displays a JSON object representing the panel.
   - **Panel data** – Displays a JSON object representing the data that was passed to the panel.
   - **DataFrame structure** – Displays the data structure of the panel, including any transformations, field configurations, and override configurations that have been applied.
5. To explore the JSON, choose **>** to expand or collapse portions of the JSON model.

Configuring repeating panels

You can configure Grafana to dynamically add panels or rows to a dashboard. A dynamic panel is a panel that the system creates based on the value of a variable. Variables dynamically change your queries across all panels in a dashboard. For more information about repeating panels, see Creating dashboards.

> **Note**
>
> Repeating panels require variables to have one or more items selected; you cannot repeat a panel zero times to hide it.

Prequisites

- Ensure that the query includes a multi-value variable.

To configure repeating panels

1. Edit the panel you want to repeat.
2. On the display options pane, choose **Panel options > Repeat options**.
3. Select a **direction**.
   - Choose **horizontal** to arrange panels side-by-side. Grafana adjusts the width of a repeated panel. You cannot mix other panels on a row with a repeated panel.
   - Choose **vertical** to arrange panels in a column. The width of repeated panels is the same as the original, repeated panel.
4. To propagate changes to all panels, reload the dashboard.

Configure standard options

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The data model used in Grafana is a columnar-oriented table structure that unifies both time series and table query results. Each column within this structure is called a *field*. A field can represent a single time series or table column.

Field options allow you to change how the data is displayed in your visualizations. Options and overrides that you apply do not change the data, they change how Grafana displays the data. When you change an option, it is applied to all fields, meaning all series or columns. For example, if you change the unit to percentage, then all fields with numeric values are displayed in percentages.

A complete list of field formatting options is included later in this topic.

> **Note**
>
> You can apply standard options to most built-in Grafana panels. Some older panels and community panels that have not updated to the new panel and data model will be missing either all or some of these field options.

To configure standard options

1. Open a dashboard. Hover over any part of the panel to display the actions menu at the top right corner of the dashboard.
2. From the actions menu, choose **Edit**.
3. In the panel display options pane, locate the **Standard options** section.
4. Select the standard options you want to apply.
5. To preview your change, select outside of the field option box you are editing or press **Enter**.

Standard options definitions

This section explains all available standard options.

You can apply standard options to most built-in Grafana panels. Some older panels and community panels that have not updated to the new panel and data model will be missing either all or some of these field options.

Most field options will not affect the visualization until you click outside of the field option box you are editing or press **Enter**.

> **Note**
>
> Grafana Labs is constantly working to add and expand options for all visualization, so all options might not be available for all visualizations.

Unit

Lets you choose what unit a field should use. Choose the **Unit** field, then drill down until you find the unit you want. The unit you select is applied to all fields except time.

Custom units

You can use the unit dropdown to also specify custom units, custom prefix or suffix and date time formats.

To select a custom unit enter the unit and select the last **Custom: xxx** option in the dropdown.

- **suffix:<suffix>** for a custom unit that should go after the value.
- **prefix:<prefix>** for a custom unit that should go before the value.
- **time:<format>** For custom date time formats, type for example `time:YYYY-MM-DD`. See format in the *Moment.js Documentation* for the format syntax and options.
- **si:<base scale><unit characters>** for custom SI units. For example: `si: mF`. This is a bit more advanced as you can specify both a unit and the source data scale. So if your source data is represented as milli (thousands of) something prefix the unit with that SI scale character.
- **count:<unit>** for a custom count unit.
- **currency:<unit>** for custom a currency unit.

You can also paste a native emoji in the unit picker and pick it as a custom unit.

String units

Grafana can sometimes be too aggressive in parsing strings and displaying them as numbers. To configure Grafana to show the original string value, create a field override and add a unit property with the **String** unit.

Scale units

By default, Grafana automatically scales the unit based on the magnitude of the value. For example, if you have a value of 0.14 kW, Grafana will display it as 140 W. Another example is that 3000 kW will be displayed as three MW. If you want to disable this behavior, you can turn off the **Scale units** switch.

Min

Lets you set the minimum value used in percentage threshold calculations. Leave blank to automatically calculate the minimum.

Max

Lets you set the maximum value used in percentage threshold calculations. Leave blank to automatically calculate the maximum.

Field min/max

By default the calculated min and max will be based on the minimum and maximum, in all series and fields. Turning field min/max on will calculate the min or max on each field individually, based on the minimum or maximum of that field.

Decimals

Specify the number of decimals Grafana includes in the rendered value. If you leave this field blank, Grafana automatically truncates the number of decimals based on the value. For example 1.1234 will display as 1.12 and 100.456 will display as 100.

To display all decimals, set the unit to **String**.

Display name

Lets you set the display title of all fields. You can use variables in the field title.

When multiple stats, fields, or series are shown, this field controls the title in each stat. You can use expressions like **${__field.name}** to use only the series name or the field name in the title.

Given a field with a name of `Temp`, and labels of `{"Loc"="PBI", "Sensor"="3"}`:

Expression syntax	Example	Renders to	Explanation
`${__field .displayName}`	Same as syntax	`Temp {Loc="PBI", Sensor="3"}`	Displays the field name, and labels in {} if they are present. If there is only one label key in

Expression syntax	Example	Renders to	Explanation
			the response, then for the label portion, Grafana displays the value of the label without the enclosing braces.
`${__field.name}`	Same as syntax	`Temp`	Displays the name of the field (without labels).
`${__field.labels}`	Same as syntax	`Loc="PBI", Sensor="3"`	Displays the labels without the name.
`${__field.labels.`*X*`}`	`${__field.labels.Loc}`	`PBI`	Displays the value of the specified label key.
`${__field.labels.__values}`	Same as Syntax	`PBI, 3`	Displays the values of the labels separated by a comma (without label keys).

If the value is an empty string after rendering the expression for a particular field, then the default display method is used.

Color scheme

The color options and their effect on the visualization depends on the visualization you are working with. Some visualizations have different color options.

You can specify a single color, or select a continuous (gradient) color scheme, based on a value. Continuous color interpolates a color using the percentage of a value relative to min and max.

Select one of the following palettes:

Color mode	Description
Single color	Specify a single color, useful in an override rule
Shades of a color	Selects shades of a single color, useful in an override rule
From thresholds	Informs Grafana to take the color from the matching threshold
Classic palette	Grafana will assign color by looking up a color in a palette by series index. Useful for Graphs and pie charts and other categorical data visualizations
Classic palette (by series name)	Grafana will assign color based on the name of the series. Useful when the series names to be vsiualized depend on the available data.
Green-Yellow-Red (by value)	Continuous color scheme
Red-Yellow-Green (by value)	Continuous color scheme
Blue-Yellow-Red (by value)	Continuous color scheme
Yellow-Red (by value)	Continuous color scheme
Blue-Purple (by value)	Continuous color scheme
Yellow-Blue (by value)	Continuous color scheme
Blues (by value)	Continuous color scheme (panel background to blue)
Reds (by value)	Continuous color scheme (panel background color to red)
Greens (by value)	Continuous color scheme (panel background color to green)

Color mode	Description
Purple (by value)	Continuous color scheme (panel background color to purple)

No value

Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

Configure a legend

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A panel includes a legend that you can use to interpret data displayed in a visualization. Each legend option adds context and clarity to the data illustrated in a visualization.

Legends are supported for the following visualizations:

- Bar chart
- Candlestick
- Histogram
- Pie chart
- State timeline
- Status history
- Time series
- Trend

Geomaps and Heatmaps also have legends, but they only provide the choice to display or not display a legend, and don't support other legend options.

Legend options

You can find the following options under the **Legend** section in the panel edit pane.

> **Note**
>
> Not all of the options listed apply to all visualizations with legends.

Visibility

Set whether the legend is displayed or not. Use the switch to toggle a legend on or off.

Mode

Set the format in which the legend is displayed. Choose from:

- **List**
- **Table**

When you format a legend as a table, other information about the legend, such as associated values or where it's located in the visualization, might be displayed as well.

Placement

Set where on the visualization a legend is displayed. Choose from:

- **Bottom**
- **Right**

Width

If you set the legend placement to **Right**, the **Width** option becomes available. Leave the field empty to allow Grafana to automatically set the legend width, or enter a vale in the field.

Values

You can add more context to a visualization by adding series data values or calculations to a legend. You can add as many values as you like. After you apply your changes, you can scroll the legend to see all values.

Changing a series color

By default, Grafana sets the color of your series data, but you can change them through the panel legend.

To change a series color

1. Navigate to the panel you want to update.
2. In the legend, select the color bar associated with the series.
3. Select a pre-set color in the **Colors** tab or set a custom color in the **Custom** tab, using the picker or RGB values.
4. Save the dashboard.

Isolating series data in a visualization

Visualizations can often be visually complex, and include many data series. You can simplify the view by removing series data from the visualization through the legend, which isolates the data you want to see. When you do this, Grafana automatically creates a new override in the **Override** tab.

To isolate series data in a visualization

1. Navigate to the panel you want to update.
2. In the legend, select the label of the series you want to isolate.

   The system removes from view all other series data.
3. To incrementally add series data back to an isolated series, press the **Ctrl** or **Command** key and select the label of the series you want to add.
4. To save your changes so that they appear to all viewers of the panel, save the dashboard.

To revert back to the default view that includes all data, click any series label twice.

Sorting series

When you format a legend as a table and add values to it, you can sort the series in the table by those values.

To sort series

1. Navigate to the panel you want to update.
2. Hover over any part of the panel you want to work on to display the menu on the top right corner of the panel.
3. From the menu, choose **Edit**.
4. Scroll to the **Legend** section of the panel edit pane.
5. Under **Values**, select the value or calculation that you want to show.

   The legend now displays values.
6. Choose the calculation name header in the legend table to sort the values in the table in ascending or descending order.

> **Note**
>
> This feature is only supported in these panels: bar chart, histogram, time series.

Configure data links

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Data links allow you to provide more granular context to your links. You can create links that include the series name or even the value under the cursor. For example, if your visualization shows

four servers, you can add a datalink to one or two of them. You can also link panels using data links.

The link itself is accessible in different ways depending on the visualization. For the time series visualization, for example, you choose a data point or line. For large area visualizations, like stat, gauge, or bar gauge, you can choose anywhere on the visualization to open the context menu.

If there is only one data link in the visualization, choosing anywhere on the visualization opens the link rather than the context menu.

Supported visualizations

- Bar chart
- Bar gauge
- Candlestick
- Canvas
- Gauge
- Geomap
- Heatmap
- Histogram
- Pie chart
- Stat
- State timeline
- Status history
- Table
- Time series
- Trend

Data link variables

Variables in data links let you send people to a detailed dashboard with preserved data filters. For example, you could use variable to specify a label, time range, series, or variable selection.

To see a list of available variables, type $ in the data link **URL** field.

You can also use template variables in your data links URLs, see Variables.

Time range panel variables

These variables allow you to include the current time range in the data link URL.

- `__url_time_range` – current dashboard's time range (i.e. `?from=now-6h&to=now`)
- `$__from` – For more information, see Global variables.
- `$__to` – For more information, see Global variables.

Series variables

Series specific variables are available under `__series` namespace:

- `__series.name` – series name to the URL

Field variables

Field-specific variables are available under `__field` namespace:

- `__field.name` – the name of the field
- `__field.labels.<LABEL>` – label's value to the URL. If your label contains dots, then use `__field.labels["<LABEL>"]` syntax.

Value variables

Value-specific variables are available under `__value` namespace:

- `__value.time` – value's timestamp (Unix ms epoch) to the URL (i.e. `?time=1560268814105`)
- `__value.raw` – raw value
- `__value.numeric` – numeric representation of a value
- `__value.text` – text representation of a value
- `__value.calc` – calculation name if the value is result of calculation

Using value-specific variable in data links can show different result depending on the set option of Tooltip mode.

Data variables

To access values from other fields use:

- `__data.fields[i]` – Value of field `i` (on the same row).
- `__data.fields["NameOfField"]` – Value of field using name instead of index.
- `__data.fields[i].labels.cluster` – Access the labels of another field.

Template variables

When linking to another dashboard that uses template variables, select variable values for whoever clicks the link.

`${var-myvar:queryparam}` – where `var-myvar` is the name of the template variable that matches one in the current dashboard that you want to use.

Variable state	Result in the created URL
selected one value	`var-myvar=value1`
selected multiple values	`var-myvar=value1&var-myvar=value2`
selected `All`	`var-myvar=All`

If you want to add all of the current dashboard's variables to the URL, then use `${__all_variables}`.

Adding a data link

You can add data links to your panels.

1. Navigate to the panel to which you want to add the data link.
2. Hover over the panel to display the menu icon in the upper-right corner.
3. From the menu, choose **Edit** to open the panel editor.
4. In the **Panel edit** pane, scroll down to the **Data links** section and expand it.
5. Choose **Add link**.
6. In the dialog box that opens, enter a **Title**. This is a human-readable label for the link, which will be displayed in the UI.
7. Enter the **URL** or variable you want to link to.

To add a data link variable, select the **URL** field and then enter $ or press Ctrl+Space or Cmd +Space to see a list of available variables.

8. If you want the link to open in a new tab, then select **Open in a new tab**.
9. Choose **Save** to save changes and close the dialog box.
10. Save your changes to the dashboard.

Configure value mappings

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

In addition to field overrides, value mapping is a technique that you can use to change how data appears in a visualization.

Values mappings bypass the unit formatting set in the Standard options of the panel editor, like color or number of decimal places displayed. When value mappings are present in a panel, Grafana displays a summary of them in the **Value mappings** section of the panel editor.

Supported visualizations

- Bar chart
- Bar gauge
- Candlestick
- Canvas
- Gauge
- Geomap
- Histogram
- Pie chart

- Stat
- State timeline
- Status history
- Table
- Time series
- Trend

Types of value mappings

Grafana supports the following value mapping types:

- **Value** – Maps specific values to a text and a color. For example, you can configure a value mapping so that all instances of the value `10` appear as `Perfection!` rather than the number. Use a **Value** mapping when you want to format a single value.
- **Range** – Maps numerical ranges to text and a color. For example, if a value is within a certain range, you can configure a range value mapping to display `Low` or `High` rather than the number. Use **Range** mapping when you want to formate multiple continuous values.
- **Regex** – Maps regular expressions to text and a color. For example, if a value is `www.example.com`, you can configure a regex value mapping so that Grafana displays `www` and truncates the domain. Use the **Regex** mapping when you want to format the text and color of a regular expression value.
- **Special** – Maps special values like `Null`, `NaN` (not a number), and Boolean values like `true` and `false` to a text and color. For example, you can configure a special value mapping so that `null` values appear as `N/A`. Use the **Special** mappiong when you want to format uncommon, Boolean, or empty value.

Adding a value mapping

You can add value mappings to your panels.

To add a value mapping

1. Navigate to the panel you want to update.
2. Hover over any part of the panel to display a menu at the top right corner of the panel.
3. From the menu, choose **Edit**.

4. In the **Value mappings** section, choose **Add value mappings**.
5. Choose **Add a new mapping**, and then select one of the following:
   - **Value** – Enter a single value to match.
   - **Range** – Enter the beginning and ending values of a range to match.
   - **Regex** – Enter a regular expression pattern to match.
   - **Special** – Choose a special value to match.
6. (Optional) Enter display text.
7. (Optional) Set the color.
8. Choose **Update** to save the value mapping.

After you've added a mapping, the **Edit value mappings** button replaces the **Add value mappings** button. Choose the edit button to add or update mappings.

Configure thresholds

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

In dashboards, a threshold is a value or limit you set for a metric that's reflected visually when it's met or exceeded. Thresholds are one way you can conditionally style and color your visualizations based on query results.

You can use thresholds to:

- Color grid lines or grid ares areas in a time-series visualization.
- Color the background or value text in a stat visualization.
- Define regions and region colors in a state timeline.
- Color lines in a time-series visualization.
- Color the gauge and threshold markers in a gauge.

- Color markers in a geomap.
- Color cell text or background in a table.

Supported visualizations

- Bar chart
- Bar gauge
- Candlestick
- Canvas
- Gauge
- Geomap
- Histogram
- Stat
- State timeline
- Status history
- Table
- Time series
- Trend

Default thresholds

On visualizations that support thresholds, Grafana has the following default threshold settings:

- 80 = red
- Base = green
- Mode = Absolute
- Show thresholds = Off (for some visualizations)

  For more information, see Show thresholds.

Thresholds options

You can set the following options to further define how thresholds look.

Thresholds value

This number is the value that triggers the threshold. You can also set the color associated with the threshold in this field.

The **Base** value represents minus infinity. By default, it's set to the color green, which is generally the "good" color.

Thresholds mode

There are two threshold modes:

- **Absolute** thresholds are defined by a number. For example, 80 on a scale of 1 to 150.
- **Percentage** thresholds are defined relative to minimum or maximum. For example, 80 percent.

Show thresholds

> **Note**
>
> This option is only supported for the bar chart, candlestick, time series, and trend visualizations.

Set if and how thresholds are shown on the visualization with the following options.

- **Off** – The thresholds are not shown.
- **As lines** – The thresholds are shown as horizontal lines on the visualization.
- **As lines (dashed)** – The thresholds are shown as dashed horizontal lines.
- **As filled regions** – The thresholds are shown as horizontal regions.
- **As filled regions and lines** – The thresholds are shown as horizontal regions separated by lines.
- **As filled regions and lines (dashed)** – The thresholds are shown as horizontal regions separated by dashed lines.

Adding a threshold

You can add as many thresholds to a visualization as you want. Grafana automatically sorts thresholds values from highest to lowest.

To add a threshold

1. Navigate to the panel you want to update.
2. Hover over any part of the panel to display the menu at the top right corner.
3. From the menu, select **Edit**.
4. Scroll to the **Thresholds** section, or enter `Thresholds` in the search bar at the top of the panel edit pane.
5. Choose **+ Add threshold**.
6. Enter a new threshold value, or use the up and down arrows at the right side of the field to increase or decrease the value incrementally.
7. Click the colored circle to the left of the threshold value to open the color picker, where you can update the threshold color.
8. Under **Thresholds mode**, select either **Absolute** or **Percentage**.
9. Under **Show thresholds**, set how the threshold is displayed, or turn it off.

To delete a threshold, navigate to the panel that contains the threshold and choose the trash icon next to the threshold you want to remove.

Configure field overrides

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Overrides allow you to customize visualization settings for specific fields or series. This is accomplished by adding an override rule that targets a particular set of fields and that can each define multiple options.

For example, you set the unit for all fields that include the text `bytes` by adding an override using the **Fields with name matching regex** matcher and then add the Unit option to the override rule.

Example 1: Format temperature

Let's assume that our result set is a data frame that consists of two fields: `time` and `temperature`.

time	temperature
2020-01-02 03:04:00	45.0
2020-01-02 03:05:00	47.0
2020-01-02 03:06:00	48.0

Each field (column) of this structure can have field options applied that alter the way its values are displayed. This means that you can, for example, set the Unit to Temperature > Celsius, resulting in the following table:

time	temperature
2020-01-02 03:04:00	45.0 °C
2020-01-02 03:05:00	47.0 °C
2020-01-02 03:06:00	48.0 °C

In addition, the decimal place is not required, so we can remove it. You can change the Decimals from **auto** to zero (**0**), resulting in the following table:

time	temperature
2020-01-02 03:04:00	45 °C
2020-01-02 03:05:00	47 °C
2020-01-02 03:06:00	48 °C

Example 2: Format temperature and humidity

Let's assume that our result set is a data frame that consists of four fields: `time`, `high temp`, `low temp`, and `humidity`.

time	high temp	low temp	humidity
2020-01-02 03:04:00	45.0	30.0	67
2020-01-02 03:05:00	47.0	34.0	68
2020-01-02 03:06:00	48.0	31.0	68

Let's add the Celsius unit and get rid of the decimal place. This results in the following table:

time	high temp	low temp	humidity
2020-01-02 03:04:00	45 °C	30 °C	67 °C
2020-01-02 03:05:00	47 °C	34 °C	68 °C
2020-01-02 03:06:00	48 °C	31 °C	68 °C

The temperature fields look good, but the humidity must now be changed. We can fix this by applying a field option override to the humidity field and change the unit to Misc > percent (0-100).

time	high temp	low temp	humidity
2020-01-02 03:04:00	45 °C	30 °C	67%
2020-01-02 03:05:00	47 °C	34 °C	68%
2020-01-02 03:06:00	48 °C	31 °C	68%

Adding a field override

A field override rule can customize the visualization settings for a specific field or series.

To add a field override

1. Edit the panel to which you want to add an override.
2. In the panel options side pane, choose **Add field override** at the bottom of the pane.
3. Select which fields an override rule will be applied to:
   - **Fields with name** – Select a field from the list of all available fields. Properties you add to a rule with this selector are only applied to this single field.
   - **Fields with name matching regex** – Specify fields to override with a regular expression. Properties you add to a rule with this selector are applied to all fields where the field name match the regex. This selects fields to override, but doesn't rename the fields; to do this, use the Rename by regex transformation.
   - **Fields with type** – Select fields by type, such as string, numeric, and so on. Properties you add to a rule with this selector are applied to all fields that match the selected type.
   - **Fields returned by query** – Select all fields returned by a specific query, such as A, B, or C. Properties you add to a rule with this selector are applied to all fields returned by the selected query.
4. Choose **Add override property**.
5. Select the field option that you want to apply.
6. Enter options by adding values in the fields. To return options to default values, delete the white text in the fields.
7. Continue to add overrides to this field by choosing **Add override property**, or you can choose **Add override** and select a different field to add overrides to.
8. When finished, choose **Save** to save all panel edits to the dashboard.

Deleting a field override

Delete a field override when you no longer need it. When you delete an override, the appearance of value defaults to its original format. This change impacts dashboards and dashboard users that rely on an affected panel.

To delete a field override

1. Edit the panel that contains the override you want to delete.
2. In panel options side pane, scroll down until you see the overrides.

3. Choose the override you want to delete and then choose the associated trash icon.

Viewing field overrides

You can view field overrides in the panel display options.

1. Edit the panel that contains the overrides you want to view.
2. In panel options side pane, scroll down until you see the overrides.

> **Note**
>
> The override settings that appear on the **All** tab are the same as the settings that appear on the **Overrides** tab.

Editing a field override

Edit a field override when you want to make changes to an override setting. The change you make takes effect immediately.

To edit a field override

1. Edit the panel that contains the overrides you want to edit.
2. In panel options side pane, scroll down until you see the overrides.
3. Locate the override that you want to change.
4. Perform any of the following:

   - Edit settings on existing overrides or field selection parameters.
   - Delete existing override properties by choosing the **X** next to the property.
   - Add an override properties by choosing **Add override property**.

Visualizations available in Grafana version 10

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana offers a variety of visualizations to support different use cases. This section of the documentation highlights the built-in visualizations, their options and typical usage.

A common panel to get started with, and to learn the basics of using panels, is the Time series panel.

Note

If you are unsure which visualization to pick, Grafana can provide visualization suggestions based on the panel query. When you select a visualization, Grafana will show a preview with that visualization applied.

- Graphs & charts
  - Time series is the default and main Graph visualization.
  - State timeline for state changes over time.
  - Status history for periodic state over time.
  - Bar chart shows any categorical data.
  - Histogram calculates and shows value distribution in a bar chart.
  - Heatmap visualizes data in two dimensions, used typically for the magnitude of a phenomenon.
  - Pie chart is typically used where proportionality is important.
  - Candlestick is typically for financial data where the focus is price/data movement.
  - Gauge is the traditional rounded visual showing how far a single metric is from a threshold.
  - Trend for datasets that have a sequential, numeric x that is not time.
  - XY Chart provides a way to visualize arbitrary x and y values in a graph.
- Stats & numbers
  - Stat for big stats and optional sparkline.
  - Bar gauge is a horizontal or vertical bar gauge.

- Misc
  - Table is the main and only table visualization.
  - Logs is the main visualization for logs.
  - Node graph for directed graphs or networks.
  - Traces is the main visualization for traces.
  - Flame graph is the main visualization for profiling.
  - Geomap helps you visualize geospatial data.
  - Datagrid allows you to create and manipulate data, and acts as a data source for other panels.
- Widgets
  - Dashboard list can list dashboards.
  - Alert list can list alerts.
  - Text can show markdown and html.
  - News can show RSS feeds.

Get more

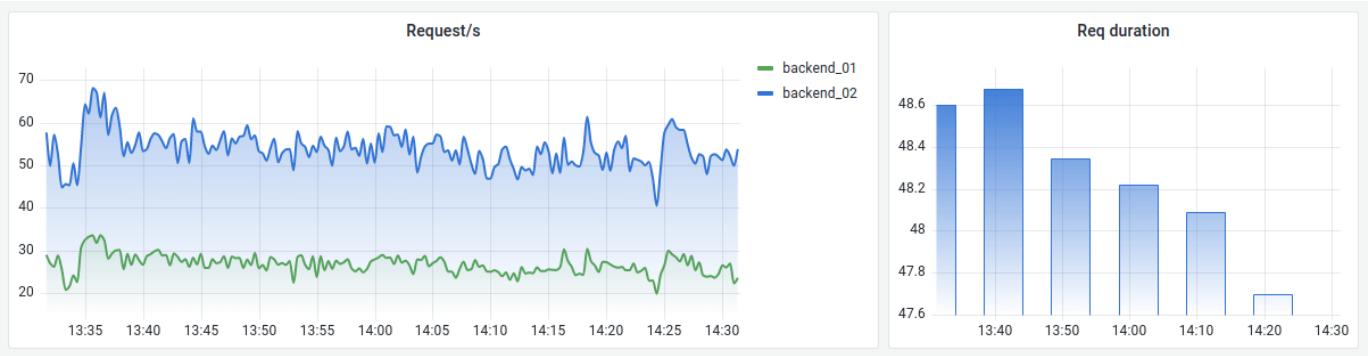
You can add more visualization types by installing panel plugins from the Find plugins with the plugin catalog.

Examples

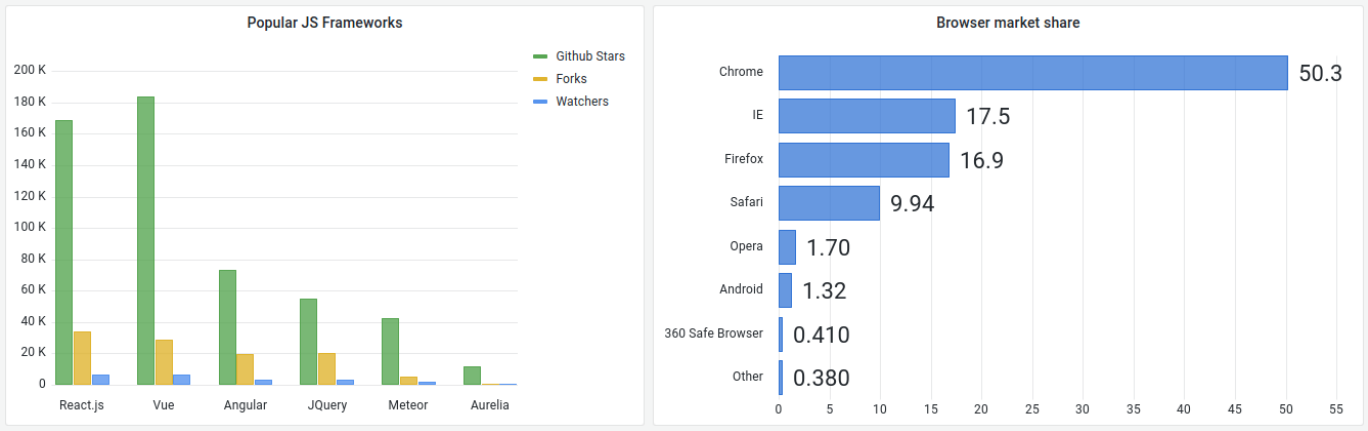
In the following sections you can find visualizations examples.

Graphs

For time based line, area, and bar charts, we recommend the default time series visualization.

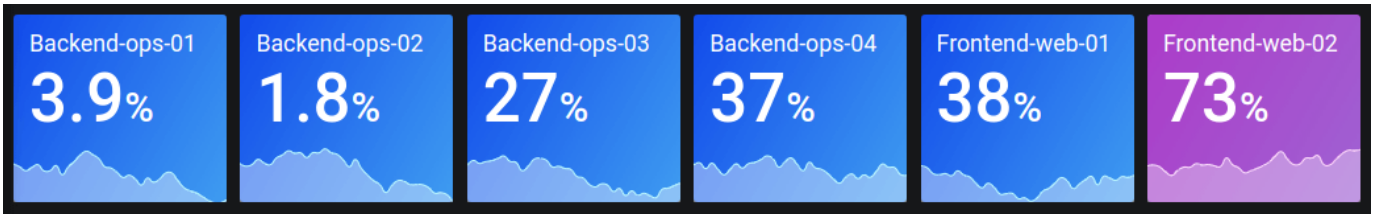


For categorical data, use a bar chart.



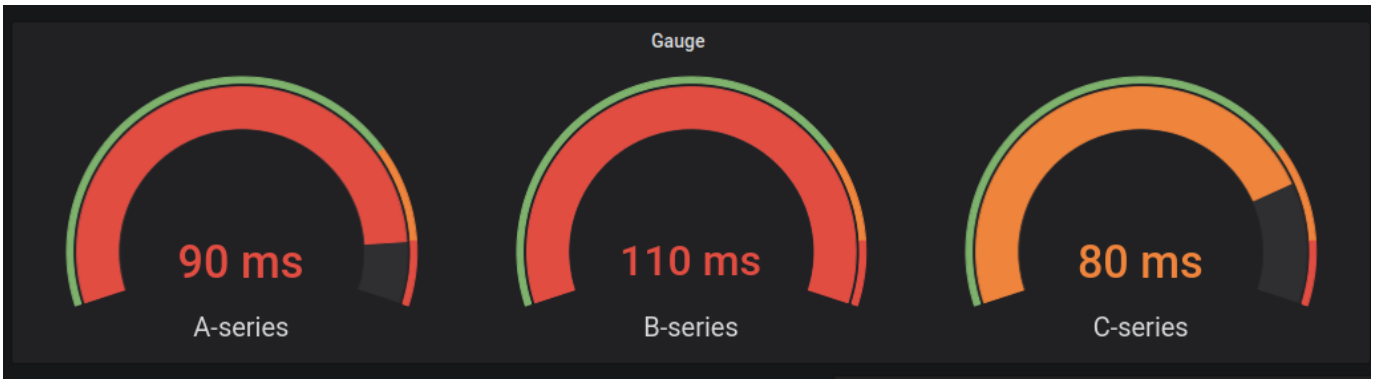
Big numbers & stats

A stat visualization shows one large stat value with an optional graph sparkline. You can control the background or value color using thresholds or color scales.

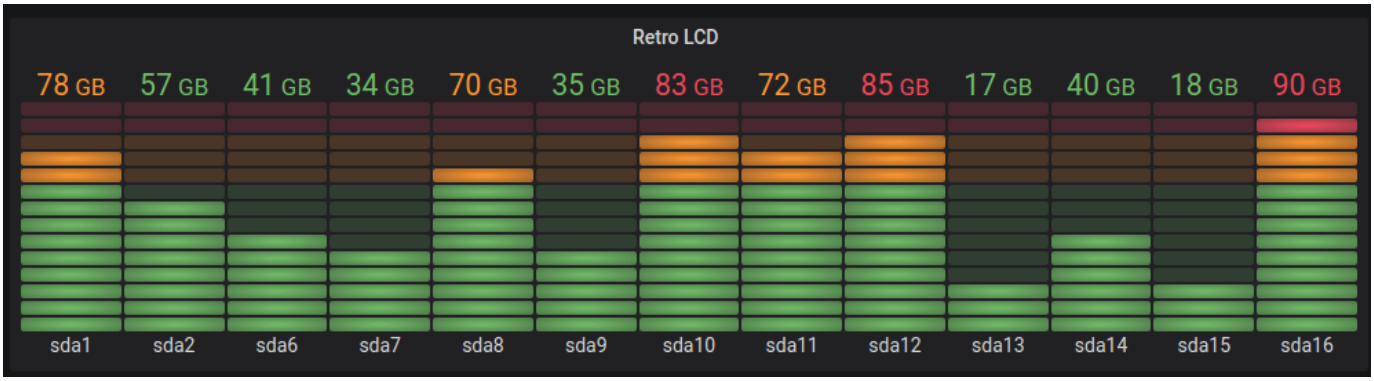


Gauge

If you want to present a value as it relates to a min and max value, you have two options. First a standard radial gauge:

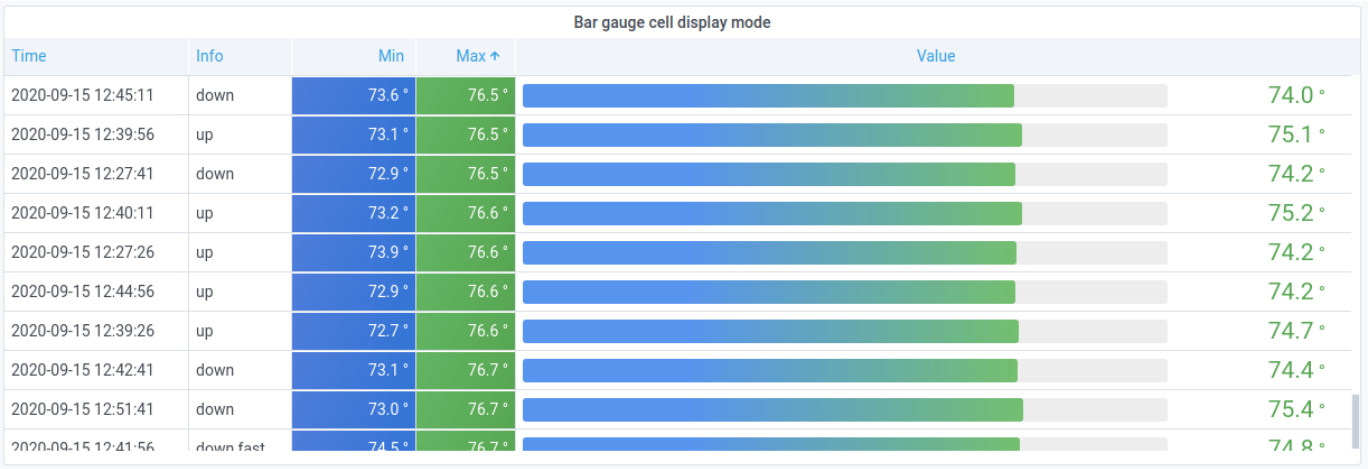


Secondly, Grafana also has a horizontal or vertical bar gauge with three distinct display modes.



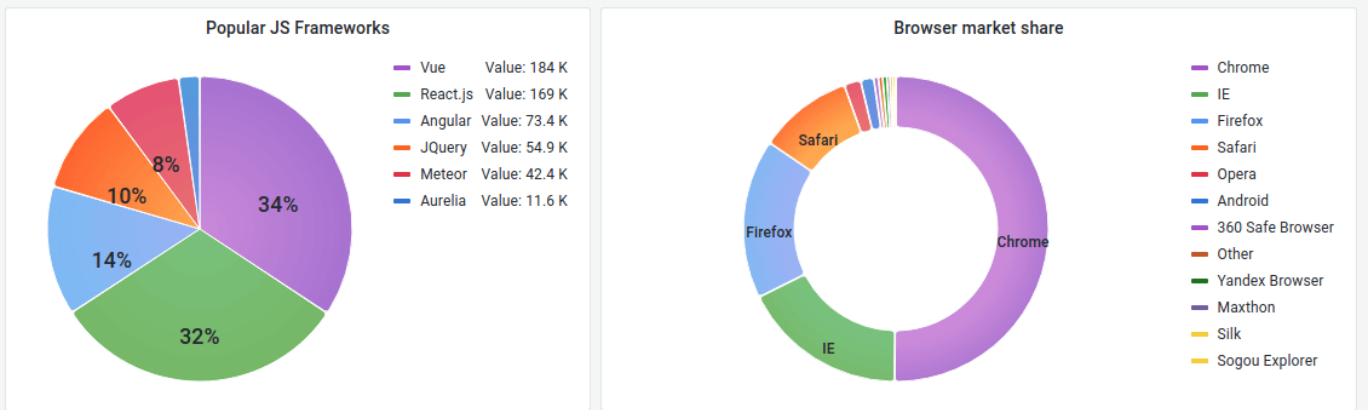
Table

To show data in a table layout, use a table visualization.



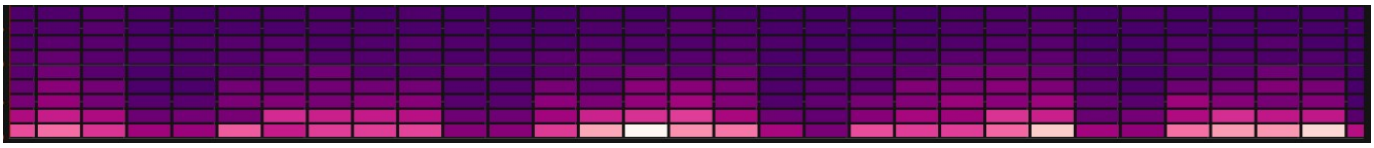
Pie chart

To display reduced series, or values in a series, from one or more queries, as they relate to each other, use a pie chart visualization.



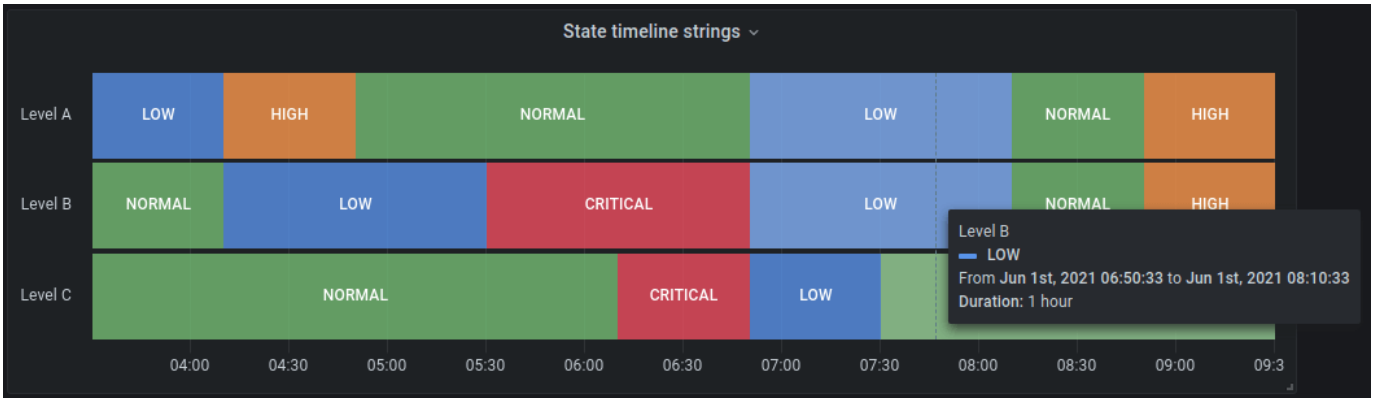
Heatmaps

To show value distribution over time, use a heatmap visualization.



State timeline

A state timeline shows discrete state changes over time. When used with time series, thresholds are used to turn numerical values into discrete state regions.



Alert list

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use alert lists to display your alerts. You can configure the list to show current state. For more information about alerts, see Alerts in Grafana version 10.

Use these settings to refine your visualization.

Options

- **Group mode** – Choose between **Default grouping** to show alert instances grouped by their alert rule, and **Custom grouping** to group alert instances by a custom set of labels.
- **Max Items** – Set the maximum number of alerts to list.
- **Sort order** – Select how to order the alerts displayed.
  - **Alphabetical (asc)** – Alphabetical order
  - **Alphabetical (desc)** – Reverse alphabetical order
  - **Importance** – By importance according to the following values, with 1 being the highest:
    - `alerting` or `firing`: 1
    - `no_data`: 2
    - `pending`: 3
    - ok: 4
    - `paused` or `inactive`: 5
  - **Time (asc)** – Newest active alert instances first.
  - **Time (desc)** – Oldest active alert instances first.
- **Alerts from this dashboard** – Show alerts only from the dashboard that the alert list is in.

Filter

These options allow you to limit alerts shown to only those that match the query, folder, or tags that you choose:

- **Alert name** – Enter an alert name query.
- **Alert instance label** – Filter alert instances using label querying. For example, `{severity="critical", instance=~"cluster-us-.+"}`.
- **Folder** – Select a folder. Only alerts from dashboards in the selected folder will be displayed.
- **Datasource** – Filter alerts from the selected data source.

State filter

Choose which alert states to display in this panel.

- Alerting / Firing

- Pending
- No data
- Normal
- Error

Annotations list

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The Annotations list shows a list of available annotations you can use to view annotated data. Various options are available to filter the list based on tags and on the current dashboard.

Annotation query

The following options control the source query for the list of annotations.

Query Filter

Use the query filter to create a list of annotations from all dashboards in your organization or the current dashboard in which this panel is located. It has the following options:

- All dashboards - List annotations from all dashboards in the current organization.
- This dashboard - Limit the list to the annotations on the current dashboard.

Time Range

Use the time range option to specify whether the list should be limited to the current time range. It has the following options:

- None - no time range limit for the annotations query.

- This dashboard - Limit the list to the time range of the dashboard where the annotation list panel is available.

Tags

Use the tags option to filter the annotations by tags. You can add multiple tags in order to refine the list.

> **Note**
>
> Optionally, leave the tag list empty and filter on the fly by selecting tags that are listed as part of the results on the panel itself.

Limit

Use the limit option to limit the number of results returned.

Display

These options control additional metadata included in the annotations panel display.

Show user

Use this option to show or hide which user created the annotation.

Show time

Use this option to show or hide the annotation creation time.

Show Tags

Use this option to show or hide the tags associated with an annotation. *NB*: You can use the tags to live-filter the annotation list on the visualization itself.

Link behavior

Link target

Use this option to chose how to view the annotated data. It has the following options.

- Panel - This option will take you directly to a full-screen view of the panel with the corresponding annotation

- Dashboard - This option will focus the annotation in the context of a complete dashboard

Time before

Use this option to set the time range before the annotation. Use duration string values like “1h” = 1 hour, “10m” = 10 minutes, etc.

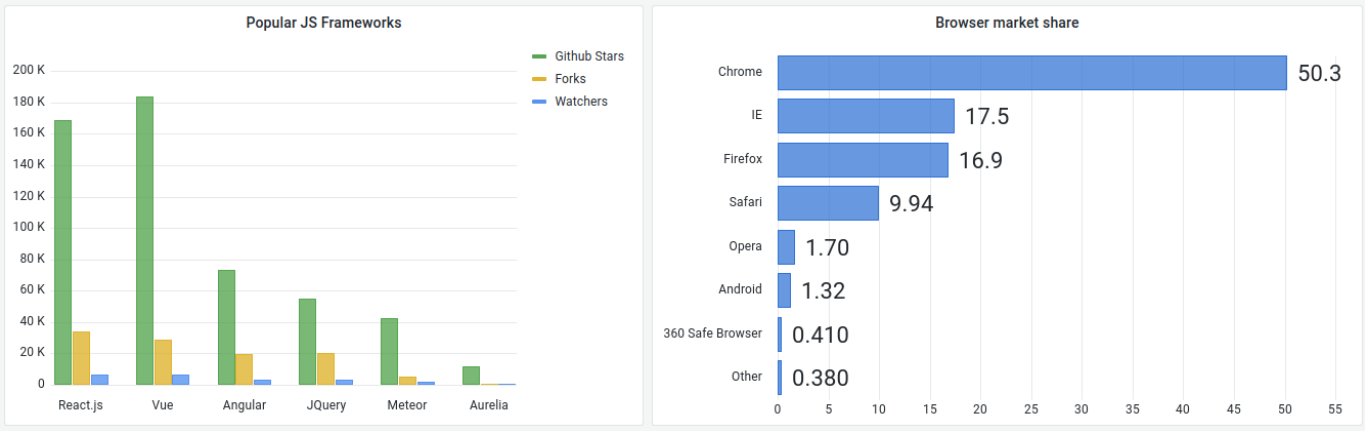
Time after

Use this option to set the time range after the annotation.

Bar chart

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Bar charts allow you to graph categorical data.



Supported data formats

Only one data frame is supported and it needs to have at least one string field that will be used as the category for an X or Y axis and one or more numerical fields. The following is an example of data formats:

Browser	Market share
Chrome	50
Internet Explorer	17.5

If you have more than one numerical field, the panel shows grouped bars.

Visualizing time series or multiple result sets

If you have multiple time series or tables, you first need to join them using a join, or reduce transform. For example, if you have multiple time series and you want to compare their last and max value, add the **Reduce** transform and specify **Max** and **Last** as options under **Calculations**.

Bar chart options

Use these options to refine your visualization.

Orientation

- **Auto** – Grafana decides the bar orientation based on the panel dimensions.
- **Horizontal** – Makes the X axis the category axis.
- **Vertical** – Makes the Y axis the category axis.

Rotate x-axis tick labels

When the graph is vertically oriented, this setting rotates the labels under the bars. This setting is useful when bar chart labels are long and overlap.

X-axis tick label maximum length

Sets the maximum length of bar chart labels. Labels longer than the maximum length are truncated with ellipses.

Bar labels minimum spacing

Sets the minimum spacing between bar labels.

Show values

Controls whether values are shown on top of or to the left of bars.

- **Auto** – Values are shown if there is space.
- **Always** – Always show values.
- **Never** – Never show values.

Stacking

Controls bar chart stacking.

- **Off** – Bars will not be stacked.
- **Normal** – Bars will be stacked on top of each other.
- **Percent** – Bars will be stacked on top of each other, and the height of each bar is the percentage of the total height of the stack.

Group width

Controls the width of groups.

- `0 = Minimum width`
- `1 = Maximum width`

Bar width

Controls the width of bars.

- `0 = Minimum width`
- `1 = Maximum width`

Bar radius

Controls the radius of the bars.

- `0 = Minimum radius`
- `0.5 = Maximum radius`

Highlight full area on hover

Controls if the entire surrounding area of the bar is highlighted when you hover over the bar.

Line width

Controls line width of the bars.

Fill opacity

Controls the fill opacity of the bars.

Gradient mode

Sets the mode of the gradient fill. Fill gradient is based on the line color. To change the color, use the standard color scheme field option.

Gradient appearance is influenced by the **Fill opacity** setting.

- **None** – no gradient fill. This is the default setting.
- **Opacity** – Transparency of the gradient is calculated based on the values on the y-axis. Opacity of the fill is increasing with the values on the Y-axis.
- **Hue** – Gradient color is generated based on the hue of the line color.

Tooltip mode

When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.

- **Single** – The hover tooltip shows only a single series, the one that you are hovering over on the visualization.
- **All** – The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden** – Do not display the tooltip when you interact with the visualization.

> **Note**
>
> You can use an override to hide individual series from the tooltip.

Text size

Enter a value to change the size of the text on your bar chart.

Legend options

Legend mode

Use these settings to define how the legend appears in your visualization. For more information, see Configure a legend.

- **List** – Displays the legend as a list. This is a default display mode of the legend.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement

Choose where to place the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend values

Choose series data values or standard calculations to show in the legend. You can have more than one. For more information, see Configure a legend.

Axis options

Use the following field settings to refine how your axes display. Some field options will not affect the visualization until you click outside of the field option box you are editing or press Enter.

Placement

Sets the placement of the Y-axis.

- **Auto** – Grafana automatically assigns Y-axis to the series. When there are two or more series with different units, then Grafana assigns the left axis to the first unit and right to the following units.
- **Left** – Display all Y-axes on the left side.
- **Right** – Display all Y-axes on the right side.
- **Hidden** – Hide all Y-axes.

To selectively hide axes, adding field overrides that targets specific fields.

Label

Set a Y-axis text label. If you have more than one Y-axis, then you can assign different labels with an override.

Width

Set a fixed width of the axis. By default, Grafana dynamically calculates the width of an axis.

By setting the width of the axis, data with different axes types can share the same display proportions. This makes it easier to compare more than one graph's worth of data because the axes are not shifted or stretched within visual proximity of each other.

Soft min and soft max

Set a soft min or soft max option for better control of Y-axis limits. By default, Grafana sets the range for the Y-axis automatically based on the dataset.

Soft min and soft max settings can prevent blips from turning into mountains when the data is mostly flat, and hard min or max derived from standard min and max field options can prevent intermittent spikes from flattening useful detail by clipping the spikes past a defined point.

You can set standard min/max options to define hard limits of the Y-axis. For more information, see Configure standard options.

Display multiple y-axes

In some cases, you may want to display multiple y-axes. For example, if you have a dataset showing both temperature and humidity over time, you may want to show two y-axes with different units for these two series.

You can do this by adding field overrides. Follow the steps as many times as required to add as many y-axes as you need.

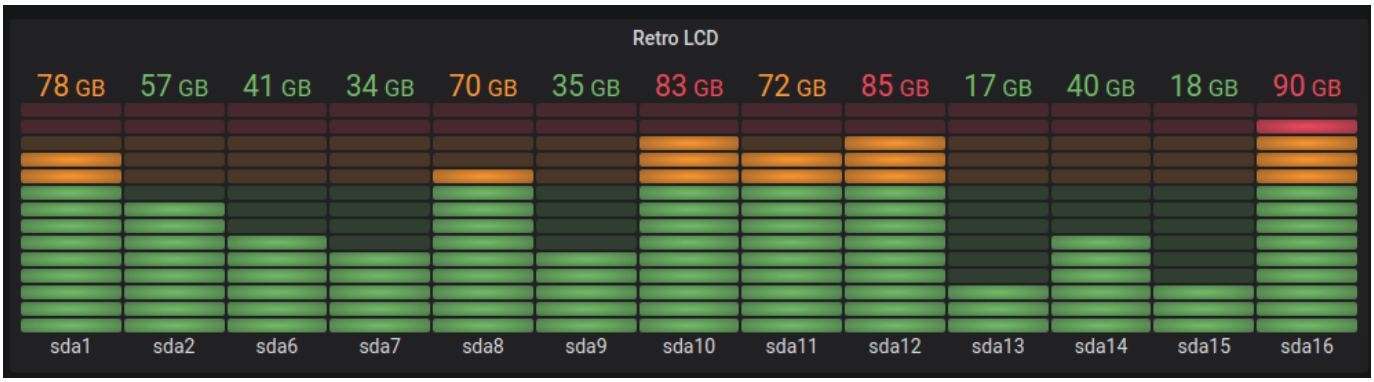
Bar gauge

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Bar gauges simplify your data by reducing every field to a single value. You choose how Grafana calculates the reduction.

This panel can show one or more bar gauges depending on how many series, rows, or columns your query returns.



Value options

Use the following options to refine how your visualization displays the value:

Show

Choose how Grafana displays your data.

Calculate

Show a calculated value based on all rows.

- **Calculation** – Select a reducer function that Grafana will use to reduce many fields to a single value. For a list of available calculations, refer to Calculation types.
- **Fields** – Select the fields display in the panel.

All values

Show a separate stat for every row. If you select this option, then you can also limit the number of rows to display.

- **Limit** – The maximum number of rows to display. Default is 5,000.
- **Fields** – Select the fields display in the panel.

Bar gauge options

Adjust how the bar gauge is displayed.

Orientation

Choose a stacking direction.

- **Auto** – Grafana selects what it thinks is the best orientation.
- **Horizontal** – Bars stretch horizontally, left to right.
- **Vertical** – Bars stretch vertically, bottom to top.

Display mode

Choose a display mode.

- **Gradient** – Threshold levels define a gradient.
- **Retro LCD** – The gauge is split into small cells that are lit or unlit.
- **Basic** – Single color based on the matching threshold.

Value display

Choose a value display mode.

- **Value color** – Value color is determined by value.
- **Text color** – Value color is the default text color.
- **Hidden** – Values are hidden.

Name placement

Choose a name placement mode.

> **Note**
>
> This option only applies when the orientation of the bar gauge is horizontal. When the bar gauge is in the vertical orientation, names are always placed at the bottom of each bar gauge.

- **Auto** – Grafana determines the best placement.
- **Top** – Names are placed on the top of each bar gauge.
- **Left** – Names are placed to the left of each bar gauge.

Show unfilled area

Select this if you want to render the unfilled region of the bars as dark gray. Not applicable to Retro LCD display mode.

Bar size

Choose a bar size mode.

- **Auto** – Grafana determines the best bar gauge size.
- **Manual** – Manually configure the bar gauge size.

Min width

Limit the minimum width of the bar column when the gauge is oriented vertically.

Automatically show x-axis scrollbar when there is a large amount of data.

> **Note**
>
> This option only applies when bar size is set to manual.

Min height

Limit the minimum height of the bar row when the gauge is oriented horizontally.

Automatically show y-axis scrollbar when there is a large amount of data.

> **Note**
>
> This option only applies when bar size is set to manual.

Max height

Limit the maximum height of the bar row when the gauge is oriented horizontally.

Automatically show y-axis scrollbar when there is a large amount of data.

> **Note**
>
> This option only applies when bar size is set to manual.

Candlestick

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The Candlestick visualization allows you to visualize data that includes a number of consistent dimensions focused on price movement. The Candlestick panel includes an Open-High-Low-Close (OHLC) mode, as well as support for additional dimensions based on time series data.



Candlestick visualizations build upon the foundation of the Time series and includes many common configuration settings.

Mode

The mode options allow you to toggle which dimensions are used for the visualization.

- **Candles** – Limit the panel dimensions to the open, high, low, and close dimensions used by candlestick visualizations.
- **Volume** – Limit the panel dimension to the volume dimension.
- **Both** – The default behavior for the candlestick panel. It includes both candlestick and volume visualizations.

Candle style

- **Candles** – The default display style, creates candle-style visualizations between the open and close dimensions.
- **OHLC Bars** – Display the four core dimensions open, high, low, and close values.

Color strategy

- **Since Open** – The default behavior. This mode will utilize the *Up* color (below) if the intra-period price movement is positive. In other words, if the value on close is greater or equal to the value on open, the *Up* color is used.
- **Since Prior Close** – An alternative display method where the color of the candle is based on the inter-period price movement or change in value. In other words, if the value on open is greater than the previous value on close, the *Up* color is used. If the value on open is lower than the previous value on close, the *Down* color is used. *This option also triggers the hollow candlestick visualization mode*. Hollow candlesticks indicate that the intra-period movement is positive (value is higher on close than on open), filled candlesticks indicate the intra-period change is negative (value is lower on close than on open). To learn more, see the explanation of the differences.

Up & down colors

The **Up color** and **Down color** options select which colors are used when the price movement is up or down. The *Color strategy* above will determine if intra-period or inter-period price movement is used to select the candle or OHLC bar color.

Open, high, low, close

The candlestick panel will attempt to map fields to the appropriate dimension.

- **Open** corresponds to the starting value of the given period.
- **High** corresponds to the highest value of the given period.
- **Low** corresponds to the lowest value of the given period.
- **Close** corresponds to the final (end) value of the given period.
- **Volume** corresponds to the sample count in the given period. (e.g. number of trades)

> **Note**
>
> The candlestick legend doesn't display these values.

To properly map these dimensions, the query results table from your data must include *at least* the following columns.

- `timestamp`
- `open`
- `high`
- `low`
- `close`

If your data can't be mapped to these dimensions for some reason (for example, because the column names aren't the same), you can map them manually using the **Open**, **High**, **Low**, and **Close** fields under the **Candlestick** options in the panel editor.

Additional fields

Additional fields

The candlestick panel is based on the time series visualization. It can visualize additional data dimensions beyond open, high, low, close, and volume. The **Include** and **Ignore** options allow it to visualize other included data such as simple moving averages, Bollinger bands and more, using the same styles and configurations available in the Time series.

Canvas

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Canvases combine the power of Grafana with the flexibility of custom elements. Canvases are extensible form-built panels that allow you to explicitly place elements within static and dynamic layouts. This empowers you to design custom visualizations and overlay data in ways that aren’t possible with standard Grafana panels, all within Grafana’s UI. If you’ve used popular UI and web design tools, then designing Canvas panels will feel very familiar.

Elements

You can add these elements on your canvas. Adding multiple, and different kinds, of elements lets you customize a visualization in ways that are not possible with any other visualization.

Metric value

The metric value element lets you easily select the data you want to display on canvas. This element has a unique *edit* mode that can be triggered either through the context menu **Edit** option or by double-clicking panel. When in edit mode you can select which field data that you want to display.

Text

The text element lets you easily add text to the canvas. The element also supports an editing mode, triggered via either double-clicking or the edit menu option in the context menu.

Ellipse

The ellipse element lets you add a basic ellipse to the canvas. An ellipse element can display text (both fixed and field data) and its background color can be changed based on data thresholds.

Rectangle

The rectangle element lets you to add a basic rectangle to the canvas. A rectangle elements can display text (both fixed and field data) and its background color can be changed based on data thresholds.

Icon

The icon element lets you add a supported icon to the canvas. Icons can have their color set based on thresholds or value mappings.

Server

The server element lets you easily represent a single server, a stack of servers, a database, or a terminal. Server elements support status color, bulb color, and a bulb blink rate, all configurable by fixed or field values.

Button

The button element lets you add a basic button to the canvas. Button elements support triggering basic, unauthenticated API calls. API settings are found in the button element editor. You can also pass template variables in the API editor.

> **Note**
>
> Choosing a button will only trigger an API call when inline editing is disabled. See Canvas editing.

Connections

When building a canvas, you can connect elements together to create more complex visualizations. You can create connections by dragging from the connection anchor of one element to the connection anchor of another element. You can also create connections to the background of the canvas. Connection anchors are displayed when you hover over an element and inline editing is turned on. To remove a connection, select the connection and then press `Delete` or `Backspace`.

You can set both the size and color of connections based on fixed or field values. To do so, enter into panel edit mode, select the connection, and modify the connection's properties in the panel editor.

Canvas editing

Inline editor

You can edit your canvas inline while in the context of dashboard mode.

Pan and zoom

You and turn on panning and zooming in a canvas. This allows you to both create and navigate more complex designs.

> **Note**
>
> Pan and zoom is currently in preview by Grafana Labs. Support is limited, and breaking changes might occur before general availability.

Context menu

The context menu lets you perform common tasks quickly and efficiently. Supported functionality includes opening and closing the inline editor, duplicating an element, deleting an element, and more.

The context menu is triggered by a right click action (or equivalent) over the panel or a given canvas element.

When right clicking *the panel*, you are able to set a background image and easily add elements to the canvas.

When right clicking *an element*, you are able to edit, delete, and duplicate the element, or modify the element's layer positioning.

Canvas Options

Inline editing

The inline editing toggle lets you lock or unlock the canvas panel. When turned off, the canvas panel becomes *locked*, freezing elements in place and preventing unintended modifications.

Data links

Canvases support data links. You can create a data link for a metric-value element and display it for all elements that use the field name by following these steps.

To create a data link for an element

1. Set an element to be tied to a field value.
2. Turn off the inline editing toggle.
3. Create an override for **Fields with name** and select the element field name from the list.
4. Choose the **+ Add override property** button.
5. Select **Datalinks > Datalinks** from the list.
6. Choose **+ Add link**, add a title and URL for the data link.
7. Hover over the element to display the data link tooltip.
8. Choose the element to be able to open the data link.

If multiple elements use the same field name, and you want to control which elements display the data link, you can create a unique field name using the Add field from calculation transform. The alias you create in the transform will appear as a field you can use with an element.

Dashboard list

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Dashboard lists allow you to display dynamic links to other dashboards. The list can be configured to use starred dashboards, recently viewed dashboards, a search query, and dashboard tags.

On each dashboard load, this panel queries the dashboard list, always providing the most up-to-date results.

Options

Use these options to refine your visualization.

- **Include current time range** – Select this option to propagate the time range of the current dashboard to the dashboard links. When the user selects a link, the linked dashboard opens with the indicated time range already set.
- **Include current template variable values** – Select this option to include template variables currently used as query parameters in a link. When the user selects a link, any matching templates in the linked dashboard are set to the values from the list. For more information, see Dashboard URL variables.
- **Starred** – Display starred dashboards in alphabetical order.
- **Recently viewed** – Display recently viewed dashboards in alphabetical order.
- **Search** – Display dashboards by search query or tags. You must enter at least one value in **Query** or **Tags**. For the **Query** and **Tags** fields, variable interpolation is supported, for example, `$my_var` or `${my_var}`.
- **Show headings** – The chosen list selection (Starred, Recently viewed, Search) is shown as a heading.
- **Max items** – Sets the maximum number of items to list per section. For example, if you left this at the default value of 10 and displayed Starred and Recently viewed dashboards, then the panel would display up to 20 total dashboards, ten in each section.

Search

These options only apply if the **Search** option is selected.

- **Query** – Enter the query you want to search by. Queries are case-insensitive, and partial values are accepted.
- **Folder** – Select the dashboard folders that you want to display.
- **Tags** – Here is where you enter the tags you want to search by. Existing tags will not appear as you type, and they *are* case sensitive.

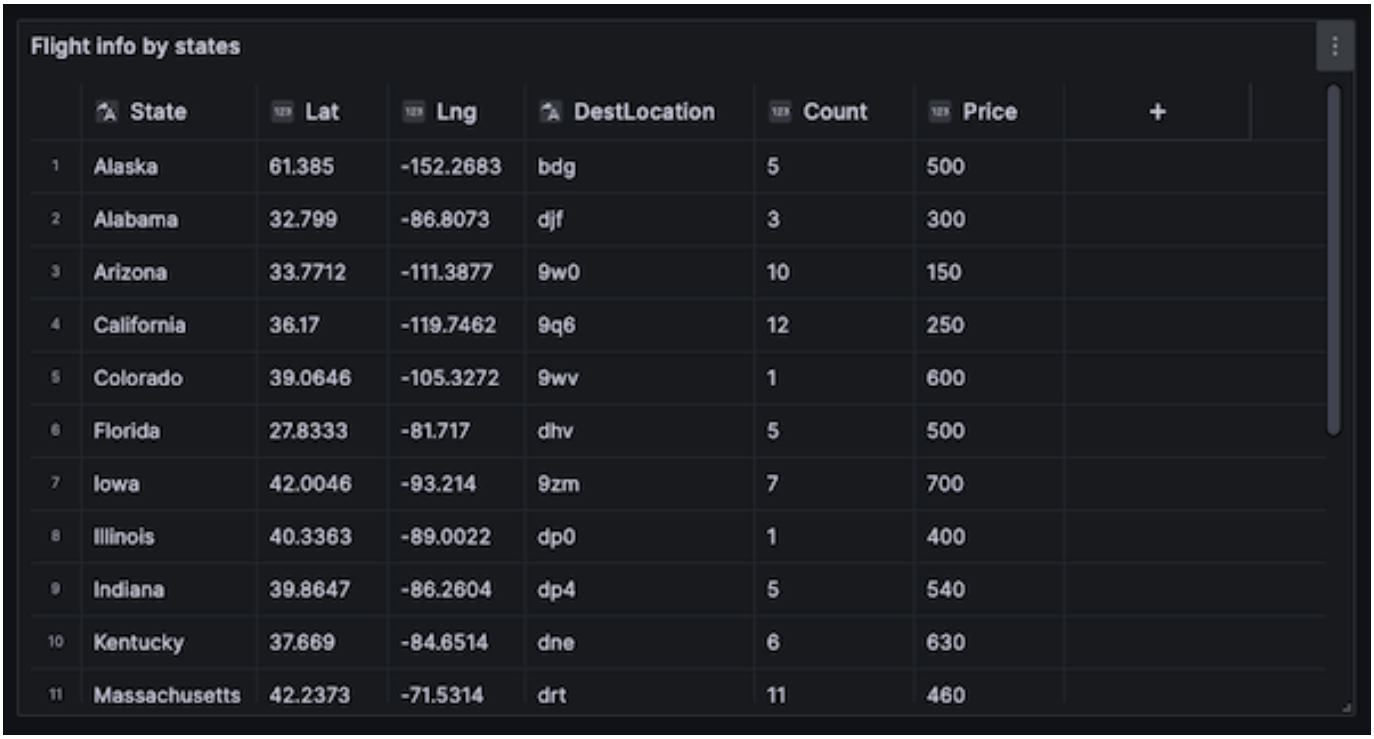
> **Note**
>
> When multiple tags and strings appear, the dashboard list displays those matching *all* conditions.

Datagrid

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

> **Note**
>
> The data grid visualization is currently in preview by Grafana Labs. Support is limited, and breaking changes might occur before general availability.

Datagrids offer you the ability to create, edit, and fine-tune data within Grafana. As such, this panel can act as a data source for other panels inside a dashboard.



	State	Lat	Lng	DestLocation	Count	Price
1	Alaska	61.385	-152.2683	bdg	5	500
2	Alabama	32.799	-86.8073	djf	3	300
3	Arizona	33.7712	-111.3877	9w0	10	150
4	California	36.17	-119.7462	9q6	12	250
5	Colorado	39.0646	-105.3272	9wv	1	600
6	Florida	27.8333	-81.717	dhv	5	500
7	Iowa	42.0046	-93.214	9zm	7	700
8	Illinois	40.3363	-89.0022	dp0	1	400
9	Indiana	39.8647	-86.2604	dp4	5	540
10	Kentucky	37.669	-84.6514	dne	6	630
11	Massachusetts	42.2373	-71.5314	drt	11	460

Through it, you can manipulate data queried from any data source, you can start from a blank slate, or you can pull data from a dragged and dropped file. You can then use the panel as a simple tabular visualization, or you can modify the data—and even remove it altogether—to create a blank slate.

Editing the dataset changes the data source to use the built-in **Grafana** data source, replacing the old data source settings and related queries, while also copying the current dataset into the dashboard model.

You can then use the panel as a data source for other panels, by using the built-in **Dashboard** data source to pull the datagrid data. This provides an interactive dashboard experience, where you can modify the data and see the changes reflected in other panels.

For more information about the **Grafana** and **Dashboard** data sources, see Special data sources.

Context menu

To provide a more streamlined experience, the datagrid has a context menu that can be accessed by right-clicking on a cell, column header, or row selector. Depending on the state of your datagrid, the context menu offers different options including the following.

- Delete or clear all rows and columns.
- Remove all existing data (rendering your datagrid blank).

- Trigger search functionality, which allows you to find keywords within the dataset.

Deleting a row or column will remove the data from the datagrid, while clearing a row or column will only remove the data from the cells, leaving the row or column intact.

Header menu

You can also access a header menu by choosing the dropdown icon next to the header title. From here, you can not only delete or clear a column, but also rename it, freeze it, or convert the field type of the column.

Selecting series

If there are multiple series, you can set the datagrid to display the preferred dataset using the **Select series** dropdown in the panel options.

Using datagrids

Datagrids offer various ways of interacting with your data. You can edit, move, clear, and remove rows and columns; use the built-in search functionality to find specific data; and convert field types or freeze horizontal scroll on a specific column.

Adding data

You can add data to a datagrid by creating a new column or row.

To add a new column

1. In an existing panel, choose the **+** button in the table header after the last column.
2. Add a name for the new column.
3. Select anywhere outside the field or press `Enter` to save the column.

Now you can add data in each cell.

To add a new row, choose a **+** button after the last row. The button is present in each cell after the last row, and choosing it triggers the creation of a new row while also activating the cell that you chose.

Editing data

You can move columns and rows as needed.

To move a column

1. Press and hold the header of the column that needs to be moved.
2. Drag the column to the desired location.
3. Release the column to finalize the move.

To move a row, select and hold the row selector from the number column situated on the far left side of the grid, and drag it to the desired location. Release the row to finalize the move.

Selecting multiple cells

You can select multiple cells by choosing a single cell, and dragging across others. This select can be used to copy the data from the selected cells or to delete them using the `Delete` key.

Deleting or clearing multiple rows or columns

To delete or clear multiple rows, you can do the following.

To delete or clear multiple rows or columns

1. Hover over the number column (to the left of the first column in the grid) to display row checkbox.
2. Select the checkboxes for the rows you want to work with. To select multiple consecutive rows, press and hold the `Shift` key while clicking the first and last row. To select non-consecutive rows, press and hold the `Ctrl` (or `Cmd`) key while clicking the desired rows.
3. Right-click (or equiavalent) to access the context menu.
4. Select **Delete rows** or **Clear rows**.

The same rules apply to columns by clicking the column headers.

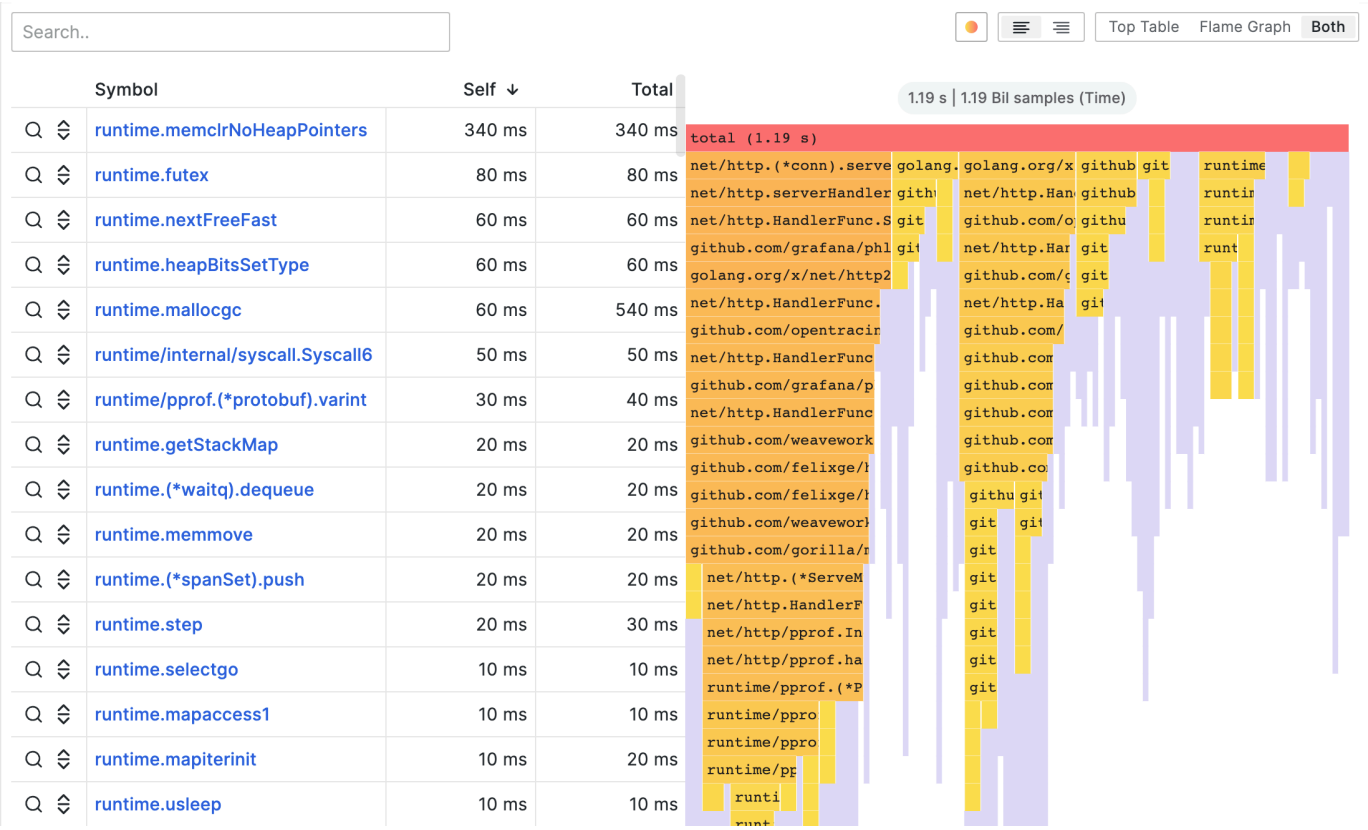
To delete all rows, use the **Select all** checkbox at the top left corner of the datagrid. This select all rows and allows you to delete them using the context menu.

Flame graph

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Flame graphs let you visualize profiling data. Using this visualization, the profile can be represented as a flame graph, table, or both.



Flame graph mode

A flame graph takes advantage of the hierarchical nature of profiling data. It condenses data into a format that allows you to easily see which code paths are consuming the most system resources, such as CPU time, allocated objects, or space when measuring memory. Each block in the flame graph represents a function call in a stack and its width represents its value.

Grayed-out sections are a set of functions that represent a relatively small value and they are collapsed together into one section for performance reasons.

You can hover over a specific function to view a tooltip that shows you additional data about that function, like the function's value, percentage of total value, and the number of samples with that function.

Drop-down actions

You can click a function to show a drop-down menu with additional actions:

- **Focus block** – When you choose **Focus block**, the block, or function, is set to 100% of the flame graph's width and all its child functions are shown with their widths updated relative to the width of the parent function. This makes it easier to drill down into smaller parts ofthe flame graph.
- **Copy function name** – When you choose **Copy function name**, the full name of the function that the block represents is copied.
- **Sandwich view** – The sandwich view allows you to show the context of the clicked function. It shows all the function's callers on the top and all the callees at the bottom. This shows the aggregated context of the function so if the function exists in multiple places in the flame graph, all the contexts are shown and aggregated in the sandwich view.

Status bar

The status bar shows metadata about the flame graph and currently applied modifications, like what part of the graph is in focus or what function is shown in sandwich view. Click the **X** in the status bar pill to remove that modification.

Toolbar

Search

You can use the search field to find functions with a particular name. All the functions in the flame graph that match the search will remain colored while the rest of the functions are grayed-out.

Color schema picker

You can switch between coloring functions by their value or by their package name to visually tie functions from the same package together.

Text align

Align text either to the left or to the right to show more important parts of the function name when it does not fit into the block.

Visualization picker

You can choose to show only the flame graph, only table, or both at the same time.

Top table mode

The top table shows the functions from the profile in table format. The table has three columns: symbols, self, and total. The table is sorted by self time by default, but can be reordered by total time or symbol name by clicking the column headers. Each row represents aggregated values for the given function if the function appears in multiple places in the profile.

There are also action buttons on the left for each row. The first button searches for the function name while second button shows the sandwich view of the function.

Data API

In order to render the flame graph, you must format the data frame data using a nested set model.

A nested set model ensures each item of the flame graph is encoded just by its nesting level as an integer value, its metadata, and by its order in the data frame. This means that the order of items is significant and needs to be correct. The ordering is a depth-first traversal of the items in the flame graph which recreates the graph without needing variable-length values in the data frame like in a children's array.

Required fields:

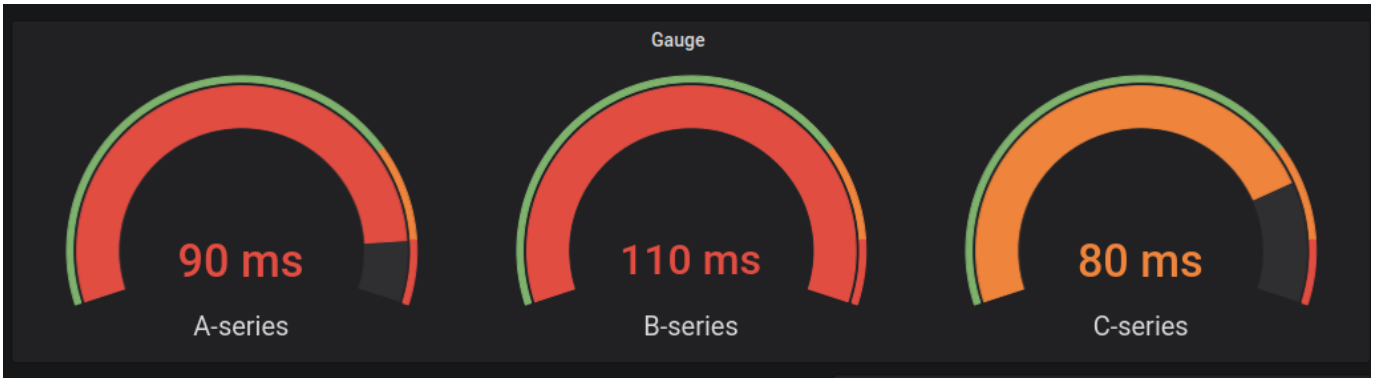
Field name	Type	Description
level	number	The nesting level of the item. In other words how many items are between this item and the top item of the flame graph.
value	number	The absolute or cumulativ e value of the item. This translates to the width of the item in the graph.
label	string	Label to be shown for the particular item.

Field name	Type	Description
self	number	Self value which is usually the cumulative value of the item minus the sum of cumulativ e values of its immediate children.

Gauge

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Gauges are single-value visualizations that can repeat a gauge for every series, column or row.



Value options

Use the following options to refine how your visualization displays the value:

Show

Choose how Grafana displays your data.

Calculate

Show a calculated value based on all rows.

- **Calculation** – Select a reducer function that Grafana will use to reduce many fields to a single value. For a list of available calculations, refer to Calculation types.
- **Fields** – Select the fields to display in the panel.

All values

Show a separate stat for every row. If you select this option, then you can also limit the number of rows to display.

- **Limit** – The maximum number of rows to display. Default is 5,000.
- **Fields** – Select the fields to display in the panel.

Gauge

Adjust how the gauge is displayed.

Orientation

Choose a stacking direction.

- **Auto** – Gauges display in rows and columns.
- **Horizontal** – Gauges display top to bottom.
- **Vertical** – Gauges display left to right.

Show threshold labels

Controls if threshold values are shown.

Show threshold markers

Controls if a threshold band is shown outside the inner gauge value band.

Gauge size

Choose a guage size mode

- **Auto** – Grafana determines the best gauge size.
- **Manual** – Manually configure the gauge size.

Min width

Set the minimum width of vertically-oriented gauges.

If you set a minimum width, the x-axis scrollbar is automatically displayed when there is a large amount of data.

> **Note**
>
> This option only applies when gauge size is set to manual.

Min height

Se the minimum height of horizontally-oriented gauges.

If you set a minimum height, the y-axis scrollbar is automatically displayed when there is a large amount of data.

> **Note**
>
> This option only applies when gauge size is set to manual.

Neutral

Set the starting value from which every gauge will be filled.

Text size

Adjust the sizes of the gauge text.

- **Title** – Enter a numeric value for the gauge title size.
- **Value** – Enter a numeric value for the gauge value size.

Geomap

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Geomaps allow you to view and customize the world map using geospatial data. You can configure various overlay styles and map view settings to easily focus on the important location-based characteristics of the data.

Note

You can add your own geospatial data on top of basemap layers provides by AWS. The basemap layers must all come from https://tiles.maps.search-services.aws.a2z.com.



Map View

The map view controls the initial view of the map when the dashboard loads.

Initial View

The initial view configures how the GeoMap panel renders when the panel is first loaded.

- **View** sets the center for the map when the panel first loads.
  - **Fit to data** fits the map view based on the data extents of Map layers and updates when data changes.

    - **Data** option allows selection of extent based on data from *All layers*, a single *Layer*, or the *Last value* from a selected layer.
    - **Layer** can be selected if fitting data from a single *Layer* or the *Last value* of a layer.
    - **Padding** sets padding in relative percent beyond data extent (not available when looking at *Last value* only).
    - **Max Zoom** sets the maximum zoom level when fitting data.
- **Coordinates** sets the map view based on:
    - **Latitude**
    - **Longitude**
- Default views are also available including:
    - **(0°, 0°)**
    - **North America**
    - **South America**
    - **Europe**
    - **Africa**
    - **West Asia**
    - **South Asia**
    - **South-East Asia**
    - **East Asia**
    - **Australia**
    - **Oceania**
- **Zoom** sets the initial zoom level.

Map layers

Geomaps support showing multiple layers. Each layer determines how you visualize geospatial data on top of the base map.

Types

There are three map layer types to choose from in the Geomap visualization.

- Markers layer renders a marker at each data point.
- Heatmap layer visualizes a heatmap of the data.

- GeoJSON layer renders static data from a GeoJSON file.
- Night / Day layer (Alpha) renders a night or day region.
- Route layer (preview) render data points as a route.
- Photos layer (preview) renders a photo at each data point.
- Network layer (preview) visualizes a network graph from the data.

There are also two experimental (or alpha) layer types.

- **Icon at last point (alpha)** renders an icon at the last data point.
- **Dynamic GeoJSON (alpha)** styles a GeoJSON file based on query results.

> **Note**
>
> Layers marked *preview* or *alpha* in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

Layer Controls

The layer controls allow you to create layers, change their name, reorder and delete layers.

- **Add layer** creates an additional, configurable data layer for the geomap. When you add a layer, you are prompted to select a layer type. You can change the layer type at any point during panel configuration.
- The layer controls allow you to rename, delete, and reorder the layers of the panel.
  - **Edit layer name** (pencil icon) renames the layer.
  - **Trash Bin** deletes the layer.
  - **Reorder** (six dots/grab handle) allows you to change the layer order. Data on higher layers will appear above data on lower layers. The visualization will update the layer order as you drag and drop to help simplify choosing a layer order.

You can add multiple layers of data to a single geomap panel in order to create rich, detailed visualizations.

Location

Geomaps needs a source of geographical data. This data comes from a database query, and there are four mapping options for your data.

- **Auto** automatically searches for location data. Use this option when your query is based on one of the following names for data fields.
  - *geohash*: `geohash`
  - *latitude*: `latitude`, `lat`
  - *longitude*: `longitude`, `lng`, `lon`
  - *lookup*: `lookup`
- **Coords** specifies that your query holds coordinate data. You will getprompted to select numeric data fields for latitude and longitude from your database query.
- **Geohash** specifies that your query holds geohash data. You will be prompted to select a string data field for the geohash from your database query.
- **Lookup** specifies that your query holds location name data that needs to be mapped to a value. You will be prompted to select the lookup field from your database query and a gazetteer. The gazetteer is the directory that is used to map your queried data to a geographical point.

Markers layer

The markers layer allows you to display data points as different marker shapes such as circles, squares, triangles, stars, and more.

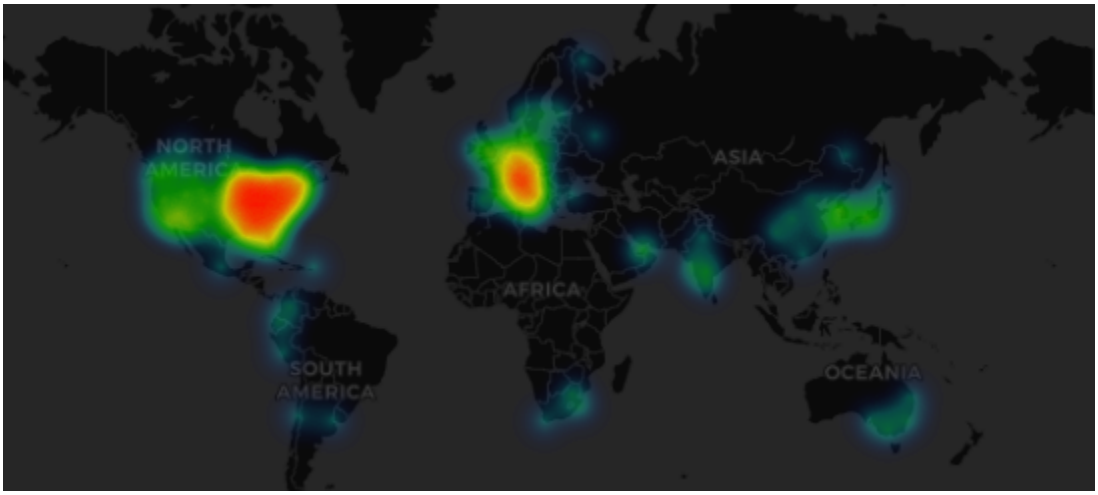
Markers have many customization options.

- **Size** configures the size of the markers. The default is `Fixed size`, which makes all marker sizes the same regardless of the data; however, there is also an option to size the markers based on the data corresponding to selected field. `Min` and `Max` marker size has to be set such that the Marker layer can scale within this range.
- **Symbol** allows you to choose the symbol, icon, or graphic to aid in providing additional visual context to your data. Choose from assets that are included with Grafana such as simple symbols or the Unicon library. You can also specify a URL containing an image asset. The image must be a scalable vector graphic (SVG).
- **Symbol Vertical Align** configures the vertical alignment of the symbol relative to the data point. Note that the symbol's rotation angle is applies first around the data point, then the vertical alignment is applied relative to the rotation of the symbol.

- **Symbol Horizontal Align** configures the horizontal alignment of the symbol relative to the data point. Note that the symbol's rotation angle is applies first around the data point, then the horizontal alignment is applied relative to the rotation of the symbol.
- **Color** configures the color of the markers. The default `Fixed color` sets all markers to a specific color. There is also an option to have conditional colors depending on the selected field data point values and the color scheme set at the **Standard options** section.
- **Fill opacity** configures the transparency of each marker.
- **Rotation angle** configures the rotation angle of each marker. The default is **Fixed value**, which makes all markers rotate to the same angle regardless of the data; however, there is also an option to set the rotation of the markers based on data corresponding to a selected field.
- **Text label** configures a text label for each marker.
- **Show legend** lets you toggle the legend for the layers.
- **Display tooltip** lets you toggle tooltips for the layer.

Heatmap layer

The heatmap layer clusters various data points to visualize locations with different densities.



To add a heatmap layer:

Choose on the dropdown menu under Data Layer and select `Heatmap`.

Similar to `Markers`, you are prompted with options to determine which data points to visualize and how you want to visualize them.

- **Weight values** configure the intensity of the heatmap clusters. **Fixed value** keeps a constant weight value throughout all data points. This value should be in the range of 0-1. Similar to Markers, there is an alternate option in the dropdown to automatically scale the weight values depending on data values.
- **Radius** configures the size of the heatmap clusters.
- **Blur** configures the amount of blur on each cluster.
- **Opacity** configures the opacity of each cluster.
- **Display tooltip** lets you toggle tooltips for the layer.

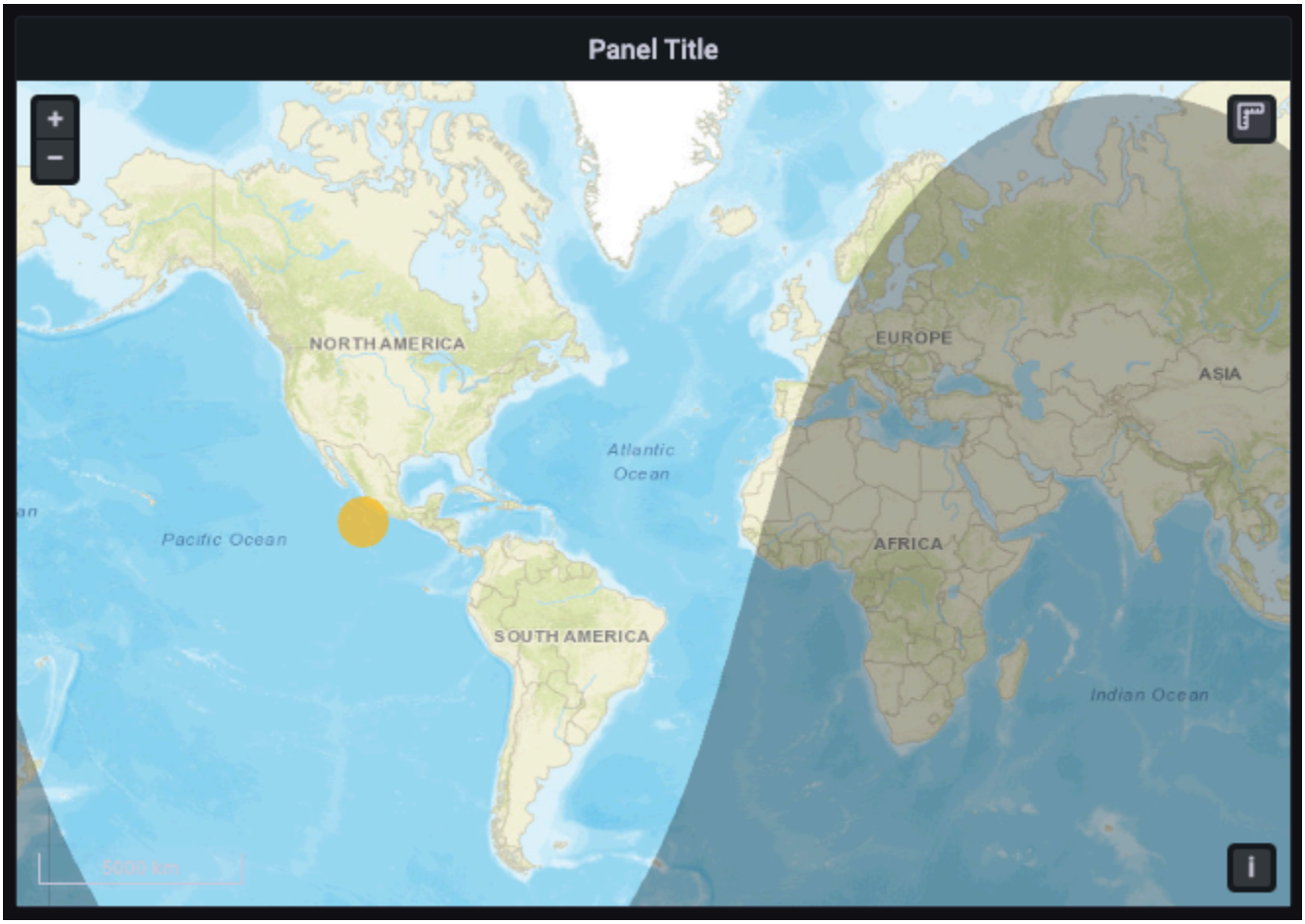
GeoJSON layer

The GeoJSON layer allows you to select and load a static GeoJSON file from the filesystem.

- **GeoJSON URL** provides a choice of GeoJSON files that ship with Grafana.
- **Default Style** controls which styles to apply when no rules above match.
  - **Color** configures the color of the default style
  - **Opacity** configures the default opacity
- **Style Rules** apply styles based on feature properties
  - **Rule** allows you to select a *feature*, *condition*, and *value* from the GeoJSON file in order to define a rule. The trash bin icon can be used to delete the current rule.
  - **Color** configures the color of the style for the current rule
  - **Opacity** configures the transparency level for the current rule.
- **Add style rule** creates additional style rules.
- **Display tooltip** lets you toggle tooltips for the layer.

Night / Day layer (Alpha)

The Night / Day layer displays night and day regions based on the current time range.



Options

- **Show** toggles the time source from panel time range.
- **Night region color** picks the color for the night region.
- **Display sun** toggles sun icon.
- **Opacity** from 0 (transparent) to 1 (opaque).
- **Display tooltip** lets you toggle tooltips for the layer.

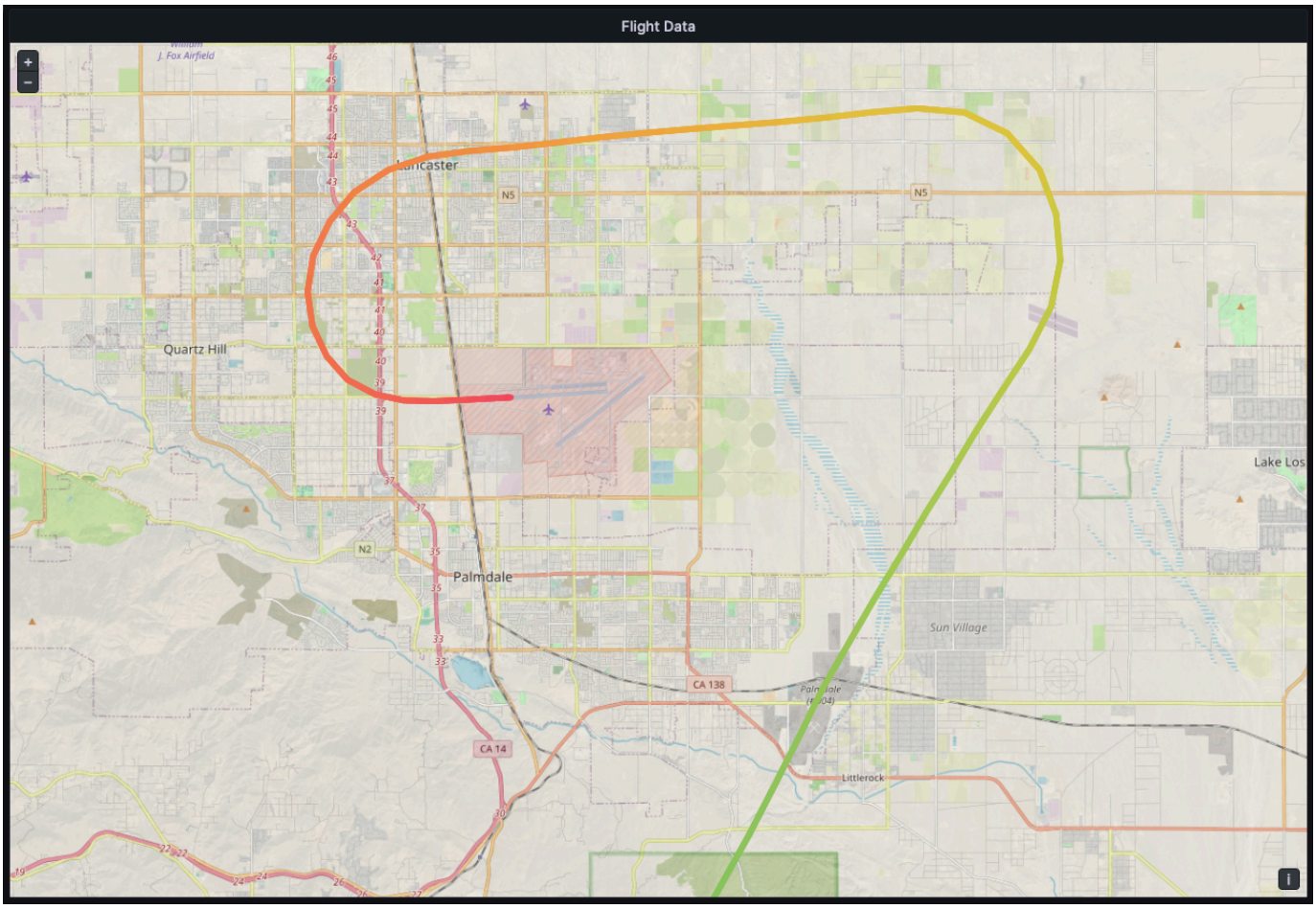
> **Note**
>
> For more information, see Extensions for OpenLayers - DayNight.

Route layer (preview)

The Route layer renders data points as a route.

Note

The Route layer is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.



Options

- **Size** sets the route thickness. Fixed value by default. When field data is selected, you can set the Min and Max range in which field data can scale.
- **Color** sets the route color. Set to fixed color by default, you can also tie the color to field data.
- **Fill opacity** configures the opacity of the route.
- **Text label** configures a text label for each route.
- **Arrow** configures the arrow styling to display along the route, in order of data.
  - **None**

- **Forward**
- **Reverse**

- **Display tooltip** lets you toggle tooltips for the layer.

> **Note**
>
> For more information, see Extensions for OpenLayers - Flow Line Style.

Photos layer (preview)

The Photos layer renders a photo at each data point.

> **Note**
>
> The Photos layer is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

Options

- **Image Source Field** lets you select a string field containing image data as a Base64 encoded image binary (`...`).
- **Kind** sets the frame style around the images. Choose from:
  - **Square**
  - **Circle**
  - **Anchored**
  - **Folio**
- **Crop** toggles whether the images are cropped to fit.
- **Shadow** toggles a box shadow behind the images.
- **Border** sets the border size around images.
- **Border color** sets the border color around images.
- **Radius** sets the overall size of images in pixels.
- **Display tooltip** lets you toggle tooltips for the layer.

> **Note**
>
> For more information, see Extensions for OpenLayers - Image Photo Style.

Network layer (preview)

The Network layer renders a network graph. This layers supports the same data format supported by the node graph visualization, with the addition of geospatial data included in the nodes data. The geospatial data is used to locate and render the nodes on the map.

> **Note**
>
> The Network layer is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.



> **Note**
>
> The Network layer is currently in public preview. Grafana Labs offers limited support, and breaking changes might occur prior to the feature being made generally available.

Options

- **Arrow** sets the arrow direction to display for each edge, with forward meaning source to target. Choose from:
  - **None**
  - **Forward**
  - **Reverse**
  - **Both**
- **Show legend** lets you toggle the legend for the layer. The legend only supports node data.
- **Display tooltip** lets you toggle tooltips for the layer.

Node styles

- **Size** configures the size of the nodes. The default is **Fixed size**, which makes all node sizes the same regardless of the data; however, there is also an option to size the nodes based on data corresponding to a selected field. **Min** and **Max** node sizes have to be set such that the nodes can scale within this range.
- **Color** configures the color of the nodes. The default is **Fixed color**, which sets all nodes to a specific color. There is also an option to have conditional colors depending on the selected field data point values and the color scheme set in the **Standard options** section.
- **Symbol** lets you choose the symbol, icon, or graphic to aid in providing additional visual context to your data. Choose from assets that are included with Grafana, such as simple symbols or the Unicon library. You can also specify a URL containing an image asset. The image must be a scalable vector graphic (SVG).
- **Fill opacity** configures the transparency of each node.
- **Rotation angle** configures the rotation angle of each node. The default is **Fixed value**, which makes all nodes rotate to the same angle regardless of the data; however, there is also an option to set the rotation of the nodes based on data corresponding to a selected field.
- **Text label** configures a text label for each node.

Edge styles

- **Size** configures the line width of the edges. The default is **Fixed size**, which makes all edge line widths the same regardless of the data; however, there is also an option to size the edges based

on data corresponding to a selected field. **Min** and **Max** edge sizes have to be set such that the edges can scale within this range.

- **Color** configures the color of the edges. The default is **Fixed color**, which sets all edges to a specific color. There is also an option to have conditional colors depending on the selected field data point values and the color scheme set in the **Standard options** section.
- **Fill opacity** configures the transparency of each edge.
- **Text label** configures a text label for each edge.

CARTO layer

CARTO layers are not supported in Amazon Managed Grafana.

XYZ tile layer

XYZ tile layers are not supported in Amazon Managed Grafana.

Open Street Map layer

Open Street Map layers (other than the default basedmap) are not supported in Amazon Managed Grafana.

ArcGIS layer

ArcGIS layers are not supported in Amazon Managed Grafana.

Map Controls

The map controls section contains various options for map information and tool overlays.

Zoom

Options for zoom controls.

Show zoom control

Displays zoom controls in the upper left corner.

Mouse wheel zoom

Turns on or off using the mouse wheel for zooming in or out.

Show attribution

Displays attribution for basemap layers on the map.

Show scale

Displays scale information in the bottom left corner.

> **Note**
>
> Displays units in [m]/[km].

Show measure tools

Displays measure tools in the upper right corner. Measurements appear only when this control is open.

- **Click** to start measuring
- **Continue clicking** to continue measurement
- **Double-click** to end measurement

> **Note**
>
> When you change measurement type or units, the previous measurement is removed from the map.
> If the control is closed and then re-opened, the most recent measurement is displayed.
> A measurement can be modified by clicking and dragging on it.

Length

Get the spherical length of a geometry. This length is the sum of the great circle distances between coordinates. For multi-part geometries, the length is the sum of the length of each part. Geometries are assumed to be in 'EPSG:3857'.

You can choose the following units for length measurements:

- **Metric (m/km)**
- **Feet (ft)**
- **Miles (mi)**

- **Nautical miles (nmi)**

Area

Get the spherical area of a geometry. This area is calculated assuming that polygon edges are segments of great circles on a sphere. Geometries are assumed to be in 'EPSG:3857'.

You can choose the following units for area measurements:

- **Square Meters (m^2)**
- **Square Kilometers (km^2)**
- **Square Feet (ft^2)**
- **Square Miles (mi^2)**
- **Acres (acre)**
- **Hectare (ha)**

Show debug

Displays debug information in the upper right corner of the map. This can be useful for debugging or validating a data source.

- **Zoom** displays the current zoom level of the map.
- **Center** displays the current **longitude**, and **latitude** of the map center.

Tooltip

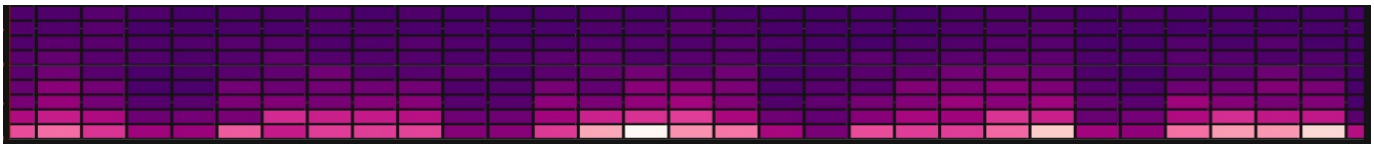
- **None** displays tooltips only when a data point is clicked.
- **Details** displays tooltips when a pointer hovers over a data point.

Heatmap

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The Heatmap panel visualization allows you to view histograms over time. For more information about histograms, refer to Introduction to histograms and heatmaps.



Calculate from data

This setting determines if the data is already a calculated heatmap (from the data source/transformer), or one that should be calculated in the panel.

X Bucket

This setting determines how the X-axis is split into buckets. You can specify a time interval in the **Size** input. For example, a time range of 1h makes the cells 1-hour wide on the X-axis.

Y Bucket

This setting determines how the Y-axis is split into buckets.

Y Bucket scale

Select one of the following Y-axis value scales:

- **linear** – Linear scale.
- **log (base 2)** – Logarithmic scale with base 2.
- **log (base 10)** – Logarithmic scale with base 10.
- **symlog** – Symlog scale.

Y Axes

Defines how the Y axis is displayed

Placement

- **Left** – On the left

- **Right** – On the right
- **Hidden** – Hidden

Unit

Unit configuration

Decimals

This setting determines decimal configuration.

Min/Max value

This setting configures the axis range.

Reverse

When selected, the axis appears in reverse order.

Display multiple y-axes

In some cases, you may want to display multiple y-axes. For example, if you have a dataset showing both temperature and humidity over time, you may want to show two y-axes with different units for these two series.

You can do this by adding field overrides. Follow the steps as many times as required to add as many y-axes as you need.

Colors

The color spectrum controls the mapping between value count (in each bucket) and the color assigned to each bucket. The leftmost color on the spectrum represents the minimum count and the color on the right most side represents the maximum count. Some color schemes are automatically inverted when using the light theme.

You can also change the color mode to Opacity. In this case, the color will not change but the amount of opacity will change with the bucket count

- **Mode**
  - **Scheme** – Bucket value represented by cell color.
    - **Scheme** – If the mode is **Scheme**, then select a color scheme.

- **opacity** – Bucket value represented by cell opacity. Opaque cell means maximum value.
  - **Color** – Cell base color.
  - **Scale** – Scale for mapping bucket values to the opacity.
    - **linear** – Linear scale. Bucket value maps linearly to the opacity.
    - **sqrt** – Power scale. Cell opacity calculated as `value ^ k`, where `k` is a configured **Exponent** value. If exponent is less than 1, you will get a logarithmic scale. If exponent is greater than 1, you will get an exponential scale. In case of 1, scale will be the same as linear.
  - **Exponent** – value of the exponent, greater than `0`.

Start/end color from value

By default, Grafana calculates cell colors based on minimum and maximum bucket values. With Min and Max you can overwrite those values. Consider a bucket value as a Z-axis and Min and Max as Z-Min and Z-Max, respectively.

- **Start** – Minimum value using for cell color calculation. If the bucket value is less than Min, then it is mapped to the “minimum” color. The series min value is the default value.
- **End** – Maximum value using for cell color calculation. If the bucket value is greater than Max, then it is mapped to the “maximum” color. The series max value is the default value.

Cell display

Use these settings to refine your visualization.

Additional display options

Tooltip

- **Show tooltip** – Show heatmap tooltip.
- **Show Histogram** – Show a Y-axis histogram on the tooltip. A histogram represents the distribution of the bucket values for a specific timestamp.
- **Show color scale** – Show a color scale on the tooltip. The color scale represents the mapping between bucket value and color.

Legend

Choose whether you want to display the heatmap legend on the visualization.

Exemplars

Set the color used to show exemplar data.

Histogram

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The histogram visualization calculates the distribution of values and presents them as a bar chart. The Y-axis and the height of each bar represent the count of values that fall into each bracket while the X-axis represents the value range.



Supported formats

Histogram visualization supports time series and any table results with one or more numerical fields.

Display options

Use these options to refine your visualizations.

Bucket size

The size of the buckets. Leave this empty for automatic bucket sizing (~10% of the full range).

Bucket offset

If the first bucket should not start at zero. A non-zero offset shifts the aggregation window. For example, 5-sized buckets that are 0–5, 5–10, 10–15 with a default 0 offset would become 2–7, 7–12, 12–17 with an offset of 2; offsets of 0, 5, or 10, in this case, would effectively do nothing. Typically, this option would be used with an explicitly defined bucket size rather than automatic. For this setting to affect, the offset amount should be greater than 0 and less than the bucket size; values outside this range will have the same effect as values within this range.

Combine series

This will merge all series and fields into a combined histogram.

Line width

Controls line width of the bars.

Fill opacity

Controls the fill opacity of the bars.

Gradient mode

Set the mode of the gradient fill. Fill gradient is based on the line color. To change the color, use the standard color scheme field option. Gradient appearance is influenced by the **Fill opacity** setting.

- **None** – No gradient fill, this is the default setting.
- **Opacity** – Transparency of the gradient is calculated based on the values on the Y-axis. Opacity of the fill is increasing with the values on the Y-axis.
- **Hue** – Gradient color is generated based on the hue of the line color.

Legend options

When the legend option is enabled, it can show either the value mappings or the threshold brackets. To show the value mappings in the legend, it's important that the Color scheme option under standard options is set to Single color or Classic palette. To see the threshold brackets in the legend, set the Color scheme to From thresholds. For more information about the legend, see Configure a legend.

Legend mode

Use these settings to define how the legend appears in your visualization.

- **List** – Displays the legend as a list. This is a default display mode of the legend.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement

Choose where to place the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend Values

Choose which of the standard calculations to show in the legend. You can have more than one. For more information, see Calculation types.

Legend calculations

Choose which calculations to show in the legend. You can select more than one.

Logs

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The logs panel visualization shows log lines from data sources that support logs, such as Elastic, Influx, and Loki. Typically, you would use this panel next to a graph panel to display the log output of a related process.

The logs panel shows the result of queries that were entered on the **Query** tab. The results of multiple queries are merged and sorted by time. You can scroll inside the panel if the data source returns more lines than can be displayed.

To limit the number of lines rendered, you can use the **Max data points** setting in the **Query options**. If it is not set, the data source will usually enforce a default limit.

Log level

For logs where a **level** label is specified, we use the value of the label to determine the log level and update color accordingly. If the log doesn't have a level label specified, we try to find out if its content matches any of the supported expressions (see below for more information). The log level is always determined by the first match. In case Grafana is not able to determine a log level, it will be visualized with **unknown** log level. For more information, see Log level.

Log details

Each log row has an extendable area with its labels and detected fields, for more robust interaction. Each field or label has a stats icon to display statistics in relation to all displayed logs.

Data links

By using data links, you can turn any part of a log message into an internal or external link. The created link is visible as a button in the **Links** section inside the **Log details** view.

Display options

Use the following settings to refine your visualization:

- **Time** – Show or hide the time column. This is the timestamp associated with the log line as reported from the data source.
- **Unique labels** – Show or hide the unique labels column, which shows only non-common labels.
- **Common labels** – Show or hide the common labels
- **Wrap lines** – Toggle line wrapping.

- **Prettify JSON** – Set this to `true` to pretty print all JSON logs. This setting does not affect logs in any format other than JSON.
- **Enable log details** – Toggle option to see the log details view for each log row. The default setting is `true`.
- **Order** – Display results in descending or ascending time order. The default is **Descending**, showing the newest logs first. Set to **Ascending** to show the oldest log lines first.

News

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The news visualization displays an RSS feed. By default, it displays articles from the Grafana Labs blog, and users can change this by entering a different RSS feed URL.

Enter the URL of an RSS in the **Display** section. This visualization type does not accept any other queries, and users should not expect to be able to ilter or query the RSS feed data in any way using this visualization.

Note

RSS feeds are loaded by the Grafana front end without a proxy. As a result, only RSS feeds that are configured with the appropriate CORS headers will load. If the RSS feed you're trying to display fails to load, consider re-hosting the RSS feed or creating your own proxy.

Node graph

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Node graphs can visualize directed graphs or networks. They use a directed force layout to effectively position the nodes, so they can help with displaying complex infrastructure maps, hierarchies, or run diagrams.

Data requirements

A node graph requires a specific shape of the data to be able to display its nodes and edges. Not every data source or query can be visualized with this graph. If you want to use this as a data source developer, see the section about data API.

A node graph consists of *nodes* and *edges*.

- A *node* is displayed as a circle. A node might represent an application, a service, or anything else that is relevant from an application perspective.
- An *edge* is displayed as a line that connects two nodes. The connection might be a request, an operation, or some other relationship between the two nodes.

Both nodes and edges can have associated metadata or statistics. The data source defines what information and values is shown, so different data sources can show different type of values or not show some values.

Nodes

Usually, nodes show two statistical values inside the node and two identifiers just below the node, usually name and type. Nodes can also show another set of values as a color circle around the node, with sections of different color representing different values that should add up to 1. For example, you can have the percentage of errors represented by red portion of the circle.

Additional details can be displayed in a context menu, which is displayed when you choose the node. There also can be additional links in the context menu that can target either other parts of the Grafana workspace or any external link.

> **Note**
>
> Node graph can show only 1,500 nodes. If this limit is crossed, a warning is visible in the upper right corner, and some nodes will be hidden. You can expand hidden parts of the graph by clicking on the **Hidden nodes** markers in the graph.

Edges

Edges can also show statistics when you hover over the edge. Similar to nodes, you can open a context menu with additional details and links by choosing the edge.

The first data source supporting this visualization is the AWS X-Ray data source for its service map feature. For more information, see Connect to an AWS X-Ray data source.

Navigating the node graph

Pan

You can pan within the node graph by choosing outside of any node or edge and dragging the pointer.

Zoom in or out

You can zoom by using the buttons on the upper left corner of the node graph, or use a mouse wheel or other scroll input with the `Ctrl` (or `Cmd`) key.

Explore hidden nodes

The number of nodes shown at a given time is limited to maintain a reasonable performance. Nodes that are outside this limit are hidden behind selectable markers that show an approximate number of hidden nodes that are connected to that edge. You can choose the marker to expand the graph around that node.

Grid view

You can switch to the grid view to have a better overview of the most interesting nodes in the graph. Grid view shows nodes in a grid without edges and can be sorted by stats shown inside the node or by stats represented by the a colored border of the nodes.

To sort the nodes, choose the stats inside the legend. The marker next to the stat name (either # or #) shows which stat is currently used for sorting and sorting direction.

Choose a node and then select the **Show in Graph layout** option to switch back to graph layout with focus on the selected node, to show it in context of the full graph.

Data API

This visualization needs a specific shape of the data to be returned from the data source in order to correctly display it.

Node Graph at minimum requires a data frame describing the edges of the graph. By default, node graph will compute the nodes and any stats based on this data frame. Optionally a second data frame describing the nodes can be sent in case there is need to show more node specific metadata. You have to set `frame.meta.preferredVisualisationType = 'nodeGraph'` on both data frames or name them `nodes` and `edges` respectively for the node graph to render.

Edges data from structure

Required fields:

Field name	Type	Description
id	string	Unique identifier of the edge.
source	string	Id of the source node.
target	string	Id of the target.

Optional fields:

Field name	Type	Description
mainstat	string/number	First stat shown in the overlay when hovering over the edge. It can be a string showing the value as is or it can be a number. If it is a number, any unit associated with that field is also shown.

Field name	Type	Description
secondarystat	string/number	Same as mainStat, but shown right under it.
detail__*	string/number	Any field prefixed with `detail__` will be shown in the header of context menu when clicked on the edge. Use `config.displayName` for a more human readable label.

Nodes data from structure

Required fields:

Field name	Type	Description
id	string	Unique identifier of the node. This ID is referenced by edge in its source and target field.

Optional fields:

Field name	Type	Description
title	string	Name of the node visible just under the node.
subtitle	string	Additional, name, type or other identifier shown under the title.
mainstat	string/number	First stat shown inside the node itself. It can either be a string showing the value as is or a number. If it is a number,

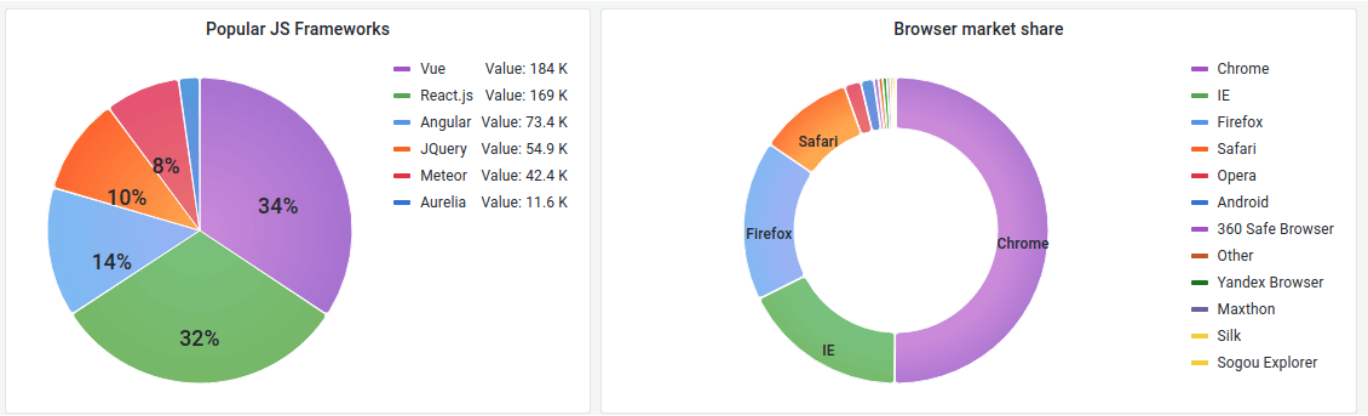
Field name	Type	Description
		any unit associated with that field is also shown.
secondarystat	string/number	Same as mainStat, but shown under it inside the node.
arc__*	number	Any field prefixed with `arc__` will be used to create the color circle around the node. All values in these fields should add up to 1. You can specify color using `config.color.fixedColor` .
detail__*	string/number	Any field prefixed with `detail__` will be shown in the header of context menu when clicked on the node. Use `config.displayName` for more human readable label.
color	string/number	Can be used to specify a single color instead of using the `arc__` fields to specify color sections. It can either be a string (it must be an acceptable HTML color string), or it can be a number, in which case the behavior depends on the `field.config.color.mode` setting. This can be used, for example, to create gradient colors controlled by a field value.

Field name	Type	Description
icon	string	Name of the icon to show inside the node instead of the default stats. Only Grafana built in icons are allowed (see the available icons here).
nodeRadius	number	Radius value in pixels. Used to manage node size.
highlighted	Boolean	Sets whether the node should be highlighted. Use, for example, to represent a specific path in the graph by highlighting several nodes and edges. Defaults to `false`.

Pie chart



This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.



The pie chart displays reduced series, or values in a series, from one or more queries, as they relate to each other, in the form of slices of a pie. The arc length, area and central angle of a slice are all proportional to the slices value, as it relates to the sum of all values. This type of chart is best used when you want a quick comparison of a small set of values in an aesthetically pleasing form.

Value options

Use the following options to refine the value in your visualization.

Show

Choose how much information to show.

- **Calculate** – Reduces each value to a single value per series.
- **All values** – Displays every value from a single series.

Calculation

Select a calculation to reduce each series when **Calculate** has been selected. For information about available calculations, refer to Calculation types.

Limit

When displaying every value from a single series, this limits the number of values displayed.

Fields

Select at least one field to display in the visualization. Each field name is available on the list, or you can select one of the following options:

- **Numeric fields** – All fields with numerical values.

- **All fields** – All fields that are not removed by transformations.
- **Time** – All fields with time values.

Pie chart options

Use these options to refine how your visualization looks.

Pie chart type

Select the pie chart display style. Can be either:

- **Pie** – A standard pie chart
- **Donut** – A pie chart with a hole in the middle

Labels

Select labels to display on the pie chart. You can select more than one.

- **Name** – The series or field name.
- **Percent** – The percentage of the whole.
- **Value** – The raw numerical value.

Labels are displayed in white over the body of the chart by default. You can select darker chart colors to make them more visible. Long names or numbers might be clipped.

Tooltip mode

When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.

- **Single** – The hover tooltip shows only a single series, the one that you are hovering over on the visualization.
- **All** – The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden** – Do not display the tooltip when you interact with the visualization.

Use an override to hide individual series from the tooltip.

Legend options

Use these settings to define how the legend appears in your visualization. For more information about the legend, refer to Configure a legend.

Legend visibility

Use the **Visibility** toggle to show or hide the legend.

Legend mode

Set the display mode of the legend.

- **List** – Displays the legend as a list. This is the default display mode of the legend.
- **Table** – Displays the legend as a table.

Legend placement

Choose where to display the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend values

Select values to display in the legend. You can select more than one.

- **Percent** – The percentage of the whole.
- **Value** – The raw numerical value.

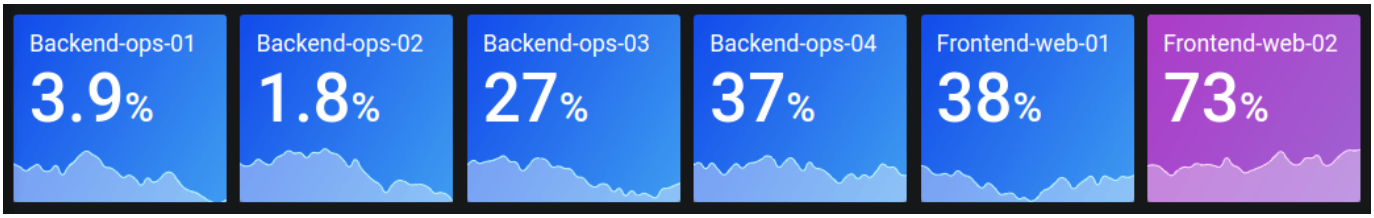
For more information about the legend, refer to Configure a legend.

Stat

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Stats show one large stat value with an optional graph sparkline. You can control the background or value color using thresholds or overrides.



By default, a Stat displays one of the following:

- Just the value for a single series or field.
- Both the value and name for multiple series or fields.

You can use the **Text mode** to control whether the text is displayed or not.

Automatic layout adjustment

The panel automatically adjusts the layout depending on available width and height in the dashboard. It automatically hides the graph (sparkline) if the panel becomes too small.

Value options

Use the following options to refine how your visualization displays the value:

Show

Choose how Grafana displays your data.

- **Calculate**

  Show a calculated value based on all rows.

  - **Calculation** – Select a reducer function that Grafana will use to reduce many fields to a single value. For a list of available calculations, see standard calculations.
  - **Fields** – Select the fields display in the visualization.
- **All values**

Show a separate stat for every row. If you select this option, then you can also limit the number of rows to display.

- **Limit** – The maximum number of rows to display. Default is 5,000.
- **Fields** – Select the fields display in the visualization.

Stat styles

Style your visualization.

Orientation

Choose a stacking direction.

- **Auto** – Grafana selects what it thinks is the best orientation.
- **Horizontal** – Bars stretch horizontally, left to right.
- **Vertical** – Bars stretch vertically, top to bottom.

Text mode

You can use the Text mode option to control what text the visualization renders. If the value is not important, only the name and color is, then change the **Text mode** to **Name**. The value will still be used to determine color and is displayed in a tooltip.

- **Auto** – If the data contains multiple series or fields, show both name and value.
- **Value** – Show only value, never name. Name is displayed in the hover tooltip instead.
- **Value and name** – Always show value and name.
- **Name** – Show name instead of value. Value is displayed in the hover tooltip.
- **None** – Show nothing (empty). Name and value are displayed in the hover tooltip.

Wide layout

Set whether wide layout is enabled or not. Wide layout is enabled by default.

- **On** – Wide layout is turned on.
- **Off** – Wide layout is turned off.

> **Note**
>
> This option is only applicable when **Text mode** is set to **Value and name**. When wide layout is turned on, the value and name are displayed side-by-side with the value on the right, if the panel is wide enough. When wide layout is turned off, the value is always rendered underneath the name.

Color mode

Select a color mode.

- **None** – No color is applied to the value.
- **Value** – Applies colors to the value and graph area.
- **Background Gradient** – Applies color to the value, graph area, and background, with a slight background gradient.
- **Background Solid** – Applies color to the value, graph area, and background, with a solid background color.

Graph mode

Select a graph and sparkline mode.

- **None** – Hides the graph and only shows the value.
- **Area** – Shows the area graph below the value. This requires that your query returns a time column.

Text alignment

Choose an alignment mode.

- **Auto** – If only a single value is shown (no repeat), then the value is centered. If multiple series or rows are shown, then the value is left-aligned.
- **Center** – Stat value is centered.

Show percent change

Set whether percent change is displayed or not. By default it is not shown.

> **Note**
>
> This option is not applicable when the **Show** setting, under **Value options**, is set to **All values**.

Text size

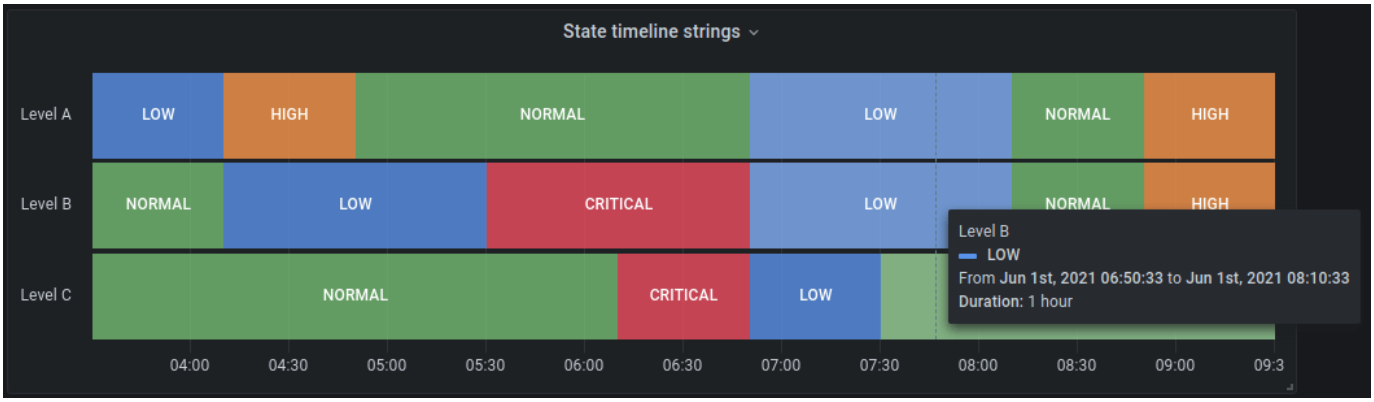
Adjust the sizes of the gauge text.

- **Title** – Enter a numeric value for the gauge title size.
- **Value** – Enter a numeric value for the gauge value size.

State timeline

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

State timelines show discrete state changes over time. Each field or series is rendered as its unique horizontal band. State regions can either be rendered with or without values. This panel works well with string or boolean states but can also be used with time series. When used with time series, the thresholds are used to turn the numerical values into discrete state regions.



State timeline options

Use these options to refine your visualizations:

Merge equal consecutive values

Controls whether Grafana merges identical values if they are next to each other.

Show values

Controls whether values are rendered inside the state regions. **Auto** will render values if there is sufficient space.

Align values

Controls value alignment inside state regions.

Row height

Controls space between rows. 1 = no space = 0.5 = 50% space.

Line width

Controls the line width of state regions.

Fill opacity

Controls the opacity of state regions.

Connect null values

Choose how null values, which are gaps in the data, appear on the graph. Null values can be connected to form a continuous line or set to a threshold above which gaps in the data are no longer connected.

- **Never** – Time series data points with gaps in the data are never connected.
- **Always** – Time series data points with gaps in the data are always connected.
- **Threshold** – Specify a threshold above which gaps in the data are no longer connected. This can be useful when the connected gaps in the data are of a known size or within a known range, and gaps outside this range should no longer be connected.

Disconnect values

Choose whether to set a threshold above which values in the data should be disconnected.

- **Never** – Time series data points in the data are never disconnected.
- **Threshold** – Specify a threshold above which values in the data are disconnected. This can be useful when desired values in the data are of a known size or within a known range, and values outside this range should no longer be connected.

Value mappings

To assign colors to boolean or string values, use Configure value mappings.

Time series data with thresholds

The visualization can be used with time series data as well. In this case, the thresholds are used to turn the time series into discrete colored state regions.

Legend options

When the legend option is enabled, it can show either the value mappings or the threshold brackets. To show the value mappings in the legend, it's important that the **Color scheme** option under Standard options is set to **Single color** or **Classic palette**. To see the threshold brackets in the legend, set the **Color scheme** to **From thresholds**.

Legend mode

Use these settings to define how the legend appears in your visualization. For more information about the legend, refer to Configure a legend.

- **List** – Displays the legend as a list. This is the default mode.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement

Choose where to display the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend values

Choose which of the standard calculations to show in the legend. You can have more than one.

Status history

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Status histories show periodic states over time. Each field or series is rendered as a horizontal row. Boxes are rendered and centered around each value.

Supported data

A status history works with string, boolean, and numerical fields or time series. A time field is required. You can use value mappings to color strings or assign text values to numerical ranges.

Display options

Use these options to refine the visualization.

Show values

Controls whether values are rendered inside the value boxes. **Auto** will render values if there is sufficient space.

Column width

Controls the width of boxes. 1 = maximum space and 0 = minimum space.

Line width

Controls line width of state regions.

Fill opacity

Controls the fill opacity of state regions.

Value mappings

To assign colors to boolean or string values, use Configure value mappings.

Time series data with thresholds

The panel can be used with time series data as well. In this case, the thresholds are used to color the boxes. You can also use gradient color schemes to color values.

Legend options

When the legend option is enabled, it can show either the value mappings or the threshold brackets. To show the value mappings in the legend, it's important that the **Color scheme** option under Standard options is set to **Single color** or **Classic palette**. To see the threshold brackets in the legend, set the **Color scheme** to **From thresholds**.

Legend mode

Use these settings to define how the legend appears in your visualization. For more information about the legend, refer to Configure a legend.

- **List** – Displays the legend as a list. This is the default mode.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement

Choose where to display the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

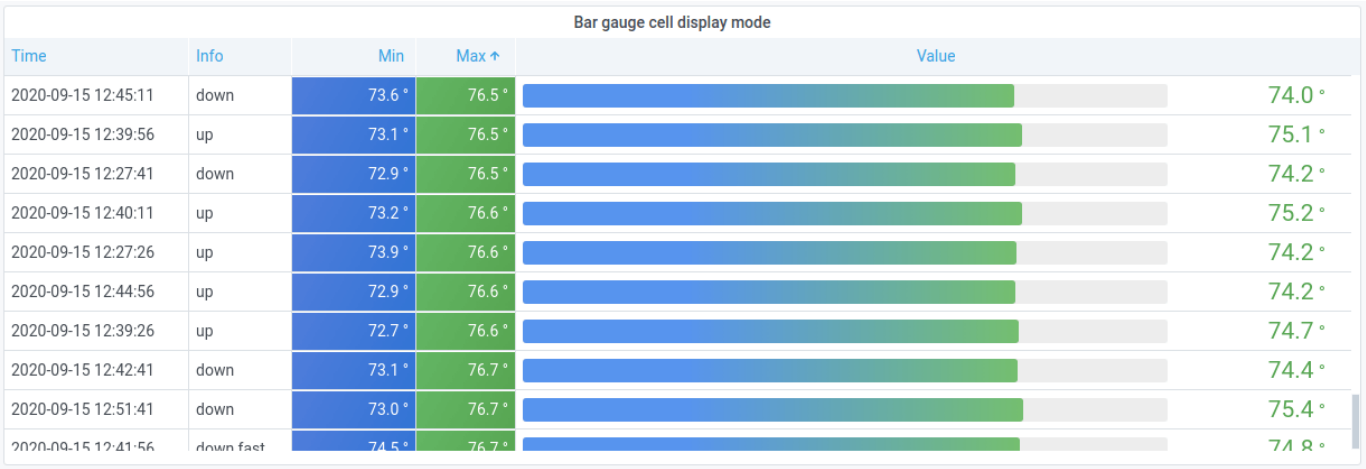
Legend values

Choose which of the standard calculations to show in the legend. You can have more than one.

Table

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Tables are very flexible, supporting multiple modes for time series and for tables, annotation, and raw JSON data. This visualization also provides date formatting, value formatting, and coloring options.



> **Note**
>
> Annotations and alerts are not supported in tables.

Sort column

Choose a column title to change the sort order from default to descending to ascending. Each time you select the column, the sort order changes to the next option in the cycle. You can sort on multiple columns by holding the `shift` key when selecting additional columns.

Table options

Show header

Show or hide column names imported from your data source.

Column width

By default, Grafana automatically calculates the column width based on the table size and the minimum column width. This field option can override the setting and define the width for all columns in pixels.

For example, if you enter `100`, all the columns will be set to 100 pixels wide (the change takes place when you exit the field).

Minimum column width

By default, the minimum width of the table column is 150 pixels. This field option can override that default and will define the new minimum column width for the table panel in pixels.

For example, if you set the minimum to 75, all the columns will scale to no smaller than 75 pixels wide.

For small-screen devices, such as smartphones or tablets, you can reduce the default `150` pixel value to `50` to allow table based panels to render correctly in dashboards.

Column alignment

Choose how Grafana should align cell contents.

- Auto (default)
- Left
- Center
- Right

Cell type

By default, Grafana automatically chooses display settings. You can override the settings by choosing one of the following options to set the default for all fields. Additional configuration is available for some cell types.

> **Note**
>
> If you set these in the **Field** tab, then the type will apply to all fields, including the time field. You can set them in the **Override** tab to apply the change to one or more fields.

Color text

If thresholds are set, then the field text is displayed in the appropriate threshold color.

Color background (gradient or solid)

If thresholds are set, then the field background is displayed in the appropriate threshold color.

Gauge

Cells can be displayed as a graphical gauge, with several different presentation types.

- *Basic* – The basic mode will show a simple gauge with the threshold levels defining the color of gauge.
- *Gradient* – The threshold levels define a gradient.
- *LCD* – The gauge is split up in small cells that are lit or unlit.

Additionally, labels displayed alongside the gauges can be set to be colored by value, match the theme text color, or be hidden.

- **Value color**
- **Text color**
- **Hidden**

JSON view

Shows value formatted as code. If a value is an object the JSON view allowing browsing the JSON object will appear on hover.

Sparkline

Shows values rendered as a sparkline. Requires time series to table data transform.

Cell value inspect

Enables value inspection from table cell. The raw value is presented in a modal window.

> **Note**
>
> Cell value inspection is only available when cell display mode is set to Auto, Color text, Color background, or JSON View.

Column filter

You can temporarily change how column data is displayed. For example, you can order values from highest to lowest or hide specific values. For more information, see Filter table columns.

Pagination

Use this option to enable or disable pagination. It is a front-end option that does not affect queries. When enabled, the page size automatically adjusts to the height of the table.

Filter table columns

If you turn on the **Column filter**, then you can filter table options.

To turn on column filtering

1. In Grafana, navigate to the dashboard with the table with the columns that you want to filter.
2. On the table panel you want to filter, open the panel editor.
3. Choose the **Field** tab.
4. In **Table** options, turn on the **Column filter** option.

A filter (funnel) icon appears next to each column title.

Filter column values

To filter column values, choose the filter (funnel) icon next to a column title. Grafana displays the filter options for that column.

Choose the check box next to the values that you want to display. Enter text in the search field at the top to show those values in the display so that you can select them rather than scroll to find them.

Select from several operators to display column values:

- **Contains** – Matches a regex pattern (operator by default).
- **Expression** – Evaluates a Boolean expression. The character $ represents the column value in the expression (for example, `$ >= 10 &&& $ <= 12`).
- **Comparison operators** – You can use the typical comparison operators: =, !=, <, <=, >, >=.

Choose the checkbox above the **Ok** and **Cancel** buttons to add or remove all displayed values from the filter.

Clear column filters

Columns with filters applied have a blue funnel displayed next to the title.

To remove the filter, choose the blue funnel icon and then select **Clear filter**.

Table footer

You can use the table footer to show calculations on fields.

After you enable the table footer, you can select the **Calculation**, and then the **Fields** that you want to calculate.

The system applies the calculation to all numeric fields if you do not select a field.

Count rows

If you want to show the number of rows in the dataset instead of the number of values in the selected fields, select the **Count** calculation and enable **Count rows**.

Text



This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Text visualizations enable you to directly include text or HTML in your dashboards. This can be used to add contextual information and descriptions or embed complex HTML.

Mode

Mode determines how embedded content appears. It has the following options:

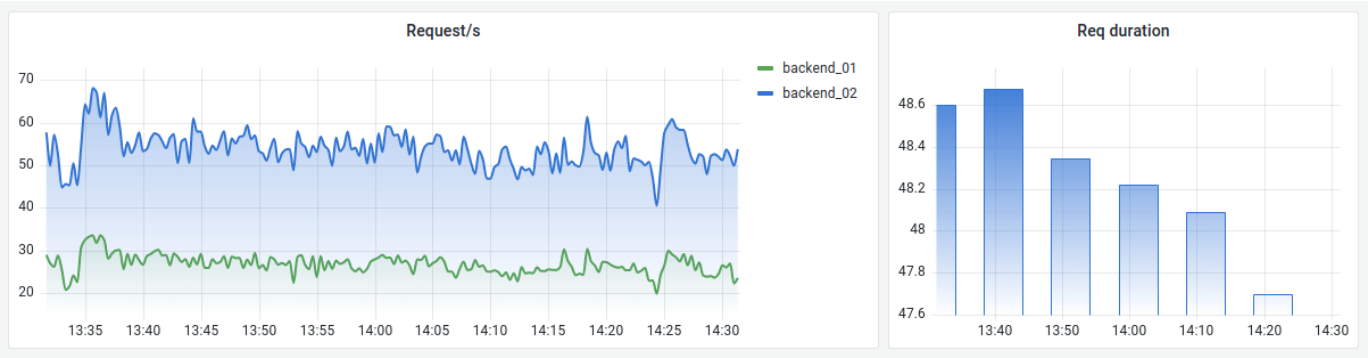
- **Markdown** – This option formats the content as markdown.
- **HTML** – This setting renders the content as sanitized HTML.
- **Code** – This setting renders content inside a read-only code editor. Select an appropriate language to apply syntax highlighting to the embedded text.

Variables

Variables in the content will be expanded for display.

Time series

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.



Time series visualizations are the default and primary way to visualize time series data as a graph. They can render series as lines, points, or bars. They're versatile enough to display almost any time-series data.

Note

You can migrate Graph panel visualizations to Time series visualizations. To migrate, on the **Panel** tab, choose **Time series visualization**. Grafana transfers all applicable settings.

Topics

- Tooltip options
- Legend options
- Graph styles
- Axis options
- Color options

Tooltip options

This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Tooltip options control the information overlay that appears when you hover over data points in the graph.

Tooltip mode

- **Single** – The hover tooltip shows only a single series, the one that you are hovering over.
- **All** – The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden** – Do not display the tooltip when you interact with the visualization.

Legend options

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Legend options control the series names and statistics that appear under or to the right of the graph.

Legend mode

Use these settings to define how the legend appears in your visualization. For more information about the legend, see Configure a legend.

- **List** – Displays the legend as a list. This is the default display mode of a legend.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement

Choose where to display the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend values

Choose which of the standards calculations to show in the legend. You can have more than one.

Graph styles

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use this option to define how to display your time series data. You can use overrides to combine multiple styles in the same graph.

- **Lines** – Display the time series as a line on a graph.
- **Bars** – Display the time series as a series of bars on a graph, one for each data point.
- **Points** – Display the time series as dots on a graph, one for each data point.

Bar alignment

Set the position of the bar relative to a data point.where the point would be drawn on the graph. Because a bar has a width, it can be placed before, after, or centered on the point. The choices for this option are:

- 
  Before – The bar is drawn before the point. The point is placed on the trailing corner of the bar.
- 
  Center – The bar is drawn around the point. The point is placed in the center of the bar. This is the default.
- 
  After – The bar is drawn after the point. the point is placed on the leading corner of the bar.

Line width

Line width is a slider that controls the thickness for series lines or the outline for bars.

Fill opacity

Sets the opacity of a fill color. Fills are used, for example, to show the area under the line in a line graph, or as the color of bars in a bar graph.

Gradient mode

Gradient mode specifies the gradient fill, which is based on the series color. To change the color, use the standard color scheme field option. For more information, see Color scheme.

The gradient mode options are:

- **None** – No gradient fill. This is the default setting.
- **Opacity** – An opacity gradient where the opacity of the fill increases as the Y-axis values increase.
- **Hue** – A subtle gradient that is based on the hue of the series color.
- **Scheme** – A color gradient defined by your color scheme. This setting is used for the fill area and the line. For more information, see Color options.

The gradient appearance is influenced by the **Fill opacity** setting.

Show points

You can configure your visualization to add points to line or bars, with the following options.

- **Auto** – Grafana determines whether to show points based on the density of the data. Points are shown when the density is low.
- **Always** – Points are shown, regardless of the data density.
- **Never** – Do not show points.

Point size

Sets the size of the points, from 1 to 40 pixels in diameter.

Line interpolation

Choose how Grafana interpolates the series line.



The options are:

- **Linear** – Points are joined by straight lines.

- **Smooth** – Points are joined by curved lines that smooths transitions between points.
- **Step before** – The line is displayed as steps between points. Points are rendered at the end of the step.
- **Step after** – The line is displayed as steps between points. Points are rendered at the beginning of the step.

Line style

Set the style of the line. To change the color, use the standard color scheme field option.

The choices for line style are:

- **Solid** – Display a solid line. This is the default setting.
- **Dash** – Display a dashed line. When you choose this option, a list appears for you to select the length and gap (length, gap) for the line dashes. Dash spacing is set to 10, 10 by default.
- **Dots** – Display dotted lines. When you choose this option, a list appears for you to select the gap length for the dot spacing. Dot spacing is set to 10 by default.

Connect null values

Choose how null values, which are gaps in the data, appear on the graph. Null values can be connected to form a continuous line, or set to a threshold above which gaps in the data are no longer connected.

The choices for how to connect null values are:

- **Never** – Time series data points with gaps in the data are never connected.
- **Always** – Time series data points with gaps in the data are always connected.
- **Threshold** – Specify a threshold above which gaps in the data are no longer connected. This can be useful when the connected gaps in the data are of a known size or within a known range, and gaps outside the range should no longer be connected.

Disconnect values

Choose whether to add a gap between values in the data that have times between them above a specified threshold.

The choices for disconnect values are are:

- **Never** – Time series data points are never disconnected.
- **Threshold** – Specify a threshold above which values in the data are disconnected. This can be useful when desired values in the data are of a known size or within a known range, and values outside this range should no longer be connected.

Stack series

Stacking allows Grafana to display series on top of each other. Be cautious when using stacking in the visualization as it can easily create misleading graphs. To read more about why stacking might not be the best approach, refer to The Issue with Stacking.

The choices for stacking are:

- **Off** – Turns off series stacking.
- **Normal** – Stacks series on top of each other.
- **100%** – Stack by percentage, where all series together add up to 100%.

Stacking series in groups

You can override the stacking behavior to stack series in groups. For more information about creating an override, see Configure field overrides.

To stack series in groups

1. Edit the panel and choose **Overrides**.
2. Create a field override for the **Stack series** option.
3. In stacking mode, select **Normal**.
4. Name the stacking group in which you want the series to appear.

   The stacking group name option is only available when you create an override.

Fill below to

The **Fill below to** option fills the area between two series. This options is only available as a series or field override.

Using this option you can fill the area between two series, rather than from the series line down to 0. For example, if you had two series called *Max* and *Min*, you could select the *Max* series and override it to **Fill below to** the *Min* series. This would fill only the area between the two series lines.

Axis options

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Options under the axis category change how the x- and y-axes are rendered. Some options do not take effect until you click outside of the field option box you are editing. You can also press `Enter`.

Time zone

Set the desired time zones to display along the x-axis.

Placement

Select the placement of the Y-axis. The options are:

- **Auto** – Automatically assigns the y-axis to the series. When there are two or more series with different units, Grafana assigns the left axis to the first unit, and the right axis to the units that follow.
- **Left** – Display all y-axes on the left side.
- **Right** – Display all y-axes on the right side.
- **Hidden** – Hide all axes.

To selectively hide axes, add a field override that targets specific fields.

Label

Set a y-axis text label. If you have more than one y-axis, then you can assign different labels using an override.

Width

Set a fixed width of the axis. By default, Grafana dynamically calculates the width of an axis.

By setting the width of the axis, data with different axes types can share the same display proportions. This setting makes it easier for you to compare more than one graph's worth of data because the axes are not shifted or stretched within visual proximity to each other.

Show grid lines

Set the axis grid line visibility.

- **Auto** – Automatically show grid lines based on the density of the data.
- **On** – Always show grid lines.
- **Off** – Never show grid lines.

Color

Set the color of the axis.

- **Text** – Set the color based on theme text color.
- **Series** – Set the color based on the series color.

Show border

Set the axis border visibility.

Scale

Set how the y-axis values scale.

- **Linear** – Divides the scale into equal parts.
- **Logarithmic** – Use a logarithmic scale. When you select this option, a list appears for you to choose a binary (base 2) or common (base 10) logarithmic scale.
- **Symlog** – Use a symmetrical logarithmic scale. When you select this option, a list appears for you to choose a binary (base 2) or common (base 10) logarithmic scale. The linear threshold option allows you to set the threshold at which the scale changes from linear to logarithmic.

Centered zero

Sets the y-axis to be centered on zero.

Soft min and soft max

Set a **Soft min** or **soft max** option for better control of y-axis limits. By default, Grafana sets the range for the y-axis automatically based on the dataset.

Soft min and soft max settings can prevent small variations in the data from being magnified when it's mostly flat. In contrast, hard min and max values help prevent obscuring useful detail in the data by clipping intermittent spikes past a specific point.

To define hard limits of the y-axis, set standard min/max options. For more information, refer to Configure standard options.

Transform

Use this option to transform the series values without affecting the values shown in the tooltip, context menus, or legend. You have two transform options:

- **Negative Y transform** – Flip the results to negative values on the Y axis.
- **Constant** – Show the first value as a constant line.

> **Note**
>
> The transform option is only available as an override.

Display multiple y-axes

There are some cases where you want to display multiple y-axes. For example, if you have a dataset showing both temperature and humidity over time, you could show two y-axes with different units for these two series.

To display multiple y-axes, add a field override. Follow the steps as many times as required to add as many y-axes as you need.

Color options

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

By default, the graph uses the standard Color scheme option to assign series colors. You can also use the legend to open the color picker by selecting the legend series color icon. Setting color this way automatically creates an override rule that set's a specific color for a specific series.

The following are additional options that you can use to override series color defaults.

Classic palette

The most common setup is to use the **Classic palette** for graphs. This scheme automatically assigns a color for each field or series based on its order. If the order of a field changes in your query, the color also changes. You can manually configure a color for a specific field using an override rule.

Single color

Use this mode to specify a color. You can also select the colored line icon next to each series in the Legend to open the color picker. This automatically creates a new override that sets the color scheme to single color and the selected color.

By value color schemes

If you select a by value color scheme like **From thresholds (by value)** or **Green-Yellow-Red (by value)**, the **Color series by** option appears. This option controls which value (Last, Min, Max) to use to assign the series its color.

Scheme gradient mode

The **Gradient mode** option located under the **Graph styles** has a mode named **Scheme**. When you enable **Scheme**, the line or bar receives a gradient color defined from the selected **Color scheme**.

From thresholds

If the **Color scheme** is set to **From thresholds (by value)** and **Gradient mode** is set to **Scheme**, then the line or bar color changes as they cross the defined thresholds. You will see only the exact colors chosen in the scheme.

Gradient color schemes

Using a gradient color scheme *without* setting the **Gradient mode** to **Scheme**, means that the colors chosen will form a gradient between the colors chosen, as the values in the series move between the thresholds set.

Traces

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Traces visualizations let you follow a request as it traverses the services in your infrastructure. The traces visualization displays traces data in a diagram that allows you to easily interpret it.

For more information about traces and how to use them, refer to the following documentation:

- Tracing in Explore
- Tempo data source
- Getting started with Tempo in the *Grafana Labs Tempo Documentation*.

Adding a panel with tracing visualizations

Once you have tracing data available in your Grafana stack, you can add tracing panels to your Grafana dashboards.

Using a dashboard variable, `traceID`, lets you create a query to show specific traces for a given trace ID. For more information about dashboard variables, refer to the Variables documentation.

Prerequisites

Before you begin, you need:

- An Amazon Managed Grafana workspace.
- A Tempo data source connected to your workspace.

To view and analyze traces data in a dashboard, you need to add the traces visualization to your dashboard and define a query using the panel editor. The query determines the data that is displayed in the visualization. For more information on the panel editor, refer to the Panel editor documentation.

This procedure uses dashboard variables and templates to allow you to enter trace IDs which can then be visualized. You'll use a variable called `traceId` and add it as a template query.

To add a traces visualization query

1. In your workspace, create a new dashboard or go to an existing dashboard where you'd like to add traces visualizations.
2. Choose **Add visualization** from a new dashboard or choose **Add Panel** on an existing dashboard.
3. Select the appropriate tracing data source.
4. In the top-right of the panel editor, choose the **Visualizations** tab, and select **Traces**.
5. Under the **Panel options**, enter a **Title** for your trace panel. For more information on the panel editor, see Configure panel options.
6. In the query editor, select the **TraceQL** query type tab.
7. Enter `${traceId}` in the TraceQL query field to create a dashboard variable. This variable is used as the template query.
8. Choose **Apply** in the panel editor to add the panel to the dashboard.
9. Go to the dashboard **Settings** and add a new variable called `traceId`, of variable type **Custom**, giving it a label, if required. Choose **Apply** to add the variable to the dashboard.
10. Verify that the panel works by using a valid trace ID for the data source used for the trace panel and editing the ID in the dashboard variable.

Adding TraceQL with table visualizations

While you can add a trace visualization to a dashboard, having to manually add trace IDs as a dashboard variable is cumbersome. It's more useful to instead be able to use TraceQL queries to search for specific types of traces and then select appropriate traces from matching results.

Prerequisites

This procedure assumes you have completed the previous procedure.

To add TraceQL with table visualizations

1. In the same dashboard where you added the trace visualization, choose **Add panel** to add a new visualization panel.
2. Select the same trace data source you used in the previous section.
3. In the top-right of the panel editor, select the **Visualizations** tab, then choose **Table**.
4. In the query editor, choose the **TraceQL** tab.
5. Under the **Panel options**, enter a **Title** for your trace panel.
6. Add an appropriate TraceQL query to search for traces that you would like to visualize in the dashboard. For example, here is a simple, static query from a server called *`my-server`*.

   ```
   { .service.name = "my-server" && .http.status_code=500 }
   ```

   You can write the TraceQL query as a template query to take advantage of other dashboard variables, if they exist. This lets you create dynamic queries based on these variables.

When results are returned from a query, the results are rendered in the panel's table.

The results in the traces visualization include links to the **Explore** page that renders the trace. You can add other links to traces in the table that fill in the `traceId` dashboard variable when selected, so that the trace is visualized in the same dashboard.

To create a set of data links in the panel, use the following procedure.

To use a variable to add other links to traces

1. In the right-side menu, under **Data links**, choose **Add link**.
2. Add a **Title** for the data link.

3. Find the path to the dashboard by looking in your browser's address bar when the full dashboard is being rendered. Because this is a link to a dashboard in the same Grafana stack, only the path of the dashboard is required.

   For example, if your path is:

   ```
   https://g-example.grafana-workspace.us-east-1.amazonaws.com/d/1234abcd5/my-dashboard?orgId=1
   ```

   Then the path to the dashboard is:

   ```
   /d/1234abcd5/my-dashboard?orgId=1
   ```

4. In the **URL** field, make a self-reference to the dashboard that contains both of the panels. This self-reference uses the value of the selected trace in the table to fill in the dashboard variable. Use the path for the dashboard from the previous step and then fill in the value of `traceId` using the selected results from the TraceQL table. The trace ID is exposed using the `traceID` data field in the returned results, so use that as the value for the dashboard variable. For example:

   ```
   /d/1234abcd5/my-dashboard?orgId=1&var-traceId=${__data.fields["traceID"]}
   ```

5. Choose **Save** to save the data link.
6. Choose **Apply** from the panel editor to apply the panel to the dashboard.
7. Save the dashboard.

You should now see a list of matching traces in the table visualization. While selecting the **TraceID** or **SpanID** fields will give you the option to either open the **Explore** page to visualize the trace or following the data link, selecting any other field (such as `Start time`, `Name`, or `Duration`) automatically follows the data link, filling in the `traceId` dashboard variable, and then shows the relevant trace in the trace panel.

Trend

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Trend visualizations should be used for datasets that have a sequential, numeric X that is not time. Some examples are function graphs, rpm/torque curves, supply/demand relationships, and elevation or heart rate plots along a race course (with x as distance or duration from start).

Trend visualizations support all visual styles and options available in the time series visualization with the following exceptions:

- No annotations or time regions
- No shared cursor (or crosshair)
- No multi-timezone x axis
- No ability to change the dashboard time range via drag-selection

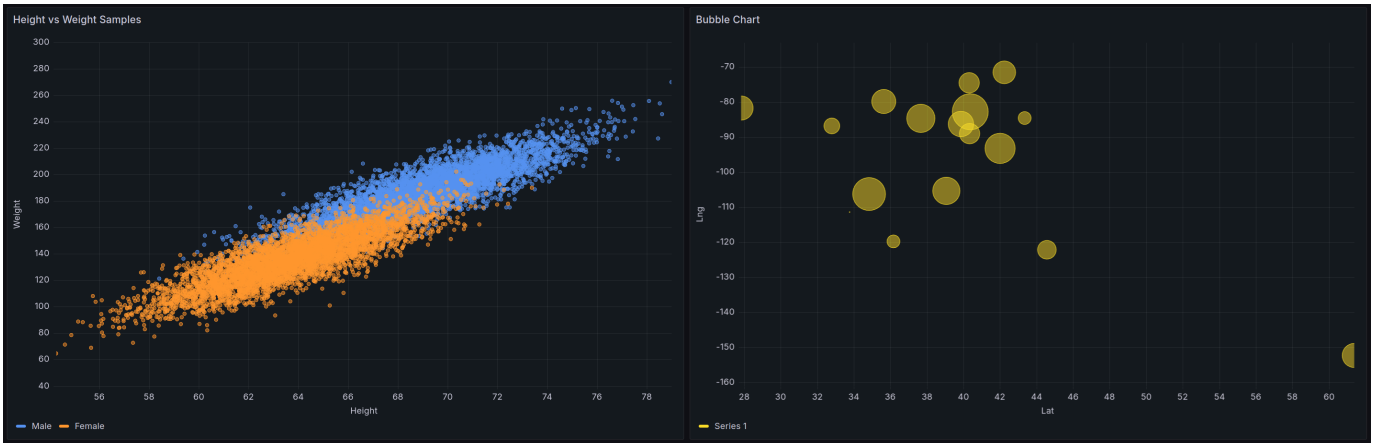
X Field selection

Use this option to select a field that contains increasing numeric values.

For example, you could represent engine power and torque versus speed where speed is plotted on the x axis and power and torque are plotted on the y axes.

XY Chart

XY charts provide a way to visualize arbitrary x and y values in a graph so that you can easily show the relationship between two variables. XY charts are typically used to create scatter plots. you can also use them to create bubble charts, where field values determine the size of each bubble.



Supported data formats

You can use any type of tabular data with at least two numeric fields in an XY chart. This type of visualization doesn't require time data.

Panel options

In the **Panel options** section of the panel editor pane, you set basic options like the panel title and description. You can also configure repeating panels in this section. For more information, see Configure panel options.

XY chart options

Series mapping

Set how series data is mapped in the chart.

- **Auto** – Automatically generates series from all available data frames (or datasets). You can filter to select only one frame.
- **Manual** – Explicitly define the series by selecting from available data frames.

Depending on your series mapping selection, the **Frame**, **X-field**, and **Y-field** options differ. The Auto and Manual series mapping sections describe these different options.

Auto series mapping options

When you select **Auto** as your series mapping mode, the following options are preconfigured, but you can also define them yourself.

- **Frame** – By default an XY chart displays all data frames. You can filter to select only one frame.

- **X-field** – Select which field X represents. By default,t his is the first number field in each data frame.
- **Y-field** – After the X-field is set, by default, all the remaining number fields in the data frame are designated as the Y-fields. You can use this option to explicitly choose which fields to use for Y.

  The series of the chart are generated from the Y-fields. To make changes to a series in an XY chart, make overrides to the Y-field. Any field you use in the Size field or Color field doesn't generate a series.

  You can also use overrides to exclude Y-fields individually. To do so, add an override with the following properties for each Y-field you want removed:
  - Override type: **Fields with name**
  - Override property: **Series > Hide in area**
  - Area: **Viz**

Manual series mapping options

When you select **Manual** as your series mode, you can add, edit, and delete series. To manage a series, select the **Series** field. To rename the series, select the series name.

In **Manual** mode, you must set the following options:

- **Frame** – Select your data frame or dataset. You can add as many frames as you want.
- **X-field** – Select which field X represents.
- **Y-field** – Select which field Y represents.

Size field

Use this option to set which field's values control the size of point in the chart. This value is relative to the min and max of all the values in the data frame.

When you select this option, you can then set the min and max point size options.

Color field

Use this option to set which field's values control the color of the points in the chart. To use the color value options under the **Standard options**, you must set this field.

Typically, this option is used when you only have one series displayed in the chart.

Show

Set how values are represented in the visualization.

- **Points** – Display values as points. When you select this option, the point size option is also displayed.
- **Lines** – Add a line between values. When you select this option, the line style and line width options are also displayed.
- **Both** – Display both points and lines.

Point size

Sets the size of all point in the chart, from one to 100 pixels in diameter. The default size is five pixels. You can set an override to set the pixel size by series (Y-field).

Min/Max point size

Use these optinos to control the minimum or maximum point size when you've set the **Size field** option. You can override these options for specific series.

Line style

Set the style of the line. To change the color, use the standard color scheme field option.

- **Solid** – Display a solid line. This is the default setting.
- **Dash** – Display a dashed line. When you choose this option, a drop-down list is displayed where you can select the length and gap setting for the line dashes. By default, the length and gap are set to `10, 10`.
- **Dots** – Display dotted lines. When you choose this option, a drop-down list is displayed where you can select dot spacing. By default, the dot spacing is set to `0, 10` (the first number represents dot length, and is always zero).

Line width

Sets the width of the lines, in pixels.

Tooltip options

Tooltip options control the information overlay that appears when you hover over data points in the graph.

Tooltip mode

- **Single** – The hover tooltip shows only a single series, the one that you are hovering over.
- **Hidden** – Do not display the tooltip when you interact with the visualization.

Use an override to hide individual series from the tooltip.

Max height

Set the maximum height of the tooltip box. The defautl is 600 pixels.

Legend options

Legend options control the series names and statistics that appear under or to the right of the graph. For more information about the legend, see Configure a legend.

Visibility

Toggle the switch to turn the legend on or off.

Mode

Use these settings to define how the legend appears in your visualization.

- **List** – Displays the legend as a list. This is the default display mode of a legend.
- **Table** – Displays the legend as a table.

Placement

Choose where to display the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Values

Choose which of the standards calculations to show in the legend. You can have more than one.

Width

Control how wide the legend is when place to the right side of the visualization. This option is only displayed if you set the legend placement to **Right**.

Axis options

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Options under the axis category change how the x- and y-axes are rendered. Some options don't take effect until you click outside of the field option box you are editing. You can also press `Enter`.

Placement (y-axis)

Select the placement of the Y-axis. The options are:

- **Auto** – Automatically assigns the y-axis to the series. When there are two or more series with different units, Grafana assigns the left axis to the first unit, and the right axis to the units that follow.
- **Left** – Display all y-axes on the left side.
- **Right** – Display all y-axes on the right side.
- **Hidden** – Hide all axes.

To selectively hide axes, add a field override that targets specific fields.

Label

Set a y-axis text label. If you have more than one y-axis, then you can assign different labels using an override.

Width

Set a fixed width of the axis. By default, Grafana dynamically calculates the width of an axis.

By setting the width of the axis, data with different axes types can share the same display proportions. This setting makes it easier for you to compare more than one graph's worth of data because the axes are not shifted or stretched within visual proximity to each other.

Show grid lines

Set the axis grid line visibility.

- **Auto** – Automatically show grid lines based on the density of the data.
- **On** – Always show grid lines.
- **Off** – Never show grid lines.

Color

Set the color of the axis.

- **Text** – Set the color based on theme text color.
- **Series** – Set the color based on the series color.

Show border

Set the axis border visibility.

Scale

Set how the y-axis values scale.

- **Linear** – Divides the scale into equal parts.
- **Logarithmic** – Use a logarithmic scale. When you select this option, a list appears for you to choose a binary (base 2) or common (base 10) logarithmic scale.
- **Symlog** – Use a symmetrical logarithmic scale. When you select this option, a list appears for you to choose a binary (base 2) or common (base 10) logarithmic scale. The linear threshold option allows you to set the threshold at which the scale changes from linear to logarithmic.

Centered zero

Sets the y-axis to be centered on zero.

Soft min and soft max

Set a **Soft min** or **soft max** option for better control of y-axis limits. By default, Grafana sets the range for the y-axis automatically based on the dataset.

Soft min and soft max settings can prevent small variations in the data from being magnified when it's mostly flat. In contrast, hard min and max values help prevent obscuring useful detail in the data by clipping intermittent spikes past a specific point.

To define hard limits of the y-axis, set standard min/max options. For more information, refer to Configure standard options.

Transform

Use this option to transform the series values without affecting the values shown in the tooltip, context menus, or legend. You have two transform options:

- **Negative Y transform** – Flip the results to negative values on the Y axis.
- **Constant** – Show the first value as a constant line.

> **Note**
>
> The transform option is only available as an override.

Display multiple y-axes

There are some cases where you want to display multiple y-axes. For example, if you have a dataset showing both temperature and humidity over time, you could show two y-axes with different units for these two series.

To display multiple y-axes, add a field override. Follow the steps as many times as required to add as many y-axes as you need.

Standard options

Standard options in the panel editor let you change how field data is displayed in your visualization. When you set a standard option, the change is applied to all fields or series. For more granular control over the display of fields, refer to Configure field overrides.

You can cusotmize the following standard options:

- **Field min/max** – Enable **Field min/max** to have Grafana calculate the min or max of each field individually, based on the minimum or maximum value of the field.
- **Color scheme** – Set single or multiple colors for your entire visualization.

For more information, see Configure standard options.

Data links

Data links allow you to link to other panels, dashboards, and external resources while maintaining the context of the source panel. You can create links that include the series name or even the value under the cursor.

For each data link, set the following options:

- **Title**
- **URL**
- **Open in new tab**

For more information, see Configure data links.

Field overrides

Overrides allow you to customize visualization settings for specific fields or series. When you add an override rule, you can target a particular set of fields and define multiple options for how those fields are displayed.

Choose from one of the following override options:

- **Fields with name** – Select a field from the list of all available fields.
- **Fields with name matching regex** – Specify fields to override with a regular expression.
- **Fields with type** – Select fields by type, such as string, numeric, or time.
- **Fields returned by query** – Select all fields returned by a specific query.
- **Fields with values** – Select all fields returned by your defined reducer condition, such as **Min**, **Max**, **Count**, or **Total**.

For more information, see Configure field overrides.

Explore in Grafana version 10

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana's dashboard UI provides functionality to build dashboards for visualization. *Explore* strips away the dashboard and panel options so that you can focus on the query. It helps you iterate until you have a working query and then you can build a dashboard from the query.

Note

If you just want to explore your data and do not want to create a dashboard, then Explore makes this much easier. If your data source supports graph and table data, then Explore shows the results both as a graph and a table. This allows you to see trends in the data and more details at the same time.

This page will get you started exploring your data. The following topics give you more details on specific features and uses for Explore.

- Query management in Explore
- Logs in Explore
- Tracing in Explore
- Correlations editor in Explore
- Inspector in Explore

Start exploring

> **Note**
>
> In order to access Explore, you must have an editor or an administrator role.

To access Explore

1. In your Grafana workspace, choose the Explore menu item from the left menu bar.

   An empty Explore tab opens.

   Alternately, to start with an existing query in a panel, choose the Explore option from the Panel menu. This opens an Explore tab with the query from the panel and allows you to tweak or iterate in the query outside of your dashboard.
2. Choose your data source from the dropdown in the top left.

   You can also choose **Open advanced data source picker** to see more options, including adding a data source (for Admins only).
3. Write the querying using the query editor provided by the selected data source.

   For more details about queries, see Query and transform data.
4. Run the query using the button in the top right corner.

Split and compare

The split view provides an easy way to compare visualizations side-by-side or to look at related data together on one page.

Top open the split view

1. In the Explore view, choose the **Split** button to duplicate the current query and split the page into two side-by-side queries.

> **Note**
>
> It is possible to select another data source for the new query which, for example, allows you to compare the same query for two different servers or to compare the staging environment to the production environment.

In split view, timepickers for both panels can be linked (if you change one, the other gets changed as well) by selecting a time-sync button attached to one of the timepickers. Linking timepickers keeps the start and the end times of the split view queries in sync. It ensures that you're looking at the same time interval in both split panels.

2. To close the newly created query, choose the **Close Split** button.

Content outline

The content outline is a side navigation bar that keeps track of the queries and visualizations you created in Explore. It allows you to navigate between them quickly.

The content outline also works in a split view. When you are in split view, the content outline is generated for each pane.

To open the content outline

1. Select the **Outline** button in the top left corner of the **Explore** screen.
2. Select any panel icon in the content outline to navigate to that panel.

Share Explore URLs

When using Explore, the URL in the browser address bar updates as you make changes to the queries. You can share or bookmark this URL.

> **Note**
>
> Explore may generate relatively long URLs. You can also generate and share a shortened link if the URL is too long for your tools.

Generating Explore URLs from external tools

Because Explore URLs have a defined structure, you can build a URL from external tools and open it in Grafana. The url structure is:

```
http://<workspace_url>/explore?
panes=<panes>&schemaVersion=<schema_version>&orgId=<org_id>
```

where:

- `org_id` is the organization ID
- `schema_version` is the schema version (should be set to the latest version, which is 1.
- `panes` is a url-encoded JSON object of panes, where each key is the pane ID and each value is an object matching the following schema:

```
{
  datasource: string; // the pane's root datasource UID, or `-- Mixed --` for mixed
 datasources
  queries: {
    refId: string; // an alphanumeric identifier for this query, must be unique
 within the pane, i.e. "A", "B", "C", etc.
    datasource: {
      uid: string; // the query's datasource UID ie: "AD7864H6422"
      type: string; // the query's datasource type-id, i.e: "loki"
    }
    // ... any other datasource-specific query parameters
  }[]; // array of queries for this pane
  range: {
    from: string; // the start time, in milliseconds since epoch
    to: string; // the end time, in milliseconds since epoch
  }
}
```

> **Note**
>
> The `from` and `to` fields also accept relative ranges as described in the Setting dashboard time range topic.

Share shortened link

The Share shortened link capability allows you to create smaller and simpler URLs of the format /goto/:uid instead of using longer URLs with query parameters. To create a shortened link to the query results, select the **Share** option in the Explore toolbar. A shortened link that is never used will automatically get deleted after seven (7) days. If a link is used at least once, it won't get deleted.

Sharing shortened links with absolute time

Short links have two options: keeping relative time (for example, from two hours ago to the current time) or absolute time (for example, from 8am to 10am). Sharing a shortened link by default will copy the time range selected, relative or absolute. Choosing the dropdown button next to the share shortened link button and selecting one of the options under **Time-Sync URL Links** will allow you to create a short link with the absolute time, meaning anyone receiving the link will see the same data you are seeing, even if they open the link at another time. This option will not affect the time range selected in your Explore view.

Query management in Explore

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can manage the queries that you have created in Explore, including a history of queries that you have run, and queries that you have starred.

Query history

Query history is a list of queries that you used in Explore. The history is stored in the Grafana database and it is not shared with other users. The retention period for queries in history is two weeks. Queries older than two weeks are automatically deleted. To open and interact with your history, select the **Query history** button in Explore.

> **Note**
>
> Starred (favorited) queries are not subject to the two weeks retention period and they are not deleted.

View query history

Query history lets you view the history of your querying. For each individual query, you can:

- Run the query.
- Create and/or edit a comment.
- Copy a query to the clipboard.
- Copy a shortened link with the query to the clipboard.
- Star (favorite) a query.

Manage favorite queries

All queries that have been starred in the Query history tab are displayed in the Starred tab. This allows you to access your favorite queries faster and to reuse these queries without typing them from scratch.

Sorting query history

By default, query history shows you the most recent queries. You can sort your history by date or by data source name in ascending or descending order.

To sort your query history

1. Select the **Sort queries by** field.
2. Select one of the following options:
   - **Newest first**
   - **Oldest first**

Filtering query history

You can filter your query history in Query history and Starred tab to a specific data source.

To filter history to a data source

1. Select the **Filter queries for specific data source(s)** field.
2. Select the data source for which you would like to filter your history. You can select multiple data sources.

> **Note**
>
> Queries ran using the Mixed data source will appear only when filtering for Mixed, and not when filtering by their individual data sources.

In the **Query history** tab it is also possible to filter queries by date using the slider:

- Use the vertical slider to filter queries by date.
- Adjust the start date by dragging the bottom handle.
- Adjust the end date by dragging the top handle.

Searching in query history

You can search in your history across queries and your comments. Search is possible for queries in the Query history tab and Starred tab.

To search in query history

1. Select the **Search queries** field.
2. Enter the term you are searching for into search field.

Query history settings

You can customize the query history in the Settings tab. Options are described in the following table.

Setting	Default value
Change the default active tab	Query history tab

> **Note**
>
> Query history settings are global, and applied to both panels in split mode.

Logs in Explore

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Explore allows you to investigate your logs in different data sources, including:

- OpenSearch
- Amazon CloudWatch
- InfluxDB
- Loki

During an infrastructure monitoring and incident response, you can dig deeper into the metrics and logs to find the cause. Explore also allows you to correlate logs with other telemetry signals such as metrics, traces, or profiles, by viewing them side-by-side.

The results of log queries are displayed as individual log lines and as a graph showing the logs volume for the selected time period.

Logs volume

When working with data sources that support a full range logs volume, Explore automatically displays a graph showing the log distribution for all the entered log queries. This feature is currently supported by OpenSearch and Loki data sources.

Note

In Loki, this full range log volume is rendered by a metric query which can be expensive depending on the time range queried. This query can be particularly challenging to process for smaller Loki installations. To mitigate this, you can use a proxy like nginx in front of Loki to set a custom timeout (for example, 10 seconds) for these queries. Log volume histogram queries can be identified by looking for queries with the HTTP header `X-Query-Tags` with value `Source=logvolhist`; these headers are added by Grafana to all log volume histogram queries.

If the data source does not support loading the full range logs volume, the logs model calculates a time series by counting log rows and organizing them into buckets based on an automatically calculated time interval. The timestamp of the first log row is used to anchor the start of the logs volume in the results. The end of the time series is anchored to the time picker's **To** range. This way, you can still analyze and visualize log data efficiently even when the data source doesn't offer full range support.

Logs

In the following sections, you will find detailed explanations of how to visualize and interact with individual logs in Explore.

Logs navigation

Logs navigation, at the right side of the log lines, can be used to easily request additional logs. You can do this by clicking the **Older logs** button at the bottom of the navigation. This is especially useful when you reach the line limit and you want to see more logs. Each request that is run from the navigation is then displayed in the navigation as separate page. Every page shows `from` and `to` timestamps of the incoming log lines. You can see previous results by clicking on each page. Explore caches the last five requests run from the logs navigation, so you're not re-running the same queries when clicking on the pages, saving time and resources.

Visualization options

You can customize how logs are displayed and select which columns are shown.

Option	Description
Time	Shows or hides the time column. This is the timestamp associated with the log line as reported from the data source.
Unique labels	Shows or hides the unique labels column that includes only non-common labels. All common labels are displayed above.
Wrap lines	Set this to `true` if you want the display to use line wrapping. If set to `false`, it will result in horizontal scrolling.
Prettify JSON	Set this to `true` to pretty print all JSON logs. This setting does not affect logs in any format other than JSON.
Deduplication	Log data can be very repetitive and Explore can help by hiding duplicate log lines. There are a few different deduplication algorithm s that you can use. **Exact** matches are done on the whole line except for date fields. **Numbers** matches are done on the line after stripping out numbers such as durations, IP addresses, and so on. **Signature** is the most aggressive deduplication as it strips all letters and numbers and matches on the remaining whitespace and punctuation.
Display results order	You can change the order of received logs from the default descending order (newest first) to ascending order (oldest first).

Download log lines

To download log results in either `txt` or `json` format, use the **Download** button. This feature allows you to save the log data for further analysis or to share it with others in a convenient and accessible format.

Log result meta information

Above the received log lines you can find essential meta information, including:

- **Number of received logs** – Indicates the total count of logs received for the current query or time range.
- **Error** – Displays possible error in your log results.
- **Common labels** – Shows common labels.
- **Total bytes processed** – Represents the cumulative size of the log data processed in bytes.

> **Note**
>
> The availability of certain meta information may depend on the data source, and as a result, you may only see some of these details for specific data sources.

Escaping newlines

Explore automatically detects some incorrectly escaped sequences in log lines, such as newlines (`\n`, `\r`) or tabs (`\t`). When it detects such sequences, Explore provides an **Escape newlines** option.

To automatically fix incorrectly escaped sequences that Explore has detected

1. Choose **Escape newlines** to replace the sequences.
2. Manually review the replacements to confirm their correctness.

Explore replaces these sequences. When it does so, the option will change from **Escape newlines** to **Remove escaping**. Evaluate the changes as the parsing may not be accurate based on the input received. You can revert the replacements by selecting **Remove escaping**.

Log level

For the logs where a `level` label is specified, we use the value of this label to determine the log level and update color of each log line accordingly. If the log doesn't have specified level label, we try to find out if its content matches any of the supported expressions (see the following table for more information). The log level is always determined by the first match. In the case where Grafana is not able to infer a log level field, it will be visualized with an unknown log level.

> **Note**
>
> If you use a Loki data source and the `level` is part of your log line, you can use parsers (JSON, logfmt, regex,..) to extract the level information into a level label that is used to determine the level value. This will allow the histogram to show the various log levels as separate bars.

Supported log levels and mapping of log level abbreviation and expressions:

Log level	Color	Supported expressions
critical	purple	emerg, fatal, alert, crit, critical
error	red	err, eror, error
warning	yellow	warn, warning
info	green	info, information, informati onal, notice
debug	blue	dbug, debug
trace	light blue	trace
unknown	grey	*

Highlight searched words

When your query includes specific words or expressions to search for, Explore will highlight these in the log lines for better visibility. This highlighting feature makes it easier to identify and focus on the relevant content in your logs.

> **Note**
>
> The ability to highlight search words may vary depending on the data source. For some data sources, the highlighting of search words may not be available.

Log details view

In Explore, each log line has an expandable section called *Log details* that can be opened by choosing the log line. The Log details view provides additional information and exploration options in the form of *Fields* and *Links* attached to the log lines, enabling a more robust interaction and analysis.

Fields

Within the Log details view, you can filter displayed fields in two ways: a positive filter (to focus on an specific field) and a negative filter (to exclude certain fields). These filters will update the corresponding query that produced the log line, adding equality and inequality expressions accordingly. If the data source has support, as is the case for Loki and OpenSearch, log details will check if the field is already present in the current query showing and active state (for positive filters only), allowing you to toggle it off the query, or changing the filter expression from positive to negative.

You can select a subset of fields to visualize in the logs list instead of the complete log line by clicking on the eye icon. Each field has a stats icon to display statistics in relation to all displayed logs.

Links

Grafana offers the functionality of data links or correlations, enabling you to convert any part of a log message into an internal or external link. These links can be used to navigate to related data or external resources, providing a seamless and convenient way to explore further information.

Log context

Log context displays additional lines of context surrounding a log entry that matches a particular search query. This can be helpful in understanding the log entry's context, and is similar to the `-C` parameter in the `grep` command.

You may encounter long lines of text that make it difficult to read and analyze the context around each log entry. This is where the **Wrap lines** toggle can come in handy. By enabling this toggle, Grafana will automatically wrap long lines of text so that they fit within the visible width of the viewer. This can make it easier to read and understand the log entries.

The **Open in split view** button allows you to execute the context query for a log entry in a split screen in the Explore view. Choosing this button will open a new Explore pane with the context query displayed alongside the log entry, making it easier to analyze and understand the surrounding context.

The log context query can also be opened in a new browser tab by pressing the `Ctrl` (or `Cmd`) key while choosing the button to open the context modal. When opened in a new tab, the previously selected filters are applied as well.

Copy log line

You can easily copy the content of a selected log line to your clipboard by choosing the **Copy log line** button.

Copy link to log line

Linking of log lines in Grafana allows you to quickly navigate to specific log entries for precise analysis. By choosing the **Copy shortlink** button for a log line, you can generate and copy a short URL that provides direct access to the exact log entry within an absolute time range. When you open the link, Grafana will automatically scroll to the corresponding log line and highlight it with a blue background, making it easy to identify and focus on the relevant information.

> **Note**
>
> This is only supported in Loki and other data sources that provide an `id` field.

Tracing in Explore

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can use Explore to visualize traces from tracing data sources.

The following data sources are supported.

- Tempo (supported ingestion formats: OpenTelemetry, Jaeger, and Zipkin)
- Jaeger
- AWS X-Ray
- Zipkin

For information on how to configure queries for the data sources listed above, refer to the documentation for specific data source.

Query editor

You can query and search tracing data using a data source's query editor.

Each data source can have its own query editor. The query editor for the Tempo data source is slightly different than the query editor for the Jaeger data source.

For information on querying each data source, refer to their documentation.

- Tempo
- Jaeger
- AWS X-Ray
- Zipkin

Trace View explanation

This section explains the elements of the Trace View dashboard.

Header

The header of the trace view has the following elements:

- Header title – Shows the name of the root span and trace ID.
- Search – Highlights spans containing the searched text.
- Metadata – Various metadata about the trace.

Minimap

Shows condensed view or the trace timeline. Drag your pointer over the minimap to zoom into smaller time range. Zooming will also update the main timeline, so it is easy to see shorter spans. Hovering over the minimap, when zoomed, will show the **Reset Selection** button which resets the zoom.

Span filters

Using span filters, you can filter your spans in teh trace timeline viewer. The more filters you add, the more specific are the filtered spans.

You can add one or more of the following filters:

- Resource service name
- Span name
- Duration
- Tags (which include tags, process tags, and log fields)

To show only the spans you have matched, choose the **Show matches only** toggle.

Timeline

Shows list of spans within the trace. Each span row consists of these components:

- Expand children button – Expands or collapses all the children spans of the selected span.
- Service name – Name of the service that logged the span.

- Operation name – Name of the operation that this span represents.
- Span duration bar – Visual representation of the operation duration within the trace.

Span details

Choosing the span row shows span details, including the following.

- Operation name
- Span metadata
- Tags – Any tags associated with this span.
- Process metadata – Metadata about the process that logged this span.
- Logs – List of logs logged by this span and associated key values. In case of Zipkin logs section shows Zipkin annotations.

Trace to logs

You can navigate from a span in a trace view directly to logs relevant for that span. This is available for Tempo, Jaeger, and Zipkin data sources. Refer to their relevant documentation for instructions on how to configure each data source.

Choose the document icon to open a split view in Explore with the configured data source and query relevant logs for the span.

Trace to metrics

> **Note**
>
> This feature is currently in beta

You can navigate form a span in a trace view directly to metrics relevant for that span. This feature is available for Tempo, Jaeger, and Zipkin data sources. Refer to their relevant documentation for details on configuration.

Trace to profiles

Using Trace to profiles, you can use Grafana's ability to correlate different signales by adding the functionality to link between traces and profiles.

Node graph

You can optionally expand the node graph for the displayed trace. Depending on the data source, this can show spans of the trace as nodes in the graph, or add some additional context, including the service graph based on the current trace.

Service Graph view

The Service Graph view visualizes the span metrics (traces data for rates, error rates, and durations (RED)) and service graphs. Once the requirements are set up, this pre-configured view is immediately available.

For more information, see Tempo data source page. You can also see the service graph view page in the *Grafana Labs Tempo documentation*.

Data API

This visualization needs a specific shape of the data to be returned from the data source in order to correctly display it.

The data source needs to return data frame and set `frame.meta.preferredVisualisationType = 'trace'`.

Data frame structure

Required fields;

Field name	Type	Description
traceID	string	Identifier for the entire trace. There should be only one trace in the data frame.
spanID	string	Identifier for the current span. SpanIDs should be unique per trace.
parentSpanID	string	SpanID of the parent span to create child parent relations hip in the trace view. Can be

Field name	Type	Description
		`undefined` for root span without a parent.
serviceName	string	Name of the service this span is part of.
serviceTags	TraceKeyValuePair[]	List of tags relevant for the service.
startTime	number	Start time of the span in millisecond epoch time.
duration	number	Duration of the span in milliseconds.

Optional fields:

Field name	Type	Description
logs	TraceLog[]	List of logs associated with the current span.
tags	TraceKeyValuePair[]	List of tags associated with the current span.
warnings	string[]	List of warnings associated with the current span.
stackTraces	string[]	List of stack traces associated with the current span.
errorIconColor	string	Color of the error icon in case span is tagged with `error: true`.

For details about the types see TraceSpanRow, TraceKeyValuePair and TraceLog on GitHub.

Correlations editor in Explore

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Correlations allow users to build a link between any two data sources. For more information, including an overview of correlations, see Correlations in Grafana version 10.

Creating a correlation

You can create correlations from the Explore page.

To create a correlation

1. In your Amazon Managed Grafana workspace, navigate to the Explore page.
2. Select a data source that you would like to be the source for a new correlation.
3. Run a query producing data in a supported visualization.

   Note

   Supported visualizations are Logs and Table.

4. Choose **+ Add** in the top toolbar and select **Add correlation** (you can also select **Correlations Editor** from the Command Palette.

   Explore is now in Correlations Editor mode indicated by a blue border and top bar. You can exit Correlations Editor by choosing **Exit** in the top bar.
5. You can now create the following new correlations for the visualization with links that are attached to the data that you can use to build a new query:

   - *Logs* – links are displayed next to field values inside log details for each log row.
   - *Table* – every table cell is a link.

6. Choose a link to add a new correlation. Links are associated with a field that is used as a result field of a correlation. For more details, see Correlation configuration.
7. In the split view that opens, use the right pane to set up the target query source of the correlation. For more details, see Target query.
8. Build a target query using variables syntax with variables from the list provided at the top of the pane. The list contains sample values from the selected data row.
9. Provide a label and description (optional). A label will be used as the name of the link inside the visualization and can contain variables.
10. Provide transformations (optional; see below for details).
11. Choose **Save** in the top toolbar to save the correlation and exit Correlations Editor mode. The link used to create the correlation is replaced with a data link in each row. When the link is selected, the query you defined will run in another pane, with the variables replaced dynamically with the values from the selected row.

Transformations

Transformations allow you to extract values that exist in a field with other data. For example, using a transformation, you can extract one portion of a log line to use in a correlation. For more details on transformations in correlations, see Correlation Transformations.

After choosing one of the generated links in the editor mode, you can add transformations by selecting **Add transformation** in the **Transformations** dropdown menu.

To use a transformation in a correlation

1. Select a field to apply the transformation to. Select the portion of the field that you want to use for the transformation. For example, a log line. Once selected, the value of this field will be used to assist you in building the transformation.
2. Select the type of the transformation. See Correlation Transformations for the options and relevant settings.
3. Based on your selection, you might see one or more variables populate, or you might need to provide more specifications in options that are displayed.
4. Select **Add transformation to correlation** to add the specified variables to the list of available variables.

> **Note**
>
> For regular expressions in this dialog box, the `mapValue` referred to in other documentation is called `Variable Name` here. Grafana highlights any text that matches the expression in the field value. Use regular expression capture groups to select what portion of the match should be extracted. When a valid regular expression is provided, the variable and the value of that variable appear below the `Variable Name` field.

Correlations examples

The following examples show how to create correlations using the Correlations Editor in Explore. If you'd like to follow these examples, make sure to set up a test data source.

Creating a text to graph correlation

This example shows how to create a correlation using Correlations Editor in Explore.

Correlations allow you to use results of one query to run a new query in any data source. In this example, you will run a query that renders tabular data. The data will be used to run a different query that yields a graph result.

To follow this example, make sure you have set up a test data source.

To create a text to graph correlation

1. In Grafana, navigate to **Explore**.
2. Select the **test data source** from the dropdown menu at the top left of the page.
3. Choose **+ Add** in the dropdown menu to the right and select **Add correlation**.
4. Explore is now in Correlations Editor mode, indicated by a blue border.
5. Select the following scenario from the scenario dropdown menu: **CSV File**.
6. Select the file, **population_by_state.csv**. Each cell is a link that you can click on to begin creating a new correlation.
7. Click on any cell in the `State` column to create a new correlation that attaches a data link to that entry. For example, select `California`.
8. In the split view, select the same data source you selected in the left pane. The helper above the query editor contains all available variables you can use the target query. Variables contain all data fields (table columns) from the selected row.

9. In the **Scenario** menu, select **CSV Metric Values**. The `String Input` field in the Query editor provides variables with population values for each year: `${1980},${2000},${2020}`. This will generate a graph using variable values.
10. In the Query Editor **Alias** field, enter `${State}`.

    Run a query to see that it produces a graph using sample values from the variables.
11. Choose **Save** to save the correlation and exit the Correlations Editor.

    After the correlation is saved, Explore will rerun the query in the left pane. By clicking a state name, the query on the right is rerun with values from the row being inserted into the CSV, thus changing the graph. The query is rerun with updated values every time you click on a state name.

You can apply the same steps to any data source. Correlations allow you to create links in visualizations to run dynamic queries based on selected data. In this example we used data returned by a query to build a new query generating different visualization using the same data source. However, you can create correlations between any data sources to create custom exploration flows.

Creating a logs to table correlation

In this example, you will create a correlation to demonstrate how to use transformations to extract values from the log line and another field.

To follow this example, make sure you have set up a test data source.

To create a logs to table correlation

1. In Grafana, navigate to **Explore**.
2. Select the **test data source** from the dropdown menu at the top left of the page.
3. Choose **+ Add** in the dropdown menu to the right and select **Add correlation**.
4. Explore is now in Correlations Editor mode, indicated by a blue border.
5. In the **Scenario** menu, select **Logs**.
6. Expand a log line to see the correlation links. Select `Correlate with hostname`.
7. Explore opens in split view. Select the same data source you selected in the left pane. The helper above the query editor contains all available variables you can use the target query.
8. Expand the transformations section, and click **Add transformation**.

9. In the **Field** dropdown menu, select **message**. The log line shows up as example data.
10. Under **Type**, select **Logfmt**. This populates the list of variables.
11. Choose **Add transformation to correlation**.
12. Choose **Add transformation** again and under **Field**, select **hostname**.
13. Under **Type**, select **Regular expression**.
14. Under **Expression**, enter the following: `-([0-9]\*)`. This selects any numbers to the right of the dash.
15. Under **Variable Name**, enter the following: `hostNumber`. This populates the list of variables.
16. Choose **Add transformation to correlation** to add it to the other variables.
17. In the data source editor, open the **Scenario** dropdown menu and select **CSV Content**.
18. In the text box below, provide the following and save the correlation:

```
time,msg,hostNumber,status
${time},${msg},${hostNumber},${status}
```

This closes the split view and reruns the left query. Expand any log line to see the correlation button. Chooseing the correlation button opens the split view with the `time` (a field), `msg` (extracted with *logfmt* from the log line), `host number` (extracted with *regex* from the `hostname`) and the `status` (extracted with *logfmt* from the log line).

Inspector in Explore

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The inspector helps you understand and troubleshoot your queries. You can inspect the raw data, export that data to a comma-separated values (CSV) file, export log results in TXT format, and view query requests.

Inspector UI

The inspector has the following tabs:

- **Stats tab** – Shows how long your query takes and how much it returns.
- **Query tab** – Shows you the requests to the server sent when Grafana queries the data source.
- **JSON tab** – Allows you to view and copy the data JSON and data frame structure JSON.
- **Data tab** – Shows the raw data returned by the query.
- **Error tab** – Shows the error. Only visible when query returns error.

Inspector tasks

You can perform a variety of tasks in the Explore inspector.

Open the Inspector

After you run the query you would like to inspect, select the **Inspector** button.

The inspector pane opens on the bottom of the screen.

Inspect raw query results

You can view raw query results, that is the data returned by the query in a table.

In the **Inspector** tab, click the **Data** tab.

For multiple queries or for queries multiple nodes, there are additional options.

- **Show data frame:** Select the result set data you want to view.
- **Series joined by time:** View the raw data from all of your queries at once, one result set per column. You can click a column heading to sort the data.

Download raw query results as CSV

Grafana generates a CSV file in your default browser download location. You can open it in the viewer of your choice.

1. In the **Inspector** tab, get raw query results by following the instructions above.
2. Refine the query settings until you can see the raw data that you want to export.
3. Choose **Download CSV**.

In order to download a CSV file specifically formatted for Excel, expand **Data options** and then turn on the **Download for Excel** toggle before you select the **Download CSV** option.

Download log results as TXT

You can generate a TXT file of the logs you are currently viewing, by selecting **Download logs** in the **Inspector**tab.

Download trace results

Based on the data source type, Grafana generates a JSON file for the trace results in one of the supported formats: Jaeger, Zipkin, or OTLP formats.

1. Open the Inspector.
2. Inspect the log query results. Refine the results until you see the raw logs that you want to export.
3. Choose **Download logs**.

Inspect query performance

The **Stats** tab displays statistics that tell you how long your query takes, how many queries you send, and the number of rows returned. This information can help you troubleshoot your queries, especially if any of the numbers are unexpectedly high or low.

Statistics are displayed in read-only format.

View JSON model

You can explore and export data as well as data frame JSON models.

To view the JSON model

1. In the Inspector panel, click the **JSON** tab.
2. From the **Select source** dropdown, choose one of the following options:
   - **Data** – Displays a JSON object representing the data that was returned to Explore.
   - **DataFrame structure** – Displays the raw result set.
3. You can expand or collapse portions of the JSON to view separate sections. You can also select the **Copy to clipboard** option to copy JSON body and paste it into another application.

View raw request and response to data source

As you are working with Explore and the Inspector tab, you can view the raw request and response data that you are generating with a query. In the Inspector, select the **Query** tab and choose **Refresh** to see the raw data.

Grafana sends the query to the server and displays the result. You can drill down on specific portions of the query, expand or collapse all of it, or copy the data to the clipboard to use in other applications.

Correlations in Grafana version 10

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can create interactive links for Explore visualizations to run queries related to presented data by setting up Correlations.

A correlation defines how data in one data source is used to query data in another data source. Some examples:

- An application name returned in a logs data source can be used to query metrics related to that application in a metrics data source.
- A user name returned by an SQL data source can be used to query logs related to that particular user in a logs data source.

Explore takes user-defined correlations to display links inside the visualizations. You can click on a link to run the related query and see results in Explore Split View.

Explore visualizations that currently support showing links based on correlations:

- Logs
- Table

You can configure correlations using the **Administration > Plugins and data > Correlations** page in Grafana or directly in Explore.

Topics

- Correlation configuration
- Create a new correlation

Correlation configuration

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Each correlation is configured with the following options:

Label

Link label, shown in the visualization.

Description

Optional description.

Source data source

The source of results that have links displayed.

Results field

Defines where the link is shown in a visualization.

Target query

The target query run when a link is clicked.

Transformations

Optional manipulations applied to the source data before it is passed to the target query.

For details about creating a correlation, see Create a correlation.

Source data source and result field

Links are shown in Explore visualizations for the results from the correlation's source data source. A link is assigned to one of the fields from the result provided in the correlation configuration (the results field). Each visualization displays fields with links in a different way.

Target query

The target query is run when a link is clicked in the visualization. You can use the query editor of the selected target data source to specify the target query. Source data results can be accessed inside the target query with variables.

Correlation Variables

You can use variables inside the target query to access the source data related to the query. Correlations use Grafana variable syntax. Variables are filled with values from the source results when the link is chosen. There are two types of variables you can use:

- Field variables (to access field values and labels).
- Correlation variables (to access field values and transformations).

Example: If source results contain a field called `employee`, the value of the field can be accessed with:

- A field variable `${__data.fields.employee}`.
- A correlation variable that maps the field value above to `${employee}`.

In addition to mapping field values to shorter variable names, more correlation variables can be created by applying transformations to existing fields.

Correlation creates a data link only if all variables have values in the selected data row. Global variables are the exception to this rule and are not required to be filled in from the returned data. These variables are interpolated automatically by data sources.

Correlation Transformations

Transformations provide a way to extract more variables out of field values. The output of transformations is a set of new variables that can be accessed as any other variable.

There are two types of transformations: logfmt and regular expression.

Each transformation uses a selected field value as the input. The output of a transformation is a set of new variables based on the type and options of the transformation.

Logfmt transformation

The logfmt transformation deconstructs a field value containing text formatted with logfmt key/value pairs. Each pair becomes a variable with the key being the name of the variable.

The logfmt transformation only requires specifying the input field name if you would like the transformation to apply to a different field than the results field. Example output variables for `field = "host=srv001 endpoint=/test app=foo"`:

name	value
host	srv001
endpoint	/test
app	foo

Regular expression transformation

The regular expression transformation deconstructs a field value based on the provided regular expression.

Regular expression transformation options:

field

Input field name

expression

Regular expression. Named capture groups are mapped to variables matching the group name. If non-named matching groups are used a variable is created out of the first match. The value overrides the variable matching the input field or a new variable is created if mapValue is provided (see examples in the following table).

mapValue

Used with simple regex groups without named matching groups. By default, the first match overrides the variable with the name of the field that is used as the input. To change that default behavior you can specify the mapValue property. The provided name is used to create a new variable. This can be useful if your target query requires both the exact value and a part of the value extracted with the transformation.

Example: Assuming the selected field name is `employee` and the field value is `John Doe`.

Various output variables based on expression and mapValue options:

expression	mapValue	output variables	comment
`/\w+ (\w+)/`	-	`employee=Doe`	No mapValue provided. The first matching is mapped to the existing field name variable (`employee`).
`/(\w+) (\w+)/`	name	`name=John`	The first matching is mapped to a new variable called `name`.
`/(?\w+) (?\w+)/`	-	`firstName=John , lastName=Doe`	When named groups are used they are the names of the output variables and mapValue is ignored.
`/(?\w+) (?\w+)/`	name	`firstName=John , lastName=Doe`	Same as above

Create a new correlation

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can create correlations in the Explore correlations editor, or using the Grafana **Administration** page in your Amazon Managed Grafana workspace.

Prerequisites

You must have permission to add new correlations. Only users with write permissions to data sources can define new correlations.

Creating a correlation in Explore's correlations editor

You can create a correlation in the Explore correlation editor. For more details, see Creating a correlation.

Creating a correlation in the Administration page

You can use the Grafana console **Administration** page to create a correlation.

To create a correlation in the Administration page

1. Go to the **Administration** section in Grafana.
2. Under **Plugins and data**, open the **Correlations** page.
3. Choose the **Add** button in the top right corner.
4. Provide a **label** for the correlation.
5. (Optional) Provide an **description**.
6. Go to the next page.
7. Provide **target data source**.
8. Provide **target query** using variables.

9. Go to the next page.
10. Provide **source data source**.
11. Provide **results field**.
12. Add transformations if you need variables that are not fields in the source data source.
13. Choose **Add** to add a new transformation.
14. Select the type of a transformation.
15. Configure transformation depending on the selected type.
16. Save the correlation.

You can edit a correlation in the same way, but when editing, you can't change the selected data sources.

Alerts in Grafana version 10

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

With Grafana v10, Amazon Managed Grafana includes access to an updated alerting system, *Grafana alerting*, that centralizes alerting information in a single, searchable view. Grafana alerting was introduced as an optional feature in Grafana v8, and GrafanaLabs has announced the removal of legacy alerting in version 11.

Note

This documentation covers Grafana alerting. For information on legacy alerting, see Classic dashboard alerts.

Grafana Alerting allows you to learn about problems in your systems moments after they occur.

Monitor your incoming metrics data or log entries and set up your Alerting system to watch for specific events or circumstances and then send notifications when those things are found.

In this way, you eliminate the need for manual monitoring and provide a first line of defense against system outages or changes that could turn into major incidents.

Using Grafana Alerting, you create queries and expressions from multiple data sources — no matter where your data is stored — giving you the flexibility to combine your data and alert on your metrics and logs in new and unique ways. You can then create, manage, and take action on your alerts from a single, consolidated view, and improve your team's ability to identify and resolve issues quickly.

With Mimir and Loki alert rules you can run alert expressions closer to your data and at massive scale, all managed by the Grafana UI you are already familiar with.

> **Note**
>
> If you are migrating from an earlier version of Grafana, where you used the legacy Grafana alerting, you might find it helpful to see the differences between the legacy alerting and the new Grafana alerting.

Key features and benefits

One page for all alerts

A single Grafana Alerting page consolidates both Grafana-managed alerts and alerts that reside in your Prometheus-compatible data source in one single place.

Multi-dimensional alerts

Alert rules can create multiple individual alert instances per alert rule, known as multi-dimensional alerts, giving you the power and flexibility to gain visibility into your entire system with just a single alert rule. You do this by adding labels to your query to specify which component is being monitored and generate multiple alert instances for a single alert rule. For example, if you want to monitor each server in a cluster, a multi-dimensional alert will alert on each CPU, whereas a standard alert will alert on the overall server.

Route alerts

Route each alert instance to a specific contact point based on labels you define. Notification policies are the set of rules for where, when, and how the alerts are routed to contact points.

Silence alerts

Silences stop notifications from getting created and last for only a specified window of time. Silences allow you to stop receiving persistent notifications from one or more alert rules. You can also partially pause an alert based on certain criteria. Silences have their own dedicated section for better organization and visibility, so that you can scan your paused alert rules without cluttering the main alerting view.

Mute timings

A mute timing is a recurring interval of time when no new notifications for a policy are generated or sent. Use them to prevent alerts from firing a specific and reoccurring period, for example, a regular maintenance period.

Similar to silences, mute timings do not prevent alert rules from being evaluated, nor do they stop alert instances from being shown in the user interface. They only prevent notifications from being created.

Design your Alerting system

Monitoring complex IT systems and understanding whether everything is up and running correctly is a difficult task. Setting up an effective alert management system is therefore essential to inform you when things are going wrong before they start to impact your business outcomes.

Designing and configuring an alert management set up that works takes time.

Here are some tips on how to create an effective alert management set up for your business:

Which are the key metrics for your business that you want to monitor and alert on?

- Find events that are important to know about and not so trivial or frequent that recipients ignore them.
- Alerts should only be created for big events that require immediate attention or intervention.
- Consider quality over quantity.

Which type of Alerting do you want to use?

- Choose between Grafana-managed Alerting or Grafana Mimir or Loki-managed Alerting; or both.

How do you want to organize your alerts and notifications?

- Be selective about who you set to receive alerts. Consider sending them to whomever is on call or a specific Slack channel.
- Automate as far as possible using the Alerting API or alerts as code (Terraform).

How can you reduce alert fatigue?

- Avoid noisy, unnecessary alerts by using silences, mute timings, or pausing alert rule evaluation.
- Continually tune your alert rules to review effectiveness. Remove alert rules to avoid duplication or ineffective alerts.
- Think carefully about priority and severity levels.
- Continually review your thresholds and evaluation rules.

Grafana alerting limitations

- When aggregating rules from other systems, the Grafana alerting system can retrieve rules from all available Amazon Managed Service for Prometheus, Prometheus, Loki, and Alertmanager data sources. It might not be able to fetch rules from other supported data sources.

> **Important**
>
> Amazon Managed Grafana has an alert evaluation timeout of 30 seconds. Queries made by alerts have a maximum duration of 30 seconds due to the high volume of queries the alert engine can generate. This timeout is not configurable. For more information, see Amazon Managed Grafana service quotas in the *AWS General Reference*.

Topics

- Overview
- Set up Alerting
- Configure alerting
- Manage your alerts

Overview

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Whether you're just starting out or you're a more experienced user of Grafana Alerting, learn more about the fundamentals and available features that help you create, manage, and respond to alerts; and improve your team's ability to resolve issues quickly.

Principles

In Prometheus-based alerting systems, you have an alert generator that creates alerts and an alert receiver that receives alerts. For example, Prometheus is an alert generator and is responsible for evaluating alert rules, while Alertmanager is an alert receiver and is responsible for grouping, inhibiting, silencing, and sending notifications about firing and resolved alerts.

Grafana Alerting is built on the Prometheus model of designing alerting systems. It has an internal alert generator responsible for scheduling and evaluating alert rules, as well as an internal alert receiver responsible for grouping, inhibiting, silencing, and sending notifications. Grafana doesn't use Prometheus as its alert generator because Grafana Alerting needs to work with many other data sources in addition to Prometheus. However, it does use Alertmanager as its alert receiver.

Alerts are sent to the alert receiver where they are routed, grouped, inhibited, silenced and notified. In Grafana Alerting, the default alert receiver is the Alertmanager embedded inside Grafana, and is referred to as the Grafana Alertmanager. However, you can use other Alertmanagers too, and these are referred to as External Alertmanagers.

Fundamentals

The following provides an overview of the different parts of Grafana alerting.

Alert rules

An alert rule is a set of criteria that determine when an alert should fire. It consists of one or more queries and expressions, a condition which needs to be met, an interval which determines how

often the alert rule is evaluated, and a duration over which the condition must be met for an alert to fire.

Alert rules are evaluated over their interval, and each alert rule can have zero, one, or any number of alerts firing at a time. The state of the alert rule is determined by its most `severe` alert, which can be one of Normal, Pending, or Firing. For example, if at least one of an alert rule's alerts are firing then the alert rule is also firing. The health of an alert rule is determined by the status of its most recent evaluation. These can be OK, Error, and NoData.

A very important feature of alert rules is that they support custom annotations and labels. These allow you to instrument alerts with additional metadata such as summaries and descriptions, and add additional labels to route alerts to specific notification policies.

Alerts

Alerts are uniquely identified by sets of key/value pairs called Labels. Each key is a label name and each value is a label value. For example, one alert might have the labels `foo=bar` and another alert might have the labels `foo=baz`. An alert can have many labels such as `foo=bar,bar=baz` but it cannot have the same label twice such as `foo=bar,foo=baz`. Two alerts cannot have the same labels either, and if two alerts have the same labels such as `foo=bar,bar=baz` and `foo=bar,bar=baz` then one of the alerts will be discarded. Alerts are resolved when the condition in the alert rule is no longer met, or the alert rule is deleted.

In Grafana Managed Alerts, alerts can be in Normal, Pending, Alerting, No Data or Error states. In Data source Managed Alerts, such as Mimir and Loki, alerts can be in Normal, Pending and Alerting, but not NoData or Error.

Contact points

Contact points determine where notifications are sent. For example, you might have a contact point that sends notifications to an email address, to Slack, to an incident management system (IRM) such as Grafana OnCall or Pagerduty, or to a webhook.

The notifications that are sent from contact points can be customized using notification templates. You can use notification templates to change the title, message, and structure of the notification. Notification templates are not specific to individual integrations or contact points.

Notification policies

Notification policies group alerts and then route them to contact points. They determine when notifications are sent, and how often notifications should be repeated.

Alerts are matched to notification policies using label matchers. These are human-readable expressions that assert if the alert's labels exactly match, do not exactly match, contain, or do not contain some expected text. For example, the matcher `foo=bar` matches alerts with the label `foo=bar` while the matcher `foo=~[a-zA-Z]+` matches alerts with any label called foo with a value that matches the regular expression `[a-zA-Z]+`.

By default, an alert can only match one notification policy. However, with the `continue` feature alerts can be made to match any number of notification policies at the same time. For more information about notification policies, see Notification Policies.

Silences and mute timings

Silences and mute timings allow you to pause notifications for specific alerts or even entire notification policies. Use a silence to pause notifications on an ad-hoc basis, such as while working on a fix for an alert; and use mute timings to pause notifications at regular intervals, such as during regularly scheduled maintenance windows.

Topics

- Data sources and Grafana alerting
- Alerting on numeric data
- Labels and annotations
- About alert rules
- Alertmanager
- Contact points
- Notifications
- Alerting high availability

Data sources and Grafana alerting

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

There are a number of data sources that are compatible with Grafana Alerting. Each data source is supported by a plugin. Grafana alerting requires that data source plugins be *backend* plugins, in order to evaluate rules using the data source, because the evaluation engine runs on the backend. Plugins must also specify that they are compatible with Grafana alerting.

Data sources are added and updated over time. The following data sources are known to be compatible with Grafana alerting.

- Connect to an Amazon CloudWatch data source
- Connect to an Azure Monitor data source
- Connect to an Amazon OpenSearch Service data source
- Connect to a Google Cloud Monitoring data source
- Connect to a Graphite data source
- Connect to an InfluxDB data source
- Connect to a Loki data source
- Connect to a Microsoft SQL Server data source
- Connect to a MySQL data source
- Connect to an OpenTSDB data source
- Connect to a PostgreSQL data source
- Connect to Amazon Managed Service for Prometheus and open-source Prometheus data sources
- Connect to a Jaeger data source
- Connect to a Zipkin data source
- Connect to a Tempo data source
- Configure a TestData data source for testing

For more detailed information about data sources and data source plugins in Amazon Managed Grafana, see Connect to data sources.

Alerting on numeric data

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This topic describes how Grafana handles alerting on numeric rather than time series data.

Among certain data sources, numeric data that is not time series can be directly alerted on, or passed into Server Side Expressions (SSE). This allows for more processing and resulting efficiency within the data source, and it can also simplify alert rules. When alerting on numeric data instead of time series data, there is no need to reduce each labeled time series into a single number. Instead labeled numbers are returned to Grafana instead.

Tabular data

This feature is supported with backend data sources that query tabular data:

- SQL data sources such as MySQL, Postgres, MSSQL, and Oracle.
- The Azure Kusto based services: Azure Monitor (Logs), Azure Monitor (Azure Resource Graph), and Azure Data Explorer.

A query with Grafana managed alerts or SSE is considered numeric with these data sources, if:

- The “Format AS” option is set to “Table” in the data source query.
- The table response returned to Grafana from the query includes only one numeric (e.g. int, double, float) column, and optionally additional string columns.

If there are string columns, then those columns become labels. The name of a column becomes the label name, and the value for each row becomes the value of the corresponding label. If multiple rows are returned, then each row should be uniquely identified their labels.

Example

For a MySQL table called “DiskSpace”:

Time	Host	Disk	PercentFree
2021-June-7	web1	/etc	3

Time	Host	Disk	PercentFree
2021-June-7	web2	/var	4
2021-June-7	web3	/var	8
...	...	...	...

You can query the data filtering on time, but without returning the time series to Grafana. For example, an alert that would trigger per Host, Disk when there is less than 5% free space:

```
SELECT Host , Disk , CASE WHEN PercentFree  < 5.0 THEN PercentFree  ELSE 0 END FROM (
       SELECT
          Host,
          Disk,
          Avg(PercentFree)
       FROM DiskSpace
       Group By
          Host,
          Disk
       Where __timeFilter(Time)
```

This query returns the following Table response to Grafana:

Host	Disk	PercentFree
web1	/etc	3
web2	/var	4
web3	/var	0

When this query is used as the **condition** in an alert rule, then the non-zero will be alerting. As a result, three alert instances are produced:

Labels	Status
{Host=web1,disk=/etc}	Alerting

Labels	Status
{Host=web2,disk=/var}	Alerting
{Host=web3,disk=/var}	Normal

Labels and annotations

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Labels and annotations contain information about an alert. Both labels and annotations have the same structure: a set of named values; however their intended uses are different. An example of label, or the equivalent annotation, might be `alertname="test"`.

The main difference between a label and an annotation is that labels are used to differentiate an alert from all other alerts, while annotations are used to add additional information to an existing alert.

For example, consider two high CPU alerts: one for `server1` and another for `server2`. In such an example, we might have a label called `server` where the first alert has the label `server="server1"` and the second alert has the label `server="server2"`. However, we might also want to add a description to each alert such as `"The CPU usage for server1 is above 75%."`, where `server1` and `75%` are replaced with the name and CPU usage of the server (please refer to the documentation on Templating labels and annotations for how to do this). This kind of description would be more suitable as an annotation.

Labels

Labels contain information that identifies an alert. An example of a label might be `server=server1`. Each alert can have more than one label, and the complete set of labels for an alert is called its label set. It is this label set that identifies the alert.

For example, an alert might have the label set `{alertname="High CPU usage",server="server1"}` while another alert might have the label set `{alertname="High CPU usage",server="server2"}`. These are two separate alerts because although their `alertname` labels are the same, their `server` labels are different.

The label set for an alert is a combination of the labels from the datasource, custom labels from the alert rule, and a number of reserved labels such as `alertname`.

Custom Labels

Custom labels are additional labels from the alert rule. Like annotations, custom labels must have a name, and their value can contain a combination of text and template code that is evaluated when an alert is fired. Documentation on how to template custom labels can be found here.

When using custom labels with templates, it is important to make sure that the label value does not change between consecutive evaluations of the alert rule as this will end up creating large numbers of distinct alerts. However, it is OK for the template to produce different label values for different alerts. For example, do not put the value of the query in a custom label as this will end up creating a new set of alerts each time the value changes. Instead use annotations.

It is also important to make sure that the label set for an alert does not have two or more labels with the same name. If a custom label has the same name as a label from the datasource then it will replace that label. However, should a custom label have the same name as a reserved label then the custom label will be omitted from the alert.

Annotations

Annotations are named pairs that add additional information to existing alerts. There are a number of suggested annotations in Grafana such as `description`, `summary`, `runbook_url`, `dashboardUId` and `panelId`. Like custom labels, annotations must have a name, and their value can contain a combination of text and template code that is evaluated when an alert is fired. If an annotation contains template code, the template is evaluated once when the alert is fired. It is not re-evaluated, even when the alert is resolved. Documentation on how to template annotations can be found here.

Topics

- How label matching works
- Labels in Grafana Alerting
- Templating labels and annotations

How label matching works

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use labels and label matchers to link alert rules to notification policies and silences. This allows for a very flexible way to manage your alert instances, specify which policy should handle them, and which alerts to silence.

A label matcher consists of 3 distinct parts, the **label**, the **value** and the **operator**.

- The **Label** field is the name of the label to match. It must exactly match the label name.
- The **Value** field matches against the corresponding value for the specified **Label** name. How it matches depends on the **Operator** value.
- The **Operator** field is the operator to match against the label value. The available operators are:

Operator	Description
=	Select labels that are exactly equal to the value.
!=	Select labels that are not equal to the value.
=~	Select labels that regex-match the value.
!~	Select labels that do not regex-match the value.

If you are using multiple label matchers, they are combined using the AND logical operator. This means that all matchers must match in order to link a rule to a policy.

Example

If you define the following set of labels for your alert:

```
{ foo=bar, baz=qux, id=12 }
```

then:

- A label matcher defined as `foo=bar` matches this alert rule.
- A label matcher defined as `foo!=bar` does *not* match this alert rule.
- A label matcher defined as `id=~[0-9]+` matches this alert rule.
- A label matcher defined as `baz!~[0-9]+` matches this alert rule.
- Two label matchers defined as `foo=bar` and `id=~[0-9]+` match this alert rule.

Exclude labels

You can also write label matchers to exclude labels.

Here is an example that shows how to exclude the label `team`. You can choose between any of these values to exclude the label.

- `team=""`
- `team!~.+`
- `team=~^$`

Labels in Grafana Alerting

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This topic explains why labels are a fundamental component of alerting.

- The complete set of labels for an alert is what uniquely identifies an alert within Grafana alerts.
- The Alertmanager uses labels to match alerts for silences and alert groups in notification policies.
- The alerting UI shows labels for every alert instance generated during evaluation of that rule.
- Contact points can access labels to dynamically generate notifications that contain information specific to the alert that is resulting in a notification.
- You can add labels to an alerting rule. Labels are manually configurable, use template functions, and can reference other labels. Labels added to an alerting rule take precedence in the event of a collision between labels (except in the case of Grafana reserved labels, see below for more information).

External Alertmanager compatibility

Grafana's built-in Alertmanager supports both Unicode label keys and values. If you are using an external Prometheus Alertmanager, label keys must be compatible with their data model. This means that label keys must only contain **ASCII letters**, **numbers**, as well as **underscores** and match the regex `[a-zA-Z_][a-zA-Z0-9_]*`. Any invalid characters will be removed or replaced by the Grafana alerting engine before being sent to the external Alertmanager according to the following rules:

- `Whitespace` will be removed.
- `ASCII characters` will be replaced with `_`.
- `All other characters` will be replaced with their lower-case hex representation. If this is the first character it will be prefixed with `_`.

> **Note**
>
> If multiple label keys are sanitized to the same value, the duplicates will have a short hash of the original label appended as a suffix.

Grafana reserved labels

> **Note**
>
> Labels prefixed with `grafana_` are reserved by Grafana for special use. If a manually configured label is added beginning with `grafana_` it will be overwritten in case of collision.

Grafana reserved labels can be used in the same way as manually configured labels. The current list of available reserved labels are:

Label	Description
grafana_folder	Title of the folder containing the alert.

Templating labels and annotations

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can use templates to include data from queries and expressions in labels and annotations. For example, you might want to set the severity label for an alert based on the value of the query, or use the instance label from the query in a summary annotation so you know which server is experiencing high CPU usage.

All templates should be written in text/template. Regardless of whether you are templating a label or an annotation, you should write each template inline inside the label or annotation that you are templating. This means you cannot share templates between labels and annotations, and instead you will need to copy templates wherever you want to use them.

Each template is evaluated whenever the alert rule is evaluated, and is evaluated for every alert separately. For example, if your alert rule has a templated summary annotation, and the alert rule has 10 firing alerts, then the template will be executed 10 times, once for each alert. You should try to avoid doing expensive computations in your templates as much as possible.

Examples

Rather than write a complete tutorial on text/template, the following examples attempt to show the most common use-cases we have seen for templates. You can use these examples verbatim, or adapt them as necessary for your use case. For more information about how to write text/template see the text/template documentation.

Print all labels, comma separated

To print all labels, comma separated, print the `$labels` variable:

```
{{ $labels }}
```

For example, given an alert with the labels `alertname=High CPU usage`, `grafana_folder=CPU alerts` and `instance=server1`, this would print:

```
alertname=High CPU usage, grafana_folder=CPU alerts, instance=server1
```

> **Note**
>
> If you are using classic conditions then `$labels` will not contain any labels from the query. Refer to the $labels variable for more information.

Print all labels, one per line

To print all labels, one per line, use a `range` to iterate over each key/value pair and print them individually. Here `$k` refers to the name and `$v` refers to the value of the current label:

```
{{ range $k, $v := $labels -}}
{{ $k }}={{ $v }}
{{ end }}
```

For example, given an alert with the labels `alertname=High CPU usage`, `grafana_folder=CPU alerts` and `instance=server1`, this would print:

```
alertname=High CPU usage
grafana_folder=CPU alerts
instance=server1
```

> **Note**
>
> If you are using classic conditions then `$labels` will not contain any labels from the query. Refer to the $labels variable for more information.

Print an individual label

To print an individual label use the `index` function with the `$labels` variable:

```
The host {{ index $labels "instance" }} has exceeded 80% CPU usage for the last 5
 minutes
```

For example, given an alert with the label `instance=server1`, this would print:

```
The host server1 has exceeded 80% CPU usage for the last 5 minutes
```

> **Note**
>
> If you are using classic conditions then `$labels` will not contain any labels from the query. Refer to the $labels variable for more information.

Print the value of a query

To print the value of an instant query you can print its Ref ID using the `index` function and the `$values` variable:

```
{{ index $values "A" }}
```

For example, given an instant query that returns the value 81.2345, this will print:

```
81.2345
```

To print the value of a range query you must first reduce it from a time series to an instant vector with a reduce expression. You can then print the result of the reduce expression by using its Ref ID instead. For example, if the reduce expression takes the average of A and has the Ref ID B you would write:

```
{{ index $values "B" }}
```

Print the humanized value of a query

To print the humanized value of an instant query use the `humanize` function:

```
{{ humanize (index $values "A").Value }}
```

For example, given an instant query that returns the value 81.2345, this will print:

```
81.234
```

To print the humanized value of a range query you must first reduce it from a time series to an instant vector with a reduce expression. You can then print the result of the reduce expression by using its Ref ID instead. For example, if the reduce expression takes the average of A and has the Ref ID B you would write:

```
{{ humanize (index $values "B").Value }}
```

Print the value of a query as a percentage

To print the value of an instant query as a percentage use the `humanizePercentage` function:

```
{{ humanizePercentage (index $values "A").Value }}
```

This function expects the value to be a decimal number between 0 and 1. If the value is instead a decimal number between 0 and 100 you can divide it by 100 either in your query or using a math expression. If the query is a range query you must first reduce it from a time series to an instant vector with a reduce expression.

Set a severity from the value of a query

To set a severity label from the value of a query use an if statement and the greater than comparison function. Make sure to use decimals (`80.0`, `50.0`, `0.0`, etc) when doing comparisons

against $values as text/template does not support type coercion. You can find a list of all the supported comparison functions here.

```
{{ if (gt $values.A.Value 80.0) -}}
high
{{ else if (gt $values.A.Value 50.0) -}}
medium
{{ else -}}
low
{{- end }}
```

Print all labels from a classic condition

You cannot use $labels to print labels from the query if you are using classic conditions, and must use $values instead. The reason for this is classic conditions discard these labels to enforce uni-dimensional behavior (at most one alert per alert rule). If classic conditions didn't discard these labels, then queries that returned many time series would cause alerts to flap between firing and resolved constantly as the labels would change every time the alert rule was evaluated.

Instead, the $values variable contains the reduced values of all time series for all conditions that are firing. For example, if you have an alert rule with a query A that returns two time series, and a classic condition B with two conditions, then $values would contain B0, B1, B2 and B3. If the classic condition B had just one condition, then $values would contain just B0 and B1.

To print all labels of all firing time series use the following template (make sure to replace B in the regular expression with the Ref ID of the classic condition if it's different):

```
{{ range $k, $v := $values -}}
{{ if (match "B[0-9]+" $k) -}}
{{ $k }}: {{ $v.Labels }}{{ end }}
{{ end }}
```

For example, a classic condition for two time series exceeding a single condition would print:

```
B0: instance=server1
B1: instance=server2
```

If the classic condition has two or more conditions, and a time series exceeds multiple conditions at the same time, then its labels will be duplicated for each condition that is exceeded:

```
B0: instance=server1
B1: instance=server2
B2: instance=server1
B3: instance=server2
```

If you need to print unique labels you should consider changing your alert rules from uni-dimensional to multi-dimensional instead. You can do this by replacing your classic condition with reduce and math expressions.

Print all values from a classic condition

To print all values from a classic condition take the previous example and replace `$v.Labels` with `$v.Value`:

```
{{ range $k, $v := $values -}}
{{ if (match "B[0-9]+" $k) -}}
{{ $k }}: {{ $v.Value }}{{ end }}
{{ end }}
```

For example, a classic condition for two time series exceeding a single condition would print:

```
B0: 81.2345
B1: 84.5678
```

If the classic condition has two or more conditions, and a time series exceeds multiple conditions at the same time, then `$values` will contain the values of all conditions:

```
B0: 81.2345
B1: 92.3456
B2: 84.5678
B3: 95.6789
```

Variables

The following variables are available to you when templating labels and annotations:

The labels variable

The `$labels` variable contains all labels from the query. For example, suppose you have a query that returns CPU usage for all of your servers, and you have an alert rule that fires when any of

your servers have exceeded 80% CPU usage for the last 5 minutes. You want to add a summary annotation to the alert that tells you which server is experiencing high CPU usage. With the `$labels` variable you can write a template that prints a human-readable sentence such as:

```
CPU usage for {{ index $labels "instance" }} has exceeded 80% for the last 5 minutes
```

> **Note**
>
> If you are using a classic condition then `$labels` will not contain any labels from the query. Classic conditions discard these labels in order to enforce uni-dimensional behavior (at most one alert per alert rule). If you want to use labels from the query in your template then follow the previous *Print all labels from a classic condition* example.

The value variable

The `$value` variable is a string containing the labels and values of all instant queries; threshold, reduce and math expressions, and classic conditions in the alert rule. It does not contain the results of range queries, as these can return anywhere from 10s to 10,000s of rows or metrics. If it did, for especially large queries a single alert could use 10s of MBs of memory and Grafana would run out of memory very quickly.

To print the `$value` variable in the summary you would write something like this:

```
CPU usage for {{ index $labels "instance" }} has exceeded 80% for the last 5 minutes:
 {{ $value }}
```

And it would look something like this:

```
CPU usage for instance1 has exceeded 80% for the last 5 minutes: [ var='A'
 labels={instance=instance1} value=81.234 ]
```

Here `var='A'` refers to the instant query with Ref ID A, `labels={instance=instance1}` refers to the labels, and `value=81.234` refers to the average CPU usage over the last 5 minutes.

If you want to print just some of the string instead of the full string then use the `$values` variable. It contains the same information as `$value`, but in a structured table, and is much easier to use then writing a regular expression to match just the text you want.

The values variable

The `$values` variable is a table containing the labels and floating point values of all instant queries and expressions, indexed by their Ref IDs.

To print the value of the instant query with Ref ID A:

```
CPU usage for {{ index $labels "instance" }} has exceeded 80% for the last 5 minutes:
 {{ index $values "A" }}
```

For example, given an alert with the labels `instance=server1` and an instant query with the value `81.2345`, this would print:

```
CPU usage for instance1 has exceeded 80% for the last 5 minutes: 81.2345
```

If the query in Ref ID A is a range query rather than an instant query then add a reduce expression with Ref ID B and replace `(index $values "A")` with `(index $values "B")`:

```
CPU usage for {{ index $labels "instance" }} has exceeded 80% for the last 5 minutes:
 {{ index $values "B" }}
```

Functions

The following functions are available to you when templating labels and annotations:

args

The `args` function translates a list of objects to a map with keys arg0, arg1 etc. This is intended to allow multiple arguments to be passed to templates.

```
{{define "x"}}{{.arg0}} {{.arg1}}{{end}}{{template "x" (args 1 "2")}}
```

```
1 2
```

externalURL

The `externalURL` function returns the external URL of the Grafana server.

```
{{ externalURL }}
```

```
https://example.com/grafana
```

graphLink

The `graphLink` function returns the path to the graphical view in Explore in Grafana version 10 for the given expression and data source.

```
{{ graphLink "{\"expr\": \"up\", \"datasource\": \"gdev-prometheus\"}" }}
```

```
/explore?left=["now-1h","now","gdev-prometheus",{"datasource":"gdev-
prometheus","expr":"up","instant":false,"range":true}]
```

humanize

The `humanize` function humanizes decimal numbers.

```
{{ humanize 1000.0 }}
```

```
1k
```

humanize1024

The `humanize1024` works similar to `humanize` but uses 1024 as the base rather than 1000.

```
{{ humanize1024 1024.0 }}
```

```
1ki
```

humanizeDuration

The `humanizeDuration` function humanizes a duration in seconds.

```
{{ humanizeDuration 60.0 }}
```

```
1m 0s
```

humanizePercentage

The `humanizePercentage` function humanizes a ratio value to a percentage.

```
{{ humanizePercentage 0.2 }}
```

```
20%
```

humanizeTimestamp

The `humanizeTimestamp` function humanizes a Unix timestamp.

```
{{ humanizeTimestamp 1577836800.0 }}
```

```
2020-01-01 00:00:00 +0000 UTC
```

match

The `match` function matches the text against a regular expression pattern.

```
{{ match "a.*" "abc" }}
```

```
true
```

pathPrefix

The `pathPrefix` function returns the path of the Grafana server.

```
{{ pathPrefix }}
```

```
/grafana
```

tableLink

The `tableLink` function returns the path to the tabular view in Explore in Grafana version 10 for the given expression and data source.

```
{{ tableLink "{\"expr\": \"up\", \"datasource\": \"gdev-prometheus\"}" }}
```

```
/explore?left=["now-1h","now","gdev-prometheus",{"datasource":"gdev-
prometheus","expr":"up","instant":true,"range":false}]
```

title

The `title` function capitalizes the first character of each word.

```
{{ title "hello, world!" }}
```

```
Hello, World!
```

toLower

The `toLower` function returns all text in lowercase.

```
{{ toLower "Hello, world!" }}
```

```
hello, world!
```

toUpper

The `toUpper` function returns all text in uppercase.

```
{{ toUpper "Hello, world!" }}
```

```
HELLO, WORLD!
```

reReplaceAll

The `reReplaceAll` function replaces text matching the regular expression.

```
{{ reReplaceAll "localhost:(.*)" "example.com:$1" "localhost:8080" }}
```

```
example.com:8080
```

About alert rules

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

An alerting rule is a set of evaluation criteria that determines whether an alert instance will fire. The rule consists of one or more queries and expressions, a condition, the frequency of evaluation, and the duration over which the condition needs to be met to start firing.

While queries and expressions select the data set to evaluate, a *condition* sets the threshold that the data must meet or exceed to create an alert.

An *interval* specifies how frequently an alerting rule is evaluated. *Duration*, when configured, indicates how long a condition must be met. The alert rules can also define alerting behavior in the absence of data.

Topics

- Alert rule types
- Recording rules
- Queries and conditions
- Alert instances
- Namespaces, folders and groups
- Alert rule evaluation
- State and health of alerting rules
- Notification templating

Alert rule types

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana supports several alert rule types. Learn more about each of the alert rule types, how they work, and decide which one is best for your use case.

Grafana managed rules

Grafana managed rules are the most flexible alert rule type. They allow you to create alerts that can act on data from any of your existing data sources.

In addition to supporting multiple data sources, you can add expressions to transform your data and express alert conditions.

In Grafana managed alerting:

- Alert rules are created within Grafana, based on one or more data sources.
- Alert rules are evaluated by the alert rule evaluation engine from within Grafana.
- Alerts are delivered using the internal Grafana Alertmanager.

> **Note**
>
> You can also configure alerts to be delivered using an external Alertmanager, or use both internal and external Alertmanagers. For more information, see Add an external alertmanager.

Data source managed rules

To create data source managed alert rules you must have a compatible Prometheus or Loki data source. You can check if your data source supports rule creation via Grafana by testing the data source and observing if the Ruler API is supported.

In data source managed alerting:

- Alert rules are created and stored within the data source itself.
- Alert rules can only be created based on Prometheus data.
- Alert rule evaluation and delivery is distributed across multiple nodes for high-availability and fault tolerance.

Choose an alert rule type

When choosing which alert rule type to use, consider the following comparison between Grafana managed alert rules and data source managed alert rules.

Feature	Grafana-managed alert rule	Loki/Mimir-managed alert rule
Create alert rules based on data from any of our supported data sources	Yes	No: You can only create alert rules that are based on Prometheus data. The data source must have the Ruler API enabled.
Mix and match data sources	Yes	No
Includes support for recording rules	No	Yes
Add expressions to transform your data and set alert conditions	Yes	No
Use images in alert notificat ions	Yes	No
Scaling	More resource intensive, depend on the database, and are likely to suffer from transient errors. They only scale vertically.	Store alert rules within the data source itself and allow for “infinite” scaling. Generate and send alert notifications from the location of your data.
Alert rule evaluation and delivery	Alert rule evaluation and delivery is done from within Grafana, using an external Alertmanager; or both.	Alert rule evaluation and alert delivery is distributed, meaning there is no single point of failure.

Recording rules

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Recording rules are only available for compatible Prometheus or Loki data sources.

A recording rule allows you to pre-compute frequently needed or computationally expensive expressions and save their result as a new set of time series. This is useful if you want to run alerts on aggregated data or if you have dashboards that query computationally expensive expressions repeatedly.

Querying this new time series is faster, especially for dashboards since they query the same expression every time the dashboards refresh.

Read more about recording rules in Prometheus.

Queries and conditions

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

In Grafana, queries play a vital role in fetching and transforming data from supported data sources, which include databases like MySQL and PostgreSQL, time series databases like Prometheus, InfluxDB and Graphite, and services like OpenSearch, Amazon CloudWatch, Azure Monitor and Google Cloud Monitoring.

For more information on supported data sources, see Data sources and Grafana alerting.

The process of executing a query involves defining the data source, specifying the desired data to retrieve, and applying relevant filters or transformations. Query languages or syntaxes specific to the chosen data source are utilized for constructing these queries.

In Alerting, you define a query to get the data you want to measure and a condition that needs to be met before an alert rule fires.

An alert rule consists of one or more queries and expressions that select the data you want to measure.

For more information on queries and expressions, see Query and transform data.

Data source queries

Queries in Grafana can be applied in various ways, depending on the data source and query language being used. Each data source's query editor provides a customized user interface that helps you write queries that take advantage of its unique capabilities.

Because of the differences between query languages, each data source query editor looks and functions differently. Depending on your data source, the query editor might provide auto-completion features, metric names, variable suggestions, or a visual query-building interface.

Some common types of query components include:

Metrics or data fields – Specify the specific metrics or data fields you want to retrieve, such as CPU usage, network traffic, or sensor readings.

Time range – Define the time range for which you want to fetch data, such as the last hour, a specific day, or a custom time range.

Filters – Apply filters to narrow down the data based on specific criteria, such as filtering data by a specific tag, host, or application.

Aggregations – Perform aggregations on the data to calculate metrics like averages, sums, or counts over a given time period.

Grouping – Group the data by specific dimensions or tags to create aggregated views or breakdowns.

> **Note**
>
> Grafana does not support alert queries with template variables. More information is available here in the Grafana Labs forums.

Expression queries

In Grafana, an expression is used to perform calculations, transformations, or aggregations on the data source queried data. It allows you to create custom metrics or modify existing metrics based on mathematical operations, functions, or logical expressions.

By leveraging expression queries, users can perform tasks such as calculating the percentage change between two values, applying functions like logarithmic or trigonometric functions, aggregating data over specific time ranges or dimensions, and implementing conditional logic to handle different scenarios.

In Alerting, you can only use expressions for Grafana-managed alert rules. For each expression, you can choose from the math, reduce, and resample expressions. These are called multi-dimensional rules, because they generate a separate alert for each series.

You can also use a classic condition, which creates an alert rule that triggers a single alert when its condition is met. As a result, Grafana sends only a single alert even when alert conditions are met for multiple series.

> **Note**
>
> Classic conditions exist mainly for compatibility reasons and should be avoided if possible.

Reduce

Aggregates time series values in the selected time range into a single value.

Math

Performs free-form math functions/operations on time series and number data. Can be used to preprocess time series data or to define an alert condition for number data.

Resample

Realigns a time range to a new set of timestamps, this is useful when comparing time series data from different data sources where the timestamps would otherwise not align.

Threshold

Checks if any time series data matches the threshold condition.

The threshold expression allows you to compare two single values. It returns `0` when the condition is false and 1 if the condition is true. The following threshold functions are available:

- Is above (x > y)
- Is below (x < y)
- Is within range (x > y1 AND x < y2)
- Is outside range (x < y1 AND x > y2)

Classic condition

Checks if any time series data matches the alert condition.

> **Note**
>
> Classic condition expression queries always produce one alert instance only, no matter how many time series meet the condition. Classic conditions exist mainly for compatibility reasons and should be avoided if possible.

Aggregations

Grafana Alerting provides the following aggregation functions to enable you to further refine your query.

These functions are available for **Reduce** and **Classic condition** expressions only.

Function	Expression	What it does
avg	Reduce / Classic	Displays the average of the values

Function	Expression	What it does
min	Reduce / Classic	Displays the lowest value
max	Reduce / Classic	Displays the highest value
sum	Reduce / Classic	Displays the sum of all values
count	Reduce / Classic	Counts the number of values in the result
last	Reduce / Classic	Displays the last value
median	Reduce / Classic	Displays the median value
diff	Classic	Displays the difference between the newest and oldest value
diff_abs	Classic	Displays the absolute value of diff
percent_diff	Classic	Displays the percentage value of the difference between newest and oldest value
percent_diff_abs	Classic	Displays the absolute value of percent_diff
count_non_null	Classic	Displays a count of values in the result set that aren't `null`

Alert condition

An alert condition is the query or expression that determines whether the alert will fire or not depending on the value it yields. There can be only one condition which will determine the triggering of the alert.

After you have defined your queries and/or expressions, choose one of them as the alert rule condition.

When the queried data satisfies the defined condition, Grafana triggers the associated alert, which can be configured to send notifications through various channels like email, Slack, or PagerDuty. The notifications inform you about the condition being met, allowing you to take appropriate actions or investigate the underlying issue.

By default, the last expression added is used as the alert condition.

Recovery threshold

To reduce the noise of flapping alerts, you can set a recovery threshold different to the alert threshold.

Flapping alerts occur when a metric hovers around the alert threshold condition and may lead to frequent state changes, resulting in too many notifications being generated.

Grafana-managed alert rules are evaluated for a specific interval of time. During each evaluation, the result of the query is checked against the threshold set in the alert rule. If the value of a metric is above the threshold, an alert rule fires and a notification is sent. When the value goes below the threshold and there is an active alert for this metric, the alert is resolved, and another notification is sent.

It can be tricky to create an alert rule for a noisy metric. That is, when the value of a metric continually goes above and below a threshold. This is called flapping and results in a series of firing - resolved - firing notifications and a noisy alert state history.

For example, if you have an alert for latency with a threshold of 1000ms and the number fluctuates around 1000 (say 980 ->1010 -> 990 -> 1020, and so on) then each of those will trigger a notification.

To solve this problem, you can set a (custom) recovery threshold, which basically means having two thresholds instead of one. An alert is triggered when the first threshold is crossed and is resolved only when the second threshold is crossed.

For example, you could set a threshold of 1000ms and a recovery threshold of 900ms. This way, an alert rule will only stop firing when it goes under 900ms and flapping is reduced.

Alert instances

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana managed alerts support multi-dimensional alerting. Each alert rule can create multiple alert instances. This is powerful if you are observing multiple series in a single expression.

Consider the following PromQL expression:

```
sum by(cpu) (
  rate(node_cpu_seconds_total{mode!="idle"}[1m])
)
```

A rule using this expression will create as many alert instances as the amount of CPUs we are observing after the first evaluation, allowing a single rule to report the status of each CPU.

Namespaces, folders and groups

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Alerts can be organized using folders for Grafana managed rules and namespaces for Mimir, Loki, or Prometheus rules and group names.

Namespaces and folders

When creating Grafana-managed rules, the folder can be used to perform access control and grant or deny access to all rules within a specific folder.

A namespace contains one or more groups. The rules within a group are run sequentially at a regular interval. The default interval is one minute. You can rename Grafana Mimi or Loki rule namespaces and groups, and edit group evaluation intervals.

Groups

The rules within a group are run sequentially at a regular interval, meaning no rules will be evaluated at the same time, and in order of appearance. The default interval is one minute. You can rename Grafana Mimir or Loki rule namespaces or Loki rule namespaces and groups, and edit group evaluation intervals.

> **Tip**
>
> If you want rules to be evaluated concurrently and with different intervals, consider storing them in different groups.

> **Note**
>
> Grafana managed alert rules are evaluated concurrently instead of sequentially.

Alert rule evaluation

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use alert rule evaluation to determine how frequently an alert rule should be evaluated and how quickly it should change its state.

To do this, you need to make sure that your alert rule is in the right evaluation group and set a pending period time that works best for your use case.

Evaluation group

Every alert rule is part of an evaluation group. Each evaluation group contains an evaluation interval that determines how frequently the alert rule is checked.

Data-source managed alert rules within the same group are evaluated one after the other, while alert rules in different groups can be evaluated simultaneously. This feature is especially useful when you want to ensure that recording rules are evaluated before any alert rules.

Grafana-managed alert rules are evaluated at the same time, regardless of alert rule group. The default evaluation interval is set at 10 seconds, which means that Grafana-managed alert rules are evaluated every 10 seconds to the closest 10-second window on the clock, for example, 10:00:00, 10:00:10, 10:00:20, and so on. You can also configure your own evaluation interval, if required.

> **Note**
>
> Evaluation groups and alerts grouping in notification policies are two separate things. Grouping in notification policies allows multiple alerts sharing the same labels to be sent in the same time message.

Pending period

By setting a pending period, you can avoid unnecessary alerts for temporary problems.

In the pending period, you select the period in which an alert rule can be in breach of the condition until it fires.

Example

Imagine you have an alert rule evaluation interval set at every 30 seconds and the pending period to 90 seconds.

Evaluation will occur as follows:

[00:30] First evaluation - condition not met.

[01:00] Second evaluation - condition breached. Pending counter starts.**Alert starts pending.**

[01:30] Third evaluation - condition breached. Pending counter = 30s. **Pending state.**

[02:00] Fourth evaluation - condition breached. Pending counter = 60s **Pending state.**

[02:30] Fifth evaluation - condition breached. Pending counter = 90s. **Alert starts firing**

If the alert rule has a condition that needs to be in breach for a certain amount of time before it takes action, then its state changes as follows:

- When the condition is first breached, the rule goes into a "pending" state.
- The rule stays in the "pending" state until the condition has been broken for the required amount of time - pending period.
- Once the required time has passed, the rule goes into a "firing" state.
- If the condition is no longer broken during the pending period, the rule goes back to its normal state.

> **Note**
>
> If you want to skip the pending state, you can simply set the pending period to 0. This effectively skips the pending period and your alert rule will start firing as soon as the condition is breached.

When an alert rule fires, alert instances are produced, which are then sent to the Alertmanager.

State and health of alerting rules

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The state and health of alerting rules help you understand several key status indicators about your alerts.

There are three key components: *alert rule state*, *alert instance state*, and *alert rule health*. Although related, each component conveys subtly different information.

Alert rule state

An alert rule can be in one of the following states:

State	Description
Normal	None of the time series returned by the evaluation engine is in a `Pending` or `Firing` state.
Pending	At least one time series returned by the evaluation engine is `Pending`.
Firing	At least one time series returned by the evaluation engine is `Firing`.

Note

Alerts will transition first to `pending` and then `firing`, thus it will take at least two evaluation cycles before an alert is fired.

Alert instance state

An alert instance can be in one of the following states:

State	Description
Normal	The state of an alert that is neither firing nor pending, everything is working correctly.
Pending	The state of an alert that has been active for less than the configured threshold duration.
Alerting	The state of an alert that has been active for longer than the configured threshold duration.
NoData	No data has been received for the configured time window.
Error	The error that occurred when attempting to evaluate an alerting rule.

Keep last state

An alert rule can be configured to keep the last state when `NoData` or `Error` state is encountered. This will both prevent alerts from firing, and from resolving and re-firing. Just like normal evaluation, the alert rule will transition from `Pending` to `Firing` after the pending period has elapsed.

Alert rule health

An alert rule can have one the following health statuses:

State	Description
Ok	No error when evaluating an alerting rule.
Error	An error occurred when evaluating an alerting rule.
NoData	The absence of data in at least one time series returned during a rule evaluation.

Special alerts for `NoData` and `Error`

When evaluation of an alerting rule produces state `NoData` or `Error`, Grafana Alerting will generate alert instances that have the following additional labels:

Label	Description
alertname	Either `DatasourceNoData` or `DatasourceError` depending on the state.
datasource_uid	The UID of the data source that caused the state.

You can handle these alerts the same way as regular alerts by adding a silence, route to a contact point, and so on.

Notification templating

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Notifications sent via contact points are built using notification templates. Grafana's default templates are based on the Go templating system where some fields are evaluated as text, while others are evaluated as HTML (which can affect escaping).

The default template default_template.go is a useful reference for custom templates.

Since most of the contact point fields can be templated, you can create reusable custom templates and use them in multiple contact points. To learn about custom notifications using templates, see Customize notifications.

Nested templates

You can embed templates within other templates.

For example, you can define a template fragment using the `define` keyword.

```
{{ define "mytemplate" }}
  {{ len .Alerts.Firing }} firing. {{ len .Alerts.Resolved }} resolved.
{{ end }}
```

You can then embed custom templates within this fragment using the `template` keyword. For example:

```
Alert summary:
{{ template "mytemplate" . }}
```

You can use any of the following built-in template options to embed custom templates.

Name	Notes
`default.title`	Displays high-level status information.
`default.message`	Provides a formatted summary of firing and resolved alerts.
`teams.default.message`	Similar to `default.messsage` , formatted for Microsoft Teams.

HTML in notification templates

HTML in alerting notification templates is escaped. We do not support rendering of HTML in the resulting notification.

Some notifiers support alternative methods of changing the look and feel of the resulting notification. For example, Grafana installs the base template for alerting emails to `<grafana-install-dir>/public/emails/ng_alert_notification.html`. You can edit this file to change the appearance of all alerting emails.

Alertmanager

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Alertmanager enables you to quickly and efficiently manage and respond to alerts. It receives alerts, handles mutings, inhibition, grouping, and routing by sending notifications out via your channel of choice, for example, email or Slack.

In Grafana, you can use the Grafana Alertmanager or an external Alertmanager. You can also run multiple alertmanagers; your decision depends on your set up and where your alerts are being generated.

Grafana Alertmanager

Grafana Alertmanager is an internal Alertmanager that is pre-configured and available for selection by default.

The Grafana Alertmanager can receive alerts from Grafana, but it cannot receive alerts from outside Grafana, for example, from Mimir or Loki.

> **Note**
>
> Inhibition rules are not supported in the Grafana Alertmanager.

External Alertmanager

If you want to use a single alertmanager to receive all your Grafana, Loki, Mimir, and Prometheus alerts, you can set up Grafana to use an external Alertmanager. This external Alertmanager can be configured and administered from within Grafana itself.

Here are two examples of when you might want to configure your own external alertmanager and send your alerts there instead of the Grafana Alertmanager:

1. You already have alertmanagers on-premise in your own Cloud infrastructure that you have set up and still want to use, because you have other alert generators, such as Prometheus.
2. You want to use both Prometheus on-premise and hosted Grafana to send alerts to the same alertmanager that runs in your Cloud infrastructure.

Alertmanagers are visible from the dropdown menu on the Alerting Contact Points, and Notification Policies pages.

If you are provisioning your data source, set the flag `handleGrafanaManagedAlerts` in the `jsonData` field to `true` to send Grafana-managed alerts to this Alertmanager.

Contact points

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Contact points contain the configuration for sending notifications. A contact point is a list of integrations, each of which sends a notification to a particular email address, service or URL. Contact points can have multiple integrations of the same kind, or a combination of integrations of different kinds. For example, a contact point could contain a Pagerduty integration; an Amazon SNS and Slack integration; or a Pagerduty integration, a Slack integration, and two Amazon SNS integrations. You can also configure a contact point with no integrations; in which case no notifications are sent.

A contact point cannot send notifications until it has been added to a notification policy. A notification policy can only send alerts to one contact point, but a contact point can be added to a number of notification policies at the same time. When an alert matches a notification policy, the alert is sent to the contact point in that notification policy, which then sends a notification to each integration in its configuration.

Contact points can be configured for the Grafana Alertmanager as well as external alertmanagers.

You can also use notification templating to customize notification messages for contact point types.

Supported contact point types

The following table lists the contact point types supported by Grafana.

Name	Type
Amazon SNS	`sns`
OpsGenie	`opsgenie`
Pager Duty	`pagerduty`
Slack	`slack`
VictorOps	`victorops`

For more information about contact points, see Configure contact points and Customize notifications.

Notifications

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Choosing how, when, and where to send your alert notifications is an important part of setting up your alerting system. These decisions will have a direct impact on your ability to resolve issues quickly and not miss anything important.

As a first step, define your contact points, which define where to send your alert notifications. A contact point is a set of one or more integrations that are used to deliver notifications. Add notification templates to contact points for reuse and consistent messaging in your notifications.

Next, create a notification policy which is a set of rules for where, when and how your alerts are routed to contact points. In a notification policy, you define where to send your alert notifications by choosing one of the contact points you created.

Alertmanagers

Grafana uses Alertmanagers to send notifications for firing and resolved alerts. Grafana has its own Alertmanager, referred to as **Grafana** in the user interface, but also supports sending notifications from other Alertmanagers too, such as the Prometheus Alertmanager. The Grafana Alertmanager uses notification policies and contact points to configure how and where a notification is sent; how often a notification should be sent; and whether alerts should all be sent in the same notification, sent in grouped notifications based on a set of labels, or as separate notifications.

Notification policies

Notification policies control when and where notifications are sent. A notification policy can choose to send all alerts together in the same notification, send alerts in grouped notifications based on

a set of labels, or send alerts as separate notifications. You can configure each notification policy to control how often notifications should be sent as well as having one or more mute timings to inhibit notifications at certain times of the day and on certain days of the week.

Notification policies are organized in a tree structure where at the root of the tree there is a notification policy called the default policy. There can be only one default policy and the default policy cannot be deleted.

Specific routing policies are children of the root policy and can be used to match either all alerts or a subset of alerts based on a set of matching labels. A notification policy matches an alert when its matching labels match the labels in the alert.

A nested policy can have its own nested policies, which allow for additional matching of alerts. An example of a nested policy could be sending infrastructure alerts to the Ops team; while a child policy might send high priority alerts to Pagerduty and low priority alerts to Slack.

All alerts, irrespective of their labels, match the default policy. However, when the default policy receives an alert it looks at each nested policy and sends the alert to the first nested policy that matches the alert. If the nested policy has further nested policies, then it can attempt to match the alert against one of its nested policies. If no nested policies match the alert then the policy itself is the matching policy. If there are no nested policies, or no nested policies match the alert, then the default policy is the matching policy.

For more detailed information about notification policies, see Notification policies.

Notification templates

You can customize notifications with templates. For example, templates can be used to change the title and message of notifications sent to Slack.

Templates are not limited to an individual integration or contact point, but instead can be used in a number of integrations in the same contact point and even integrations across different contact points. For example, a Grafana user can create a template called `custom_subject_or_title` and use it for both templating subjects in Pager Duty and titles of Slack messages without having to create two separate templates.

All notifications templates are written in Go's templating language, and are in the Contact points tab on the Alerting page.

For more detailed information about customizing notifications, see Customize notifications.

Silences

You can use silences to mute notifications from one or more firing rules. Silences do not stop alerts from firing or being resolved, or hide firing alerts in the user interface. A silence lasts as long as its duration which can be configured in minutes, hours, days, months or years.

For more detailed information about using silences, see Silencing alert notifications.

Notification policies

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Notification policies provide you with a flexible way of routing alerts to various different receivers. Using label matchers, you can modify alert notification delivery without having to update every individual alert rule.

In this section, you will learn more about how notification policies work and are structured, so that you can make the most out of setting up your notification policies.

Policy tree

Notification policies are *not* a list, but rather are structured according to a tree structure. This means that each policy can have child policies, and so on. The root of the notification policy tree is called the **Default notification policy**.

Each policy consists of a set of label matchers (0 or more) that specify which labels they are or aren't interested in handling.

For more information about label matching, see How label matching works.

> **Note**
>
> If you haven't configured any label matchers for your notification policy, your notification policy will match *all* alert instances. This may prevent child policies from being evaluated unless you have enabled **Continue matching siblings** on the notification policy.

Routing

To determine which notification policy will handle which alert instances, you have to start by looking at the existing set of notification policies, starting with the default notification policy.

If no policies other than the default policy are configured, the default policy will handle the alert instance.

If policies other than the default policy are defined, it will evaluate those notification policies in the order they are displayed.

If a notification policy has label matchers that match the labels of the alert instance, it will descend in to its child policies and, if there are any, will continue to look for any child policies that might have label matchers that further narrow down the set of labels, and so forth until no more child policies have been found.

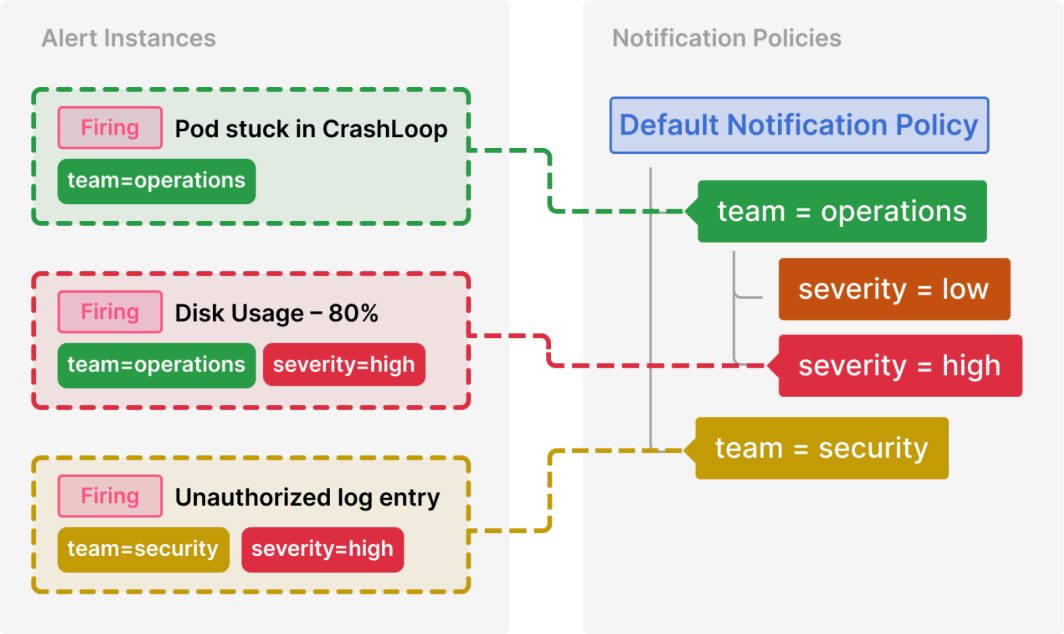
If no child policies are defined in a notification policy or if none of the child policies have any label matchers that match the alert instance's labels, the parent notification policy is used.

As soon as a matching policy is found, the system does not continue to look for other matching policies. If you want to continue to look for other policies that may match, enable **Continue matching siblings** on that particular policy.

Lastly, if none of the notification policies are selected the default notification policy is used.

Routing example

Here is an example of a relatively simple notification policy tree and some alert instances.



Here's a breakdown of how these policies are selected:

Pod stuck in CrashLoop does not have a `severity` label, so none of its child policies are matched. It does have a `team=operations` label, so the first policy is matched.

The `team=security` policy is not evaluated since we already found a match and **Continue matching siblings** was not configured for that policy.

Disk Usage – 80% has both a `team` and `severity` label, and matches a child policy of the operations team.

Unauthorized log entry has a `team` label but does not match the first policy (`team=operations`) since the values are not the same, so it will continue searching and match the `team=security` policy. It does not have any child policies, so the additional `severity=high` label is ignored.

Inheritance

In addition to child policies being a useful concept for routing alert instances, they also inherit properties from their parent policy. This also applies to any policies that are child policies of the default notification policy.

The following properties are inherited by child policies:

- Contact point
- Grouping options

- Timing options
- Mute timings

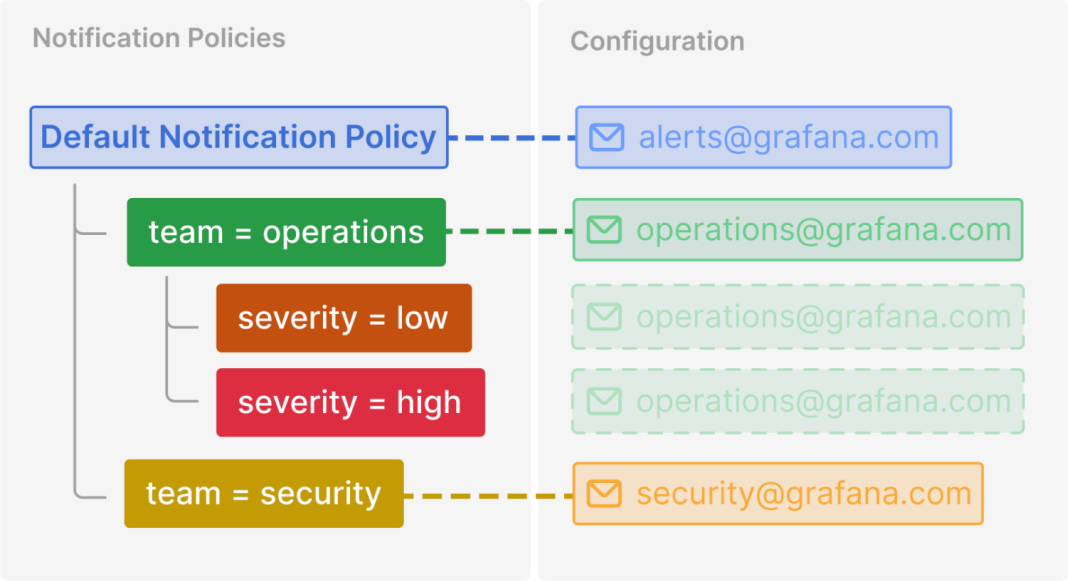
Each of these properties can be overwritten by an individual policy should you wish to override the inherited properties.

To inherit a contact point from the parent policy, leave it blank. To override the inherited grouping options, enable **Override grouping**. To override the inherited timing options, enable **Override general timings**.

Inheritance example

The example below shows how the notification policy tree from our previous example allows the child policies of the `team=operations` to inherit its contact point.

In this way, we can avoid having to specify the same contact point multiple times for each child policy.



Additional configuration options

Grouping

Grouping is an important feature of Grafana Alerting as it allows you to batch relevant alerts together into a smaller number of notifications. This is particularly important if notifications are delivered to first-responders, such as engineers on-call, where receiving lots of notifications in a short period of time can be overwhelming and in some cases can negatively impact a first-

responders ability to respond to an incident. For example, consider a large outage where many of your systems are down. In this case, grouping can be the difference between receiving 1 phone call and 100 phone calls.

You choose how alerts are grouped together using the Group by option in a notification policy. By default, notification policies in Grafana group alerts together by alert rule using the `alertname` and `grafana_folder` labels (since alert names are not unique across multiple folders). Should you wish to group alerts by something other than the alert rule, change the grouping to any other combination of labels.

Disable grouping

Should you wish to receive every alert as a separate notification, you can do so by grouping by a special label called `...`. This is useful when your alerts are being delivered to an automated system instead of a first-responder.

A single group for all alerts

Should you wish to receive all alerts together in a single notification, you can do so by leaving Group by empty.

Timing options

The timing options decide how often notifications are sent for each group of alerts. There are three timers that you need to know about: Group wait, Group interval, and Repeat interval.

Group wait

Group wait is the amount of time Grafana waits before sending the first notification for a new group of alerts. The longer Group wait is the more time you have for other alerts to arrive. The shorter Group wait is the earlier the first notification will be sent, but at the risk of sending incomplete notifications. You should always choose a Group wait that makes the most sense for your use case.

Default 30 seconds

Group interval

Once the first notification has been sent for a new group of alerts, Grafana starts the Group interval timer. This is the amount of time Grafana waits before sending notifications about changes to the group. For example, another firing alert might have just been added to the group while an

existing alert might have resolved. If an alert was too late to be included in the first notification due to Group wait, it will be included in subsequent notifications after Group interval. Once Group interval has elapsed, Grafana resets the Group interval timer. This repeats until there are no more alerts in the group after which the group is deleted.

Default 5 minutes

Repeat interval

Repeat interval decides how often notifications are repeated if the group has not changed since the last notification. You can think of these as reminders that some alerts are still firing. Repeat interval is closely related to Group interval, which means your Repeat interval must not only be greater than or equal to Group interval, but also must be a multiple of Group interval. If Repeat interval is not a multiple of Group interval it will be coerced into one. For example, if your Group interval is 5 minutes, and your Repeat interval is 9 minutes, the Repeat interval will be rounded up to the nearest multiple of 5 which is 10 minutes.

Default 4 hours

Alerting high availability



This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Amazon Managed Grafana is configured for high availability, including running multiple instances across multiple availability zones for each workspace that you create.

Grafana Alerting uses the Prometheus model of separating the evaluation of alert rules from the delivering of notifications. In this model the evaluation of alert rules is done in the alert generator and the delivering of notifications is done in the alert receiver. In Grafana Alerting, the alert generator is the Scheduler and the receiver is the Alertmanager.

With high availability configurations, all alert rules are evaluated on all instances. You can think of the evaluation of alert rules as being duplicated. This is how Grafana Alerting makes sure that as

long as at least one Grafana instance is working, alert rules will still be evaluated and notifications for alerts will still be sent. You will see this duplication in state history, and is a good way to tell if you are using high availability.

Set up Alerting

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Configure the features and integrations that you need to create and manage your alerts.

Prerequisites

Before you set up alerting, you must do the following.

- Configure your data sources.
- Ensure that the data source you choose are compatible with and supported by Grafana alerting.

To set up alerting

1. Configure alert rules.
   - Create Grafana-managed or data-source managed alert rules and recording rules.
2. Configure contact points.
   - Check the default contact point, and update the contact for your system.
   - Optionally, add new contact points and integrations.
3. Configure notification policies
   - Check the default notification policy, and update for your system.
   - Optionally, add additional nested policies.
   - Optionally, add labels and label matchers to control alert routing.

The following topics give you more information about additional configuration options, including configuring external alert managers and routing Grafana-managed alerts outside of Grafana.

Topics

- Migrating classic dashboard alerts to Grafana alerting
- Adding an external Alertmanager
- Provisioning Grafana Alerting resources

Migrating classic dashboard alerts to Grafana alerting

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Workspaces that choose not to use Grafana alerting use the Classic dashboard alerts. To switch to the new Grafana alerting, you must opt in to the feature. To see details about the differences between classic dashboard alerting and Grafana alerting, see Grafana alerting vs legacy dashboard alerting. GrafanaLabs has announced that classic dashboard alerts will be removed in version 11.

When you are using classic dashboard alerting, Amazon Managed Grafana shows you a preview of Grafana alerting where you can review and modify your upgraded alerts before finalizing the upgrade.

Previewing Grafana alerts

You can preview your alerts in Grafana alerts before migrating. In the preview, you can make changes to the alerts that will change the migration.

To preview your Grafana alerting migration

1. Sign into your Grafana workspace.
2. From the left menu, choose **Alerting (legacy)** to view your current alerts.

3. From the left menu, choose **Alerting upgrade** to view your alerts in Grafana alerting.

   From this view, you can see what your alerts will look like after migration.

> **Note**
>
> From this view, you can also make changes that will affect your migration. To undo any changes you make, choose **Reset upgrade** at the top right of the upgrade page.

When you are ready to upgrade your alerts, see the next section.

Migrating to Grafana alerting system

You can configure your Amazon Managed Grafana instance to use Grafana alerting using the AWS Management Console, the AWS CLI, or the Amazon Managed Grafana API. For details about how to configure Amazon Managed Grafana, including turning Grafana alerting on or off, see Configure a Amazon Managed Grafana workspace.

When Grafana alerting is turned on, existing classic dashboard alerts migrate in a format compatible with the Grafana alerting. In the Alerting page of your Grafana instance, you can view the migrated alerts alongside new alerts. With Grafana alerting, your Grafana-managed alert rules send multiple notifications rather than a single alert when they are matched.

Read and write access to classic dashboard alerts and Grafana alerts are governed by the permissions of the folders storing them. During migration, classic dashboard alert permissions are matched to the new rules permissions as follows:

- If the original alert's dashboard has permissions, migration creates a folder named with this format `Migrated {"dashboardUid": "UID", "panelId": 1, "alertId": 1}` to match permissions of the original dashboard (including the inherited permissions from the folder).
- If there are no dashboard permissions and the dashboard is under a folder, then the rule is linked to this folder and inherits its permissions.
- If there are no dashboard permissions and the dashboard is under the General folder, then the rule is linked to the General Alerting folder, and the rule inherits the default permissions.

> **Note**
>
> Since there is no `Keep Last State` option for `NoData` in Grafana alerting, this option becomes `NoData` during the classic rules migration. Option `Keep Last State` for `Error` handling is migrated to a new option `Error`. To match the behavior of the `Keep Last State`, in both cases, during the migration Amazon Managed Grafana automatically creates a silence for each alert rule with a duration of one year.

Notification channels are migrated to an Alertmanager configuration with the appropriate routes and receivers. Default notification channels are added as contact points to the default route. Notification channels not associated with any Dashboard alert go to the `autogen-unlinked-channel-recv` route.

Limitations

- Grafana alerting system can retrieve rules from all available Prometheus, Loki, and Alertmanager data sources. It might not be able to fetch alerting rules from other supported data sources.
- Migrating back and forth between Grafana alerts and the classic dashboard alerting can result in data loss for features supported in one system, but not the other.

> **Note**
>
> If you migrate back to the classic dashboard alerting, you lose all changes made to alerting configuration made while you had Grafana alerting enabled, including any new alert rules that were created.

Grafana alerting vs legacy dashboard alerting

Introduced in Grafana 8, Grafana alerting has several enhancements over classic dashboard alerting.

Multi-dimensional alerting

You can now create alerts that give you system-wide visibility with a single alerting rule. Generate multiple alert instances from a single alert rule. For example, you can create a rule to monitor the disk usage of multiple mount points on a single host. The evaluation engine returns multiple time series from a single query, with each time series identified by its label set.

Create alerts outside of Dashboards

Unlike legacy dashboard alerts, Grafana alerts allow you to create queries and expressions that combine data from multiple sources in unique ways. You can still link dashboards and panels to alerting rules using their ID and quickly troubleshoot the system under observation.

Since unified alerts are no longer directly tied to panel queries, they do not include images or query values in the notification email. You can use customized notification templates to view query values.

Create Loki and Grafana Mimir alerting rules

In Grafana Alerting, you can manage Loki and Grafana Mimir alerting rules using the same UI and API as your Grafana managed alerts.

View and search for alerts from Prometheus compatible data sources

Alerts for Prometheus compatible data sources are now listed under the Grafana alerts section. You can search for labels across multiple data sources to quickly find relevant alerts.

Special alerts for alert state NoData and Error

Grafana Alerting introduced a new concept of the alert states. When evaluation of an alerting rule produces state NoData or Error, Grafana Alerting will generate special alerts that will have the following labels:

- `alertname` with value DatasourceNoData or DatasourceError depending on the state.
- `rulename` name of the alert rule the special alert belongs to.
- `datasource_uid` will have the UID of the data source that caused the state.
- All labels and annotations of the original alert rule

You can handle these alerts the same way as regular alerts by adding a silence, route to a contact point, and so on.

> **Note**
>
> If the rule uses many data sources and one or many returns no data, the special alert will be created for each data source that caused the alert state.

Adding an external Alertmanager

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Set up Grafana to use an external Alertmanager as a single Alertmanager to receive all of your alerts. This external Alertmanager can then be configured and administered from within Grafana itself.

Note

You can't use Amazon Managed Service for Prometheus as an external Alertmanager.

Once you have added the alertmanager, you can use the Grafana Alerting UI to manage silences, contact points, and notification policies. A dropdown option in these pages allows you to switch between alertmanagers.

External alertmanagers are configured as data sources using Grafana Configuration from the main Grafana navigation menu. This enables you to manage the contact points and notification policies of external alertmanagers from within Grafana and also encrypts HTTP basic authentication credentials that were previously visible when configuring external alertmanagers by URL.

Note

Starting with Grafana 9.2, the URL configuration of external alertmanagers from the Admin tab on the Alerting page is deprecated. It will be removed in a future release.

To add an external Alertmanager

1. Choose **Connections** from the main left menu.

2. Search for `Alertmanager`.
3. Choose the **Create a new data source** button.
4. Fill out the fields on the page, as required.

   If you are provisioning your data source, set the flag `handleGrafanaManagedAlerts` in the `jsonData` field to `true` to send Grafana-managed alerts to this Alertmanager.

   > **Note**
   >
   > Prometheus, Grafana Mimir, and Cortex implementations of Alertmanager are supported. For Prometheus, contact points and notification policies are read-only in the Grafana Alerting UI.

5. Choose **Save & test**.

Provisioning Grafana Alerting resources

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Alerting infrastructure is often complex, with many pieces of the pipeline that often live in different places. Scaling this across multiple teams and organizations is an especially challenging task. Grafana Alerting provisioning makes this process easier by enabling you to create, manage, and maintain your alerting data in a way that best suits your organization.

There are two options to choose from:

1. Provision your alerting resources using the Alerting Provisioning HTTP API.

   > **Note**
   >
   > Typically, you cannot edit API-provisioned alert rules from the Grafana UI.

In order to enable editing, add the x-disable-provenance header to the following requests when creating or editing your alert rules in the API:

```
POST /api/v1/provisioning/alert-rules
PUT /api/v1/provisioning/alert-rules/{UID}
```

2. Provision your alerting resources using Terraform.

Note

Currently, provisioning for Grafana Alerting supports alert rules, contact points, mute timings, and templates. Provisioned alerting resources using file provisioning or Terraform can only be edited in the source that created them and not from within Grafana or any other source. For example, if you provision your alerting resources using files from disk, you cannot edit the data in Terraform or from within Grafana.

Topics

- Create and manage alerting resources using Terraform
- Viewing provisioned alerting resources in Grafana

Create and manage alerting resources using Terraform

This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use Terraform's Grafana Provider to manage your alerting resources and provision them into your Grafana system. Terraform provider support for Grafana Alerting makes it easy to create, manage, and maintain your entire Grafana Alerting stack as code.

For more information on managing your alerting resources using Terraform, refer to the Grafana Provider documentation in the Terraform documentation.

Complete the following tasks to create and manage your alerting resources using Terraform.

1. Create an API key for provisioning.
2. Configure the Terraform provider.
3. Define your alerting resources in Terraform.
4. Run `terraform apply` to provision your alerting resources.

Prerequisites

- Ensure you have the grafana/grafana Terraform provider 1.27.0 or higher.
- Ensure you are using Grafana 9.1 or higher. If you created your Amazon Managed Grafana instance with Grafana version 9, this will be true.

Create an API key for provisioning

You can create a normal Grafana API key to authenticate Terraform with Grafana. Most existing tooling using API keys should automatically work with the new Grafana Alerting support. For information specifically about creating keys for use with Terraform, see Using Terraform for Amazon Managed Grafana automation.

To create an API key for provisioning

1. Create a new service account for your CI pipeline.
2. Assign the role “Access the alert rules Provisioning API.”
3. Create a new service account token.
4. Name and save the token for use in Terraform.

Alternatively, you can use basic authentication. To view all the supported authentication formats, see Grafana authentication in the Terraform documentation.

Configure the Terraform provider

Grafana Alerting support is included as part of the Grafana Terraform provider.

The following is an example you can use to configure the Terraform provider.

```
terraform {
    required_providers {
        grafana = {
            source = "grafana/grafana"
            version = ">= 1.28.2"
        }
    }
}

provider "grafana" {
    url = <YOUR_GRAFANA_URL>
    auth = <YOUR_GRAFANA_API_KEY>
}
```

Provision contact points and templates

Contact points connect an alerting stack to the outside world. They tell Grafana how to connect to your external systems and where to deliver notifications. There are over fifteen different integrations to choose from. This example uses a Slack contact point.

To provision contact points and templates

1. Copy this code block into a .tf file on your local machine. Replace *<slack-webhook-url>* with your Slack webhook URL (or other contact point details).

   This example creates a contact point that sends alert notifications to Slack.

   ```
   resource "grafana_contact_point" "my_slack_contact_point" {
       name = "Send to My Slack Channel"

       slack {
           url = <slack-webhook-url>
           text = <<EOT
   {{ len .Alerts.Firing }} alerts are firing!

   Alert summaries:
   {{ range .Alerts.Firing }}
   {{ template "Alert Instance Template" . }}
   {{ end }}
   EOT
       }
   }
   ```

2. Enter text for your notification in the text field.

   The `text` field supports Go-style templating. This enables you to manage your Grafana Alerting notification templates directly in Terraform.

3. Run the command `terraform apply`.
4. Go to the Grafana UI and check the details of your contact point.

   You cannot edit resources provisioned via Terraform from the UI. This ensures that your alerting stack always stays in sync with your code.

5. Click **Test** to verify that the contact point works correctly.

> **Note**
>
> You can re-use the same templates across many contact points. In the example above, a shared template is embedded using the statement `{{ template "Alert Instance Template" . }}`
> This fragment can then be managed separately in Terraform:

```
resource "grafana_message_template" "my_alert_template" {
    name = "Alert Instance Template"

    template = <<EOT
{{ define "Alert Instance Template" }}
Firing: {{ .Labels.alertname }}
Silence: {{ .SilenceURL }}
{{ end }}
EOT
}
```

Provision notification policies and routing

Notification policies tell Grafana how to route alert instances, as opposed to where. They connect firing alerts to your previously defined contact points using a system of labels and matchers.

To provision notification policies and routing

1. Copy this code block into a .tf file on your local machine.

In this example, the alerts are grouped by `alertname`, which means that any notifications coming from alerts which share the same name, are grouped into the same Slack message.

If you want to route specific notifications differently, you can add sub-policies. Sub-policies allow you to apply routing to different alerts based on label matching. In this example, we apply a mute timing to all alerts with the label a=b.

```
resource "grafana_notification_policy" "my_policy" {
    group_by = ["alertname"]
    contact_point = grafana_contact_point.my_slack_contact_point.name

    group_wait = "45s"
    group_interval = "6m"
    repeat_interval = "3h"

    policy {
        matcher {
            label = "a"
            match = "="
            value = "b"
        }
        group_by = ["..."]
        contact_point = grafana_contact_point.a_different_contact_point.name
        mute_timings = [grafana_mute_timing.my_mute_timing.name]

        policy {
            matcher {
                label = "sublabel"
                match = "="
                value = "subvalue"
            }
            contact_point = grafana_contact_point.a_third_contact_point.name
            group_by = ["..."]
        }
    }
}
```

2. In the mute_timings field, link a mute timing to your notification policy.
3. Run the command `terraform apply`.
4. Go to the Grafana UI and check the details of your notification policy.

> **Note**
>
> You cannot edit resources provisioned from Terraform from the UI. This ensures that your alerting stack always stays in sync with your code.

5. Click **Test** to verify that the notification point is working correctly.

Provision mute timings

Mute timings provide the ability to mute alert notifications for defined time periods.

To provision mute timings

1. Copy this code block into a .tf file on your local machine.

   In this example, alert notifications are muted on weekends.

```
resource "grafana_mute_timing" "my_mute_timing" {
    name = "My Mute Timing"

    intervals {
        times {
          start = "04:56"
          end = "14:17"
        }
        weekdays = ["saturday", "sunday", "tuesday:thursday"]
        months = ["january:march", "12"]
        years = ["2025:2027"]
    }
}
```

2. Run the command `terraform apply`.
3. Go to the Grafana UI and check the details of your mute timing.
4. Reference your newly created mute timing in a notification policy using the `mute_timings` field. This will apply your mute timing to some or all of your notifications.

> **Note**
>
> You cannot edit resources provisioned from Terraform from the UI. This ensures that your alerting stack always stays in sync with your code.

5. Click **Test** to verify that the mute timing is working correctly.

Provision alert rules

Alert rules enable you to alert against any Grafana data source. This can be a data source that you already have configured, or you can define your data sources in Terraform alongside your alert rules.

To provision alert rules

1. Create a data source to query and a folder to store your rules in.

   In this example, the Configure a TestData data source for testing data source is used.

   Alerts can be defined against any backend datasource in Grafana.

   ```
   resource "grafana_data_source" "testdata_datasource" {
       name = "TestData"
       type = "testdata"
   }

   resource "grafana_folder" "rule_folder" {
       title = "My Rule Folder"
   }
   ```

2. Define an alert rule.

   For more information on alert rules, refer to how to create Grafana-managed alerts.

3. Create a rule group containing one or more rules.

   In this example, the `grafana_rule_group` resource group is used.

   ```
   resource "grafana_rule_group" "my_rule_group" {
       name = "My Alert Rules"
       folder_uid = grafana_folder.rule_folder.uid
   ```

```
    interval_seconds = 60
    org_id = 1

    rule {
        name = "My Random Walk Alert"
        condition = "C"
        for = "0s"

        // Query the datasource.
        data {
            ref_id = "A"
            relative_time_range {
                from = 600
                to = 0
            }
            datasource_uid = grafana_data_source.testdata_datasource.uid
            // `model` is a JSON blob that sends datasource-specific data.
            // It's different for every datasource. The alert's query is defined
 here.
            model = jsonencode({
                intervalMs = 1000
                maxDataPoints = 43200
                refId = "A"
            })
        }

        // The query was configured to obtain data from the last 60 seconds. Let's
 alert on the average value of that series using a Reduce stage.
        data {
            datasource_uid = "__expr__"
            // You can also create a rule in the UI, then GET that rule to obtain
 the JSON.
            // This can be helpful when using more complex reduce expressions.
            model = <<EOT
{"conditions":[{"evaluator":{"params":[0,0],"type":"gt"},"operator":
{"type":"and"},"query":{"params":["A"]},"reducer":{"params":
[],"type":"last"},"type":"avg"}],"datasource":
{"name":"Expression","type":"__expr__","uid":"__expr__"},"expression":"A","hide":false,"int
EOT
            ref_id = "B"
            relative_time_range {
                from = 0
                to = 0
            }
```

```
        }

        // Now, let's use a math expression as our threshold.
        // We want to alert when the value of stage "B" above exceeds 70.
        data {
            datasource_uid = "__expr__"
            ref_id = "C"
            relative_time_range {
                from = 0
                to = 0
            }
            model = jsonencode({
                expression = "$B > 70"
                type = "math"
                refId = "C"
            })
        }
    }
}
```

4. Go to the Grafana UI and check your alert rule.

   You can see whether the alert rule is firing. You can also see a visualization of each of the alert rule's query stages.

   When the alert fires, Grafana routes a notification through the policy you defined.

   For example, if you chose Slack as a contact point, Grafana's embedded Alertmanager automatically posts a message to Slack.

Viewing provisioned alerting resources in Grafana

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can verify that your alerting resources were created in Grafana.

To view your provisioned resources in Grafana

1. Open your Grafana instance.
2. Navigate to Alerting.
3. Click an alerting resource folder, for example, Alert rules.

   Provisioned resources are labeled **Provisioned**, so that it is clear that they were not created manually.

> **Note**
>
> You cannot edit provisioned resources from Grafana. You can only change the resource properties by changing the provisioning file and restarting Grafana or carrying out a hot reload. This prevents changes being made to the resource that would be overwritten if a file is provisioned again or a hot reload is carried out.

Configure alerting

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Configure the features and integrations that you need to create and manage your alerts.

Topics

- Configure Grafana managed alert rules
- Configure data source managed alert rules
- Configure recording rules
- Configure contact points

- Configure notification policies

Configure Grafana managed alert rules

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana-managed rules are the most flexible alert rule type. They allow you to create alerts that can act on data from any of our supported data sources. In addition to supporting multiple data sources, you can also add expressions to transform your data and set alert conditions. Using images in alert notifications is also supported. This is the only type of rule that allows alerting from multiple data sources in a single rule definition.

Multiple alert instances can be created as a result of one alert rule (also known as multi-dimensional alerting).

Grafana managed alert rules can only be edited or deleted by users with Edit permissions for the folder storing the rules.

If you delete an alerting resource created in the UI, you can no longer retrieve it. To make a backup of your configuration and to be able to restore deleted alerting resources, create your alerting resources using Terraform, or the Alerting API.

In the following procedures, we'll go through the process of creating your Grafana-managed alert rules.

To create a Grafana-managed alert rule, use the in-workspace alert creation flow and follow these steps to help you.

Set alert rule name

1. Choose **Alerting** -> **Alert rules** -> **+ New alert rule**.
2. Enter a name to identify your alert rule.

This name is displayed in the alert rule list. It is also the `alertname` label for every alert instance that is created from this rule.

Next, define a query to get the data you want to measure and a condition that needs to be met before an alert rule fires.

To define the query and condition

1. Select a data source.
2. From the **Options** dropdown, specify a time range.

   > **Note**
   >
   > Grafana Alerting only supports fixed relative time ranges, for example, `now-24hr: now`.
   > It does not support absolute time ranges: `2021-12-02 00:00:00 to 2021-12-05 23:59:592` or semi-relative time ranges: `now/d to: now`.

3. Add a query.

   To add multiple queries, choose **Add query**.

   All alert rules are managed by Grafana by default. If you want to switch to a data source-managed alert rule, click **Switch to data source-managed alert rule**.

4. Add one or more expressions.

   1. For each expression, select either **Classic condition** to create a single alert rule, or choose from the **Math**, **Reduce**, and **Resample** options to generate a separate alert for each series.

      > **Note**
      >
      > When using Prometheus, you can use an instant vector and built-in functions, so you don't need to add additional expressions.

   2. Choose **Preview** to verify that the expression is successful.

5. [Optional] To add a recovery threshold, turn the **Custom recovery threshold** toggle on and fill in a value for when your alert rule should stop firing.

You can only add one recovery threshold in a query and it must be the alert condition.

6. Choose **Set as alert condition** on the query or expression you want to set as your alert condition.

Use alert rule evaluation to determine how frequently an alert rule should be evaluated and how quickly it should change its state.

To do this, you need to make sure that your alert rule is in the right evaluation group and set a pending period time that works best for your use case.

To set alert evaluation behavior

1. Select a folder or choose **+ New folder**.
2. Select an evaluation group or click **+ New evaluation group**.

   If you are creating a new evaluation group, specify the interval for the group.

   All rules within the same group are evaluated concurrently over the same time interval.
3. Enter a pending period.

   The pending period is the period in which an alert rule can be in breach of the condition until it fires.

   Once a condition is met, the alert goes into the **Pending** state. If the condition remains active for the duration specified, the alert transitions to the **Firing** state, else it reverts to the **Normal** state.
4. Turn on pause alert notifications, if required.

> **Note**
>
> Pause alert rule evaluation to prevent noisy alerting while tuning your alerts. Pausing stops alert rule evaluation and does not create any alert instances. This is different to mute timings, which stop notifications from being delivered, but still allow for alert rule evaluation and the creation of alert instances.
> You can pause alert rule evaluation to prevent noisy alerting while tuning your alerts. Pausing stops alert rule evaluation and does not create any alert instances. This is

different to mute timings, which stop notifications from being delivered, but still allow for alert rule evaluation and the creation of alert instances.

5. In **Configure no data and error handling**, configure alerting behavior in the absence of data.

   Use the guidelines later in this section.

Add labels to your alert rules to set which notification policy should handle your firing alert instances.

All alert rules and instances, irrespective of their labels, match the default notification policy. If there are no nested policies, or no nested policies match the labels in the alert rule or alert instance, then the default notification policy is the matching policy.

To configure notifications

1. Add labels if you want to change the way your notifications are routed.

   Add custom labels by selecting existing key-value pairs from the drop down, or add new labels by entering the new key or value.
2. Preview your alert instance routing set up.

   Based on the labels added, alert instances are routed to the notification policies displayed.

   Expand each notification policy to view more details.
3. Choose **See details** to view alert routing details and a preview.

Add annotations to provide more context on the alert in your alert notification message.

Annotations add metadata to provide more information on the alert in your alert notification message. For example, add a **Summary** annotation to tell you which value caused the alert to fire or which server it happened on.

To add annotations

1. [Optional] Add a summary.

   Short summary of what happened and why.
2. [Optional] Add a description.

Description of what the alert rule does.

3. [Optional] Add a Runbook URL.

   Webpage where you keep your runbook for the alert

4. [Optional] Add a custom annotation
5. [Optional] Add a dashboard and panel link.

   Links alerts to panels in a dashboard.

6. Choose **Save rule**.

Single and multi-dimensional rule

For Grafana managed alerts, you can create a rule with a classic condition or you can create a multi-dimensional rule.

- **Rule with classic condition**

  Use the classic condition expression to create a rule that triggers a single alert when its condition is met. For a query that returns multiple series, Grafana does not track the alert state of each series. As a result, Grafana sends only a single alert even when alert conditions are met for multiple series.

- **Multi-dimensional rule**

  To generate a separate alert for each series, create a multi-dimensional rule. Use `Math`, `Reduce`, or `Resample` expressions to create a multi-dimensional rule. For example:

  - Add a `Reduce` expression for each query to aggregate values in the selected time range into a single value (not needed for rules using numeric data).
  - Add a `Math` expression with the condition for the rule. Not needed in case a query or a reduce expression already returns `0` if rule should not fire, or a positive number if it should fire. Some examples: `$B > 70` if it should fire in case value of B query/expression is more than 70. `$B < $C * 100` in case it should fire if value of B is less than value of C multiplied by 100. If queries being compared have multiple series in their results, series from different queries are matched if they have the same labels or one is a subset of the other.

> **Note**
>
> Grafana does not support alert queries with template variables. More information is available at https://community.grafana.com/t/template-variables-are-not-supported-in-alert-queries-while-setting-up-alert/2514.

Configure no data and error handling

Configure alerting behavior when your alert rule evaluation returns no data or an error.

> **Note**
>
> Alert rules that are configured to fire when an evaluation returns no data or error only fire when the entire duration of the evaluation period has finished. This means that rather than immediately firing when the alert rule condition is breached, the alert rule waits until the time set as the **For** field has finished and then fires, reducing alert noise and allowing for temporary data availability issues.

If your alert rule evaluation returns no data, you can set the state on your alert rule to appear as follows:

No Data	Description
No Data	Creates a new alert `DatasourceNoData` with the name and UID of the alert rule, and UID of the datasource that returned no data as labels.
Alerting	Sets alert rule state to `Alerting`. The alert rule waits until the time set in the **For** field has finished before firing.
Ok	Sets alert rule state to `Normal`.

If your evaluation returns an error, you can set the state on your alert rule to appear as follows:

Error	Description
Error	Creates an alert instance `DatasourceError` with the name and UID of the alert rule, and UID of the datasource that returned no data as labels.
Alerting	Sets alert rule state to `Alerting`. The alert rule waits until the time set in the **For** field has finished before firing.
Ok	Sets alert rule state to `Normal`.

Resolve stale alert instances

An alert instance is considered stale if its dimension or series has disappeared from the query results entirely for two evaluation intervals.

Stale alert instances that are in the `Alerting`/`NoData`/`Error` states are automatically marked as `Resolved` and the `grafana_state_reason` annotation is added to the alert instance with the reason `MissingSeries`.

Create alerts from panels

Create alerts from any panel type. This means you can reuse the queries in the panel and create alerts based on them.

1. Navigate to a dashboard in the **Dashboards** section.
2. In the top right corner of the panel, choose the three dots (ellipses).
3. From the dropdown menu, select **More...** and then choose **New alert rule**.

This will open the alert rule form, allowing you to configure and create your alert based on the current panel's query.

Configure data source managed alert rules

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Create alert rules for an external Grafana Mimir or Loki instance that has ruler API enabled; these are called data source managed alert rules.

Note

Alert rules for an external Grafana Mimir or Loki instance can be edited or deleted by users with Editor or Admin roles.
If you delete an alerting resource created in the UI, you can no longer retrieve it. To make a backup of your configuration and to be able to restore deleted alerting resources, create your alerting resources using Terraform, or the Alerting API.

Prerequisites

- Verify that you have write permission to the Prometheus or Loki data source. Otherwise, you will not be able to create or update Grafana Mimir managed alert rules.
- For Grafana Mimir and Loki data sources, enable the Ruler API by configuring their respective services.
  - **Loki** - The `local` rule storage type, default for the Loki data source, supports only viewing of rules. To edit rules, configure one of the other rule storage types.
  - **Grafana Mimir** - use the `/prometheus` prefix. The Prometheus data source supports both Grafana Mimir and Prometheus, and Grafana expects that both the Query API and Ruler API are under the same URL. You cannot provide a separate URL for the Ruler API.

> **Note**
>
> If you do not want to manage alert rules for a particular Loki or Prometheus data source, go to its settings and clear the **Manage alerts via alerting UI** checkbox.

In the following procedures, we'll guide you through the process of creating your data source managed alert rules.

To create a data source-managed alert rule, use the in-workspace alert creation flow and follow these steps to help you.

To set the alert rule name

1. Choose **Alerting** -> **Alert rules** -> **+ New alert rule**.
2. Enter a name to identify your alert rule.

   This name is displayed in the alert rule list. It is also the `alertname` label for every alert instance that is created from this rule.

Define a query to get the data you want to measure and a condition that needs to be met before an alert rule fires.

To define query and condition

1. All alert rules are managed by Grafana by default. To switch to a data source managed alert rule, choose **Switch to data source-managed alert rule**.
2. Select a data source from the drop-down list.

   You can also choose **Open advanced data source picker** to see more options, including adding a data source (Admins only).
3. Enter a PromQL or LogQL query.
4. Choose **Preview alerts**.

Use alert rule evaluation to determine how frequently an alert rule should be evaluated and how quickly it should change its state.

To set alert evaluation behavior

1. Select a namespace or choose **+ New namespace**.
2. Select an evaluation group or choose **+ New evaluation group**.

   If you are creating a new evaluation group, specify the interval for the group.

   All rules within the same group are evaluated sequentially over the same time interval.
3. Enter a pending period.

   The pending period is the period in which an alert rule can be in breach of the condition until it fires.

   Once a condition is met, the alert goes into the `Pending` state. If the condition remains active for the duration specified, the alert transitions to the `Firing` state, else it reverts to the `Normal` state.

Add labels to your alert rules to set which notification policy should handle your firing alert instances.

All alert rules and instances, irrespective of their labels, match the default notification policy. If there are no nested policies, or no nested policies match the labels in the alert rule or alert instance, then the default notification policy is the matching policy.

Configure notifications

- Add labels if you want to change the way your notifications are routed.

  Add custom labels by selecting existing key-value pairs from the drop down, or add new labels by entering the new key or value.

Add annotations to provide more context on the alert in your alert notifications.

Annotations add metadata to provide more information on the alert in your alert notifications. For example, add a `Summary` annotation to tell you which value caused the alert to fire or which server it happened on.

To add annotations

1. [Optional] Add a summary.

Short summary of what happened and why.

2. [Optional] Add a description.

   Description of what the alert rule does.

3. [Optional] Add a Runbook URL.

   Webpage where you keep your runbook for the alert

4. [Optional] Add a custom annotation
5. [Optional] Add a dashboard and panel link.

   Links alerts to panels in a dashboard.

6. Choose **Save rule**.

Configure recording rules

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can create and manage recording rules for an external Grafana Mimir or Loki instance. Recording rules calculate frequently needed expressions or computationally expensive expressions in advance and save the result as a new set of time series. Querying this new time series is faster, especially for dashboards since they query the same expression every time the dashboards refresh.

> **Note**
>
> Recording rules are run as instance rules, and run every 10 seconds.

Prerequisites

- Verify that you have write permissions to the Prometheus or Loki data source. You will be creating or updating alerting rules in your data source.
- For Grafana Mimir and Loki data sources, enable the ruler API by configuring their respective services.
  - **Loki** – The `local` rule storage type, default for the Loki data source, supports only viewing of rules. To edit rules, configure one of the other storage types.
  - **Grafana Mimir** – Use the `/prometheus` prefix. The Prometheus data source supports both Grafana Mimir and Prometheus, and Grafana expects that both the Query API and Ruler API are under the same URL. You cannot provide a separate URL for the Ruler API.

> **Note**
>
> If you do not want to manage alerting rules for a particular Loki or Prometheus data source, go to its settings and clear the **Manage alerts via Alerting UI** check box.

To create recording rules

1. From your Grafana console, in the Grafana menu, choose **Alerting**, **Alert rules**.
2. Choose **New recording rule**.
3. Set rule name.

   The recording rule name must be a Prometheus metric name and contain no whitespace.
4. Define query
   - Select your Loki or Prometheus data source.
   - Enter a query.
5. Add namespace and group.
   - From the **Namespace** dropdown, select an existing rule namespace or add a new one. Namespaces can contain one or more rule groups and only have an organizational purpose.
   - From the **Group** dropdown, select an existing group within the selected namespace or add a new one. Newly created rules are appended to the end of the group. Rules within a group are run sequentially at a regular interval, with the same evaluation time.
6. Add labels.

- Add custom labels selecting existing key-value pairs from the dropdown, or add new labels by entering the new key or value.

7. Choose **Save rule** to save the rule, or **Save rule and exit** to save the rule and go back to the Alerting page.

Configure contact points

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use contact points to define how your contacts are notified when an alert rule fires.

Note

You can create and edit contact points for Grafana managed alerts. Contact points for data source managed alerts are read-only.

Working with contact points

The following procedures show how to add, edit, delete, and test a contact point.

To add a contact point

1. In the left-side menu, choose **Alerting**.
2. Choose **Contact points**.
3. From the **Choose Alertmanager** dropdown, select an Alertmanager. The Grafana Alertmanager is selected by default.
4. On the **Contact Points** tab, choose **+ Add contact point**.
5. Enter a **Name** for the contact point.

6. From **Integration**, choose a type, and fill out the mandatory fields based on that type. For example, if you choose Slack, enter the Slack channels and users who should be contacted.
7. If available for the contact point you selected, choose any desired **Optional settings** to specify additional settings.
8. Under **Notification settings**, optionally select **Disable resolved message** if you do not want to be notified when an alert resolves.
9. To add another contact point integration, choose **Add contact point integration** and repeat the steps for each contact point type needed.
10. Save your changes.

To edit a contact point

1. In the left-side menu, choose **Alerting**.
2. Choose **Contact points** to see a list of existing contact points.
3. Select the contact point to edit, then choose **Edit**.
4. Update the contact point, and then save your changes.

You can delete contact points that are not in use by a notification policy.

To delete a contact point

1. In the left-side menu, choose **Alerting**.
2. Choose **Contact points** to open the list of existing contact points.
3. On the **Contact points**, select the contact point to delete, then choose **More**, **Delete**.
4. In the confirmation dialog box, choose **Yes, delete**.

> **Note**
>
> If the contact point is in use by a notification policy, you must delete the notification policy or edit it to use a different contact point before deleting the contact point.

After your contact point is created, you can send a test notification to verify that it is configured properly.

To send a test notification

1. In the left-side menu, choose **Alerting**.
2. Choose **Contact points** to open the list of existing contact points.
3. On the **Contact points**, select the contact point to test, then choose **Edit**. You can also create a new contact point if needed.
4. Choose **Test** to open the contact point testing dialog.
5. Choose whether to send a predefined test notification or choose **Custom** to add your own custom annotations and labels in the test notification.
6. Choose **Send test notification** to test the alert with the given contact points.

Configure contact point integrations

Configure contact point integrations in Grafana to select your preferred communication channel for receiving notifications when your alert rules are firing. Each integration has its own configuration options and setup process. In most cases, this involves providing an API key or a Webhook URL.

Once configured, you can use integrations as part of your contact points to receive notifications whenever your alert changes its state. In this section, we'll cover the basic steps to configure an integration, using PagerDuty as an example, so you can start receiving real-time alerts and stay on top of your monitoring data.

List of supported integrations

The following table lists the contact point types supported by Grafana.

Name	Type
Amazon SNS	`sns`
OpsGenie	`opsgenie`
Pager Duty	`pagerduty`
Slack	`slack`
VictorOps	`victorops`

Configuring PagerDuty for alerting

To set up PagerDuty, you must provide an integration key. Provide the following details.

Setting	Description		
Integration Key	Integration key for PagerDuty		
Severity	Level for dynamic notifications. Default is `critical`.		
Custom Details	Additional details about the event		

The `CustomDetails` field is an object containing arbitrary key-value pairs. The user-defined details are merged with the ones used by default.

The default values for `CustomDetails` are:

```
{
 "firing":       `{{ template "__text_alert_list" .Alerts.Firing }}`,
 "resolved":     `{{ template "__text_alert_list" .Alerts.Resolved }}`,
 "num_firing":   `{{ .Alerts.Firing | len }}`,
 "num_resolved": `{{ .Alerts.Resolved | len }}`,
}
```

In case of duplicate keys, the user-defined details overwrite the default ones.

Configure notification policies

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Notification policies determine how alerts are routed to contact points.

Policies have a tree structure, where each policy can have one or more nested policies. Each policy, except for the default policy, can also match specific alert labels.

Each alert is evaluated by the default policy and subsequently by each nested policy.

If you enable the `Continue matching subsequent sibling nodes` option for a nested policy, then evaluation continues even after one or more matches. A parent policy's configuration settings and contact point information govern the behavior of an alert that does not match any of the child policies. A default policy governs any alert that does not match a nested policy.

For more information on notification policies, see Notifications.

The following procedures show you how to create and manage notification policies.

To edit the default notification policy

1. In the left-side menu, choose **Alerting**.
2. Choose **Notification policies**.
3. From the **Choose Alertmanager** dropdown, select the Alertmanager you want to edit.
4. In the **Default policy** section, choose **...**, **Edit**.
5. In **Default contact point**, update the contact point where notifications should be sent for rules when alert rules do not match any specific policy.
6. In **Group by**, choose the labels to group alerts by. If multiple alerts are matched for this policy, then they are grouped by these labels. A notification is sent per group. If the field is empty (the default), then all notifications are sent in a single group. Use a special label, `...` to group alerts by all labels (which effectively disables grouping).
7. In **Timing options**, select from the following options.
   - **Group wait** – Time to wait to buffer alerts of the same group before sending an initial notification. The default is 30 seconds.
   - **Group interval** – Minimum time interval between two notifications for a group. The default is 5 minutes.
   - **Repeat interval** – Minimum time interval before resending a notification if no new alerts were added to the group. The default is 4 hours.
8. Choose **Save** to save your changes.

To create a new notification policy, you need to follow its tree structure. New policies created on the trunk of the tree (the default policy), are the tree branches. Each branch can have their own nested policies. This is why you will always be adding a new **nested** policy, either under the default policy, or under an already nested policy.

To add a new nested policy

1. In the left-side menu, choose **Alerting**.
2. Choose **Notification policies**.
3. From the **Choose Alertmanager** dropdown, select the Alertmanager you want to edit.
4. To add a top level specific policy, go to the Specific routing section (eitehr to the default policy, or to antoher existing policy in which you would like to add a new nested policy) and choose **+ New nested policy**.
5. In the matching labels section, add one or more rules for matching alert labels.
6. In the **Contact point** dropdown, select the contact point to send notifications to if an alert matches only this specific policy and not any of the nested policies.
7. Optionally, enable **Continue matching subsequent sibling nodes** to continue matching sibling policies even after the alert matched the current policy. When this option is enabled, you can get more than one notification for one alert.
8. Optionally, enable **Override grouping** to specify the same grouping as the default policy. If the option is not enabled, the default policy grouping is used.
9. Optionally, enable **Override general timings** to override the timing options configured in the group notification policy.
10. Choose **Save policy** to save your changes.

To edit a nested policy

1. In the left-side menu, choose **Alerting**.
2. Choose **Notification policies**.
3. Select the policy that you want to edit, then choose **...**, **Edit**.
4. Make any changes (as when adding a nested policy).
5. Save your changes.

Searching for policies

You can search within the tree of policies by *Label matchers* or *contact points*.

- To search by contact point, enter a partial or full name of a contact point in the **Search by contact point** field. The policies that use that contact point will be highlighted in the user interface.
- To search by label, enter a valid label matcher in the **Search by matchers** input field. Multiple matchers can be entered, separated by a comma. For example, a valid matcher input could be `severity=high, region=~EMEA|NA`.

> **Note**
>
> When searching by label, all matched policies will be exact matches. Partial matches and regex-style matches are not supported.

Manage your alerts

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Once you have set up your alert rules, contact points, and notification policies, you can use Grafana alerting to manage your alerts in practice.

Topics

- Customize notifications
- Manage contact points
- Silencing alert notifications
- View and filter alert rules
- Mute timings
- View the state and health of alert rules

- View and filter by alert groups
- View notification errors

Customize notifications

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Customize your notifications with notifications templates.

You can use notification templates to change the title, message, and format of the message in your notifications.

Notification templates are not tied to specific contact point integrations, such as Amazon SNS or Slack. However, you can choose to create separate notification templates for different contact point integrations.

You can use notification templates to:

- Add, remove, or re-order information in the notification including the summary, description, labels and annotations, values, and links
- Format text in bold and italic, and add or remove line breaks

You cannot use notification templates to:

- Change the design of notifications in instant messaging services such as Slack and Microsoft Teams

Topics

- Using Go's templating language
- Create notification templates

- Using notification templates
- Template reference

Using Go's templating language

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You write notification templates in Go's templating language, text/template.

This section provides an overview of Go's templating language and writing templates in text/template.

Dot

In text/template there is a special cursor called dot, and is written as `.`. You can think of this cursor as a variable whose value changes depending where in the template it is used. For example, at the start of a notification template `.` refers to the `ExtendedData` object, which contains a number of fields including `Alerts`, `Status`, `GroupLabels`, `CommonLabels`, `CommonAnnotations` and `ExternalURL`. However, dot might refer to something else when used in a `range` over a list, when used inside a `with`, or when writing feature templates to be used in other templates. You can see examples of this in Create notification templates, and all data and functions in the Template reference.

Opening and closing tags

In text/template, templates start with `{{` and end with `}}` irrespective of whether the template prints a variable or runs control structures such as if statements. This is different from other templating languages such as Jinja where printing a variable uses `{{` and `}}` and control structures use `{%` and `%}`.

Print

To print the value of something use {{ and }}. You can print the value of dot, a field of dot, the result of a function, and the value of a variable. For example, to print the Alerts field where dot refers to ExtendedData you would write the following:

```
{{ .Alerts }}
```

Iterate over alerts

To print just the labels of each alert, rather than all information about the alert, you can use a range to iterate the alerts in ExtendedData:

```
{{ range .Alerts }}
{{ .Labels }}
{{ end }}
```

Inside the range dot no longer refers to ExtendedData, but to an Alert. You can use {{ .Labels }} to print the labels of each alert. This works because {{ range .Alerts }} changes dot to refer to the current alert in the list of alerts. When the range is finished dot is reset to the value it had before the start of the range, which in this example is ExtendedData:

```
{{ range .Alerts }}
{{ .Labels }}
{{ end }}
{{/* does not work, .Labels does not exist here */}}
{{ .Labels }}
{{/* works, cursor was reset */}}
{{ .Status }}
```

Iterate over annotations and labels

Let's write a template to print the labels of each alert in the format The name of the label is $name, and the value is $value, where $name and $value contain the name and value of each label.

Like in the previous example, use a range to iterate over the alerts in .Alerts such that dot refers to the current alert in the list of alerts, and then use a second range on the sorted labels so dot is updated a second time to refer to the current label. Inside the second range use .Name and .Value to print the name and value of each label:

```
{{ range .Alerts }}
{{ range .Labels.SortedPairs }}
The name of the label is {{ .Name }}, and the value is {{ .Value }}
{{ end }}
{{ range .Annotations.SortedPairs }}
The name of the annotation is {{ .Name }}, and the value is {{ .Value }}
{{ end }}
{{ end }}
```

The index functions

To print a specific annotation or label use the `index` function.

```
{{ range .Alerts }}
The name of the alert is {{ index .Labels "alertname" }}
{{ end }}
```

If statements

You can use if statements in templates. For example, to print `There are no alerts` if there are no alerts in `.Alerts` you would write the following:

```
{{ if .Alerts }}
There are alerts
{{ else }}
There are no alerts
{{ end }}
```

With

With is similar to if statements, however unlike if statements, `with` updates dot to refer to the value of the with:

```
{{ with .Alerts }}
There are {{ len . }} alert(s)
{{ else }}
There are no alerts
{{ end }}
```

Variables

Variables in text/template must be created within the template. For example, to create a variable called `$variable` with the current value of dot you would write the following:

```
{{ $variable := . }}
```

You can use `$variable` inside a range or `with` and it will refer to the value of dot at the time the variable was defined, not the current value of dot.

For example, you cannot write a template that use `{{ .Labels }}` in the second range because here dot refers to the current label, not the current alert:

```
{{ range .Alerts }}
{{ range .Labels.SortedPairs }}
{{ .Name }} = {{ .Value }}
{{/* does not work because in the second range . is a label not an alert */}}
There are {{ len .Labels }}
{{ end }}
{{ end }}
```

You can fix this by defining a variable called `$alert` in the first range and before the second range:

```
{{ range .Alerts }}
{{ $alert := . }}
{{ range .Labels.SortedPairs }}
{{ .Name }} = {{ .Value }}
{{/* works because $alert refers to the value of dot inside the first range */}}
There are {{ len $alert.Labels }}
{{ end }}
{{ end }}
```

Range with index

You can get the index of each alert within a range by defining index and value variables at the start of the range:

```
{{ $num_alerts := len .Alerts }}
{{ range $index, $alert := .Alerts }}
This is alert {{ $index }} out of {{ $num_alerts }}
{{ end }}
```

Define templates

You can define templates that can be used within other templates, using `define` and the name of the template in double quotes. You should not define templates with the same name as other templates, including default templates such as `__subject`, `__text_values_list`, `__text_alert_list`, `default.title` and `default.message`. Where a template has been created with the same name as a default template, or a template in another notification template, Grafana might use either template. Grafana does not prevent, or show an error message, when there are two or more templates with the same name.

```
{{ define "print_labels" }}
{{ end }}
```

Execute templates

You can execute defined template within your template using `template`, the name of the template in double quotes, and the cursor that should be passed to the template:

```
{{ template "print_labels" . }}
```

Pass data to templates

Within a template dot refers to the value that is passed to the template.

For example, if a template is passed a list of firing alerts then dot refers to that list of firing alerts:

```
{{ template "print_alerts" .Alerts }}
```

If the template is passed the sorted labels for an alert then dot refers to the list of sorted labels:

```
{{ template "print_labels" .SortedLabels }}
```

This is useful when writing reusable templates. For example, to print all alerts you might write the following:

```
{{ template "print_alerts" .Alerts }}
```

Then to print just the firing alerts you could write this:

```
{{ template "print_alerts" .Alerts.Firing }}
```

This works because both `.Alerts` and `.Alerts.Firing` are lists of alerts.

```
{{ define "print_alerts" }}
{{ range . }}
{{ template "print_labels" .SortedLabels }}
{{ end }}
{{ end }}
```

Comments

You can add comments with `{{/*` and `*/}}`:

```
{{/* This is a comment */}}
```

To prevent comments from adding line breaks use:

```
{{- /* This is a comment with no leading or trailing line breaks */ -}}
```

Indentation

You can use indentation, both tabs and spaces, and line breaks, to make templates more readable:

```
{{ range .Alerts }}
  {{ range .Labels.SortedPairs }}
    {{ .Name }} = {{ .Value }}
  {{ end }}
{{ end }}
```

However, indentation in the template will also be present in the text. Next we will see how to remove it.

Remove spaces and line breaks

In text/template use `{{-` and `-}}` to remove leading and trailing spaces and line breaks.

For example, when using indentation and line breaks to make a template more readable:

```
{{ range .Alerts }}
  {{ range .Labels.SortedPairs }}
    {{ .Name }} = {{ .Value }}
  {{ end }}
{{ end }}
```

The indentation and line breaks will also be present in the text:

```
    alertname = "Test"

    grafana_folder = "Test alerts"
```

You can remove the indentation and line breaks from the text changing `}}` to `-}}` at the start of each range:

```
{{ range .Alerts -}}
  {{ range .Labels.SortedPairs -}}
    {{ .Name }} = {{ .Value }}
  {{ end }}
{{ end }}
```

The indentation and line breaks in the template are now absent from the text:

```
alertname = "Test"
grafana_folder = "Test alerts"
```

Create notification templates

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Create reusable notification templates to send to your contact points.

You can add one or more templates to your notification template.

Your notification template name must be unique. You cannot have two templates with the same name in the same notification template or in different notification templates. Avoid defining templates with the same name as default templates, such as: `__subject`, `__text_values_list`, `__text_alert_list`, `default.title` and `default.message`.

In the Contact points tab, you can see a list of your notification templates.

Creating notification templates

To create a notification template

1. Choose **Alerting**, **Contact points**.
2. Choose the **Notification Templates** tab, and then **+ Add notification template**.
3. Choose a name for the notification template, such as `email.subject`.
4. Write the content of the template in the content field.

   For example:

   ```
   {{ if .Alerts.Firing -}}
      {{ len .Alerts.Firing }} firing alerts
      {{ end }}
      {{ if .Alerts.Resolved -}}
      {{ len .Alerts.Resolved }} resolved alerts
      {{ end }}
   ```

5. Save your changes.

   `{{ define "email.subject" }}` (where `email.subject` is the name of your template) and `{{ end }}` is automatically added to the start and end of the content.

To create a notification template that contains more than one template

1. Choose **Alerting**, **Contact points**.
2. Choose the **Notification Templates** tab, and then **+ Add notification template**.
3. Enter a name for the overall notification template. For example, `email`.
4. Write each template in the Content field, including `{{ define "`*`name-of-template`*`" }}` and `{{ end }}` at the start and end of each template. You can use descriptive names for each of the templates in the notification template, for example, `email.subject` or `email.message`. In this case, do not reuse the name of the notification template you entered above.

   Later sections show detailed examples for templates you might create.

5. Click Save.

Preview notification templates

Preview how your notification templates will look before using them in your contact points, helping you understand the result of the template you are creating as well as giving you a chance to fix any errors before saving the template.

> **Note**
>
> Notification previews are only available for Grafana Alertmanager.

To preview your notification templates

1. Choose **Alerting**, **Contact points**.
2. Choose the **Notification Templates** tab, and then **+ Add notification template**, or edit an existing template.
3. Add or update your template content.

   Default data is provided and you can add or edit alert data to it as well as alert instances. You can add alert data directly in the Payload data window itself, or click **Select alert instances** or **Add custom alerts**.
4. [Optional] To add alert data from existing alert instances:

   1. Choose **Select alert instances**.
   2. Hover over the alert instances to view more information about each alert instance/
   3. Choose **Confirm** to add the alert instance to the payload.
5. [Optional] To add alert data using the Alert data editor, choose **Add custom data**:

   1. Add annotations, custom labels, or set a dashboard or panel.
   2. Toggle Firing or resolved, depending on whether you want to add firing or resolved alerts to your notification.
   3. Choose **Add alert data**.
   4. Choose **Refresh preview** to see what your template content will look like and the corresponding payload data.

   If there are any errors in your template, they are displayed in the Preview and you can correct them before saving.

6. Save your changes.

Creating a template for the subject of message

Create a template for the subject of an email that contains the number of firing and resolved alerts, as in this example:

```
1 firing alerts, 0 resolved alerts
```

To create a template for the subject of an email

1. Create a template called `email.subject` with the following content:

   ```
   {{ define "email.subject" }}
   {{ len .Alerts.Firing }} firing alerts, {{ len .Alerts.Resolved }} resolved alerts
   {{ end }}
   ```

2. Use the template when creating your contact point integration by putting it into the **Subject** field with the `template` keyword.

   ```
   {{ template "email.subject" . }}
   ```

Creating a template for the message of an email

Create a template for the message of an email that contains a summary of all firing and resolved alerts, as in this example:

```
There are 2 firing alerts, and 1 resolved alerts

Firing alerts:

- alertname=Test 1 grafana_folder=GrafanaCloud has value(s) B=1
- alertname=Test 2 grafana_folder=GrafanaCloud has value(s) B=2

Resolved alerts:

- alertname=Test 3 grafana_folder=GrafanaCloud has value(s) B=0
```

To create a template for the message of an email

1. Create a notification template called `email` with two templates in the content: `email.message_alert` and `email.message`.

   The `email.message_alert` template is used to print the labels and values for each firing and resolved alert while the `email.message` template contains the structure of the email.

```
{{- define "email.message_alert" -}}
{{- range .Labels.SortedPairs }}{{ .Name }}={{ .Value }} {{ end }} has value(s)
{{- range $k, $v := .Values }} {{ $k }}={{ $v }}{{ end }}
{{- end -}}

{{ define "email.message" }}
There are {{ len .Alerts.Firing }} firing alerts, and {{ len .Alerts.Resolved }}
 resolved alerts

{{ if .Alerts.Firing -}}
Firing alerts:
{{- range .Alerts.Firing }}
- {{ template "email.message_alert" . }}
{{- end }}
{{- end }}

{{ if .Alerts.Resolved -}}
Resolved alerts:
{{- range .Alerts.Resolved }}
- {{ template "email.message_alert" . }}
{{- end }}
{{- end }}

{{ end }}
```

2. Use the template when creating your contact point integration by putting it into the **Text Body** field with the `template` keyword.

```
{{ template "email.message" . }}
```

Creating a template for the title of a Slack message

Create a template for the title of a Slack message that contains the number of firing and resolved alerts, as in the following example:

```
1 firing alerts, 0 resolved alerts
```

To create a template for the title of a Slack message

1. Create a template called `slack.title` with the following content:

   ```
   {{ define "slack.title" }}
   {{ len .Alerts.Firing }} firing alerts, {{ len .Alerts.Resolved }} resolved alerts
   {{ end }}
   ```

2. Execute the template from the title field in your contact point integration.

   ```
   {{ template "slack.title" . }}
   ```

Creating a template for the content of a Slack message

Create a template for the content of a Slack message that contains a description of all firing and resolved alerts, including their labels, annotations, and Dashboard URL.

> **Note**
>
> This template is for Grafana managed alerts only. To use the template for data source managed alerts, delete the references to DashboardURL and SilenceURL. For more information about configuring Prometheus notifications, see the Prometheus documentation on notifications.

```
1 firing alerts:

[firing] Test1
Labels:
- alertname: Test1
- grafana_folder: GrafanaCloud
Annotations:
- description: This is a test alert
```

```
Go to dashboard: https://example.com/d/dlhdLqF4z?orgId=1

1 resolved alerts:

[firing] Test2
Labels:
- alertname: Test2
- grafana_folder: GrafanaCloud
Annotations:
- description: This is another test alert
Go to dashboard: https://example.com/d/dlhdLqF4z?orgId=1
```

To create a template for the content of a Slack message

1. Create a template called `slack` with two templates in the content: `slack.print_alert` and `slack.message`.

   The `slack.print_alert` template is used to print the labels, annotations, and DashboardURL while the `slack.message` template contains the structure of the notification.

   ```
   {{ define "slack.print_alert" -}}
   [{{.Status}}] {{ .Labels.alertname }}
   Labels:
   {{ range .Labels.SortedPairs -}}
   - {{ .Name }}: {{ .Value }}
   {{ end -}}
   {{ if .Annotations -}}
   Annotations:
   {{ range .Annotations.SortedPairs -}}
   - {{ .Name }}: {{ .Value }}
   {{ end -}}
   {{ end -}}
   {{ if .DashboardURL -}}
     Go to dashboard: {{ .DashboardURL }}
   {{- end }}
   {{- end }}

   {{ define "slack.message" -}}
   {{ if .Alerts.Firing -}}
   {{ len .Alerts.Firing }} firing alerts:
   {{ range .Alerts.Firing }}
   {{ template "slack.print_alert" . }}
   {{ end -}}
   ```

```
{{ end }}
{{ if .Alerts.Resolved -}}
{{ len .Alerts.Resolved }} resolved alerts:
{{ range .Alerts.Resolved }}
{{ template "slack.print_alert" .}}
{{ end -}}
{{ end }}
{{- end }}
```

2. Execute the template from the text body field in your contact point integration:

```
{{ template "slack.message" . }}
```

Template both email and Slack with shared templates

Instead of creating separate notification templates for each contact point, such as email and Slack, you can share the same template.

For example, if you want to send an email with this subject and Slack message with this title `1 firing alerts, 0 resolved alerts`, you can create a shared template.

To create a shared template

1. Create a template called `common.subject_title` with the following content:

```
{{ define "common.subject_title" }}
{{ len .Alerts.Firing }} firing alerts, {{ len .Alerts.Resolved }} resolved alerts
{{ end }}
```

2. For email, run the template from the subject field in your email contact point integration:

```
{{ template "common.subject_title" . }}
```

3. For Slack, run the template from the title field in your Slack contact point integration:

```
{{ template "common.subject_title" . }}
```

Using notification templates

Use templates in contact points to customize your notifications.

To use a template when creating a contact point

1. From the **Alerting** menu, choose the **Contact points** tab to see a list of existing contact points.
2. Choose **New**. Alternately, you can edit an existing contact point by choosing the **Edit** icon.
3. Enter the templates you wish to use in a field, such as **Message** or **Subject**. To enter a template, use the form `{{ template "template_name" . }}`, replacing *template_name* with the name of the template you want to use.
4. Choose **Save contact point**.

Template reference

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This section provides reference information for creating your templates.

Alert (type)

The alert type contains the following data.

Name	Kind	Description	Example
Status	string	`firing` or `resolved`.	`{{ .Status }}`
Labels	KeyValue	A set of labels attached to the alert.	`{{ .Labels }}`
Annotations	KeyValue	A set of annotations attached to the alert.	`{{ .Annotations }}`

Name	Kind	Description	Example
Values	KeyValue	The values of all expressions, including Classic Conditions	`{{ .Values }}`
StartsAt	time.Time	Time the alert started firing.	`{{ .StartsAt }}`
EndsAt	time.Time	Only set if the end time of an alert is known. Otherwise set to a configurable timeout period from the time since the last alert was received.	`{{ .EndsAt }}`
GeneratorURL	string	A back link to Grafana or external Alertmanager.	`{{ .GeneratorURL }}`
SilenceURL	string	A link to silence the alert (with labels for this alert pre-filled). Only for Grafana managed alerts.	`{{ .SilenceURL}}`
DashboardURL	string	Link to grafana dashboard, if alert rule belongs to one. Only for Grafana managed alerts.	`{{ .DashboardURL }}`

Name	Kind	Description	Example
PanelURL	string	Link to grafana dashboard panel, if alert rule belongs to one. Only for Grafana managed alerts.	`{{ .PanelURL }}`
Fingerprint	string	Fingerprint that can be used to identify the alert.	`{{ .Fingerprint }}`
ValueString	string	A string that contains the labels and value of each reduced expression in the alert.	`{{ .ValueString }}`

ExtendedData

The ExtendedData object contains the following properties.

Name	Kind	Description	Example
Receiver	`string`	The name of the contact point sending the notification.	`{{ .Receiver }}`
Status	`string`	The status is `firing` if at least one alert is firing, otherwise `resolved`.	`{{ .Status }}`
Alerts	`[]Alert`	List of all firing and resolved alerts in this notification.	`There are {{ len .Alerts }} alerts`

Name	Kind	Description	Example
Firing alerts	`[]Alert`	List of all firing alerts in this notification.	`There are {{ len .Alerts.Firing }} firing alerts`
Resolved alerts	`[]Alert`	List of all resolved alerts in this notification.	`There are {{ len .Alerts.Resolved }} resolved alerts`
GroupLabels	`KeyValue`	The labels that group these alerts int his notification.	`{{ .GroupLabels }}`
CommonLabels	`KeyValue`	The labels common to all alerts in this notification.	`{{ .CommonLabels }}`
CommonAnnotations	`KeyValue`	The annotations common to all alerts in this notification.	`{{ .CommonAnnotations }}`
ExternalURL	`string`	A link to the Grafana workspace or Alertmanager that sent this notification.	`{{ .ExternalURL }}`

KeyValue type

The `KeyValue` type is a set of key/value string pairs that represent labels and annotations.

In addition to direct access of the data stored as a `KeyValue`, there are also methods for sorting, removing, and transforming the data.

Name	Arguments	Returns	Notes	Example
SortedPairs		Sorted list of key and value string pairs		`{{ .Annotations.SortedPairs }}`
Remove	[]string	KeyValue	Returns a copy of the Key/Value map without the given keys.	`{{ .Annotations.Remove "summary" }}`
Names		[]string	List of names	`{{ .Names }}`
Values		[]string	List of values	`{{ .Values }}`

Time

Time is from the Go `time` package. You can print a time in a number of different formats. For example, to print the time that an alert fired in the format `Monday, 1st January 2022 at 10:00AM`, you write the following template:

```
{{ .StartsAt.Format "Monday, 2 January 2006 at 3:04PM" }}
```

You can find a reference for Go's time format here.

Manage contact points

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The **Contact points** list view lists all existing contact points and notification templates.

On the **Contact points** tab, you can:

- Search for names and types of contact points and integrations.
- View all existing contact points and integrations.
- View how many notification policies each contact point is being used for, and navigate directly to the linked notification policies.
- View the status of notification deliveries.
- Export individual contact points or all contact points in JSON, YAML, or Terraform format.
- Delete contact points that are not in use by a notification policy.

On the **Notification templates** tab, you can:

- View, edit, copy, or delete existing notification templates.

Silencing alert notifications

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can suppress alert notifications with a *silence*. A silence only stops notifications from being created: Silences do not prevent alert rules from being evaluated, and they do not stop alerting instances from being shown in the user interface. When you silence an alert, you specify a window of time for it to be suppressed.

Note

To suppress alert notifications at regular time intervals, for example, during regular maintenance periods, use Mute timings rather than silences.

To add a silence

1. From your Grafana console, in the Grafana menu, choose **Alerting**.

2. Choose **Silences**.
3. Choose an Alertmanager from the **Alertmanager** dropdown.
4. Choose **Create Silence**.
5. Select the start and end date in **Silence start and end** to indicate when the silence should go into effect and when it should end.
6. As an alternative to setting an end time, in **Duration**, specify how long the silence is enforced. This automatically updates the end time in the **Silence start and end** field.
7. In the **Label** and **Value** fields, enter one or more *Matching Labels*. Matchers determine which rules the silence applies to. Any matching alerts (in firing state), will show in the **Affected alerts instances** field.
8. Optionally, add a **Comment** describing the silence.
9. Choose **Submit**.

To edit a silence

1. From your Grafana console, in the Grafana menu, choose **Alerting**.
2. Choose **Silences** to view the list of existing silences.
3. Find the silence you want to edit, then choose **Edit** (pen icon).
4. Make any desired changes, then choose **Submit** to save your changes.

You can edit an existing silence by choosing the **Edit** icon (pen).

To create a URL link to a silence form

When linking to a silence form, provide the default matching labels and comment via `matcher` and `comment` query parameters. The `matcher` parameter should be in the following format `[label][operator][value]` where the `operator` parameter can be one of the following: = (equals, not regex), `!=` (not equals, not regex), =~ (equals, regex), `!~` (not equals, regex). The URL can contain many query parameters with the key `matcher`. For example, to link to silence form with matching labels `severity=critical` & `cluster!~europe-.*` and comment `Silence critical EU alerts`, create a URL `https://mygrafana/alerting/silence/new?matcher=severity%3Dcritical&matcher=cluster!~europe-*&comment=Silence%20critical%20EU%20alert`.

To link to a new silence page for an external Alertmanager, add an `alertmanager` query parameter

To remove a silence

1. From your Grafana console, in the Grafana menu, choose **Alerting**.
2. Choose **Silences** to view the list of existing silences.
3. Select the silence that you want to end, and choose **Unsilence**. This ends the alert suppression.

> **Note**
>
> Unsilencing ends the alert suppression, as if the end time was set for the current time. Silences that have ended (automatically or manually) are retained and listed for five days. You cannot remove a silence from the list manually.

View and filter alert rules

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The **Alerting** page lists alerting rules. By default, rules are grouped by types of data sources. The **Grafana** section lists rules managed by Grafana. Alert rules for Prometheus compatible data sources are also listed here. You can view alerting rules for Prometheus compatible data sources but you cannot edit them.

The Mimir/Cortex/Loki rules section lists all rules for Mimir, Cortex, or Loki data sources. Cloud alert rules are also listed in this section.

When managing large volumes of alerts, you can use extended alert rule search capabilities to filter on folders, evaluation groups, and rules. Additionally, you can filter alert rules by their properties like labels, state, type, and health.

View alert rules

Using Grafana alerts, you can view all of your alerts in one page.

To view alerting details

1. From your Grafana console, in the Grafana menu, choose **Alerting**, **Alert rules**. By default, the list view is displayed.
2. In **View as**, you can toggle between the Grouped, List, and State views by choosing the option you prefer.
3. Expand the rule row to view the rule labels, annotations, data sources, the rule queries, and a list of alert instances resulting from the rule.

From this page, you can also make copies of an alert rule to help you reuse existing rules.

Export alert rules

You can export rules to YAML or JSON in the Grafana workspace.

- Choose the **Export rule group** icon next to each alert rule group to export to YAML, JSON, or Terraform.
- Choose **Export rules** to export all Grafana managed alert rules to YAML, JSON, or Terraform.
- Choose **More**, **Modify export** next to each individual alert rule within a group to edit provisioned alert rules and export a modified version.

View query definitions for provisioned alerts

View read-only query definitions for provisioned alerts. Check quickly if your alert rule queries are correct, without diving into your "as-code" repository for rule definitions.

Grouped view

Grouped view shows Grafana alert rules grouped by folder and Loki or Prometheus alert rules grouped by `namespace + group`. This is the default rule list view, intended for managing rules. You can expand each group to view a list of rules in this group. Expand a rule further to view its details. You can also expand action buttons and alerts resulting from the rule to view their details.

State view

State view shows alert rules grouped by state. Use this view to get an overview of which rules are in what state. Each rule can be expanded to view its details. Action buttons and any alerts generated by this rule, and each alert can be further expanded to view its details.

Filtering alert rules

You can filter the alerting rules that appear on the **Alerting** page in several ways.

To filter alert rules

1. From **Select data sources**, select a data source.. You can see alert rules that query the selected data source.
2. In **Search by label**, enter search criteria using label selectors. For example, `environment=production;region=~US|EU,severity!=warning`.
3. From **Filter alerts by state**, select an alerting state you want to see. You can see alerting rules that match that state. Rules matching other states are hidden.

Mute timings

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A mute timing is a recurring interval of time when no new notifications for a policy are generated or sent. Use them to prevent alerts from firing a specific and reoccurring period, for example, a regular maintenance period.

Similar to silences, mute timings do not prevent alert rules from being evaluated, nor do they stop alert instances from being shown in the user interface. They only prevent notifications from being created.

You can configure Grafana managed mute timings as well as mute timings for an external Alertmanager data source.

Mute timings vs Silences

The following table highlights the differences between mute timings and silences.

Mute timing	Silence
Uses time interval definitions that can reoccur.	Has a fixed start and end time.
Is created and then added to notification policies.	Uses labels to match against an alert to determine whether to silence or not.

Adding a mute timing

You can create mute timings in your Grafana workspace.

To add a mute timing

1. From your Grafana console, in the Grafana menu, choose **Alerting**.
2. Choose **Notification policies**, and then select the **Mute Timings** tab.
3. From the **Alertmanager** dropdown, select the Alertmanager you want to edit.
4. Choose the **+ Add mute timing** button.
5. Fill out the form to create a time interval to match against for your mute timing.
6. Save your mute timing.

Adding a mute timing to a notification policy

Once you have a mute timing, you use it by adding it to notification policy that you want to mute at regular intervals.

To add a mute timing to a notification policy

1. From your Grafana console, in the Grafana menu, choose **Alerting**.
2. Choose **Notification policies**, and then select the **Notification Policies** tab.
3. Select the notification policy you would like to add the mute timing to, and choose **...**, **Edit**.
4. From the **Mute timings** dropdown, select the mute timings you would like to add to the policy.
5. Save your changes.

Time intervals

A time interval is a specific duration during which alerts are suppressed. The duration typically consists of a specific time range and the days of the week, month, or year.

Support time interval options are:

- **Time range** – The time inclusive of the start and exclusive of the end time (in UTC, if no location has been selected, otherwise local time.
- **Location** – Sets the location for the timing—the time range is displayed in local time for the location.
- **Days of the week** – The day or range of days of the week. For example, `monday:thursday`.
- **Days of the month** – The dates within a month. Values can range from `1-31`. Negative values specify days of the month in reverse order, so `-1` represents the last day of the month.
- **Months** – The months of the year in either numerical of full calendar month name. For example, `1, may:august`.
- **Years** – The year or years for the interval. For example, `2023:2024`.

Each of these elements can be a list, and at least one item in the element must be satisfied to be a match. Fields also support ranges, using `:`. For example, `monday:thursday`.

If a field is left blank, any moment of time will match the field. For an instant of to match a complete time intervale, all fields must match. A mute timing can contain multiple time intervals.

If you want to specify an exact duration, specify all the options needed for that duration. For example, if you want to create a time interval for the first Monday of the month, for March, June, September, and December, between the hours of 12:00 and 24:00 UTC, your time interval specification could be:

- Time range:
  - Start time: `12:00`
  - End time: `24:00`
- Days of the week: `monday`
- Months: `3, 6, 9, 12`
- Days of the month: `1:7`

View the state and health of alert rules

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The state and health of alert rules gives you several key status indicators about your alerts.

There are three components:

- Alert rule state
- Alert instance state
- Alert rule health

Although related, each component conveys subtly different information.

To view the state and health of your alert rules

1. From your Grafana console, in the Grafana menu, choose **Alerting**.
2. Choose **Alert rules** to view the list of existing alerts.
3. Choose an alert rule to view its state and health.

Alert rule state

An alert rule can be in any of the following states:

State	Description		
Normal	None of the time series returned by the evaluation engine		

State	Description
	is in a `pending` or `firing` state.
Pending	At least one time series returned by the evaluation engine is `pending`.
Firing	At least one time series returned by the evaluation engine is `firing`.

> **Note**
>
> Alerts transition first to `pending` and then `firing`, thus it takes at least two evaluation cycles before an alert is fired.

Alert instance state

An alert instance can be in any of the following states:

State	Description
Normal	The state of an alert that is neither `pending` nor `firing`. Everything is working as expected.
Pending	The state of an alert that has been active for less than the

State	Description		
	configured threshold duration.		
Alerting	The state of an alert that has been active for longer than the configured threshold duration.		
No data	No data has been received for the configured time window.		
Alerting	An error occurred when attempting to evaluate an alerting rule.		

Keep last state

An alert rule can be configured to keep the last state when a `NoData` or `Error` state is encountered. This will both prevent alerts from firing, and from resolving and re-firing. Just like normal evaluation, the alert rule will transition from `pending` to `firing` after the pending period has elapsed.

Alert rule health

An alert rule can have one of the following health statuses.

State	Description		
Ok	No errors when evaluating the alert rule.		

State	Description		
Error	An error occurred when evaluating the alert rule.		
NoData	The absence of data in at least one time series returned during a rule evaluation.		
{status}, KeepLast	The rule would have received another status, but was configured to keep the last state of the alert rule.		

Special alerts for NoData and Error

When evaluation of an alert rule produces the state `NoData` or `Error`, Grafana alerting will generate alert instances that have the following additional labels.

Label	Description		
alertname	Either `DatasourceNoData` or `DatasourceError`, depending on the state.		
datasource_uid	The UID of the data source that caused the state.		

> **Note**
>
> You will need to set the no data or error handling to `NoData` or `Error` in the alert rule, as described in the Configure Grafana managed alert rules topic, to generate the additional labels.

You can handle these alerts the same way as regular alerts, including adding silences, routing to a contact point, and so on.

View and filter by alert groups

> This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Alert groups show grouped alerts from an Alertmanager instance. By default, alert rules are grouped by the label keys for the default policy in notification policies. Grouping common alert rules into a single alert group prevents duplicate alert rules from being fired.

You can view alert groups and also filter for alert rules that match specific criteria.

To view alert groups

1. From your Grafana console, in the Grafana menu, choose **Alerting**.
2. Choose **Groups** to view existing groups.
3. From the **Alertmanager** dropdown, select an external Alertmanager as your data source.
4. From the **Custom group by** dropdown, select a combination of labels to view a grouping other than the default. This is useful for debugging and verifying your grouping of notification policies.

If an alert does not contain labels specified either in the grouping of the root policy or the custom grouping, then the alert is added to a catch all group with a header of `No grouping`.

You can filter alerts by label or state of the alerts.

To filter by label

- In **Search**, enter an existing label to view alerts matching the label.

  For example, `environment=production,region=~US|EU,severity!=warning`.

To filter by state

- In **States**, select from Active, Suppressed, or Unprocessed states to view alerts matching your selected state. All other alerts are hidden.

View notification errors

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 10.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

View notification errors and understand why they failed to be sent or were not received.

Note

This feature is only supported for Grafana Alertmanager.

To view notification errors

1. From the left menu, choose **Alerting** then **Contact points**.

   If any contact points are failing, a message at the right-hand corner of the workspace tells you that there are errors, and how many.

2. Select a contact point to view the details of errors for that contact point.

Error details are displayed if you hover over the Error icon.

If a contact point has more than one integration, you see all errors for each of the integrations listed.

3. In the Health column, check the status of the notification.

   This can be either OK, No attempts, or Error.

Working in Grafana version 9

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

When you create your Grafana workspace, you have the option of which version of Grafana to use. The following topics describe using a Grafana workspace that uses version 9 of Grafana.

Topics

- Dashboards in Grafana version 9
- Panels and visualizations in Grafana version 9
- Explore in Grafana version 9
- Alerts in Grafana version 9

Dashboards in Grafana version 9

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A dashboard is a set of one or more panels organized and arranged into one or more rows. Grafana ships with a variety of panels making it easy to construct the right queries, and customize the visualization so that you can create the perfect dashboard for your need. Each panel can interact with data from any configured Connect to data sources.

Dashboard snapshots are static. Queries and expressions cannot be re-run from snapshots. As a result, if you update any variables in your query or expression, it will not change your dashboard data.

Topics

- Using dashboards
- Building dashboards
- Managing dashboards
- Sharing dashboards and panels
- Managing playlists
- Adding and managing dashboard variables
- Assessing dashboard usage
- Searching Dashboards in Grafana version 9

Using dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This topic provides an overview of dashboard features and shortcuts, and describes how to use dashboard search.

Features

You can use dashboards to customize the presentation of your data in the following ways.

Feature	Description
1. Home	Click the Grafana home icon to be redirected to the home page configured in the Grafana instance.
2. Title	When you click the dashboard title, you can search for dashboard contained in the current folder.
3. Sharing a dashboard	Use this option to share the current dashboard by link or snapshot. You can also export the dashboard definition from the share modal.
4. Adding a new panel	Use this option to add a panel, dashboard row, or library panel to the current dashboard.
5. Dashboard settings	Use this option to change dashboard name, folder, and tags and manage variables and annotation queries. For more information about dashboard settings, see Modifying dashboard settings.
6. Time picker dropdown	Click to select relative time range options and set custom absolute time ranges. • You can change the **Timezone** and **fiscal year** settings from the time range controls

Feature	Description
	by clicking the **Change time settings** button. • Time settings are saved on a per-dashboard basis.
7. Zooming out time range	Click to zoom out the time range. For more information about how to use time range controls, see Setting dashboard time range.
8. Refreshing dashboard	Click to immediately trigger queries and refresh dashboard data.
9. Refreshing dashboard time interval	Click to select a dashboard auto refresh time interval.
10. View mode	Click to display the dashboard on a large screen such as a TV or a kiosk. View mode hides irrelevant information such as navigatio n menus.
11. Dashboard panel	The primary building block of a dashboard is the panel. To add a new panel, dashboard row, or library panel, click **Add panel**. • Library panels can be shared among many dashboards. • To move a panel, drag the panel header to another location. • To resize a panel, click and drag the lower right corner of the panel.
12. Graph legend	Change series colors, y-axis, and series visibilit y directly from the legend.
13. Dashboard search	Click **Search** to search for dashboards by name or panel title.

Feature	Description
14. Dashboard row	A dashboard row is a logical divider within a dashboard that groups panels together. • Rows can be collapsed or expanded allowing you to hide parts of the dashboard. • Panels inside a collapsed row do not issue queries. • Use repeating rows to create rows dynamically based on a template variable.

Keyboard shortcuts

Grafana has a number of keyboard shortcuts available. To display all keyboard shortcuts available to you, press **?** or **h** on your keyboard.

- `Ctrl+S` saves the current dashboard.
- `f` opens the dashboard finder/search.
- `d+k` toggles kiosk mode (hides the menu).
- `d+e` expands all rows.
- `d+s` opens dashboard settings.
- `Ctrl+K` opens the command palette.
- `Esc` exits panel when in fullscreen view or edit mode. Also returns you to the dashboard from the dashboard settings.

Focused panel

To use shortcuts targeting a specific panel, hover over a panel with your pointer.

- `e` toggles panel edit view
- `v` toggles panel fullscreen view
- `ps` opens panel share feature
- `pd` duplicates panel
- `pr` removes panel

- `p l` toggles panel legend

Setting dashboard time range

Grafana provides several ways to manage the time ranges of the data being visualized, for dashboard, panels and also for alerting.

This section describes supported time units and relative ranges, the common time controls, dashboard-wide time settings, and panel-specific time settings.

Time units and relative ranges

Grafana supports the following time units: `s (seconds)`, `m (minutes)`, `h (hours)`, `d (days)`, `w (weeks)`, `M (months)`, `Q (quarters)`, and `y (years)`.

The minus operator enables you to step back in time, relative to now. If you want to display the full period of the unit (day, week, or month), append `/<time unit>` to the end. To view fiscal periods, use `fQ (fiscal quarter)` and `fy (fiscal year)` time units.

The plus operator enables you to step forward in time, relative to now. For example, you can use this feature to look at predicted data in the future.

The following table provides example relative ranges.

Example relative range	From	To
Last 5 minutes	`now-5m`	now
The day so far	`now/d`	now
This week	`now/w`	`now/w`
This week so far	`now/w`	now
This month	`now/M`	`now/M`
This month so far	`now/M`	now
Previous Month	`now-1M/M`	`now-1M/M`
This year so far	`now/Y`	now

Example relative range	From	To
This Year	now/Y	now/Y
Previous fiscal year	now-1y/fy	now-1y/fy

Note

Grafana Alerting does not support now+n for future timestamps and now-1n/n for *start of n until end of n* timestamps.

Common time range controls

The dashboard and panel time controls have a common user interface. The following describes common time range controls.

- Current time range, also called the *time picker*, shows the time range currently displayed in the dashboard or panel you are viewing. Hover your cursor over the field to see the exact time stamps in the range and their source (such as the local browser). Click the *current time range* to change it. You can change the current time using a *relative time range*, such as the last 15 minutes, or an absolute time range, such as `2020-05-14 00:00:00` to `2020-05-15 23:59:59`.
- The **relative time range** can be selected from the **Relative time ranges** list. You can filter the list using the input field at the top. Some examples of time ranges include *Last 30 minutes*, *Last 12 hours*, *Last 7 days*, *Last 2 years*, *Yesterday*, *Day before yesterday*, *This day last week*, *Today so far*, *This week so far*, and *This month so far*.
- **Absolute time range** can be set in two ways: Typing exact time values or relative time values into the **From** and **To** fields and clicking **Apply time range**, or clicking a date or date range from the calendar displayed when you click the **From** or **To** field. To apply your selections, click **Apply time range**.

Other time range features

1. To zoom out, click **Cmd+Z** or **Ctrl+Z**. Click the icon to view a larger time range in the dashboard or panel visualization.

2. To use the zoom in feature, click and drag to select the time range in the visualization that you want to view.

> **Note**
>
> Zooming in is only applicable to graph visualizations.

Refresh dashboards

Click the **Refresh dashboard** icon to immediately run every query on the dashboard and refresh the visualizations. Grafana cancels any pending requests when you trigger a refresh.

By default, Grafana does not automatically refresh the dashboard. Queries run on their own schedule according to the panel settings. However, if you want to regularly refresh the dashboard, then click the down arrow next to the **Refresh dashboard** icon and then select a refresh interval.

Control the time range using a URL

You can control the time range of a dashboard by providing the following query parameters in the dashboard URL.

- `from` defines the lower limit of the time range, specified in ms epoch, or relative time.
- `to` defines the upper limit of the time range, specified in ms epoch, or relative time.
- `time` and `time.window` defines a time range from `time-time.window/2` to `time+time.window/2`. Both parameters should be specified in ms. For example ? `time=150000000000&time.window=10000` results in 10s time range from 1499999995000 to 1500000005000.

Building dashboards

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

After you create a Grafana workspace and sign in, you can create dashboards and modify settings to suit your needs.

Topics

- Creating dashboards
- Add or edit panels
- Modifying dashboard settings
- Dashboard URL variables
- Adding a library panel to your dashboard
- Managing dashboard version history
- Managing dashboard links
- Dashboard JSON model

Creating dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Creating a dashboard

Dashboards and panels allow you to show your data in visual form using Grafana. Each panel needs at least one query to display a visualization. Before you get started, complete the following prerequisites.

- Ensure that you have the proper permissions. For more information about permissions, see Users, teams, and permissions.

- Identify the dashboard to which you want to add the panel.
- Understand the query language of the target data source.
- Ensure that data source for which you are writing a query has been added.

To create a dashboard:

1. Sign into Grafana, hover your cursor over **Dashboard**, and click **+ New Dashboard**.
2. Click **Add a new panel**.
3. In the first line of the **Query** tab, click the dropdown list and select a data source.
4. Write or construct a query in the query language of your data source.
5. In the **Visualization** list, select a visualization type. Grafana displays a preview of your query results with the visualization applied. For more information, see Visualizations options.
6. Adjust panel settings in the following ways.
   - Configure value mappings
   - Visualization-specific options
   - Override field values
   - Configure thresholds
   - Configure standard options

   > **Note**
   >
   > Most visualizations need some adjustment before they properly display the information you need.

7. Add a note to describe the visualization (or describe your changes) and then click **Save** in the upper-right corner of the page.

   > **Note**
   >
   > Notes are helpful if you need to revert the dashboard to a previous version.

Configuring repeating rows

You can configure Grafana to dynamically add panels or rows to a dashboard based on the value of a variable. Variables dynamically change your queries across all rows in a dashboard. For more information about repeating panels, see Configure repeating panels.

You can also repeat rows if you have variables set with `Multi-value` or `Include all values` selected.

Before you get started, ensure that the query includes a multi-value variable, then you should complete the following steps.

1. On the dashboard home page, click **Add panel**.
2. On the **Add a panel** dialog box, click **Add a new row**.
3. Hover over the row title and click the cog icon.
4. On the **Row Options** dialog box, add a title and select the variable for which you want to add repeating rows.

> **Note**
>
> To provide context to dashboard users, add the variable to the row title.

To move a panel

1. Open the dashboard.

1. Click the panel title and drag the panel to the new location. You can place a panel on a dashboard in any location.

To resize a panel

1. Open the dashboard.
2. To adjust the size of the panel, click and drag the lower-right corner of the panel. You can size a dashboard panel to suits your needs.

Add or edit panels

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

After you have created a dashboard, you can add, edit, or remove panels at any time.

- **View dashboard**: To view a dashboard, from the **Home** menu, select **Dashboards**, then choose the dashboard you want to view. You might have to expand the folder that contains the dashboard.
- **Add panel**: To add a panel to a dashboard, choose the **Add panel** icon in the menu bar near the top of the page.
- **Edit panel**To edit an existing panel on a dashboard, choose the menu icon that appears when you hover over the panel, and then choose **Edit**.
- **Remove panel**To remove an existing panel on a dashboard, choose the menu icon that appears when you hover over the panel, and then choose **Remove**.

Modifying dashboard settings

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The dashboard settings page enables you to:

- Edit general dashboard properties, including time settings.
- Add annotation queries.
- Add dashboard variables.
- Add links.
- View the dashboard JSON model

To access the dashboard setting page:

1. Open a dashboard in edit mode.
2. Click **Dashboard settings** (gear icon) located at the top of the page.

Modifying dashboard time settings

Adjust dashboard time settings when you want to change the dashboard timezone, the local browser time, and specify auto-refresh time intervals.

To modify dashboard time settings

1. On the **Dashboard** settings page, select **General**.
2. Navigate to the **Time Options** section.
3. Specify time settings according to the following descriptions.
4. Timezone specifies the local time zone of the service or system that you are monitoring. This can be helpful when monitoring a system or service that operates across several time zones.

    - Grafana uses the *default* selected time zone for the user profile, team, or organization. If no time zone is specified for the user profile, a team the user is a member of, or the organization, then Grafana uses the local browser time.
    - The time zone configured for the viewing user browser, the *local browser time*, is used. This is usually the same time zone as set on the computer.
    - Use standard ISO 8601 time zones, including UTC.

- **Auto-refresh** customizes the options displayed for relative time and the auto-refresh options Entries are comma separated and accept any valid time unit.
- **Now delay** overrides the now time by entering a time delay. Use this option to accommodate known delays in data aggregation to avoid null values.

- **Hide time picker** removes the Grafana time picker display.

> **Note**
>
> To have time controls, your data must include a time column. See the documentation for your specific data source for more information about including a time column.

Adding an annotation query

An annotation query is a query that queries for events. These events can be visualized in graphs across the dashboard as vertical lines along with a small icon you can hover over to see the event information.

To add n annotation query

1. On the **Dashboard settings** page, select **Annotations**.
2. Select **Add annotation query**.
3. Enter a name and select a data source.
4. Complete the rest of the form to build a query and annotation.

The query editor UI changes based on the data source that you select. see the Data source documentation for details on how to construct a query.

Adding a variable

Variables enable you to create more interactive and dynamic dashboards. Instead of hard-coding things like server, application, and sensor names in your metric queries, you can use variables in their place. Variables are displayed as dropdown lists at the top of the dashboard. These dropdowns make it easy to change the data being displayed in your dashboard.

For more information about variables, see Variables.

1. On the **Dashboard settings** page, click **Variable** in the left side section menu and then the **Add variable** button.
2. In the **General** section, the the name of the variable. This is the name that you will later use in queries.
3. Select a variable **Type**.

> **Note**
>
> The variable type that you select impacts which fields that you populate on the page.

4. Define the variable and click **Update**.

Adding a link

Dashboard links enable you to place links to other dashboards and websites directly below the dashboard header. Links provide for easy navigation to other, related dashboards and content.

1. On the **Dashboard settings** page, click **Links** in the left side section menu and then the **Add link** button.
2. Enter title and and in the **Type** field, select **Dashboard** or **Link**.
3. To add a dashboard link, add an optional tag, select any of the dashboard link Options, and click **Apply**.

> **Note**
>
> Tags are useful creating a dynamic dropdown of dashboards that all have a specific tag.

4. To add a link, add a URL and tooltip text that appears when the user hovers over the link, select an icon that appears next to the link, and select any of the dashboard link options.

View dashboard JSON model

A dashboard in Grafana is represented by a JSON object, which stores metadata of its dashboard. Dashboard metadata includes dashboard properties, metadata from panels, template variables, panel queries, and so on.

To view a dashboard JSON model, on the **Dashboard settings** page, click **JSON**.

For more information about the JSON fields, see JSON fields.

Dashboard URL variables

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana can apply variable values passed as query parameters in dashboard URLs. For more information, see Manage dashboard links and Templates and variables.

Passing variables as query parameters

Grafana interprets query string parameters prefixed with `var-` as variables in the given dashboard.

For example, in this URL:

```
https://${your-domain}/path/to/your/dashboard?var-example=value
```

The query parameter `var-example=value` represents the dashboard variable example with a value of `value`.

Passing multiple values for a variable

To pass multiple values, repeat the variable parameter once for each value.

```
https://${your-domain}/path/to/your/dashboard?var-example=value1&var-example=value2
```

Grafana interprets `var-example=value1&var-example=value2` as the dashboard variable example with two values: `value1` and `value2`.

Adding variables to dashboard links

Grafana can add variables to dashboard links when you generate them from a dashboard's settings. For more information and steps to add variables, see Manage dashboard links.

Passing ad hoc filters

Ad hoc filters apply key or value filters to all metric queries that use a specified data source. For more information, see Ad hoc filters.

To pass an ad hoc filter as a query parameter, use the variable syntax to pass the ad hoc filter variable, and also provide the key, the operator as the value, and the value as a pipe-separated list.

For example, in this URL:

`https://${your-domain}/path/to/your/dashboard?var-adhoc=example_key|=|example_value`

The query parameter `var-adhoc=key|=|value` applies the ad hoc filter configured as the adhoc dashboard variable using the `example_key` key, the = operator, and the `example_value` value.

> **Note**
>
> When sharing URLs with ad hoc filters, remember to encode the URL. In the above example, replace the pipes (|) with `%7C` and the equality operator (=) with `%3D`.

Controlling time range using the URL

To set a dashboard's time range, use the `from`, `to`, `time`, and `time.window` query parameters. Because these are not variables, they do not require the `var-` prefix.

Adding a library panel to your dashboard

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A library panel is a reusable panel that you can use in any dashboard. When you change a library panel, the change propagates to all instances of where the panel is used. Library panels streamline reuse of panels across multiple dashboards.

You can save a library panel in a folder alongside saved dashboards.

Creating a library panel

When you create a library panel, the panel on the source dashboard is converted to a library panel as well. You need to save the original dashboard after a panel is converted.

1. Open a panel in edit mode.
2. In the panel display options, click the down arrow option to bring changes to the visualization.
3. To open the **Create** dialog box, click the **Library panels** option, and then click **Create library panel**.
4. In **Library panel name**, enter the name.
5. In **Save in folder**, select the folder to save the library panel.
6. To save your changes, click **Create library panel**.
7. To save the dashboard, click **Save**.

After a library panel is created, you can modify the panel using any dashboard on which it appears. After you save the changes, all instances of the library panel reflect these modifications.

Adding a library panel to a dashboard

Add a Grafana library panel to a dashboard when you want to provide visualizations to other dashboard users.

1. Hover over the **Dashboards** option on the left menu, then select **New dashboard** from the dropdown options. The **Add panel** dialog box will open.
2. Click the **Add a panel** from the panel library option. You will see a list of your library panels.
3. Filter the list or search to find the panel you want to add.
4. Click a panel to add it to the dashboard.

Unlinking a library panel

Unlink a library panel when you want to make a change to the panel and not affect other instances of the library panel.

1. Hover over **Dashboard** on the left menu, and then click **Library panels**.
2. Select a library panel that is being used in different dashboards.
3. Select the panel that you want to unlink.
4. Click the title of the panel and then click **Edit**. The panel will open in edit mode.
5. Click the **Unlink** option on the top right corner of the page.

Viewing a list of library panels

Unlink a library panel when you want to make a change to the panel and not affect other instances of the library panel.

1. Hover over the **Dashboard** option on the left menu, then click **Library panels**. You can see a list of previously defined library panels.
2. Search for a specific library panel if you know its name. You can also filter the panels by folder or type.

Deleting a library panel

Delete a library panel when you no longer need it.

1. Hover over **Dashboard** on the left menu, and select **Library panels**.
2. Select the panel that you want to delete.
3. Click the delete icon next to the library name.

Managing dashboard version history

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Whenever you save a version of your dashboard, a copy of that version is saved so that previous versions of your dashboard are never lost. A list of these versions is available by entering the dashboard settings and then selecting **Versions** in the left side menu.

The dashboard version history feature lets you compare and restore to previously saved dashboard versions.

Comparing two dashboard versions

To compare two dashboard versions, select the two versions from the list that you wish to compare. Click **Compare versions** to view the diff between the two versions.

Upon clicking the button, you'll be brought to the diff view. By default, you'll see a textual summary of the changes.

If you want to view the diff of the raw JSON that represents your dashboard, you can do that as well by clicking the **View JSON Diff** button at the bottom.

If you want to restore to the version you are diffing against, you can do so by clicking the **Restore to version <x>** button in the top right.

Restoring to a previously saved dashboard version

If you need to restore to a previously saved dashboard version, you can either click the **Restore** button on the right of a row in the dashboard version list, or click the **Restore to version <x>** button appearing in the diff view. Clicking the button will bring up the following pop-up prompting you to confirm the restoration.

After restoring to a previous version, a new version will be created containing the same exact data as the previous version, only with a different version number. This is indicated in the **Notes column** for the row in the new dashboard version. This is done simply to ensure your previous dashboard versions are not affected by the change.

Managing dashboard links

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can use links to navigate between commonly used dashboards or to connect others to your visualizations. Links let you create shortcuts to other dashboards, panels, and even external websites.

Grafana supports dashboard links, panel links, and data links. Dashboard links are displayed at the top of the dashboard. Panel links are accessible by clicking an icon on the top left corner of the panel.

Choosing which link to use

Start by figuring out how you're currently navigating between dashboards. If you're often jumping between a set of dashboards and struggling to find the same context in each, links can help optimize your workflow.

The next step is to figure out which link type is right for your workflow. Even though all the link types in Grafana are used to create shortcuts to other dashboards or external websites, they work in different contexts.

- If the link relates to most if not all of the panels in the dashboard, use dashboard links.
- If you want to drill down into specific panels, use panel links.
- If you want to link to an external site, you can use either a dashboard link or a panel link.
- If you want to drill down into a specific series, or even a single measurement, use data links.

Controlling time range using the URL

To control the time range of a panel or dashboard, you can provide query parameters in the dashboard URL:

- `from` defines lower limit of the time range, specified in ms epoch.
- `to` defines upper limit of the time range, specified in ms epoch.
- `time` and `time.window` defines a time range from `time-time.window/2` to `time+time.window/2`. Both params should be specified in ms. For example ? `time=1500000000000&time.window=10000` will result in 10s time range from 1499999995000 to 1500000005000.

Dashboard links

When you create a dashboard link, you can include the time range and current template variables to directly jump to the same context in another dashboard. This way, you don't have to worry whether the person you send the link to is looking at the right data. For other types of links, see Data link variables.

Dashboard links can also be used as shortcuts to external systems, such as submitting a GitHub issue with the current dashboard name.

After adding a dashboard link, it will show up in the upper-right corner of your dashboard.

Adding links to dashboards

Add links to other dashboards at the top of your current dashboard.

1. While viewing the dashboard you want to link, click the gear at the top of the screen to open **Dashboard settings**.
2. Click **Links** and then click **Add Dashboard Link** or **New**.
3. In **Type**, select **dashboards**.
4. Select link options from the following.
   - **With tags**: Enter tags to limit the linked dashboards to only the ones with the tags you enter. Otherwise, Grafana includes links to all other dashboards.
   - **As dropdown**: If you are linking to lots of dashboards, then you probably want to select this option and add an optional title to the dropdown. Otherwise, Grafana displays the dashboard links side by side across the top of your dashboard.
   - **Time range**: Select this option to include the dashboard time range in the link. When the user clicks the link, the linked dashboard will open with the indicated time range already set.
   - **Variable values**: Select this option to include template variables currently used as query parameters in the link. When the user clicks the link, any matching templates in the linked dashboard are set to the values from the link. For more information, see Dashboard URL variables.
   - **Open in new tab**: Select this option if you want the dashboard link to open in a new tab or window.
5. Click **Add**.

Adding a URL link to a dashboard

Add a link to a URL at the top of your current dashboard. You can link to any available URL, including dashboards, panels, or external sites. You can even control the time range to ensure the user is zoomed in on the right data in Grafana.

1. While viewing the dashboard you want to link, click the gear at the top of the screen to open **Dashboard settings**.

2. Click **Links** and then click **Add Dashboard Link** or **New**.
3. In Type, select **Link**.
4. Select link options from the following.
   - **URL**: Enter the URL you want to link to. Depending on the target, you might want to include field values. For more information, see this Github example.
   - **Title**: Enter the title you want the link to display.
   - **Tooltip**: Enter the tooltip you want the link to display.
   - **Icon**: Choose the icon that you want displayed with the link.
   - **Time range**: Select this option to include the dashboard time range in the link. When the user clicks the link, the linked dashboard will open with the indicated time range set.
     - `from` defines lower limit of the time range, specified in ms epoch.
     - `to` defines upper limit of the time range, specified in ms epoch.
     - `time` and `time.window` defines a time range from `time-time.window/2` to `time+time.window/2`. Both params should be specified in ms. For example ? `time=1500000000000&time.window=10000` will result in 10s time range from 1499999995000 to 1500000005000.
   - **Variable values**: Select this option to include template variables currently used as query parameters in the link. When the user clicks the link, any matching templates in the linked dashboard are set to the values from the link.

     The variable format is as follows:

     ```
     https://${you-domain}/path/to/your/dashboard?var-${template-varable1}=value1&var-{template-variable2}=value2
     ```
   - **Open in a new tab**: Select this option if you want the dashboard link to open in a new tab or window
5. Click **Add**.

Updating a dashboard link

To change or update an existing dashboard link, follow this procedure.

1. In **Dashboard settings,** on the **Links** tab, click the existing link that you want to edit.
2. Change the settings and then click **Update**.

Duplicating a dashboard link

To duplicate an existing dashboard link, click the duplicate icon next to the existing link that you want to duplicate.

Deleting a dashboard link

To delete an existing dashboard link, click the trash icon next to the duplicate icon that you want to delete.

Panel links

Each panel can have its own set of links that are shown in the upper left corner of the panel. You can link to any available URL, including dashboards, panels, or external sites. You can even control the time range to ensure the user is zoomed in on the right data in Grafana.

To see available panel links, click the icon on the top left corner of a panel.

- **Adding a panel link**: Each panel can have its own set of links that are shown in the upper left corner of the panel. You can link to any available URL, including dashboards, panels, or external sites. You can even control the time range to ensure the user is zoomed in on the right data in Grafana. Click the icon on the top left corner of a panel to see available panel links.
  1. Hover your cursor over the panel that you want to add a link to and then press e. Or click the dropdown arrow next to the panel title and then click **Edit**.
  2. On the **Panel** tab, scroll down to the **Links** section.
  3. Expand **Links** and then click **Add link**.
  4. Enter a **Title**. **Title** is a human-readable label for the link that will be displayed in the UI.
  5. Enter the **URL** you want to link to. You can even add one of the template variables defined in the dashboard. Press `Ctrl+Space` or `Cmd+Space` and click in the URL field to see the available variables. By adding template variables to your panel link, the link sends the user to the right context, with the relevant variables already set.

     You can also use time variables.

     - `from` defines the lower limit of the time range, specified in ms epoch.
     - `to` defines the upper limit of the time range, specified in ms epoch.
     - `time` and `time.window` defines a time range from `time-time.window/2` to `time+time.window/2`. Both parameters should be specified in ms. For example

`?time=1500000000000&time.window=10000` results in 10s time range from 1499999995000 to 1500000005000.

- **Updating a panel link**
  1. On the **Panel** tab, find the link you want to make changes to.
  2. Click the **Edit** (pencil) icon to open the Edit link window.
  3. Make any necessary changes.
  4. Click **Save** to save changes and close the window.
  5. Click **Save** in the upper right to save your changes to the dashboard.
- **Deleting a panel link**
  1. On the **Panel** tab, find the link that you want to make changes to.
  2. Click the **X** icon next to the link you want to delete.
  3. Click **Save** in the upper right to save your changes to the dashboard.

Dashboard JSON model

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A dashboard in Grafana is represented by a JSON object, which stores metadata of its dashboard. Dashboard metadata includes dashboard properties, metadata from panels, template variables, and panel queries.

To view the JSON of a dashboard.

1. Navigate to a dashboard.
2. In the top navigation menu, click the **Dashboard settings** (gear) icon.
3. Click **JSON Model**.

JSON fields

When a user creates a new dashboard, a new dashboard JSON object is initialized with the following fields.

> **Note**
>
> In the following JSON, id is shown as null, which is the default value assigned to it until a dashboard is saved. After a dashboard is saved, an integer value is assigned to the `id` field.

```
{
  "id": null,
  "uid": "cLV5GDCkz",
  "title": "New dashboard",
  "tags": [],
  "style": "dark",
  "timezone": "browser",
  "editable": true,
  "graphTooltip": 1,
  "panels": [],
  "time": {
    "from": "now-6h",
    "to": "now"
  },
  "timepicker": {
    "time_options": [],
    "refresh_intervals": []
  },
  "templating": {
    "list": []
  },
  "annotations": {
    "list": []
  },
  "refresh": "5s",
  "schemaVersion": 17,
  "version": 0,
  "links": []
}
```

The following describes each field in the dashboard JSON.

Name	Usage
id	unique numeric identifier for the dashboard (generated by the db)
uid	unique dashboard identifier that can be generated by anyone. string (8-40)
title	current title of dashboard
tags	tags associated with dashboard, an array of strings
style	theme of dashboard, such as *dark* or *light*
timezone	timezone of dashboard, such as *utc* or *browser*
editable	if a dashboard is editable or not
graphTooltip	0 for no shared crosshair or tooltip (default), 1 for shared crosshair, 2 for shared crosshair and shared tooltip
time	time range for dashboard, such as *last 6 hours* or *last 7 days*
timepicker	timepicker metadata, see timepicker section for details
templating	templating metadata, see templating section for details
annotations	annotations metadata, see annotations for how to add them
refresh	auto-refresh interval
schemaVersion	version of the JSON schema (integer), incremented each time a Grafana update brings changes to said schema

Name	Usage
version	version of the dashboard (integer), incremented each time the dashboard is updated
panels	panels array (see below for detail)

Panels

Panels are the building blocks of a dashboard. It consists of data source queries, type of graphs, aliases, and more. Panel JSON consists of an array of JSON objects, each representing a different panel. Most of the fields are common for all panels but some fields depend on the panel type. The following is an example of panel JSON of a text panel.

```
"panels": [
  {
    "type": "text",
    "title": "Panel Title",
    "gridPos": {
      "x": 0,
      "y": 0,
      "w": 12,
      "h": 9
    },
    "id": 4,
    "mode": "markdown",
    "content": "# title"
  }
```

Panel size and position

The gridPos property describes the panel size and position in grid coordinates.

- w: 1–24 (the width of the dashboard is divided into 24 columns)
- h: In grid height units, each represents 30 pixels.
- x: The x position, in same unit as w.
- y: The y position, in same unit as h.

The grid has a negative gravity that moves up panels if there is empty space above a panel.

Timepicker

```
"timepicker": {
    "collapse": false,
    "enable": true,
    "notice": false,
    "now": true,
    "refresh_intervals": [
      "5s",
      "10s",
      "30s",
      "1m",
      "5m",
      "15m",
      "30m",
      "1h",
      "2h",
      "1d"
    ],
    "status": "Stable",
    "type": "timepicker"
  }
```

Templating

The `templating` field contains an array of template variables with their saved values along with some other metadata.

```
"templating": {
    "enable": true,
    "list": [
       {
        "allFormat": "wildcard",
        "current":  {
          "tags": [],
          "text": "prod",
          "value": "prod"
        },
        "datasource": null,
        "includeAll": true,
        "name": "env",
        "options": [
           {
```

```
            "selected": false,
            "text": "All",
            "value": "*"
          },
           {
            "selected": false,
            "text": "stage",
            "value": "stage"
          },
           {
            "selected": false,
            "text": "test",
            "value": "test"
          }
        ],
        "query": "tag_values(cpu.utilization.average,env)",
        "refresh": false,
        "type": "query"
      },
       {
        "allFormat": "wildcard",
        "current":  {
          "text": "apache",
          "value": "apache"
        },
        "datasource": null,
        "includeAll": false,
        "multi": false,
        "multiFormat": "glob",
        "name": "app",
        "options": [
           {
            "selected": true,
            "text": "tomcat",
            "value": "tomcat"
          },
           {
            "selected": false,
            "text": "cassandra",
            "value": "cassandra"
          }
        ],
        "query": "tag_values(cpu.utilization.average,app)",
        "refresh": false,
```

```
        "regex": "",
        "type": "query"
      }
    ]
  }
```

Managing dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A dashboard is a set of one or more panels that visually presents your data in one or more rows.

For more information about creating dashboards, see Add and organize panels.

Creating dashboard folders

Folders help you organize and group dashboards, which is useful when you have many dashboards or multiple teams using the same Grafana instance.

Prerequisites

Ensure that you have Grafana Admin permissions. For more information about dashboard permissions, see Dashboard permissions.

To create a dashboard folder

1. Sign in to Grafana and on the side menu, click **Dashboards > New folder**
2. Enter a unique name and click **Create**.

> **Note**
>
> When you save a dashboard, you can either select a folder for the dashboard to be saved in or create a new folder.

Managing dashboards and folders

On the **Manage dashboards and folders** page, you can:

- Create a folder
- Create a dashboard
- Move dashboards into folders
- Delete multiple dashboards
- Navigate to a folder page where you can assign folder and dashboard permissions

Dashboard folder page

You can complete the following tasks on the **Dashboard Folder** page:

- Move or delete dashboards in a folder.
- Rename a folder (available under the **Settings** tab).
- Assign permissions to folders (which are inherited by the dashboards in the folder).

To navigate to the dashboard folder page, click the cog appears when you hover over a folder in the dashboard search result list or the **Manage dashboards and folders** page.

Dashboard permissions

You can assign permissions to a folder. Any permissions you assign are inherited by the dashboards in the folder. An Access Control List (ACL) is used where **Organization Role**, **Team**, and a **User** can be assigned permissions.

See permissions for more information.

Exporting and importing dashboards

You can use the Grafana UI or the HTTP API to export and import dashboards.

Exporting a dashboard

The dashboard export action creates a Grafana JSON file that contains everything you need, including layout, variables, styles, data sources, queries, and so on, so that you can later import the dashboard.

> **Note**
>
> Grafana downloads a JSON file to your local machine.

1. Open the dashboard that you want to export.
2. Select the share icon.
3. Choose **Export**.
4. Choose **Save to file.**

Making a dashboard portable

If you want to export a dashboard for others to use, you can add template variables for things like a metric prefix (use a constant variable) and server name.

A template variable of the type `Constant` will automatically be hidden in the dashboard, and will also be added as a required input when the dashboard is imported.

Importing a dashboard

1. Choose **Dashboards** in the side menu.
2. Choose **New**, then select **Import** from the dropdown menu.
3. Perform one of the following steps.
   - Upload a dashboard JSON file.
   - Paste a Grafana.com dashboard URL.
   - Paste dashboard JSON text directly into the text area.

   The import process enables you to change the name of the dashboard, pick the data source you want the dashboard to use, and specify any metric prefixes (if the dashboard uses any).

Troubleshooting dashboards

This section provides information to help you solve common dashboard problems.

Dashboard is slow

If your dashboard is slow, consider the following:

- Are you trying to render dozens (or hundreds or thousands) of time-series on a graph? This can cause the browser to lag. Try using functions like highestMax (in Graphite) to reduce the returned series.

- Sometimes the series names can be very large. This causes larger response sizes. Try using alias to reduce the size of the returned series names.
- Are you querying many time-series or for a long range of time? Both of these conditions can cause Grafana or your data source to pull in a lot of data, which can slow it down.
- It could be high load on your network infrastructure. If the slowness isn't consistent, this might be the problem.

Dashboard refresh rate issues

By default, Grafana queries your data source every 30 seconds. Setting a low refresh rate on your dashboards puts unnecessary stress on the backend. In many cases, querying this frequently isn't necessary because the data isn't being sent to the system such that changes would be seen.

If you have this issue, the following solutions are recommended.

- Do not enable auto-refreshing on dashboards, panels, or variables unless you need it. Users can refresh their browser manually, or you can set the refresh rate for a time period that makes sense (such as very ten minutes or every hour).
- If it is required, then set the refresh rate to once a minute. Users can always refresh the dashboard manually.
- If your dashboard has a longer time period (such as one week), then automated refreshing might not be necessary.

Handling or rendering null data is wrong or confusing

Some applications publish data intermittently. For example, they only post a metric when an event occurs. By default, Grafana graphs connect lines between the data points.

Sharing dashboards and panels

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana enables you to share dashboards and panels with other users within an organization and in certain situations, publicly on the Web. You can share using:

- A direct link
- A snapshot
- An export link (for dashboards only)

You must have an authorized viewer permission to see an image rendered by a direct link.

The same permission is also required to view embedded links unless you have anonymous access permission enabled for your Grafana instance.

When you share a panel or dashboard as a snapshot, a snapshot (which is a panel or dashboard at the moment you take the snapshot) is publicly available on the web. Anyone with a link to it can access it. Because snapshots do not require any authorization to view, Grafana removes information related to the account it came from, as well as any sensitive data from the snapshot.

Sharing a dashboard

You can share a dashboard as a direct link or as a snapshot. You can also export a dashboard.

Note

If you change a dashboard, ensure that you save the changes before sharing.

1. Navigate to the home page of your Grafana instance.
2. Click on the share icon in the top navigation.

   The share dialog box will open and show the **Link** tab.

Sharing a direct link

The **Link** tab shows the current time range, template variables, and the default theme. You can also share a shortened URL.

1. Click **Copy**. This action copies the default or the shortened URL to the clipboard.
2. Send the copied URL to a Grafana user with authorization to view the link.

Publishing a snapshot

A dashboard snapshot shares an interactive dashboard publicly. Grafana strips sensitive data such as queries (metric, template, and annotation) and panel links, leaving only the visible metric data and series names embedded in the dashboard. Dashboard snapshots can be accessed by anyone with the link.

You can publish snapshots to your local instance.

1. Click **Local Snapshot**.
2. Grafana generates a link of the snapshot. Copy the snapshot link, and share it either within your organization or publicly on the web.

Exporting a dashboard

Grafana dashboards can easily be exported and imported. For more information, see Export and import dashboards.

Sharing a panel

You can share a panel as a direct link, or as a snapshot. You can also create library panels using the **Share** option on any panel.

1. Click a panel title to open the panel menu.
2. Click **Share**. The share dialog box will open and show the **Link** tab.

Using a direct link

The **Link** tab shows the current time range, template variables, and the default theme. You can optionally enable a shortened URL to share.

1. Click **Copy** to copy the default or the shortened URL to the clipboard.
2. Send the copied URL to a Grafana user with authorization to view the link.
3. You also optionally click **Direct link rendered image** to share an image of the panel.

Querying string parameters for server-side rendered images

- **width**: Width in pixels. The default is 800.
- **height**: Height in pixels. The default is 400.
- **tz**: Timezone in the format `UTC%2BHH%3AMM` where HH and MM are offset in hours and minutes after UTC.
- **timeout**: Number of seconds. The timeout can be increased if the query for the panel needs more than the default 30 seconds.
- **scale**: Numeric value to configure device scale factor. Default is 1. Use a higher value to produce more detailed images (higher DPI). Supported in Grafana v7.0+.

Publishing a snapshot

A panel snapshot shares an interactive panel publicly. Grafana strips sensitive data leaving only the visible metric data and series names embedded in the dashboard. Panel snapshots can be accessed by anyone with the link

You can publish snapshots to your local instance.

1. In the **Share Panel** dialog box, click **Snapshot** to open the tab.
2. Click **Local Snapshot**. Grafana generates the link of the snapshot.
3. Copy the snapshot link, and share it either within your organization or publicly on the web.

If you created a snapshot by mistake, click **Delete snapshot** to remove the snapshot from your Grafana instance.

Creating a library panel

To create a library panel from the **Share Panel** dialog box.

1. Click **Library panel**.
2. In **Library panel name**, enter the name.
3. In **Save in folder**, select the folder in which to save the library panel. By default, the **General** folder is selected.
4. Click **Create library** panel to save your changes.
5. Click **Save** to save the dashboard.

Managing playlists

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A *playlist* is a list of dashboards that are displayed in a sequence. You might use a playlist to build situational awareness or to present your metrics to your team or visitors. Grafana automatically scales dashboards to any resolution, which makes them perfect for large screens. You can access the playlist feature from Grafana's side menu in the **Dashboards** submenu.

Accessing, sharing, and controlling a playlist

Use the information in this section to access existing playlists. Start and control the display of a playlist using one of the five available modes.

Accessing a playlist

1. Hover your cursor over Grafana's side menu.
2. Click **Playlists**.

   You will see a list of existing playlists.

Starting a playlist

You can start a playlist in five different view modes. View mode determines how the menus and navigation bar appear on the dashboards.

By default, each dashboard is displayed for the amount of time entered in the **Interval** field, which you set when you create or edit a playlist. After you start a playlist, you can control it with the navigation bar at the top of the page.

The playlist displays each dashboard for the time specified in the `Interval` field, set when creating or editing a playlist. After a playlist starts, you can control it using the navigation bar at the top of your screen.

1. Access the playlist page to see a list of existing playlist.
2. Find the playlist that you want to start, then click **Start playlist**.

   The start playlist dialog box will open.
3. Select one of the five playlist modes available based on the information in the following table.
4. Click Start.

Mode	Description
Normal mode	• The side menu remains visible. • The navigation bar, row, and panel controls appear at the top of the screen.
TV mode	• The side menu and dashboard submenu (including variable dropdowns and dashboard links) are hidden or removed. • The navigation bar, row, and panel controls appear at the top of the screen. • Enabled automatically after one minute of user inactivity. • Enable it manually using the `d v` sequence shortcut, or by appending the parameter `?inactive` to the dashboard URL.

Mode	Description
	• Disable it with any pointer movement or keyboard action.
TV mode (with auto fit panels)	• The navigation bar, row, and panel controls appear at the top of the screen. • Dashboard panels automatically adjust to optimize space on screen.
Kiosk mode	• The side menu, navigation bar, ro and panel controls are completely hidden/removed from view. • You can enable it manually using the d v sequence shortcut after the playlist has started. • You can disable it manually with the same shortcut.
Kiosk mode (with auto fit panels)	• The side menu, navigation bar, row, and panel controls are completely hidden/re moved from view. • Dashboard panels automatically adjust to optimize space on screen.

Controlling a playlist

You can control a playlist in **Normal** or **TV** mode after it has started, using the navigation bar at the top of your screen. Press the `Esc` key in your keyboard to stop the playlist.

Button	Action
Next (double-right arrow)	Advances to the next dashboard.
Back (left arrow)	Returns to the previous dashboard.

Button	Action
Stop (square)	Ends the playlist, and exits to the current dashboard.
Cycle view mode (monitor icon)	Rotates the display of the dashboards in different view modes.
Time range	Displays data within a time range. It can be set to display the last 5 minutes up to 5 years ago, or a custom time range, using the down arrow.
Refresh (circle arrow)	Reloads the dashboard, to display the current data. It can be set to reload automatically every 5 seconds to 1 day, using the dropdown arrow.

Creating a playlist

You can create a playlist to present dashboards in a sequence with a set order and time interval between dashboards.

1. Click **New playlist** on the playlist page.
2. Enter a descriptive name in the **Name** text box.
3. Enter a time interval in the **Interval** text box.

> **Note**
>
> The dashboards you add are listed in a sequential order.

4. In **Dashboards**, add existing dashboards to the playlist using the **Add by title** and **Add by tag** dropdown options.
5. Optionally:
   - Search for a dashboard by its name, a regular expression, or a tag.
   - Filter your results by starred status or tags.
   - Rearrange the order of the dashboard you have added using the up and down arrow icon.
   - Remove a dashboard from the playlist by clicking the **X** icon beside dashboard.

6. Click **Save** to save your changes.

Saving a playlist

You can save a playlist and add it to your **Playlists** page, where you can start it.

> **Important**
>
> Ensure all the dashboards that you want to appear in your playlist are added when creating or editing the playlist before saving it.

1. To access the playlist feature, hover your cursor over Grafana's side menu.
2. Click **Playlists** to view the playlists available to you.
3. Click on the playlist of your choice.
4. Edit the playlist.
5. Check that the playlist has a **Name**, **Interval**, and at least one **Dashboard** added to it.
6. Click **Save** to save your changes.

Editing or deleting a playlist

You can edit a playlist by updating its name, interval time, and by adding, removing, and rearranging the order of dashboards.

Editing a playlist

1. Click **Edit playlist** on the playlist page.
2. Update the name and time interval, then add or remove dashboards from the playlist using instructions in Create a playlist, above.
3. Click **Save** to save your changes.

Deleting a playlist

1. Click **Playlists**.
2. Click **Remove** next to the playlist you want to delete.

Rearranging dashboard order

1. Next to the dashboard you want to move, click the up or down arrow.
2. Click **Save** to save your changes.

Removing a dashboard

1. Click **Remove** to remove a dashboard from the playlist.
2. Click **Save** to save your changes.

Sharing a playlist in view mode

You can share a playlist by copying the link address on the view mode you prefer, and pasting the URL to your destination.

1. From the **Dashboards** submenu, click **Playlists**.
2. Click **Start playlist** next to the playlist you want to share.
3. In the dropdown, right click the view mode you prefer.
4. Click **Copy Link Address** to copy the URL to your clipboard.
5. Paste the URL to your destination.

Adding and managing dashboard variables

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A variable is a placeholder for a value. You can use variables in metric queries and in panel titles. So when you change the value, using the dropdown at the top of the dashboard, your panel's metric queries will change to reflect the new value.

Variables allow you to create more interactive and dynamic dashboards. Instead of hard-coding things like server, application, and sensor names in your metric queries, you can use variables in their place. Variables are displayed as dropdown lists at the top of the dashboard. These dropdowns make it easy to change the data being displayed in your dashboard.

These can be especially useful for administrators who want to allow Grafana viewers to quickly adjust visualizations but do not want to give them full editing permissions. Grafana Viewers can use variables.

Variables and templates also allow you to single-source dashboards. If you have multiple identical data sources or servers, you can make one dashboard and use variables to change what you are viewing. This simplifies maintenance and upkeep enormously.

Templates

A template is any query that contains a variable. For example, if you were administering a dashboard to monitor several servers, you could make a dashboard for each server, or you could create one dashboard and use panels with template queries, such as the following.

```
wmi_system_threads{instance=~"$server"}
```

Variable values are always synced to the URL using the syntax var-<varname>=value.

Examples

Variables are listed in dropdown lists across the top of the screen. Select different variables to see how the visualizations change.

To see variable settings, navigate to **Dashboard Settings > Variables**. Click a variable in the list to see its settings.

Variables can be used in titles, descriptions, text panels, and queries. Queries with text that starts with $ are templates. Not all panels will have template queries.

Variable best practices

- Variable dropdown lists are displayed in the order they are listed in the variable list in **Dashboard settings**.
- Put the variables that you will change often at the top, so they will be shown first (far left on the dashboard).

Add and manage variables

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The following table lists the types of variables shipped with Grafana.

Variable type	Description
Query	Query-generated list of values such as metric names, server names, sensor IDs, data centers, and so on. Add a query variable.
Custom	Define the variable options manually using a comma-separated list. Add a custom variable.
Text box	Display a free text input field with an optional default value. Add a text box variable.
Constant	Define a hidden constant. Add a constant variable.
Data source	Quickly change the data source for an entire dashboard. Add a data source variable.
Interval	Interval variables represent time spans. Add an interval variable.
Ad hoc filters	Key-value filters that are automatically added to all metric queries for a data source (Prometheus, Loki, InfluxDB, and Elasticsearch only). Add ad hoc filters.

Variable type	Description
Global variables	Built-in variables that can be used in expressio ns in the query editor. Refer to Global variables.
Chained variables	Variable queries can contain other variables. Refer to Chained variables.

Entering General options

You must enter general options for any type of variable that you create.

To enter general options

1. Navigate to the dashboard you want to make a variable for and select the **Dashboard settings** (gear) icon at the top of the page.
2. On the **Variables** tab, select **New**.
3. Enter a **Name** for the variable.
4. In the **Type** list, select **Query**.
5. (Optional) In **Label**, enter the display name of the variable dropdown.

   If you don't enter a display name, then the dropdown label is the variable name.
6. Choose a **Hide** option:

   - **No selection (blank):** The variable dropdown displays the variable **Name** or **Label** value.
   - **Label:** The variable dropdown only displays the selected variable value and a down arrow.
   - **Variable:** No variable dropdown is displayed on the dashboard.

Adding a query variable

Query variables enable you to write a data source query that can return a list of metric names, tag values, or keys. For example, a query variable might return a list of server names, sensor IDs, or data centers. The variable values change as they dynamically fetch options with a data source query.

Query variables are generally only supported for strings. If your query returns numbers or any other data type, you might need to convert them to strings in order to use them as variables. For the Azure data source, for example, you can use the tostring function for this purpose.

Query expressions can contain references to other variables and in effect create linked variables. Grafana detects this and automatically refreshes a variable when one of its linked variables change.

> **Note**
>
> Query expressions are different for each data source. For more information, refer to the documentation for your data source.

To add a query variable

1. Enter general options, as above.
2. In the **Data source** list, select the target data source for the query.
3. In the **Refresh** list, select when the variable should update options.
   - **On Dashboard Load:** Queries the data source every time the dashboard loads. This slows down dashboard loading, because the variable query needs to be completed before dashboard can be initialized.
   - **On Time Range Change:** Queries the data source when the dashboard time range changes. Only use this option if your variable options query contains a time range filter or is dependent on the dashboard time range.
4. In the **Query** field, enter a query.
   - The query field varies according to your data source. Some data sources have custom query editors.
   - If you need more room in a single input field query editor, then hover your cursor over the lines in the lower right corner of the field and drag downward to expand.
5. (Optional) In the **Regex** field, type a regex expression to filter or capture specific parts of the names returned by your data source query. To see examples, refer to Filter variables with regex.
6. In the **Sort** list, select the sort order for values to be displayed in the dropdown list. The default option, **Disabled**, means that the order of options returned by your data source query will be used.

7. (Optional) Enter Selection Options.
8. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
9. Select **Add** to add the variable to the dashboard.

Adding a custom variable

Use a *custom* variable for a value that does not change, such as a number or a string.

For example, if you have server names or Region names that never change, then you might want to create them as custom variables rather than query variables. Because they do not change, you might use them in chained variables rather than other query variables. That would reduce the number of queries Grafana must send when chained variables are updated.

To add a custom variable

1. Enter general options, as above.
2. In the

   Values separated by comma list, enter the values for this variable in a comma-separated list. You can include numbers, strings, or key-value pairs separated by a space and a colon. For example, `key1 : value1,key2 : value2`.
3. (Optional) Enter Selection Options.
4. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
5. Select **Add** to add the variable to the dashboard.

Adding a text box variable

Text box variables display a free text input field with an optional default value. This is the most flexible variable, because you can enter any value. Use this type of variable if you have metrics with high cardinality or if you want to update multiple panels in a dashboard at the same time.

To add a text box variable

1. Enter general options, as above.

2. (Optional) In the **Default value** field, select the default value for the variable. If you do not enter anything in this field, then Grafana displays an empty text box for users to type text into.
3. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
4. Select **Add** to add the variable to the dashboard.

Adding a constant variable

Constant variables enable you to define a hidden constant. This is useful for metric path prefixes for dashboards you want to share. When you export a dashboard, constant variables are converted to import options.

Constant variables are *not* flexible. Each constant variable only holds one value, and it cannot be updated unless you update the variable settings.

Constant variables are useful when you have complex values that you need to include in queries but don't want to retype in every query. For example, if you had a server path called `i-0b6a61efe2ab843gg`, then you could replace it with a variable called `$path_gg`.

To add a constant variable

1. Enter general options, as above.
2. In the **Value** field, enter the variable value. You can enter letters, numbers, and symbols. You can even use wildcards if you use raw format.
3. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
4. Select **Add** to add the variable to the dashboard.

Adding a data source variable

Data source variables enable you to quickly change the data source for an entire dashboard. They are useful if you have multiple instances of a data source, perhaps in different environments.

To add a data source variable

1. Enter general options, as above.
2. In the **Type** list, select the target data source for the variable.

3. (Optional) In **Instance name filter**, enter a regex filter for which data source instances to choose from in the variable value dropdown list. Leave this field empty to display all instances.
4. (Optional) Enter Selection Options.
5. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
6. Select **Add** to add the variable to the dashboard.

Adding an interval variable

Use an *interval* variable to represents time spans such as `1m`,`1h`, or `1d`. You can think of them as a dashboard-wide *group by time* command. Interval variables change how the data is grouped in the visualization. You can also use the Auto Option to return a set number of data points per time span.

You can use an interval variable as a parameter to group by time (for InfluxDB), date histogram interval (for Elasticsearch), or as a summarize function parameter (for Graphite).

To add an interval variable

1. Enter general options, as above.
2. In the **Values** field, enter the time range intervals that you want to appear in the variable dropdown list. The following time units are supported: `s (seconds)`, `m (minutes)`, `h (hours)`, `d (days)`, `w (weeks)`, `M (months)`, and `y (years)`. You can also accept or edit the default values: `1m,10m,30m,1h,6h,12h,1d,7d,14d,30d`.
3. (Optional) Turn on the **Auto Option** if you want to add the `auto` option to the list. This option allows you to specify how many times the current time range should be divided to calculate the current `auto` time span. If you turn it on, then two more options appear:
   - **Step count -** Select the number of times the current time range will be divided to calculate the value, similar to the **Max data points** query option. For example, if the current visible time range is 30 minutes, then the `auto` interval groups the data into 30 one-minute increments. The default value is 30 steps.
   - **Min Interval -** The minimum threshold below which the step count intervals will not divide the time. To continue the 30 minute example, if the minimum interval is set to 2m, then Grafana would group the data into 15 two-minute increments.
4. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.

5. Select **Add** to add the variable to the dashboard.

Interval variable examples

The following example shows a template variable `myinterval` in a Graphite function:

```
summarize($myinterval, sum, false)
```

Adding ad hoc filters

Ad hoc filters enable you to add key-value filters that are automatically added to all metric queries that use the specified data source. Unlike other variables, you do not use ad hoc filters in queries. Instead, you use ad hoc filters to write filters for existing queries.

> **Note**
>
> Ad hoc filter variables only work with Prometheus, Loki, InfluxDB, and Elasticsearch data sources.

1. Enter general options, as above.
2. In the **Data source** list, select the target data source.
3. Select **Add** to add the variable to the dashboard.

Create ad hoc filters

Ad hoc filters are one of the most complex and flexible variable options available. Instead of a regular list of variable options, this variable allows you to build a dashboard-wide ad hoc query. Filters you apply in this manner are applied to all panels on the dashboard.

Configure variable selection options

Selection Options are a feature you can use to manage variable option selections. All selection options are optional, and they are off by default.

Multi-value variables

Interpolating a variable with multiple values selected is tricky as it is not straight forward how to format the multiple values into a string that is valid in the given context where the variable

is used. Grafana tries to solve this by allowing each data source plugin to inform the templating interpolation engine what format to use for multiple values.

> **Note**
>
> The **Custom all value** option on the variable must be blank for Grafana to format all values into a single string. If it is left blank, then Grafana concatenates (adds together) all the values in the query. For example, `value1,value2,value3`. If a custom `all` value is used, then instead the value will be `*` or `all`.

Multi-value variables with a Graphite data source

Graphite uses glob expressions. A variable with multiple values would, in this case, be interpolated as `{host1,host2,host3}` if the current variable value was *host1*, *host2*, and *host3*.

Multi-value variables with a Prometheus or InfluxDB datasource

InfluxDB and Prometheus use regex expressions, so the same variable would be interpolated as `(host1|host2|host3)`. Every value would also be regex escaped. If not, a value with a regex control character would break the regex expression.

Multi-value variables with an Elastic data source

Elasticsearch uses lucene query syntax, so the same variable would be formatted as `("host1" OR "host2" OR "host3")`. In this case, every value must be escaped so that the value only contains lucene control words and quotation marks.

Troubleshoot multi-value variables

Automatic escaping and formatting can cause problems and it can be tricky to grasp the logic behind it. Especially for InfluxDB and Prometheus where the use of regex syntax requires that the variable is used in regex operator context.

If you do not want Grafana to do this automatic regex escaping and formatting, then you must do one of the following:

- Turn off the **Multi-value** or **Include All option** options.
- Use the raw variable format.

Include All option

Grafana adds an `All` option to the variable dropdown list. If a user selects this option, then all variable options are selected.

Custom all value

This option is only visible if the **Include All option** is selected.

Enter regex, globs, or lucene syntax in the **Custom all value** field to define the value of the `All` option.

By default the `All` value includes all options in a combined expression. This can become very long and can have performance problems. Sometimes it can be better to specify a custom all value, like a wildcard regex.

To have custom regex, globs, or lucene syntax in the **Custom all value** option, it is never escaped so you will have to think about what is a valid value for your data source.

Global variables

Grafana has global built-in variables that can be used in expressions in the query editor. This topic lists them in alphabetical order and defines them. These variables are useful in queries, dashboard links, panel links, and data links.

$__dashboard

This variable is the name of the current dashboard.

$__from and $__to

Grafana has two built-in time range variables: `$__from` and `$__to`. They are currently always interpolated as epoch milliseconds by default, but you can control date formatting.

Syntax	Example result	Description
`${__from}`	1594671549254	Unix millisecond epoch
`${__from:date}`	2020-07-13T20:19:09.254Z	No args, defaults to ISO 8601/RFC 3339
`${__from:date:iso}`	2020-07-13T20:19:09.254Z	ISO 8601/RFC 3339

Syntax	Example result	Description
`${__from:date:seconds}`	1594671549	Unix seconds epoch
`${__from:date:YYYY-MM}`	2020-07	Any custom date format that does not include the : character

The syntax above also works with `${__to}`.

$__interval

You can use the `$__interval` variable as a parameter to group by time (for InfluxDB, MySQL, Postgres, MSSQL), Date histogram interval (for Elasticsearch), or as a *summarize* function parameter (for Graphite).

Grafana automatically calculates an interval that can be used to group by time in queries. When there are more data points than can be shown on a graph, the queries can be made more efficient by grouping by a larger interval. For example, if you are looking at a graph of 3 months worth of data, you might not be able to see detail at the minute level. Grouping by the hour or day makes the query more efficient without affecting what the graph shows. The `$__interval` is calculated using the time range and the width of the graph (the number of pixels).

Approximate Calculation: `(to - from) / resolution`

For example, when the time range is 1 hour and the graph is full screen, then the interval might be calculated to `2m` - points are grouped in 2 minute intervals. If the time range is 6 months and the graph is full screen, then the interval might be `1d` (1 day) - points are grouped by day.

In the InfluxDB data source, the legacy variable `$interval` is the same variable. `$__interval` should be used instead.

The InfluxDB and Elasticsearch data sources have `Group by time interval` fields that are used to hard code the interval or to set the minimum limit for the `$__interval` variable (by using the > syntax -> `>10m`).

$__interval_ms

This variable is the `$__interval` variable in milliseconds, not a time interval formatted string. For example, if the `$__interval` is `20m` then the `$__interval_ms` is `1200000`.

$__org

This variable is the ID of the current organization. `${__org.name}` is the name of the current organization.

$__user

`${__user.id}` is the ID of the current user. `${__user.login}` is the login handle of the current user. `${__user.email}` is the email for the current user.

$__range

Currently only supported for Prometheus and Loki data sources. This variable represents the range for the current dashboard. It is calculated by `to - from`. It has a millisecond and a second representation called `$__range_ms` and `$__range_s`.

$__rate_interval

Currently only supported for Prometheus data sources. The `$__rate_interval` variable is meant to be used in the rate function.

$timeFilter or $__timeFilter

The `$timeFilter` variable returns the currently selected time range as an expression. For example, the time range interval `Last 7 days` expression is `time > now() - 7d`.

This is used in several places, including:

- The WHERE clause for the InfluxDB data source. Grafana adds it automatically to InfluxDB queries when in Query Editor mode. You can add it manually in Text Editor mode: `WHERE $timeFilter`.
- Log Analytics queries in the Azure Monitor data source.
- SQL queries in MySQL, Postgres, and MSSQL.
- The `$__timeFilter` variable is used in the MySQL data source.

Chained variables

Chained variables, also called *linked variables* or *nested variables*, are query variables with one or more other variables in their variable query. This section explains how chained variables work and provides links to example dashboards that use chained variables.

Chained variable queries are different for every data source, but the premise is the same for all. You can use chained variable queries in any data source that allows them.

Extremely complex linked templated dashboards are possible, 5 or 10 levels deep. Technically, there is no limit to how deep or complex you can go, but the more links you have, the greater the query load.

Best practices and tips

The following practices will make your dashboards and variables easier to use.

Creating new linked variables

- Chaining variables create parent/child dependencies. You can envision them as a ladder or a tree.
- The easiest way to create a new chained variable is to copy the variable that you want to base the new one on. In the variable list, click the **Duplicate variable** icon to the right of the variable entry to create a copy. You can then add on to the query for the parent variable.
- New variables created this way appear at the bottom of the list. You might need to drag it to a different position in the list to get it into a logical order.

Variable order

You can change the orders of variables in the dashboard variable list by clicking the up and down arrows on the right side of each entry. Grafana lists variable dropdowns left to right according to this list, with the variable at the top on the far left.

- List variables that do not have dependencies at the top, before their child variables.
- Each variable should follow the one it is dependent on.
- Remember there is no indication in the UI of which variables have dependency relationships. List the variables in a logical order to make it easy on other users (and yourself).

Complexity consideration

The more layers of dependency you have in variables, the longer it will take to update dashboards after you change variables.

For example, if you have a series of four linked variables (country, Region, server, metric) and you change a root variable value (country), then Grafana must run queries for all the dependent variables before it updates the visualizations in the dashboard.

Manage variables

The variables page lets you add variables and manage existing variables. It also allows you to inspect variables and identify whether a variable is being referenced (or used) in other variables or dashboard.

Move: You can move a variable up or down the list using drag and drop.

Clone: To clone a variable, click the clone icon from the set of icons on the right. This creates a copy of the variable with the name of the original variable prefixed with `copy_of_`.

Delete: To delete a variable, click the trash icon from the set of icons on the right.

Filter variables with regex

Using the Regex Query option, you filter the list of options returned by the variable query or modify the options returned.

This page shows how to use regex to filter/modify values in the variable dropdown.

Using the Regex Query Option, you filter the list of options returned by the Variable query or modify the options returned. For more information, refer to the Mozilla guide on Regular expressions.

The following examples show filtering on the following list of options

```
backend_01
backend_02
backend_03
backend_04
```

Filter so that only the options that end with `01` or `02` are returned

Regex:

```
/
```

```
(
01|02
)
$/
```

Result:

```
backend_01
backend_02
```

Filter and modify the options using a regex capture group to return part of the text

Regex:

```
/.*
(
01|02
)
/
```

Result:

```
01
02
```

Filter and modify - Prometheus Example

List of options:

```
up{instance="demo.robustperception.io:9090",job="prometheus"} 1 1521630638000
up{instance="demo.robustperception.io:9093",job="alertmanager"} 1 1521630638000
up{instance="demo.robustperception.io:9100",job="node"} 1 1521630638000
```

Regex:

```
/. *instance="
(
[^"]*
)
.*/
```

Result:

```
demo.robustperception.io:9090
demo.robustperception.io:9093
demo.robustperception.io:9100
```

Filter and modify using named text and value capture groups

Using named capture groups, you can capture separate 'text' and 'value' parts from the options returned by the variable query. This allows the variable dropdown list to contain a friendly name for each value that can be selected.

For example, when querying the node_hwmon_chip_names Prometheus metric, the chip_name is a lot friendlier that the chip value. So the following variable query result:

```
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_0",chip_name="enp216s0f0np0"} 1
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_1",chip_name="enp216s0f0np1"} 1
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_2",chip_name="enp216s0f0np2"} 1
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_3",chip_name="enp216s0f0np3"} 1
```

Passed through the following Regex:

```
/chip_name="(?<text>[ ^ " ] + ) |chip=" (?<value >[ ^ " ] + )/g
```

Would produce the following dropdown list:

```
Display Name          Value
------------          -------------------------
enp216s0f0np0         0000:d7:00_0_0000:d8:00_0
enp216s0f0np1         0000:d7:00_0_0000:d8:00_1
enp216s0f0np2         0000:d7:00_0_0000:d8:00_2
enp216s0f0np3         0000:d7:00_0_0000:d8:00_3
```

Only text and value capture group names are supported.

The variables page lets you easily identify whether a variable is being referenced (or used) in other variables or dashboard.

Any variable that is referenced or used has a green check mark next to it, while unreferenced variables have an orange caution icon next to them. In addition, all referenced variables have a

dependency icon next to the green check mark. You can select the icon to view the dependency map. The dependency map can be moved. You can zoom in or out with the mouse wheel or equivalent.

Variable syntax

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Panel titles and metric queries can refer to variables using two different syntaxes.

- `$varname` – This syntax is easy to read, but it does not allow you to use a variable in the middle of a word.

  Example: `apps.frontend.$server.requests.count`
- `${var_name}` – Use this syntax when you want to use a variable in the middle of an expression.
- `${var_name:<format>}` – This format gives you more control over how Grafana interprets values. For more information, see *Advanced variable format options*.
- `[[varname]]` – Do not use. This syntax is old and has been deprecated. It will be removed in a future release.

Before queries are sent to your data source the query is *interpolated*, meaning the variable is replaced with its current value. During interpolation, the variable value might be *escaped* in order to conform to the syntax of the query language and where it is used. For example, a variable used in a regex expression in an InfluxDB or Prometheus query will be regex escaped.

Advanced variable format options

The formatting of the variable interpolation depends on the data source, but there are some situations where you might want to change the default formatting.

For example, the default for the MySQL data source is to join multiple values as comma-separated with quotes: `'server01','server02'`. In some cases, you might want to have a comma-separated string without quotes: `server01,server02`. You can make that happen with advanced variable formatting options listed below.

General syntax

Syntax: `${var_name:option}`

If any invalid formatting option is specified, then `glob` is the default/fallback option.

CSV

Formats variables with multiple values as a comma-separated string.

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:csv}'
Interpolation result:  'test1,test2'
```

Distributed - OpenTSDB

Formats variables with multiple values in custom format for OpenTSDB.

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:distributed}'
Interpolation result:  'test1,servers=test2'
```

Doublequote

Formats single- and multi-valued variables into a comma-separated string, escapes " in each value by \" and quotes each value with ".

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:doublequote}'
Interpolation result:  '"test1","test2"'
```

Glob - Graphite

Formats variables with multiple values into a glob (for Graphite queries).

```
servers = [ 'test1',  'test2' ]
```

```
String to interpolate:  '${servers:glob}'
Interpolation result:  '{test1,test2}'
```

JSON

Formats variables with multiple values as a comma-separated string.

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:json}'
Interpolation result:  '["test1", "test2"]'
```

Lucene - Elasticsearch

Formats variables with multiple values in Lucene format for Elasticsearch.

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:lucene}'
Interpolation result:  '("test1" OR "test2")'
```

Percentencode

Formats single and multivalued variables for use in URL parameters.

```
servers = [ 'foo()bar BAZ',  'test2' ]
String to interpolate:  '${servers:percentencode}'
Interpolation result:  'foo%28%29bar%20BAZ%2Ctest2'
```

Pipe

Formats variables with multiple values into a pipe-separated string.

```
servers = [ 'test1.',  'test2' ]
String to interpolate:  '${servers:pipe}'
Interpolation result:  'test1.|test2'
```

Raw

Turns off data source-specific formatting, such as single quotes in an SQL query.

```
servers = [ 'test.1',  'test2' ]
```

```
String to interpolate:  '${var_name:raw}'
Interpolation result:  'test.1,test2'
```

Regex

Formats variables with multiple values into a regex string.

```
servers = [ 'test1.',  'test2' ]
String to interpolate:  '${servers:regex}'
Interpolation result:  '(test1\.|test2)'
```

Singlequote

Formats single- and multi-valued variables into a comma-separated string, escapes `'` in each value by `\'` and quotes each value with `'`.

```
servers = [ 'test1',  'test2' ]
String to interpolate:  '${servers:singlequote}'
Interpolation result:  "'test1','test2'"
```

Sqlstring

Formats single- and multi-valued variables into a comma-separated string, escapes `'` in each value by `''` and quotes each value with `'`.

```
servers = [ "test'1",  "test2" ]
String to interpolate:  '${servers:sqlstring}'
Interpolation result:  "'test''1','test2'"
```

Text

Formats single- and multi-valued variables into their text representation. For a single variable, it will just return the text representation. For multi-valued variables, it will return the text representation combined with +.

```
servers = [ "test1",  "test2" ]
String to interpolate:  '${servers:text}'
Interpolation result:  "test1 + test2"
```

Query parameters

Formats single- and multi-valued variables into their query parameter representation. Example: `var-foo=value1&var-foo=value2`

```
servers = [ "test1",  "test2" ]
String to interpolate:  '${servers:queryparam}'
Interpolation result:  "var-servers=test1&var-servers=test2"
```

Assessing dashboard usage

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Usage insights enable you to have a better understanding of how your Grafana instance is used.

The usage insights feature collects a number of aggregated data and stores them in the database.

- Dashboard views (aggregated and per user)
- Data source errors
- Data source queries

The aggregated data provides you access to several features, including dashboard and data source insights, presence indicator, sorting dashboards by using insights data, and visualizing usage insight data in a dashboard.

This feature also generates detailed logs that can be exported to Loki.

Dashboard and data source insights

For every dashboard and data source, you can access usage information.

Dashboard insights

To see dashboard usage information, click **Dashboard insights** in the top bar.

Dashboard insights show the following information.

- **Stats**: The number of daily queries and errors for the past 30 days.
- **Users & activity**: The daily view count for the last 30 days; last activities on the dashboard and recent users (with a limit of 20).

Data source insights

Data source insights provide information about how a data source has been used in the past 30 days, such as:

- Queries per day
- Errors per day
- Query load time per day (averaged in ms)

To find data source insights:

1. Go to the **Data source** list view.
2. Click on **Data source**.
3. Click the **Insights** tab.

Presence indicator

When you sign in and look at a dashboard, you can know who is looking at the same dashboard as you are through a presence indicator, which displays avatars of users who have recently interacted with the dashboard. The default timeframe is 10 minutes. To see the user's name, hover over the user's avatar. The avatars come from Gravatar based on the user's email.

When there are more active users on a dashboard than can fit within the presence indicator, click the **+X** icon. Doing this will open dashboard insights, which contain more details about recent user activity.

Sorting dashboards by using insights data

In the search view, you can use insights data to help you find most-used, broken, and unused dashboards.

- Errors total
- Errors 30 days
- Views total
- Views 30 days

Searching Dashboards in Grafana version 9

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can search for dashboards by dashboard name and by panel title. When you search for dashboards, the system returns all dashboards available within the Grafana instance, even if you do not have permission to view the contents of the dashboard.

Search dashboards using dashboard name

Enter any part of the dashboard name in the search bar. The search returns results for any partial string match in real-time, as you type.

Dashboard search is:

- Real-time
- *Not* case sensitive
- Functional across stored and file based dashboards.

Tip

You can use your keyboard arrow keys to navigate the results and press `Enter` to open the selected dashboard.

Search dashboards using panel title

You can search for a dashboard by the title of a panel that appears in a dashboard. If a panel's title matches your search query, the dashboard appears in the search results.

Filter dashboard search results by tags

Tags are a great way to organize your dashboards, especially as the number of dashboards grow. You can add and manage tags in dashboard **Settings**.

When you select multiple tags, Grafana shows dashboards that include all selected tags.

To filter dashboard search result by a tag, complete one of the following steps:

- To filter dashboard search results by tag, choose a tag that appears in the right column of the search results.

  You can continue filtering by choosing additional tags.
- To see a list of all available tags, click the **Filter by tags** dropdown menu and select a tag.

  All tags will be shown, and when you select a tag, the dashboard search will be instantly filtered.

> **Tip**
>
> When using only a keyboard, press the `tab` key and navigate to the **Filter by tag** dropdown menu, press the down arrow key to activate the menu and locate a tag, and press `Enter` to select the tag.

Panels and visualizations in Grafana version 9

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The *panel* is the basic visualization building block in Grafana. Each panel has a query editor specific to the data source selected in the panel. The query editor allows you to build a query that returns the data you want to visualize.

There are a wide variety of styling and formatting options for each panel. Panels can be dragged, dropped, and resized to rearrange them on the dashboard.

Before you add a panel, ensure that you have configured a data source.

Additional panel types might be available by installing additional plugins to your workspace.

- For details about using specific data sources, see Connect to data sources.

Topics

- Panel editor overview
- Configure panel options
- Configure standard options
- Query and transform data
- Configure thresholds
- Configure data links
- Configure field overrides
- Configure value mappings
- Configure a legend
- Calculation types
- Annotating visualizations
- The panel inspect view
- Visualizations available in Grafana version 9

Panel editor overview

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This section describes the areas of the Grafana panel editor.

- Panel header: The header section lists the dashboard in which the panel appears and the following controls:
  - **Dashboard settings (gear) icon:** Click to access the dashboard settings.
  - **Discard:** Discards changes you have made to the panel since you last saved the dashboard.
  - **Save:** Saves changes you made to the panel.
  - **Apply:** Applies changes you made and closes the panel editor, returning you to the dashboard. You will have to save the dashboard to persist the applied changes.
- Visualization preview: The visualization preview section contains the following options:
  - **Table view:** Convert any visualization to a table so you can see the data. Table views are helpful for troubleshooting. This view only contains the raw data. It does not include transformations you might have applied to the data or the formatting options available in the Table visualization.
  - **Fill:** The visualization preview fills the available space. If you change the width of the side pane or height of the bottom pane the visualization changes to fill the available space.
  - **Actual:** The visualization preview will have the exact size as the size on the dashboard. If not enough space is available, the visualization will scale down preserving the aspect ratio.
  - **Time range controls: Default** is either the browser local timezone or the timezone selected at a higher level.
- Data section: The data section contains tabs where you enter queries, transform your data, and create alert rules (if applicable).
  - **Query tab:** Select your data source and enter queries here.
  - **Transform tab:** Apply data transformations.
  - **Alert tab:** Write alert rules.
- Panel display options: The display options section contains tabs where you configure almost every aspect of your data visualization.

Open the panel inspect drawer

The inspect drawer helps you understand and troubleshoot your panels. You can view the raw data for any panel, export that data to a comma-separated values (CSV) file, view query requests, and export panel and data JSON.

> **Note:** Not all panel types include all tabs. For example, dashboard list panels do not have raw data to inspect, so they do not display the Stats, Data, or Query tabs.

The panel inspector consists of the following options:

- The panel inspect drawer displays opens a drawer on the right side. Click the arrow in the upper right corner to expand or reduce the drawer pane.
- **Data tab -** Shows the raw data returned by the query with transformations applied. Field options such as overrides and value mappings are not applied by default.
- **Stats tab -** Shows how long your query takes and how much it returns.
- **JSON tab -** Allows you to view and copy the panel JSON, panel data JSON, and data frame structure JSON. This is useful if you are provisioning or administering Grafana.
- **Query tab -** Shows you the requests to the server sent when Grafana queries the data source.
- **Error tab -** Shows the error. Only visible when query returns error.

Configure panel options

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A Grafana panel is the user interface you use to define a data source query, and transform and format data that appears in visualizations.

A panel editor includes a query builder and a series of options that you can use to transform data and add information to your panels.

This topic describes how to:

- Open a panel for editing
- Add a panel title and description
- View a panel JSON model
- Add repeating rows and panels

Edit a panel

After you add a panel to a dashboard, you can open it at any time to change change or update queries, add data transformation, and change visualization settings.

1. Open the dashboard that contains the panel you want to edit.
2. Hover over any part of the panel to display the actions menu on the top right corner.
3. Click the menu and select **Edit**.

   To use a keyboard shortcut to open the panel, hover over the panel and press **e**.

   The panel opens in edit mode.

Add a title and description to a panel

Add a title and description to a panel to share with users any important information about the visualization. For example, use the description to document the purpose of the visualization.

1. Edit a panel.
2. In the panel display options pane, locate the **Panel options** section.
3. Enter a **Title**.

   Text entered in this field appears at the top of your panel in the panel editor and in the dashboard.
4. Write a description of the panel and the data you are displaying.

   Text entered in this field appears in a tooltip in the upper-left corner of the panel.

   You can use variables you have defined in the **Title** and **Description** field, but not global variables.

View a panel JSON model

Explore and export panel, panel data, and data frame JSON models.

1. Open the dashboard that contains the panel.
2. Hover over any part of the panel to display the actions menu on the top right corner.
3. Click the menu and select **Inspect > Panel JSON**.
4. In the **Select source** field, select one of the following options:
   - **Panel JSON:** Displays a JSON object representing the panel.
   - **Panel data:** Displays a JSON object representing the data that was passed to the panel.
   - **DataFrame structure:** Displays the raw result set with transformations, field configurations, and override configurations applied.
5. To explore the JSON, click **>** to expand or collapse portions of the JSON model.

Configure repeating panels

You can configure Grafana to dynamically add panels or rows to a dashboard. A dynamic panel is a panel that the system creates based on the value of a variable. Variables dynamically change your queries across all panels in a dashboard.

> **Note**
>
> Repeating panels require variables to have one or more items selected; you cannot repeat a panel zero times to hide it.

Before you begin:

- Ensure that the query includes a multi-value variable.

To configure repeating panels:

1. Edit the panel you want to repeat.
2. On the display options pane, click **Panel options > Repeat options**.
3. Select a **direction**.

- Choose **horizontal** to arrange panels side-by-side. Grafana adjusts the width of a repeated panel. Currently, you cannot mix other panels on a row with a repeated panel.
- Choose **vertical** to arrange panels in a column. The width of repeated panels is the same as the original, repeated panel.

4. To propagate changes to all panels, reload the dashboard.

Configure standard options

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The data model used in Grafana is a columnar-oriented table structure that unifies both time series and table query results. Each column within this structure is called a *field*. A field can represent a single time series or table column.

Field options allow you to change how the data is displayed in your visualizations. Options and overrides that you apply do not change the data, they change how Grafana displays the data. When you change an option, it is applied to all fields, meaning all series or columns. For example, if you change the unit to percentage, then all fields with numeric values are displayed in percentages.

For a complete list of field formatting options, refer to Standard options definitions.

Note

You can apply standard options to most built-in Grafana panels. Some older panels and community panels that have not updated to the new panel and data model will be missing either all or some of these field options.

1. Open a dashboard, click the panel title, and click **Edit**.

2. In the panel display options pane, locate the **Standard options** section.
3. Select the standard options you want to apply.
4. To preview your change, click outside of the field option box you are editing or press **Enter**.

Standard options definitions

This section explains all available standard options.

You can apply standard options to most built-in Grafana panels. Some older panels and community panels that have not updated to the new panel and data model will be missing either all or some of these field options.

Most field options will not affect the visualization until you click outside of the field option box you are editing or press Enter.

> **Note**
>
> We are constantly working to add and expand options for all visualization, so all options might not be available for all visualizations.

Unit

Lets you choose what unit a field should use. Click in the **Unit** field, then drill down until you find the unit you want. The unit you select is applied to all fields except time.

Custom units

You can use the unit dropdown to also specify custom units, custom prefix or suffix and date time formats.

To select a custom unit enter the unit and select the last **Custom: xxx** option in the dropdown.

- **suffix:<suffix>** for custom unit that should go after value.
- **prefix:<prefix>** for custom unit that should go before value.
- **time:<format>** For custom date time formats type for example **time:YYYY-MM-DD**. See formats for the format syntax and options.

- **si:<base scale><unit characters>** for custom SI units. For example: **si: mF**. This one is a bit more advanced as you can specify both a unit and the source data scale. So if your source data is represented as milli (thousands of) something prefix the unit with that SI scale character.
- **count:<unit>** for a custom count unit.
- **currency:<unit>** for custom a currency unit.

You can also paste a native emoji in the unit picker and pick it as a custom unit.

String units

Grafana can sometimes be too aggressive in parsing strings and displaying them as numbers. To configure Grafana to show the original string value, create a field override and add a unit property with the **String** unit.

Min

Lets you set the minimum value used in percentage threshold calculations. Leave blank for auto calculation based on all series and fields.

Max

Lets you set the maximum value used in percentage threshold calculations. Leave blank for auto calculation based on all series and fields.

Decimals

Specify the number of decimals Grafana includes in the rendered value. If you leave this field blank, Grafana automatically truncates the number of decimals based on the value. For example 1.1234 will display as 1.12 and 100.456 will display as 100.

To display all decimals, set the unit to **String**.

Display name

Lets you set the display title of all fields. You can use variables.

When multiple stats, fields, or series are shown, this field controls the title in each stat. You can use expressions like **${__field.name}** to use only the series name or the field name in title.

Given a field with a name of Temp, and labels of {"Loc"="PBI", "Sensor"="3"}

Express n syntax	Exam	Renders to	Explanation
${__fiel .display ame}	Same as syntax	**Temp {Loc="PB ", Sensor=' "}**	Displays the field name, and labels in **{}** if they are present. If there is only one label key in the response, then for the label portion, Grafana displays the value of the label without the enclosing braces.
${__fiel .name}	Same as syntax	**Temp**	Displays the name of the field (without labels).
${__fiel .labels}	Same as syntax	**Loc="PB , Sensor=' "**	Displays the labels without the name.
${__fiel .labels. **}**	**${__fi .label oc}**	**PBI**	Displays the value of the specified label key.
${__fiel .labels. _values	Same as Synta	**PBI, 3**	Displays the values of the labels separated by a comma (without label keys).

If the value is an empty string after rendering the expression for a particular field, then the default display method is used.

Color scheme

The color options and their effect on the visualization depends on the visualization you are working with. Some visualizations have different color options.

You can specify a single color, or select a continuous (gradient) color schemes, based on a value. Continuous color interpolates a color using the percentage of a value relative to min and max.

Select one of the following palettes:

Color mode	Description
Single color	Specify a single color, useful in an override rule
From thresholds	Informs Grafana to take the color from the matching threshold
Classic palette	Grafana will assign color by looking up a color in a palette by series index. Useful for Graphs and pie charts and other categorical data visualizations
Green-Yellow-Red (by value)	Continuous color scheme
Blue-Yellow-Red (by value)	Continuous color scheme
Blues (by value)	Continuous color scheme (panel background to blue)
Reds (by value)	Continuous color scheme (panel background color to blue)
Greens (by value)	Continuous color scheme (panel background color to blue)
Purple (by value)	Continuous color scheme (panel background color to blue)

No value

Enter what Grafana should display if the field value is empty or null. The default value is a hyphen (-).

Query and transform data

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana supports many types of data sources. Data source *queries* return data that Grafana can *transform* and visualize. Each data source uses its own query language, and data source plugins each implement a query-building user interface called a query editor.

About queries

Grafana panels communicate with data sources via queries, which retrieve data for the visualization. A query is a question written in the query language used by the data source.

You can configure query frequency and data collection limits in the panel's data source options. Grafana supports up to 26 queries per panel.

You can find more information about each data source's query language in the Data sources section.

Query editors

Each data source's *query editor* provides a customized user interface that helps you write queries that take advantage of its unique capabilities.

Because of the differences between query languages, each data source query editor looks and functions differently. Depending on your data source, the query editor might provide auto-completion features, metric names, variable suggestions, or a visual query-building interface.

For details on a specific data source's unique query editor features, refer to its documentation:

- For data sources included with Grafana, see Built-in data sources.

- For data sources included with Grafana Enterprise editiion, see Connect to Enterprise data sources.

Query syntax

Data sources use different query languages to request data. For details on a specific data source's unique query language, refer to its documentation.

PostgreSQL example:

```
SELECT hostname FROM host WHERE region IN($region)
```

PromQL example:

```
query_result(max_over_time(<metric>[${__range_s}s]) != <state>)
```

Special data sources

Grafana also includes three special data sources: **Grafana**, **Mixed**, and **Dashboard**. For details, refer to Data sources

Navigate the query tab

A panel's **Query** tab consists of the following elements:

- **Data source selector** – Selects the data source to query.
- **Query options:** – Sets maximum data retrieval parameters and query run time intervals.
- **Query inspector button:** – Opens the query inspector panel, where you can view and optimize your query.
- **Query editor list:** – Lists the queries you've written.
- **Expressions:** – Uses the expression builder to create alert expressions. For more information about expressions, see Write expression queries.

Add a query

A query returns data that Grafana visualizes in dashboard panels. When you create a panel, Grafana automatically selects the default data source.

To add a query

1. Edit the panel to which you're adding a query.
2. Choose the **Query** tab.
3. Choose the **Data source** dropdown menu and select a data source.
4. Choose **Query options** to configure the maximum number of data points you need. For more information about query options, see Query options.
5. Write the query using the query editor.
6. Choose **Apply**.

Grafana queries the data source and visualizes the data.

Manage queries

Grafana organizes queries in collapsible query rows. Each query row contains a query editor and is identified with a letter (A, B, C, and so on).

To manage your queries, you can copy queries, hide queries, remove a query, reorder queries, and toggle help for the query editor.

Query options

Choose **Query options** next to the data source selector to see settings for the selected data source. Changes you make here affect only queries made in this panel.

Grafana sets defaults that are shown in dark gray text. Changes are displayed in white text. To return a field to the default setting, delete the white text from the field.

Panel data source query options include:

- **Max data points** – If the data source supports it, this sets the maximum number of data points for each series returned. If the query returns more data points than the max data points setting, then the data source reduces the number of points returned by aggregating them together by average, max, or another function.

  You can limit the number of points to improve query performance or smooth the visualized line. The default value is the width (or number of pixels) of the graph, because you can only visualize as many data points as the graph panel has room to display.

- **Min interval** – Sets a minimum limit for the automatically calculated interval, which is typically the minimum scrape interval. If a data point is saved every 15 seconds, you don't benefit from having an interval lower than that. You can also set this to a higher minimum than the scrape interval to retrieve queries that are more coarse-grained and well-functioning.
- **Interval** – Sets a time span that you can use when aggregating or grouping data points by time.

  Grafana automatically calculates an appropriate interval that you can use as a variable in templated queries. The variable is measured in either seconds (`$__interval`) or milliseconds (`$__interval_ms`).

  Intervals are typically used in aggregation functions like sum or average.
  For example, this is a Prometheus query that uses the interval variable:
  `rate(http_requests_total[$__interval])`.

  This automatic interval is calculated based on the width of the graph. As the user zooms out on a visualization, the interval grows, resulting in a more coarse-grained aggregation. Likewise, if the user zooms in, the interval decreases, resulting in a more fine-grained aggregation.

  For more information, see Global variables.
- **Relative time** – Overrides the relative time range for individual panels, which causes them to be different than what is selected in the dashboard time picker in the top-right corner of the dashboard. You can use this to show metrics from different time periods or days on the same dashboard.

  > **Note**
  >
  > Panel time overrides have no effect when the dashboard's time range is absolute.

Example	Relative time field
Last 5 minutes	`now-5m`
The day so far	`now/d`
Last 5 days	`now-5d/d`
This week so far	`now/w`

Example	Relative time field
Last 2 years	`now-2y/y`

- **Time shift** – Overrides the time range for individual panels by shifting its start and end relative to the time picker. For example, you can shift the time range for the panel to be two hours earlier than the dashboard time picker.

> **Note**
>
> Panel time overrides have no effect when the dashboard's time range is absolute.

Example	Time shift field
Last entire week	`1w/w`
Two entire weeks ago	`2w/w`
Last entire month	`1M/M`
This entire year	`1d/y`
Last entire year	`1y/y`

- **Cache timeout** – *(Visible only if available in the data source)* Overrides the default cache timeout if your time series store has a query cache. Specify this value as a numeric value in seconds.

Write expression queries

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Server-side expressions enable you to manipulate data returned from queries with math and other operations. Expressions create new data and do not manipulate the data returned by data sources.

About expressions

Server-side expressions allow you to manipulate data returned from queries with math and other operations. Expressions create new data and do not manipulate the data returned by data sources, aside from some minor data restructuring to make the data acceptable input for expressions.

Using expressions

Expressions are primarily used by Grafana Alerting. The processing is done server-side, so expressions can operate without a browser session. However, expressions can also be used with backend data sources and visualization.

> **Note**
>
> Expressions do not work with legacy dashboard alerts.

Expressions are meant to augment data sources by enabling queries from different data sources to be combined or by providing operations unavailable in a data source.

> **Note**
>
> When possible, you should do data processing inside the data source. Copying data from storage to the Grafana server for processing is inefficient, so expressions are targeted at lightweight data processing.

Expressions work with data source queries that return time series or number data. They also operate on multiple-dimensional data. For example, a query that returns multiple series, where each series is identified by labels or tags.

An individual expression takes one or more queries or other expressions as input and adds data to the result. Each individual expression or query is represented by a variable that is a named identifier known as its RefID (e.g., the default letter A or B).

To reference the output of an individual expression or a data source query in another expression, this identifier is used as a variable.

Types of expressions

Expressions work with two types of data.

- A collection of time series.
- A collection of numbers, where each number is an item.

Each collection is returned from a single data source query or expression and represented by the RefID. Each collection is a set, where each item in the set is uniquely identified by its dimensions which are stored as labels or key-value pairs.

Data source queries

Server-side expressions only support data source queries for backend data sources. The data is generally assumed to be labeled time series data. In the future we intended to add an assertion of the query return type (number or time series) data so expressions can handle errors better.

Data source queries, when used with expressions, are run by the expression engine. When it does this, it restructures data to be either one time series or one number per data frame. So for example if using a data source that returns multiple series on one frame in the table view, you might notice it looks different when run with expressions.

Currently, the only non-time series format (number) is supported when using data frames are you have a table response that returns a data frame with no time, string columns, and one number column:

Loc	Host	Avg_CPU
MIA	A	1
NYC	B	2

The example above will produce a number that works with expressions. The string columns become labels and the number column the corresponding value. For example `{"Loc": "MIA", "Host": "A"}` with a value of 1.

Operations

You can use the following operations in expressions: math, reduce, and resample.

Math

Math is for free-form math formulas on time series or number data. Math operations take numbers and time series as input and changes them to different numbers and time series.

Data from other queries or expressions are referenced with the RefID prefixed with a dollar sign, for example `$A`. If the variable has spaces in the name, then you can use a brace syntax like `${my variable}`.

Numeric constants can be in decimal (`2.24`), octal (with a leading zero like `072`), or hex (with a leading 0x like `0x2A`). Exponentials and signs are also supported (e.g., `-0.8e-2`).

Operators

The arithmetic (+, binary and unary `-`, `*`, `/`, `%`, exponent `**`), relational (<, >, ==, `!=`, >=, <=), and logical (&&, `||`, and unary `!`) operators are supported.

How the operation behaves with data depends on if it is a number or time series data.

With binary operations, such as `$A + $B` or `$A || $B`, the operator is applied in the following ways depending on the type of data:

- If both `$A` and `$B` are a number, then the operation is performed between the two numbers.
- If one variable is a number, and the other variable is a time series, then the operation between the value of each point in the time series and the number is performed.
- If both `$A` and `$B` are time series data, then the operation between each value in the two series is performed for each time stamp that exists in both `$A` and `$B`. The `Resample` operation can be used to line up time stamps.

Summary:

- Number OP number = number
- Number OP series = series
- Series OP series = series

Because expressions work with multiple series or numbers represented by a single variable, binary operations also perform a union (join) between the two variables. This is done based on the identifying labels associated with each individual series or number.

So if you have numbers with labels like `{host=web01}` in `$A` and another number in `$B` with the same labels then the operation is performed between those two items within each variable, and the result will share the same labels. The rules for the behavior of this union are as follows:

- An item with no labels will join to anything.
- If both `$A` and `$B` each contain only one item (one series, or one number), they will join.
- If labels are exact math they will join.
- If labels are a subset of the other, for example an item in `$A` is labeled `{host=A,dc=MIA}` and an item in `$B` is labeled `{host=A}` they will join.
- If within a variable such as `$A` there are different tag keys for each item, the join behavior is undefined.

The relational and logical operators return 0 for false 1 for true.

Math Functions

While most functions exist in the own expression operations, the math operation does have some functions that similar to math operators or symbols. When functions can take either numbers or series, than the same type as the argument will be returned. When it is a series, the operation of performed for the value of each point in the series.

abs

abs returns the absolute value of its argument which can be a number or a series. For example `abs(-1)` or `abs($A)`.

is_inf

is_inf takes a number or a series and returns `1` for `Inf` values (negative or positive) and `0` for other values. For example `is_inf($A)`.

> **Note**
>
> If you need to specifically check for negative infinity for example, you can do a comparison like `$A == infn()`.

is_nan

is_nan takes a number or a series and returns 1 for NaN values and 0 for other values. For example `is_nan($A)`. This function exists because NaN is not equal to NaN.

is_null

is_null takes a number or a series and returns 1 for null values and 0 for other values. For example `is_null($A)`.

is_number

is_number takes a number or a series and returns 1 for all real number values and 0 for other values (which are null, Inf+, Inf-, and NaN). For example `is_number($A)`.

log

Log returns the natural logarithm of its argument which can be a number or a series. If the value is less than 0, NaN is returned. For example `log(-1)` or `log($A)`.

inf, infn, nan, and null

The inf, infn, nan, and null functions all return a single value of the name. They primarily exist for testing. Example: `null()`.

round

Round returns a rounded integer value. For example, `round(3.123)` or `round($A)`.

ceil

Ceil rounds the number up to the nearest integer value. For example, `ceil(3.123)` returns 4.

floor

Floor rounds the number down to the nearest integer value. For example, `floor(3.123)` returns 3.

Reduce

Reduce takes one or more time series returned from a query or an expression and turns each series into a single number. The labels of the time series are kept as labels on each outputted reduced number.

Fields:

- **Function** – The reduction function to use

- **Input** – The variable (refID (such as A)) to resample
- **Mode** – Allows control behavior of reduction function when a series contains non-numerical values (null, NaN, +-Inf)

Reduction Functions

Count

Count returns the number of points in each series.

Mean

Mean returns the total of all values in each series divided by the number of points in that series. In `strict` mode if any values in the series are null or nan, or if the series is empty, NaN is returned.

Min and Max

Min and Max return the smallest or largest value in the series respectively. In `strict` mode if any values in the series are null or nan, or if the series is empty, NaN is returned.

Sum

Sum returns the total of all values in the series. If series is of zero length, the sum will be 0. In `strict` mode if there are any NaN or Null values in the series, NaN is returned.

Last

Last returns the last number in the series. If the series has no values then returns NaN.

Reduction Modes

Strict

In Strict mode the input series is processed as is. If any values in the series are non-numeric (null, NaN or +-Inf), NaN is returned.

Drop Non-Numeric

In this mode all non-numeric values (null, NaN or +-Inf) in the input series are filtered out before running the reduction function.

Replace Non-Numeric

In this mode all non-numeric values are replaced by a pre-defined value.

Resample

Resample changes the time stamps in each time series to have a consistent time interval. The main use case is so you can resample time series that do not share the same timestamps so math can be performed between them. This can be done by resample each of the two series, and then in a Math operation referencing the resampled variables.

Fields:

- **Input** – The variable of time series data (refID (such as A)) to resample
- **Resample** to – The duration of time to resample to, for example `10s`. Units can be `s` seconds, `m` for minutes, `h` for hours, `d` for days, `w` for weeks, and `y` for years.
- **Downsample** – The reduction function to use when there are more than one data point per window sample. See the reduction operation for behavior details.
- **Upsample** – The method to use to fill a window sample that has no data points.
  - **pad** fills with the last know value
  - **backfill** with next known value
  - **fillna** to fill empty sample windows with NaNs

Write an expression

If your data source supports them, then Grafana displays the **Expression** button and shows any existing expressions in the query editor list.

To write an expression

1. Open the panel.
2. Below the query, choose **Expression**.
3. In the **Operation** field, select the type of expression you want to write.
4. Write the expression.
5. Choose **Apply**.

Special cases

When any queried data source returns no series or numbers, the expression engine returns `NoData`. For example, if a request contains two data source queries that are merged by an expression, if

`NoData` is returned by at least one of the data source queries, then the returned result for the entire query is `NoData`. For more information about how Grafana Alerting processes `NoData` results, see Handling no data or error cases.

Share query results with another panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana let you use the query result from one panel for any other panel in the dashboard. Sharing query results across panels reduces the number of queries made to your data source, which can improve the performance of your dashboard.

The Dashboard data source lets you select a panel in your dashboard that contains the queries you want to share the results for. Instead of sending a separate query for each panel, Grafana sends one query and other panels use the query results to construct visualizations.

This strategy can drastically reduce the number of queries being made when you for example have several panels visualizing the same data.

To share query results

1. Create a dashboard.
2. Change the title to `Source panel`. You'll use this panel as a source for the other panels.
3. Define the query or queries that you want to share.

   If you don't have a data source available, use the **TestData** data source, which returns a random time series that you can use for testing.
4. Add a second panel and select the **Dashboard** data source in the query editor.
5. In the **Use results from panel list**, select the first panel you created.

All queries defined in the source panel are now available to the new panel. Queries made in the source panel can be shared with multiple panels.

You can click on any of the queries to go to the panel where they are defined.

Transform data

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Transformations are a powerful way to manipulate data returned by a query before the system applies a visualization. Using transformations, you can:

- Rename fields
- Join time series data
- Perform mathematical operations across queries
- Use the output of one transformation as the input to another transformation

For users that rely on multiple views of the same dataset, transformations offer an efficient method of creating and maintaining numerous dashboards.

You can also use the output of one transformation as the input to another transformation, which results in a performance gain.

Note

Sometimes the system cannot graph transformed data. When that happens, click the **Table view** toggle above the visualization to switch to a table view of the data. This can help you understand the final result of your transformations.

Transformation types

Grafana provides a number of ways that you can transform data. There is a complete list of transformation functions below.

Order of transformations

When there are multiple transformations, Grafana applies them in the order they are listed. Each transformation creates a result set that then passes on to the next transformation in the processing pipeline.

The order in which Grafana applies transformations directly impacts the results. For example, if you use a Reduce transformation to condense all the results of one column into a single value, then you can only apply transformations to that single value.

Add a transformation function to data

The following steps guide you in adding a transformation to data.

To add a transformation to a panel

1. Navigate to the panel where you want to add one or more transformations.
2. Choose the panel title and then click **Edit**.
3. Choose the **Transform** tab.
4. Choose a transformation. A transformation row appears where you configure the transformation options.
5. To apply another transformation, Choose **Add transformation**. This transformation acts on the result set returned by the previous transformation.

Debug a transformation

To see the input and the output result sets of the transformation, choose the bug icon on the right side of the transformation row.

The input and output results sets can help you debug a transformation.

Delete a transformation

We recommend that you remove transformations that you don't need. When you delete a transformation, you remove the data from the visualization.

Prerequisites:

Identify all dashboards that rely on the transformation and inform impacted dashboard users.

To delete a transformation

1. Open a panel for editing.
2. Choose the **Transform** tab.
3. Choose the trash icon next to the transformation you want to delete.

Transformation functions

You can perform the following transformations on your data.

Add field from calculation

Use this transformation to add a new field calculated from two other fields. Each transformation allows you to add one new field.

- **Mode** - Select a mode:
  - **Reduce row** – Apply selected calculation on each row of selected fields independently.
  - **Binary option** – Apply basic math operation(sum, multiply, etc) on values in a single row from two selected fields.
- **Field name** – Select the names of fields you want to use in the calculation for the new field.
- **Calculation** – If you select **Reduce row** mode, then the **Calculation** field appears. Click in the field to see a list of calculation choices you can use to create the new field. For information about available calculations, refer to Calculation types.
- **Operation** – If you select **Binary option** mode, then the **Operation** fields appear. These fields allow you to do basic math operations on values in a single row from two selected fields. You can also use numerical values for binary operations.
- **Alias** – (Optional) Enter the name of your new field. If you leave this blank, then the field will be named to match the calculation.
- **Replace all fields** – (Optional) Select this option if you want to hide all other fields and display only your calculated field in the visualization.

Concatenate fields

This transformation combines all fields from all frames into one result. Consider these two queries.

Query A:

Temp	Uptime
15.4	1230233

Query B:

AQI	Errors
3.2	5

After you concatenate the fields, the data frame would be:

Temp	Uptime	AQI	Errors
15.4	1230233	3.2	5

Config from query results

This transformation allow you to select one query and from it extract standard options like **Min**, **Max**, **Unit** and **Thresholds** and apply it to other query results. This enables dynamic query driven visualization configuration.

If you want to extract a unique config for every row in the config query result then try the rows to fields transformation.

Options

- **Config query** – Select the query that returns the data you want to use as configuration.
- **Apply to** – Select what fields or series to apply the configuration to.
- **Apply to options** – Usually a field type or field name regex depending on what option you selected in **Apply to**.

Convert field type

This transformation changes the field type of the specified field.

- **Field** – Select from available fields
- **as** – Select the FieldType to convert to
  - **Numeric** – attempts to make the values numbers
  - **String** – will make the values strings
  - **Time** – attempts to parse the values as time
    - Will show an option to specify a DateFormat as input by a string like yyyy-mm-dd or DD MM YYYY hh:mm:ss
  - **Boolean** – will make the values Boolean

For example the following query could be modified by selecting the time field, as Time, and Date Format as YYYY.

Time	Mark	Value
7/1/2017	above	25
8/2/2018	below	22
9/2/2019	below	29
10/4/2020	above	22

The result:

Time	Mark	Value
1/1/2017	above	25
1/1/2018	below	22
1/1/2019	below	29
1/1/2020	above	22

Filter data by name

Use this transformation to remove portions of the query results.

Grafana displays the **Identifier** field, followed by the fields returned by your query.

You can apply filters in one of two ways:

- Enter a regex expression.
- Choose a field to toggle filtering on that field. Filtered fields are displayed with dark gray text, unfiltered fields have white text.

Filter data by query

Use this transformation in panels that have multiple queries, if you want to hide one or more of the queries.

Grafana displays the query identification letters in dark gray text. Click a query identifier to toggle filtering. If the query letter is white, then the results are displayed. If the query letter is dark, then the results are hidden.

> **Note**
>
> This transformation is not available for Graphite because this data source does not support correlating returned data with queries.

Filter data by value

This transformation allows you to filter your data directly in Grafana and remove some data points from your query result. You have the option to include or exclude data that match one or more conditions you define. The conditions are applied on a selected field.

This transformation is useful if your data source does not natively filter by values. You might also use this to narrow values to display if you are using a shared query.

The available conditions for all fields are:

- **Regex** – Match a regex expression

- **Is Null** – Match if the value is null
- **Is Not Null** – Match if the value is not null
- **Equal** – Match if the value is equal to the specified value
- **Different** – match if the value is different than the specified value

The available conditions for number fields are:

- **Greater** – Match if the value is greater than the specified value
- **Lower** – Match if the value is lower than the specified value
- **Greater or equal** – Match if the value is greater or equal
- **Lower or equal** – Match if the value is lower or equal
- **Range** – Match a range between a specified minimum and maximum, min and max included

Consider the following data set:

Time	Temperature	Altitude
7/7/2020 11:34:23 AM	32	101
7/7/2020 11:34:22 AM	28	125
7/7/2020 11:34:21 AM	26	110
7/7/2020 11:34:20 AM	23	98
7/7/2020 10:32:24 AM	31	95
7/7/2020 10:31:22 AM	20	85
7/7/2020 9:30:57 AM	19	101

If you **Include** the data points that have a temperature below 30°C, the configuration will look as follows:

- **Filter Type** – `Include`
- **Condition** – Rows where `Temperature` matches `Lower Than 30`

And you will get the following result, where only the temperatures below 30°C are included:

Time	Temperature	Altitude
7/7/2020 11:34:22 AM	28	125
7/7/2020 11:34:21 AM	26	110
7/7/2020 11:34:20 AM	23	98
7/7/2020 10:31:22 AM	20	85
7/7/2020 9:30:57 AM	19	101

You can add more than one condition to the filter. For example, you might want to include the data only if the altitude is greater than 100. To do so, add that condition to the following configuration:

- Filter type – `Include` rows that `Match All` conditions
- Condition 1 – Rows where `Temperature` matches `Lower` than `30`
- Condition 2 – Rows where `Altitude` matches `Greater` than `100`

When you have more than one condition, you can choose if you want the action (include / exclude) to be applied on rows that **Match all** conditions or **Match any** of the conditions you added.

In the example above we chose **Match all** because we wanted to include the rows that have a temperature lower than 30 AND an altitude higher than 100. If we wanted to include the rows that have a temperature lower than 30 OR an altitude higher than 100 instead, then we would select **Match any**. This would include the first row in the original data, which has a temperature of 32°C (does not match the first condition) but an altitude of 101 (which matches the second condition), so it is included.

Conditions that are invalid or incompletely configured are ignored.

Group by

This transformation groups the data by a specified field (column) value and processes calculations on each group. Click to see a list of calculation choices.

Here's an example of original data.

Time	Server ID	CPU Temperature	Server Status
7/7/2020 11:34:20 AM	server 1	80	Shutdown
7/7/2020 11:34:20 AM	server 3	62	OK
7/7/2020 10:32:20 AM	server 2	90	Overload
7/7/2020 10:31:22 AM	server 3	55	OK
7/7/2020 9:30:57 AM	server 3	62	Rebooting
7/7/2020 9:30:05 AM	server 2	88	OK
7/7/2020 9:28:06 AM	server 1	80	OK
7/7/2020 9:25:05 AM	server 2	88	OK
7/7/2020 9:23:07 AM	server 1	86	OK

This transformation goes in two steps. First you specify one or multiple fields to group the data by. This will group all the same values of those fields together, as if you sorted them. For instance if we group by the Server ID field, then it would group the data this way:

Time	Server ID	CPU Temperature	Server Status
7/7/2020 11:34:20 AM	server 1	80	Shutdown
7/7/2020 9:28:06 AM	server 1	80	OK
7/7/2020 9:23:07 AM	server 1	86	OK
7/7/2020 10:32:20 AM	server 2	90	Overload

Time	Server ID	CPU Temperature	Server Status
7/7/2020 9:30:05 AM	server 2	88	OK
7/7/2020 9:25:05 AM	server 2	88	OK
7/7/2020 11:34:20 AM	server 3	62	OK
7/7/2020 10:31:22 AM	server 3	55	OK
7/7/2020 9:30:57 AM	server 3	62	Rebooting

All rows with the same value of Server ID are grouped together.

After choosing which field you want to group your data by, you can add various calculations on the other fields, and apply the calculation to each group of rows. For instance, we could want to calculate the average CPU temperature for each of those servers. So we can add the *mean* calculation applied on the CPU Temperature field to get the following:

Server ID	CPU Temperature (mean)
server 1	82
server 2	88.6
server 3	59.6

And we can add more than one calculation. For instance:

- For field Time, we can calculate the *Last* value, to know when the last data point was received for each server
- For field Server Status, we can calculate the *Last* value to know what is the last state value for each server
- For field Temperature, we can also calculate the *Last* value to know what is the latest monitored temperature for each server

We would then get:

Server ID	CPU Temperature (mean)	CPU Temperature (last)	Time (last)	Server Status (last)
server 1	82	80	7/7/2020 11:34:20 AM	Shutdown
server 2	88.6	90	7/7/2020 10:32:20 AM	Overload
server 3	59.6	62	7/7/2020 11:34:20 AM	OK

This transformation enables you to extract key information from your time series and display it in a convenient way.

Join by field

Use this transformation to join multiple results into a single table. This is especially useful for converting multiple time series results into a single wide table with a shared time field.

Inner join

An inner join merges data from multiple tables where all tables share the same value from the selected field. This type of join excludes data where values do not match in every result.

Use this transformation to combine the results from multiple queries (combining on a passed join field or the first time column) into one result, and drop rows where a successful join cannot occur.

In the following example, two queries return table data. It is visualized as two separate tables before applying the inner join transformation.

Query A:

Time	Job	Uptime
7/7/2020 11:34:20 AM	node	25260122

Time	Job	Uptime
7/7/2020 11:24:20 AM	postgre	123001233
7/7/2020 11:14:20 AM	postgre	345001233

Query B:

Time	Server	Errors
7/7/2020 11:34:20 AM	server 1	15
7/7/2020 11:24:20 AM	server 2	5
7/7/2020 11:04:20 AM	server 3	10

The result after applying the inner join transformation looks like the following:

Time	Job	Uptime	Server	Errors
7/7/2020 11:34:20 AM	node	25260122	server 1	15
7/7/2020 11:24:20 AM	postgre	123001233	server 2	5

Outer join

An outer join includes all data from an inner join and rows where values do not match in every input. While the inner join joins Query A and Query B on the time field, the outer join includes all rows that don't match on the time field.

In the following example, two queries return table data. It is visualized as two tables before applying the outer join transformation.

Query A:

Time	Job	Uptime
7/7/2020 11:34:20 AM	node	25260122
7/7/2020 11:24:20 AM	postgre	123001233
7/7/2020 11:14:20 AM	postgre	345001233

Query B:

Time	Server	Errors
7/7/2020 11:34:20 AM	server 1	15
7/7/2020 11:24:20 AM	server 2	5
7/7/2020 11:04:20 AM	server 3	10

The result after applyign the outer join transformation looks like the following:

Time	Job	Uptime	Server	Errors
7/7/2020 11:04:20 AM			server 3	10
7/7/2020 11:14:20 AM	postgre	345001233		
7/7/2020 11:34:20 AM	node	25260122	server 1	15
7/7/2020 11:24:20 AM	postgre	123001233	server 2	5

Labels to fields

This transformation changes time series results that include labels or tags into a table where each label keys and values are included in the table result. The labels can be displayed either as columns or as row values.

Given a query result of two time series:

- Series 1 – labels Server=Server A, Datacenter=EU
- Series 2 – labels Server=Server B, Datacenter=EU

In **Columns** mode, the result looks like this:

Time	Server	Datacenter	Value
7/7/2020 11:34:20 AM	Server A	EU	1
7/7/2020 11:34:20 AM	Server B	EU	2

In “Rows” mode, the result has a table for each series and show each label value like this:

label	value
Server	Server A
Datacenter	EU

label	value
Server	Server B
Datacenter	EU

Value field name

If you selected Server as the **Value field name**, then you would get one field for every value of the Server label.

Time	Datacenter	Server A	Server B
7/7/2020 11:34:20 AM	EU	1	2

Merging behavior

The labels to fields transformer is internally two separate transformations. The first acts on single series and extracts labels to fields. The second is the merge transformation that joins all the results into a single table. The merge transformation tries to join on all matching fields. This merge step is required and cannot be turned off.

> **Note**
>
> The *merge* transformation can be used on its own, and is described in detail below.

To illustrate this, here is an example where you have two queries that return time series with no overlapping labels.

- Series 1 – labels Server=ServerA
- Series 2 – labels Datacenter=EU

This will first result in these two tables:

Time	Server	Value
7/7/2020 11:34:20 AM	ServerA	10

Time	Datacenter	Value
7/7/2020 11:34:20 AM	EU	20

After merge:

Time	Server	Value	Datacenter
7/7/2020 11:34:20 AM	ServerA	10	
7/7/2020 11:34:20 AM		20	EU

Merge

Use this transformation to combine the result from multiple queries into one single result. This is helpful when using the table panel visualization. Values that can be merged are combined into the same row. Values are mergeable if the shared fields contain the same data.

In the example below, we have two queries returning table data. It is visualized as two separate tables before applying the transformation.

Query A:

Time	Job	Uptime
7/7/2020 11:34:20 AM	node	25260122
7/7/2020 11:24:20 AM	postgre	123001233

Query B:

Time	Job	Errors
7/7/2020 11:34:20 AM	node	15
7/7/2020 11:24:20 AM	postgre	5

Here is the result after applying the Merge transformation:

Time	Job	Errors	Uptime
7/7/2020 11:34:20 AM	node	15	25260122
7/7/2020 11:24:20 AM	postgre	5	123001233

Organize fields

Use this transformation to rename, reorder, or hide fields returned by the query.

> **Note**
>
> This transformation only works in panels with a single query. If your panel has multiple queries, then you must either apply an Outer join transformation or remove the extra queries.

Grafana displays a list of fields returned by the query. You can:

- Change field order by hovering your cursor over a field. The cursor turns into a hand and then you can drag the field to its new place.
- Hide or show a field by clicking the eye icon next to the field name.
- Rename fields by typing a new name in the Rename box.

Partition by values

This transformation can help eliminate the need for multiple queries to the same datasource with different `WHERE` clauses when graphing multiple series. Consider a metrics SQL table with the following data:

Time	Region	Value
10/20/2022 12:00:00 PM	US	1520
10/20/2022 12:00:00 PM	EU	2936

Time	Region	Value
10/20/2022 1:00:00 AM	US	1327
10/20/2022 1:00:00 AM	EU	912

Prior to v9.3, if you wanted to plot a red trendline for US and a blue one for EU in the same TimeSeries panel, you would likely have to split this into two queries:

```
SELECT Time, Value FROM metrics WHERE Time > '2022-10-20' AND Region='US'
SELECT Time, Value FROM metrics WHERE Time > '2022-10-20' AND Region='EU'
```

This also requires you to know ahead of time which regions actually exist in the metrics table.

With the *Partition by values* transformer, you can now issue a single query and split the results by unique values in one or more columns (`fields`) of your choosing. The following example uses `Region`.

```
SELECT Time, Region, Value FROM metrics WHERE Time > '2022-10-20'
```

Time	Region	Value
10/20/2022 12:00:00 PM	US	1520
10/20/2022 1:00:00 AM	US	1327

Time	Region	Value
10/20/2022 12:00:00 PM	EU	2936
10/20/2022 1:00:00 AM	EU	912

Reduce

The *Reduce* transformation applies a calculation to each field in the frame and return a single value. Time fields are removed when applying this transformation.

Consider the input:

Query A:

Time	Temp	Uptime
7/7/2020 11:34:20 AM	12.3	256122
7/7/2020 11:24:20 AM	15.4	1230233

Query B:

Time	AQI	Errors
7/7/2020 11:34:20 AM	6.5	15
7/7/2020 11:24:20 AM	3.2	5

The reduce transformer has two modes:

- **Series to rows** - Creates a row for each field and a column for each calculation.
- **Reduce fields** - Keeps the existing frame structure, but collapses each field into a single value.

For example, if you used the **First** and **Last** calculation with a **Series to rows** transformation, then the result would be:

Field	First	Last
Temp	12.3	15.4
Uptime	256122	1230233
AQI	6.5	3.2
Errors	15	5

The Reduce fields with the Last calculation, results in two frames, each with one row:

Query A:

Temp	Uptime
15.4	1230233

Query B:

AQI	Errors
3.2	5

Rename by regex

Use this transformation to rename parts of the query results using a regular expression and replacement pattern.

You can specify a regular expression, which is only applied to matches, along with a replacement pattern that support back references. For example, let's imagine you're visualizing CPU usage per host and you want to remove the domain name. You could set the regex to `([^\.]+)\..+` and the replacement pattern to `$1`, `web-01.example.com` would become `web-01`.

Rows to fields

The rows to fields transformation converts rows into separate fields. This can be useful as fields can be styled and configured individually. It can also use additional fields as sources for dynamic field configuration or map them to field labels. The additional labels can then be used to define better display names for the resulting fields.

This transformation includes a field table which lists all fields in the data returned by the config query. This table gives you control over what field should be mapped to each config property (the *Use as** option). You can also choose which value to select if there are multiple rows in the returned data.

This transformation requires:

- One field to use as the source of field names.

By default, the transform uses the first string field as the source. You can override this default setting by selecting **Field name** in the **Use as** column for the field you want to use instead.

- One field to use as the source of values.

  By default, the transform uses the first number field as the source. But you can override this default setting by selecting **Field value** in the **Use as** column for the field you want to use instead.

Useful when visualizing data in:

- Gauge
- Stat
- Pie chart

Map extra fields to labels

If a field does not map to config property Grafana will automatically use it as source for a label on the output field-

Example:

Name	DataCenter	Value
ServerA	US	100
ServerB	EU	200

Output:

ServerA (labels: DataCenter: US)	ServerB (labels: DataCenter: EU)
10	20

The extra labels can now be used in the field display name provide more complete field names.

If you want to extract config from one query and apply it to another you should use the config from query results transformation.

Example

Input:

Name	Value	Max
ServerA	10	100
ServerB	20	200
ServerC	30	300

Output:

ServerA (config: max=100)	ServerB (config: max=200)	ServerC (config: max=300)
10	20	30

As you can see each row in the source data becomes a separate field. Each field now also has a max config option set. Options like Min, Max, Unit and Thresholds are all part of field configuration and if set like this will be used by the visualization instead of any options manually configured in the panel editor options pane.

Prepare time series

Prepare time series transformation is useful when a data source returns time series data in a format that isn't supported by the panel you want to use.

This transformation helps you resolve this issue by converting the time series data from either the wide format to the long format or the other way around.

Select the **Multi-frame time series** option to transform the time series data frame from the wide to the long format.

Select the **Wide time series** option to transform the time series data frame from the long to the wide format.

Series to rows

Use this transformation to combine the result from multiple time series data queries into one single result. This is helpful when using the table panel visualization.

The result from this transformation will contain three columns: Time, Metric, and Value. The Metric column is added so you easily can see from which query the metric originates from. Customize this value by defining Label on the source query.

In the example below, we have two queries returning time series data. It is visualized as two separate tables before applying the transformation.

Query A:

Time	Temperature
7/7/2020 11:34:20 AM	25
7/7/2020 10:31:22 AM	22
7/7/2020 9:30:05 AM	19

Query B:

Time	Humidity
7/7/2020 11:34:20 AM	24
7/7/2020 10:32:20 AM	29
7/7/2020 9:30:57 AM	33

Here is the result after applying the Series to rows transformation.

Time	Metric	Value
7/7/2020 11:34:20 AM	Temperature	25

Time	Metric	Value
7/7/2020 11:34:20 AM	Humidity	22
7/7/2020 10:32:20 AM	Humidity	29
7/7/2020 10:31:22 AM	Temperature	22
7/7/2020 9:30:57 AM	Humidity	33
7/7/2020 9:30:05 AM	Temperature	19

Sort by

This transformation will sort each frame by the configured field, When `reverse` is checked, the values will return in the opposite order.

Limit

Use this transformation to limit the number of rows displayed.

In the example below, we have the following response from the data source:

Time	Metric	Value
7/7/2020 11:34:20 AM	Temperature	25
7/7/2020 11:34:20 AM	Humidity	22
7/7/2020 10:32:20 AM	Humidity	29
7/7/2020 10:31:22 AM	Temperature	22
7/7/2020 9:30:57 AM	Humidity	33
7/7/2020 9:30:05 AM	Temperature	19

Here is the result after adding a Limit transformation with a value of ‘3’:

Time	Metric	Value
7/7/2020 11:34:20 AM	Temperature	25
7/7/2020 11:34:20 AM	Humidity	22
7/7/2020 10:32:20 AM	Humidity	29

Troubleshoot queries

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This page provides information to solve common dashboard problems.

I get different results when I rearrange my functions

Function order is very important. Just like in math, the order that you place your functions can affect the result.

Inspect your query request and response

The most common problems are related to the query and response from your data source. Even if it looks like a bug or visualization issue in Grafana, it is almost always a problem with the data source query or the data source response. Start by inspecting your panel query and response.

For more information, refer to Inspect request and response data.

My query is slow

How many data points is your query returning? A query that returns lots of data points will be slow. Try this:

- In **Query options**, limit the **Max data points** returned.
- In **Query options**, increase the **Min interval** time.
- In your query, use a `group by` function.

Configure thresholds

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This section includes information about using thresholds in your visualizations. You'll learn about thresholds and their defaults, how to add or delete a threshold, and adding a threshold to a legacy panel.

About thresholds

A threshold is a value that you specify for a metric that is visually reflected in a dashboard when the threshold value is met or exceeded.

Thresholds provide one method for you to conditionally style and color your visualizations based on query results. You can apply thresholds to most, but not all, visualizations. For more information about visualizations, refer to Visualization panels.

You can use thresholds to:

- Color grid lines or grid ares areas in the Time-series visualization
- Color lines in the Time-series visualization
- Color the background or value text in the Stat visualization
- Color the gauge and threshold markers in the Gauge visualization
- Color markers in the Geomap visualization
- Color cell text or background in the Table visualization

- Define regions and region colors in the State timeline visualization

There are two types of thresholds:

- **Absolute** thresholds are defined by a number. For example, 80 on a scale of 1 to 150.
- **Percentage** thresholds are defined relative to minimum or maximum. For example, 80 percent.

Default thresholds

On visualizations that support it, Grafana sets default threshold values of:

- 80 = red
- Base = green
- Mode = Absolute

The **Base** value represents minus infinity. It is generally the "good" color.

Add or delete a threshold

You can add as many thresholds to a panel as you want. Grafana automatically sorts thresholds values from highest to lowest.

Delete a threshold when it is no longer relevant to your business operations. When you delete a threshold, the system removes the threshold from all visualizations that include the threshold.

1. To add a threshold:
    1. Edit the panel to which you want to add a threshold.
    2. In the options side pane, locate the **Thresholds** section and click **+ Add threshold**.
    3. Select a threshold color, number, and mode. Threshold mode applies to all thresholds on this panel.
    4. For a time-series panel, select a **Show thresholds** option.
2. To delete a threshold, navigate to the panel that contains the threshold and click the trash icon next to the threshold you want to remove.

Add a threshold to a legacy graph panel

In the Graph panel visualization, thresholds enable you to add lines or sections to a graph to make it easier to recognize when the graph crosses a threshold.

1. Navigate to the graph panel to which you want to add a threshold.
2. On the **Panel** tab, click **Thresholds**.
3. Click **Add threshold**.
4. Complete the following fields:
   - **T1 -** Both values are required to display a threshold.
     - **lt** or **gt** - Select **lt** for less than or **gt** for greater than to indicate what the threshold applies to.
     - **Value -** Enter a threshold value. Grafana draws a threshold line along the Y-axis at that value.
   - **Color -** Choose a condition that corresponds to a color, or define your own color.
     - **custom -** You define the fill color and line color.
     - **critical -** Fill and line color are red.
     - **warning -** Fill and line color are yellow.
     - **ok -** Fill and line color are green.
   - **Fill -** Controls whether the threshold fill is displayed.
   - **Line -** Controls whether the threshold line is displayed.
   - **Y-Axis -** Choose **left** or **right**.
5. Click **Save** to save the changes in the dashboard.

Configure data links

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can use data link variables or data links to create links between panels.

Data link variables

You can use variables in data links to refer to series fields, labels, and values. For more information about data links, see Data links.

To see a list of available variables, type **$** in the data link **URL** field to see a list of variables that you can use.

> **Note**
>
> These variables changed in 6.4, so if you have an older version of Grafana, then use the version picker to select docs for an older version of Grafana.

You can also use template variables in your data links URLs, see Adding and managing dashboard variables.

Time range panel variables

These variables allow you to include the current time range in the data link URL.

- **__url_time_range** - current dashboard's time range (i.e. **?from=now-6h&to=now**)
- **$__from and $__to** - For more information, see Global variables.

Series variables

Series specific variables are available under **__series** namespace:

- **__series.name** - series name to the URL

Field variables

Field-specific variables are available under **__field** namespace:

- **__field.name** - the name of the field
- **__field.labels.<LABEL>** - label's value to the URL. If your label contains dots, then use **__field.labels["<LABEL>"]** syntax.

Value variables

Value-specific variables are available under **__value** namespace:

- **__value.time** - value's timestamp (Unix ms epoch) to the URL (i.e. **?time=1560268814105**)
- **__value.raw** - raw value
- **__value.numeric** - numeric representation of a value
- **__value.text** - text representation of a value
- **__value.calc** - calculation name if the value is result of calculation

Template variables

When linking to another dashboard that uses template variables, select variable values for whoever clicks the link.

${var-myvar:queryparam} - where **var-myvar** is the name of the template variable that matches one in the current dashboard that you want to use.

Variable state	Result in the created URL
selected one value	**var-myvar=value1**
selected multiple values	**var-myvar=value1&var-myvar=value2**
selected **All**	**var-myvar=All**

If you want to add all of the current dashboard's variables to the URL, then use **${__all_variables}**.

Data links

Data links allow you to provide more granular context to your links. You can create links that include the series name or even the value under the cursor. For example, if your visualization showed four servers, you could add a data link to one or two of them.

The link itself is accessible in different ways depending on the visualization. For the Graph you need to click on a data point or line, for a panel like Stat, Gauge, or Bar Gauge you can click anywhere on the visualization to open the context menu.

You can use variables in data links to send people to a detailed dashboard with preserved data filters. For example, you could use variables to specify a time range, series, and variable selection. For more information, see Data link variables.

Typeahead suggestions

When creating or updating a data link, press Cmd+Space or Ctrl+Space on your keyboard to open the typeahead suggestions to more easily add variables to your URL.

Add a data link

1. Hover your cursor over the panel that you want to add a link to and then press **e**. Or click the dropdown arrow next to the panel title and then click **Edit**.
2. On the Field tab, scroll down to the Data links section.
3. Expand Data links and then click **Add link**.
4. Enter a **Title**. **Title** is a human-readable label for the link that will be displayed in the UI.
5. Enter the **URL** you want to link to.

   You can even add one of the template variables defined in the dashboard. Click in the **URL** field and then type **$** or press Ctrl+Space or Cmd+Space to see a list of available variables. By adding template variables to your panel link, the link sends the user to the right context, with the relevant variables already set. For more information, see Data link variables.
6. If you want the link to open in a new tab, then select **Open in a new tab**.
7. Click **Save** to save changes and close the window.
8. Click **Save** in the upper right to save your changes to the dashboard.

Update a data link

1. On the Field tab, find the link that you want to make changes to.
2. Click the Edit (pencil) icon to open the Edit link window.
3. Make any necessary changes.
4. Click **Save** to save changes and close the window.
5. Click **Save** in the upper right to save your changes to the dashboard.

Delete a data link

1. On the Field tab, find the link that you want to delete.
2. Click the **X** icon next to the link you want to delete.
3. Click **Save** in the upper right to save your changes to the dashboard.

Configure field overrides

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Overrides allow you to customize visualization settings for specific fields or series. This is accomplished by adding an override rule that targets a particular set of fields and that can each define multiple options.

For example, you set the unit for all fields that include the text "bytes" by adding an override using the **Fields with name matching regex** matcher and then add the Unit option to the override rule.

Example 1: Format temperature

Let's assume that our result set is a data frame that consists of two fields: time and temperature.

time	temperature
2020-01-02 03:04:00	45.0
2020-01-02 03:05:00	47.0
2020-01-02 03:06:00	48.0

Each field (column) of this structure can have field options applied that alter the way its values are displayed. This means that you can, for example, set the Unit to Temperature > Celsius, resulting in the following table:

time	temperature
2020-01-02 03:04:00	45.0 °C
2020-01-02 03:05:00	47.0 °C
2020-01-02 03:06:00	48.0 °C

In addition, the decimal place is not required, so we can remove it. You can change the Decimals from **auto** to zero (**0**), resulting in the following table:

time	temperature
2020-01-02 03:04:00	45 °C
2020-01-02 03:05:00	47 °C
2020-01-02 03:06:00	48 °C

Example 2: Format temperature and humidity

Let's assume that our result set is a data frame that consists of four fields: time, high temp, low temp, and humidity.

time	high temp	low temp	humidity
2020-01-02 03:04:00	45.0	30.0	67
2020-01-02 03:05:00	47.0	34.0	68
2020-01-02 03:06:00	48.0	31.0	68

Let's add the Celsius unit and get rid of the decimal place. This results in the following table:

time	high temp	low temp	humidity
2020-01-02 03:04:00	45 °C	30 °C	67 °C
2020-01-02 03:05:00	47 °C	34 °C	68 °C
2020-01-02 03:06:00	48 °C	31 °C	68 °C

The temperature fields look good, but the humidity must now be changed. We can fix this by applying a field option override to the humidity field and change the unit to Misc > percent (0-100).

time	high temp	low temp	humidity
2020-01-02 03:04:00	45 °C	30 °C	67%
2020-01-02 03:05:00	47 °C	34 °C	68%
2020-01-02 03:06:00	48 °C	31 °C	68%

Add a field override

A field override rule can customize the visualization settings for a specific field or series.

1. Edit the panel to which you want to add an override.
2. In the panel options side pane, click **Add field override** at the bottom of the pane.
3. Select which fields an override rule will be applied to:

   - **Fields with name:** Select a field from the list of all available fields. Properties you add to a rule with this selector are only applied to this single field.
   - **Fields with name matching regex:** Specify fields to override with a regular expression. Properties you add to a rule with this selector are applied to all fields where the field name match the regex.
   - **Fields with type:** Select fields by type, such as string, numeric, and so on. Properties you add to a rule with this selector are applied to all fields that match the selected type.

- **Fields returned by query:** Select all fields returned by a specific query, such as A, B, or C. Properties you add to a rule with this selector are applied to all fields returned by the selected query.

4. Click **Add override property**.
5. Select the field option that you want to apply.
6. Enter options by adding values in the fields. To return options to default values, delete the white text in the fields.
7. Continue to add overrides to this field by clicking **Add override property**, or you can click **Add override** and select a different field to add overrides to.
8. When finished, click **Save** to save all panel edits to the dashboard.

Delete a field override

Delete a field override when you no longer need it. When you delete an override, the appearance of value defaults to its original format. This change impacts dashboards and dashboard users that rely on an affected panel.

1. Edit the panel that contains the override you want to delete.
2. In panel options side pane, scroll down until you see the overrides.
3. Click the override you want to delete and then click the associated trash icon.

View field overrides

You can view field overrides in the panel display options.

1. Edit the panel that contains the overrides you want to view.
2. In panel options side pane, scroll down until you see the overrides.

   The override settings that appear on the **All** tab are the same as the settings that appear on the **Overrides** tab.

Edit a field override

Edit a field override when you want to make changes to an override setting. The change you make takes effect immediately.

1. Edit the panel that contains the overrides you want to edit.

2. In panel options side pane, scroll down until you see the overrides.
3. Locate the override that you want to change.
4. Perform any of the following:

   - Edit settings on existing overrides or field selection parameters.
   - Delete existing override properties by clicking the **X** next to the property.
   - Add an override properties by clicking **Add override property**.

Configure value mappings

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

In addition to field overrides, value mapping is a technique that you can use to change the visual treatment of data that appears in a visualization.

Values mapped via value mappings bypass the unit formatting. This means that a text value mapped to a numerical value is not formatted using the configured unit.

If value mappings are present in a panel, then Grafana displays a summary in the side pane of the panel editor.

> **Note**
>
> The new value mappings are not compatible with some visualizations, such as Graph (old), Text, and Heatmap.

Types of value mappings

Grafana supports the following value mappings:

- **Value:** Maps text values to a color or different display text. For example, you can configure a value mapping so that all instances of the value **10** appear as **Perfection!** rather than the number.
- **Range:** Maps numerical ranges to a display text and color. For example, if a value is within a certain range, you can configure a range value mapping to display **Low** or **High** rather than the number.
- **Regex:** Maps regular expressions to replacement text and a color. For example, if a value is **www.example.com**, you can configure a regex value mapping so that Grafana displays **www** and truncates the domain.
- **Special** Maps special values like **Null**, **NaN** (not a number), and boolean values like **true** and **false** to a display text and color. For example, you can configure a special value mapping so that **null** values appear as **N/A**.

You can also use the dots on the left to drag and reorder value mappings in the list.

Map a value

Map a value when you want to format a single value.

1. Open a panel for which you want to map a value.
2. In panel display options, locate the **Value mappings** section and click **Add value mappings**.
3. Click **Add a new mapping** and then select **Value**.
4. Enter the value for Grafana to match.
5. (Optional) Enter display text.
6. (Optional) Set the color.
7. Click **Update** to save the value mapping.

Map a range

Map a range of values when you want to format multiple, continuous values.

1. Edit the panel for which you want to map a range of values.
2. In panel display options, in the **Value mappings** section, click **Add value mappings**.
3. Click **Add a new mapping** and then select **Range**.
4. Enter the beginning and ending values in the range for Grafana to match.

5. (Optional) Enter display text.
6. (Optional) Set the color.
7. Click **Update** to save the value mapping.

Map a regular expression

Map a regular expression when you want to format the text and color of a regular expression value.

1. Edit the panel for which you want to map a regular expression.
2. In the **Value mappings** section of the panel display options, click **Add value mappings**.
3. Click **Add a new mapping** and then select **Regex**.
4. Enter the regular expression pattern for Grafana to match.
5. (Optional) Enter display text.
6. (Optional) Set the color.
7. Click **Update** to save the value mapping.

Map a special value

Map a special value when you want to format uncommon, boolean, or empty values.

1. Edit the panel for which you want to map a special value.
2. In panel display options, locate the **Value mappings** section and click **Add value mappings**.
3. Click **Add a new mapping** and then select **Special**.
4. Select the special value for Grafana to match.
5. (Optional) Enter display text.
6. (Optional) Set the color.
7. Click **Update** to save the value mapping.

Edit a value mapping

You can change a value mapping at any time.

1. Edit the panel that contains the value mapping you want to edit.
2. In the panel display options, in the **Value mappings** section, click **Edit value mappings**.
3. Make the changes and click **Update**.

Configure a legend



This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A panel includes a legend that you can use to interpret data displayed in a visualization. Each legend option adds context and clarity to the data illustrated in a visualization.

Isolating series data in a visualization

Visualizations can often be visually complex, and include many data series. You can simplify the view by removing series data from the visualization, which isolates the data you want to see. Grafana automatically creates a new override in the **Override** tab.

When you apply your changes, the visualization changes appear to all users of the panel.

To isolate series data in a visualization

1. Open the panel.
2. In the legend, select the label of the series you want to isolate.

   The system removes from view all other series data.
3. To incrementally add series data to an isolated series, press the **Ctrl** or **Command** key and select the label of the series you want to add.
4. To revert back to the default view that includes all data, click any series label twice.
5. To save your changes so that they appear to all viewers of the panel, select **Apply**.

This topic currently applies to the following visualizations:

- Bar chart
- Histogram

- Pie chart
- State timeline
- Status history
- Time series

Adding values to a legend

As way to add more context to a visualization, you can add series data values to a legend. You can add as many values as you'd like; after you apply your changes, you can horizontally scroll the legend to see all values.

To add values to a legend

1. Edit a panel.
2. In the panel display options pane, locate the **Legend** section.
3. In the **Legend values** field, select the values you want to appear in the legend.
4. Choose **Apply** to save your changes are navigate back to the dashboard.

Changing a series color

By default, Grafana specifies the color of your series data, which you can change.

To change a series color

1. Edit a panel.
2. In the legend, select the color bar associated with the series.
3. Select a pre-set color or a custom color from the color palette.
4. Choose **Apply** to save your changes are navigate back to the dashboard.

Sort series

You can change legend mode to **Table** and choose Calculation types to be displayed in the legend. Select the calculation name header in the legend table to sort the values in the table in ascending or descending order.

The sort order affects the positions of the bars in the Bar chart panel as well as the order of stacked series in the Time series and Bar chart panels.

Note

This feature is only supported in these panels: Bar chart, Histogram, Time series, XY Chart.

Calculation types

This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The following table contains a list of calculations you can perform in Grafana. You can find these calculations in the **Transform** tab and in the bar gauge, gauge, and stat visualizations.

Calculation	Description
All nulls	True when all values are null
All zeros	True when all values are 0
Change count	Number of times the field's value changes
Count	Number of values in a field
Delta	Cumulative change in value, only counts increments
Difference	Difference between first and last value of a field
Difference percent	Percentage change between first and last value of a field
Distinct count	Number of unique values in a field
First (not null)	First, not null value in a field

Calculation	Description
Max	Maximum value of a field
Mean	Mean value of all values in a field
Variance	Variance of all values in a field
StdDev	Standard deviation of all values in a field
Min	Minimum value of a field
Min (above zero)	Minimum, positive value of a field
Range	Difference between maximum and minimum values of a field
Step	Minimal interval between values of a field
Total	Sum of all values in a field

Annotating visualizations

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Annotations provide a way to mark points on the graph with rich events. When you hover over an annotation, you can get event description and event tags. The text field can include links to other systems with more detail.

Native annotations

Grafana comes with a native annotation store and the ability to add annotation events directly from the graph panel or through the HTTP API.

Adding an annotation

1. In the dashboard, click on the **Time series** panel. A context menu will appear.
2. In the context menu, click **Add annotation**.
3. Add an annotation description, and optionally, tags.
4. Click **Save**.

Alternatively, to add an annotation, `Ctrl+Click` or `Cmd+Click` on the **Time series** panel and the **Add annotation** popover will appear.

Adding region annotation

1. In the dashboard, `Ctrl+Click` or `Cmd+Click`on the **Time series** panel.
2. In the context menu, click on **Add annotation**.
3. Add an annotation description, and optionally, tags.
4. Click **Save**.

Editing an annotation

1. In the dashboard, hover over an annotation indicator on the **Time series** panel.
2. Click on the edit (pencil) icon in the annotation tooltip.
3. Modify the description, and optionally, tags.
4. Click **Save**.

Deleting an annotation

1. In the dashboard, hover over an annotation indicator on the **Time series** panel.
2. Click on the trash icon in the annotation tooltip.

Built-in query

After you added an annotation, they will still be visible. This is due to the built in annotation query that exists on all dashboards. This annotation query will fetch all annotation events that originate from the current dashboard and show them on the panel where they were created. This includes alert state history annotations. You can stop annotations from being fetched and drawn

by opening the **Annotations** settings (through Dashboard cogs menu) and modifying the query named `Annotations & Alerts (Built-in)`.

When you copy a dashboard using the **Save As** feature, it will get a new dashboard id so annotations created on source dashboard will no longer be visible on the copy. You can still show them if you add a new **Annotation Query** and filter by tags. This only works if the annotations on the source dashboard had tags to filter by.

Querying by tag

You can create new queries to fetch annotations from the native annotation store through the `-- Grafana --` data source by setting **Filter by** to `Tags`.

Grafana v8.1 and later versions also support typeahead of existing tags, provide at least one tag.

For example, create an annotation query name `outages` and specify a tag `outage`. This query will show all annotations (from any dashboard or through API) with the outage tag. If multiple tags are defined in an annotation query, then Grafana will only show annotations matching all the tags. To modify the behavior, enable `Match any`, and Grafana will show annotations that contain any one of the tags you provided.

In Grafana v5.3+ it's possible to use template variables in the tag query. So if you have a dashboard showing stats for different services and a template variable that dictates which services to show, you can now use the same template variable in your annotation query to only show annotations for those services.

Querying other data sources

Annotation events are fetched through annotation queries. To add a new annotation query to a dashboard open the dashboard settings menu, then select **Annotations**. This will open the dashboard annotations settings view. To create a new annotation query hit the **New** button.

Specify a name for the annotation query. This name is given to the toggle (check box) that will allow you to enable or disable showing annotation events from this query. For example you might have two annotation queries named `Deploys` and `Outages`. The toggle will allow you to decide what annotations to show.

Annotation query details

The annotation query options are different for each data source. For information about annotations in a specific data source, see the specific data source topic.

The panel inspect view



This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The panel inspect view, which you can open via the panel menu, helps you understand and troubleshoot your panels. You can inspect the raw data for any Grafana panel, export that data to a comma-separated values (CSV) file, view query requests, and export panel and data JSON.

> **Note**
>
> Not all panel types include all tabs. For example, dashboard list panels do not have raw data to inspect, so they do not display the Stats, Data, or Query tabs.

The panel inspector consists of the following options:

1. The panel inspector displays **Inspect: [NameOfPanelBeingInspected]** at the top of the pane. Click the arrow in the upper right corner to expand or reduce the pane.
2. **Data tab -** Shows the raw data returned by the query with transformations applied. Field options such as overrides and value mappings are not applied by default.
3. **Stats tab -** Shows how long your query takes and how much it returns.
4. **JSON tab -** Allows you to view and copy the panel JSON, panel data JSON, and data frame structure JSON. This is useful if you are provisioning or administering Grafana.
5. **Query tab -** Shows you the requests to the server sent when Grafana queries the data source.
6. **Error tab -** Shows the error. Only visible when query returns error.

Download raw query results

Grafana generates a CSV file that contains your data, including any transformations to that data. You can choose to view the data before or after the panel applies field options or field option overrides.

1. Edit the panel that contains the query data you want to download.
2. In the query editor, click **Query Inspector**.
3. Click **Data**.

   If your panel contains multiple queries or queries multiple nodes, then you have additional options.

   - **Select result**: Choose which result set data you want to view.
   - **Transform data**
   - **Join by time**: View raw data from all your queries at once, one result set per column. Click a column heading to reorder the data.
4. To see data before the system applies field overrides, click the **Formatted data** toggle.
5. To download a CSV file specifically formatted for Excel, click the **Download for Excel** toggle .
6. Click **Download CSV**.

Inspect query performance

The **Stats** tab displays statistics that tell you how long your query takes, how many queries you send, and the number of rows returned. This information can help you troubleshoot your queries, especially if any of the numbers are unexpectedly high or low.

1. Edit the panel that contains the query with performance you want to inspect.
2. In the query editor, click **Query Inspector**.
3. Click **Stats**.

Statistics are displayed in read-only format.

Inspect query request and response

Inspect query request and response data when you want to troubleshoot a query that returns unexpected results, or fails to return expected results.

1. Edit the panel that contains the query you want to export.
2. In the query editor, click **Query Inspector**.
3. Click **Refresh**.

   The panel populates with response data.
4. Make adjustments, as necessary and re-run the query.
5. To download the query request and response data, click the **Copy to clipboard** icon and paste the results into another application.

Visualizations available in Grafana version 9

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana offers a variety of visualizations to support different use cases. This section of the documentation highlights the built-in panels, their options and typical usage.

A common panel to get started with, and to learn the basics of using panels, is the Time series panel panel.

Topics

- Alert list panel
- Annotations panel
- Bar chart panel
- Bar gauge
- Candlestick panel
- Canvas panel
- Clock panel

- Dashboard list
- Gauge panel
- Geomap panel
- Graph panel
- Heatmap panel
- Histogram panel
- Logs panel
- News panel
- Node graph panel
- Pie chart panel
- Plotly panel
- Sankey panel
- Scatter panel
- Stat panel
- State timeline panel
- Status history panel
- Table panel
- Text panel
- Time series panel
- Traces panel (Beta)
- WindRose

Alert list panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The alert list panel displays your dashboards alerts. You can configure the list to show current state or recent state changes. For more information about alerts, see Alerts in Grafana version 9.

Use these settings to refine your visualization.

Options

- **Group mode** – Choose **Default grouping** to show alert instances grouped by their alert rule, or **Custom grouping** to group alert instances by a custom set of labels.
- **Max Items** – Set the maximum number of alerts to list.
- **Sort order** – Select how to order the alerts displayed.
  - **Alphabetical (asc)** – Alphabetical order
  - **Alphabetical (desc)** – Reverse alphabetical order
  - **Importance** – By importance according to the following values, with 1 being the highest:
    - `alerting` or `firing`: 1
    - `no_data`: 2
    - `pending`: 3
    - ok: 4
    - `paused` or `inactive`: 5
- **Alerts from this dashboard** – Show alerts only from the dashboard that the alert list is in.

Filter

Use the following options to filter the alerts to match the query, folder, or tags that you choose:

- **Alert name** – Enter an alert name query.
- **Alert instance label** – Filter alert instances using label querying. For example, `{severity="critical", instance=~"cluster-us-.+"}`.
- **Folder** – Select a folder. Only alerts from dashboards in the selected folder will be displayed.
- **Datasource** – Filter alerts from the selected data source.

State filter

Choose which alert states to display in this panel.

- Alerting / Firing

- Pending
- No data
- Normal
- Error

Annotations panel

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The Annotations panel shows a list of available annotations you can use to view annotated data. Various options are available to filter the list based on tags and on the current dashboard.

Annotation query

The following options control the source query for the list of annotations.

Query Filter

Use the query filter to create a list of annotations from all dashboards in your organization or the current dashboard in which this panel is located. It has the following options:

- All dashboards - List annotations from all dashboards in the current organization.
- This dashboard - Limit the list to the annotations on the current dashboard.

Time Range

Use the time range option to specify whether the list should be limited to the current time range. It has the following options:

- None - no time range limit for the annotations query.

- This dashboard - Limit the list to the time range of the dashboard where the annotation list panel is available.

Tags

Use the tags option to filter the annotations by tags. You can add multiple tags in order to refine the list.

> **Note**
>
> Optionally, leave the tag list empty and filter on the fly by selecting tags that are listed as part of the results on the panel itself.

Limit

Use the limit option to limit the number of results returned.

Display

These options control additional metadata included in the annotations panel display.

Show user

Use this option to show or hide which user created the annotation.

Show time

Use this option to show or hide the time the annotation creation time.

Show Tags

Use this option to show or hide the tags associated with an annotation. *NB*: You can use the tags to live-filter the annotation list on the panel itself.

Link behavior

Link target

Use this option to chose how to view the annotated data. It has the following options.

- Panel - This option will take you directly to a full-screen view of the panel with the corresponding annotation

- Dashboard - This option will focus the annotation in the context of a complete dashboard

Time before

Use this option to set the time range before the annotation. Use duration string values like “1h” = 1 hour, “10m” = 10 minutes, etc.

Time after

Use this option to set the time range after the annotation.

Bar chart panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This panel visualization allows you to graph categorical data.

Supported data formats

Only one data frame is supported and it needs to have at least one string field that will be used as the category for an X or Y axis and one or more numerical fields. The following is an example of data formats:

Browser	Market share
Chrome	50
Internet Explorer	17.5

If you have more than one numerical field, the panel shows grouped bars.

Visualizing time series or multiple result sets

If you have multiple time series or tables, you first need to join them using a join, or reduce transform. For example, if you have multiple time series and you want to compare their last and max value, add the **Reduce** transform and specify **Max** and **Last** as options under **Calculations**.

Bar chart options

Use these options to refine your visualizations:

Orientation

- **Auto** – Grafana decides the bar orientation based on the panel dimensions.
- **Horizontal** – Makes the X axis the category axis.
- **Vertical** – Makes the Y axis the category axis.

X-axis tick label maximum length sets the maximum length of bar chart labels. Labels longer than the maximum length are truncated (with ellipses).

Bar labels minimum spacing sets the minimum spacing between bar labels.

Show values

Controls whether values are shown on top of or to the left of bars.

- **Auto** – Values are shown if there is space.
- **Always** – Always show values.
- **Never** – Never show values.

Stacking

Controls bar chart stacking.

- **Off** – Bars will not be stacked.
- **Normal** – Bars will be stacked on top of each other.
- **Percent** – Bars will be stacked on top of each other, and the height of each bar is the percentage of the total height of the stack (all bar stacks will be the same height, adding up to 100 percent).

Group width controls the width of groups. 0=min and 1=max width.

Bar width controls the width of bars. 0=min and 1=max width.

Bar radius controls the radius of the bars, 0 = minimum and 0.5 = maximum radius.

Highlight full area on hover controls if the surrounding area of the bar is highlighted when you hover over the bar with a pointer.

Line width controls line width of the bars.

Fill opacity controls the fill opacity of the bars.

Gradient mode sets the mode of the gradient fill. Fill gradient is based on the line color. To change the color, use the standard color scheme field option. Gradient appearance is influenced by the **Fill opacity** setting.

- **None** – no gradient fill, this is the default setting.
- **Opacity** – Transparency of the gradient is calculated based on the values on the y-axis. Opacity of the fill is increasing with the values on the Y-axis.
- **Hue** – Gradient color is generated based on the hue of the line color.

Tooltip mode – When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.

- **Single** – The hover tooltip shows only a single series, the one that you are hovering over on the visualization.
- **All** – The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden** – Do not display the tooltip when you interact with the visualization.

> **Note**
>
> You can use an override to hide individual series from the tooltip.

Legend mode – Use these settings to refine how the legend appears in your visualization. For more information, see Configure a legend.

- **List** – Displays the legend as a list. This is a default display mode of the legend.

- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement – Choose where to place the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend calculations – Choose which of the standard calculations to show in the legend. You can have more than one.

Text size – Enter a value to change the size of the text on your bar chart.

Axis – Use the following field settings to refine how your axes display. Some field options will not affect the visualization until you click outside of the field option box you are editing or press Enter.

- **Placement** – Sets the placement of the Y-axis.
- **Auto** – Grafana automatically assigns Y-axis to the series. When there are two or more series with different units, then Grafana assigns the left axis to the first unit and right to the following units.
- **Left** – Display all Y-axes on the left side.
- **Right** – Display all Y-axes on the right side.
- **Hidden** – Hide all Y-axes.
- **Label** – Set a Y-axis text label. If you have more than one Y-axis, then you can assign different labels with an override.
- **Width** – Set a fixed width of the axis. By default, Grafana dynamically calculates the width of an axis.

  By setting the width of the axis, data with different axes types can share the same display proportions. This makes it easier to compare more than one graph's worth of data because the axes are not shifted or stretched within visual proximity of each other.
- **Soft min and soft max** – Set a soft min and soft max option for better control of Y-axis limits. By default, Grafana sets the range for the Y-axis automatically based on the dataset.

  Soft min and soft max settings can prevent blips from turning into mountains when the data is mostly flat, and hard min or max derived from standard min and max field options can prevent intermittent spikes from flattening useful detail by clipping the spikes past a defined point.

You can set standard min/max options to define hard limits of the Y-axis.

Bar gauge

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The bar gauge simplifies your data by reducing every field to a single value. You choose how Grafana calculates the reduction.

This panel can show one or more bar gauges depending on how many series, rows, or columns your query returns.

Value options

Use the following options to refine how your visualization displays the value:

Show – Choose how Grafana displays your data.

Calculate – Show a calculated value based on all rows.

- **Calculation** – Select a reducer function that Grafana will use to reduce many fields to a single value. For a list of available calculations, refer to Calculation types.
- **Fields** – Select the fields display in the panel.

All values – Show a separate stat for every row. If you select this option, then you can also limit the number of rows to display.

- **Limit** – The maximum number of rows to display. Default is 5,000.
- **Fields** – Select the fields display in the panel.

Bar gauge options

Adjust how the bar gauge is displayed.

Orientation – Choose a stacking direction.

- **Auto** – Grafana selects what it thinks is the best orientation.
- **Horizontal** – Bars stretch horizontally, left to right.
- **Vertical** – Bars stretch vertically, bottom to top.

Display mode – Choose a display mode.

- **Gradient** – Threshold levels define a gradient.
- **Retro LCD** – The gauge is split into small cells that are lit or unlit.
- **Basic** – Single color based on the matching threshold.

Show unfilled area – Select this if you want to render the unfilled region of the bars as dark gray. Not applicable to Retro LCD display mode.

Min width

Limit the minimum width of the bar column in the vertical direction.

Automatically show x-axis scrollbar when there is a large amount of data.

Min height

Limit the minimum height of the bar row in the horizontal direction.

Automatically show y-axis scrollbar when there is a large amount of data.

Candlestick panel



This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The Candlestick panel allows you to visualize data that includes a number of consistent dimensions focused on price movement. The Candlestick panel includes an Open-High-Low-Close (OHLC) mode, as well as support for additional dimensions based on time series data.

The Candlestick panel builds upon the foundation of the Time series panel and includes many common configuration settings.

Mode

The mode options allow you to toggle which dimensions are used for the visualization.

- **Candles** limits the panel dimensions to the open, high, low, and close dimensions used by candlestick visualizations.
- **Volume** limits the panel dimension to the volume dimension.
- **Both** is the default behavior for the candlestick panel. It includes both candlestick and volume visualizations.

Candle style

- **Candles** is the default display style and creates candle-style visualizations between the open and close dimensions.
- **OHLC Bars** displays the four core dimensions open, high, low, and close values.

Color strategy

- **Since Open** is the default behavior. This mode will utilize the *Up* color (below) if the intra-period price movement is positive. In other words, if the value on close is greater or equal to the value on open, the *Up* color is used.
- **Since Prior Close** is an alternative display method where the color of the candle is based on the interperiod price movement or change in value. In other words, if the value on open is greater than the previous value on close, the *Up* color is used. If the value on open is lower than the previous value on close, the *Down* color is used. *This option also triggers the hollow candlestick visualization mode*. Hollow candlesticks indicate that the intra-period movement is positive (value is higher on close than on open), filled candlesticks indicate the intra-period change

is negative (value is lower on close than on open). To learn more, see the explanation of the differences.

Up & Down Colors

The **Up color** and **Down color** options select which colors are used when the price movement is up or down. The *Color strategy* above will determine if intra-period or inter-period price movement is used to select the candle or OHLC bar color.

Open, High, Low, Close

The candlestick panel will attempt to map fields to the appropriate dimension. The **Open**, **High**, **Low**, and **Close** options allow you to map your data to these dimensions if the panel is unable to do so.

> **Note**
>
> These values are hidden from the legend.

- **Open** corresponds to the starting value of the given period.
- **High** corresponds to the highest value of the given period.
- **Low** corresponds to the lowest value of the given period.
- **Close** corresponds to the final (end) value of the given period.
- **Volume** corresponds to the sample count in the given period. (e.g. number of trades)

Additional fields

The candlestick panel is based on the time series panel. It can visualize additional data dimensions beyond open, high, low, close, and volume The **Include** and **Ignore** options allow the panel to visualize other included data such as simple moving averages, Bollinger bands and more, using the same styles and configurations available in the Time series panel.

Canvas panel

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Canvas is a new panel that combines the power of Grafana with the flexibility of custom elements. Canvas visualizations are extensible form-built panels that allow you to explicitly place elements within static and dynamic layouts. This empowers you to design custom visualizations and overlay data in ways that aren't possible with standard Grafana panels, all within Grafana's UI. If you've used popular UI and web design tools, then designing Canvas panels will feel very familiar.

Elements

Metric value

The metric value element enables you to easily select the data you want to display on canvas. This element has a unique "edit" mode that can be triggered either through the context menu "Edit" option or by double-clicking. When in edit mode you can select which field data that you want to display.

Text

The text element enables you to easily add text to the canvas. The element also supports an editing mode, triggered via either double-clicking or the edit menu option in the context menu.

Rectangle

The rectangle element enables you to add a basic rectangle to the canvas. Rectangle elements support displaying text (both fixed and field data) as well as can change background color based on data thresholds.

Icon

The icon element enables you to add a supported icon to the canvas. Icons can have their color set based on thresholds or value mappings.

Canvas Editing

Inline editor

Canvas introduces a new editing experience. You can now edit your canvas panel inline while in the context of dashboard mode.

Context menu

The context menu gives you access to common tasks. Supported functionality includes opening and closing the inline editor, duplicating an element, deleting an element, and more.

The context menu is triggered by a right click action over the panel or a given canvas element. When right clicking the panel, you are able to set a background image and easily add elements to the canvas.

When right clicking an element, you are able to edit, delete, and duplicate the element, or modify the element's layer positioning.

Canvas Options

Inline editing

The inline editing toggle enables you to lock or unlock the canvas panel. When turned off the canvas panel becomes *locked*, freezing elements in place and preventing unintended modifications.

Clock panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The clock panel shows the current time or a countdown. It updates every second.

- **Mode** – The default is **time**. If you choose **countdown**, set the **Countdown Deadline** to start the countdown.
- **12 or 24 hour** – The options for showing the time are 12-hour format and 24-hour format.

- **Timezone** – The time zones are supplied by the moment timezone library. The default is the time zone on your computer.
- **Countdown Deadline** – Specify the time and date to count down to, if you have set **Mode** to **countdown**.
- **Countdown End Text** – Specify the text to show when the countdown ends.
- **Date/Time formatting options** – Customize the font size, weight, and date/time formatting. If you are showing a countdown and don't want to see the seconds ticking down, change the time format to `hh:mm` for the 24-hour clock or `h:mm A` for the 12-hour clock. For a complete list of options, see Display.
- **Bg Color** – Select a background color for the clock.

Dashboard list

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The dashboard list visualization allows you to display dynamic links to other dashboards. The list can be configured to use starred dashboards, recently viewed dashboards, a search query, and dashboard tags.

On each dashboard load, this panel queries the dashboard list, always providing the most up-to-date results.

Options

Use these options to refine your visualization.

- **Starred** – Display starred dashboards in alphabetical order.
- **Recently viewed** – Display recently viewed dashboards in alphabetical order.

- **Search** – Display dashboards by search query or tags. You must enter at least one value in **Query** or **Tags**. For the **Query** and **Tags** fields, variable interpolation is supported, for example, `$my_var` or `${my_var}`.
- **Show headings** – The chosen list selection (Starred, Recently viewed, Search) is shown as a heading.
- **Max items** – Sets the maximum number of items to list per section. For example, if you left this at the default value of 10 and displayed Starred and Recently viewed dashboards, then the panel would display up to 20 total dashboards, ten in each section.

Search

These options only apply if the **Search** option is selected.

- **Query** – Enter the query you want to search by. Queries are case-insensitive, and partial values are accepted.
- **Folder** – Select the dashboard folders that you want to display.
- **Tags** – Here is where you enter your tags you want to search by. Existing tags will not appear as you type, and they *are* case sensitive.

> **Note**
>
> When multiple tags and strings appear, the dashboard list displays those matching *all* conditions.

Gauge panel

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Gauge is a single-value visualization that can repeat a gauge for every series, column or row.

Value options

Use the following options to refine how your visualization displays the value:

Show

Choose how Grafana displays your data.

Calculate

Show a calculated value based on all rows.

- **Calculation** – Select a reducer function that Grafana will use to reduce many fields to a single value. For a list of available calculations, refer to Calculation types.
- **Fields** – Select the fields to display in the panel.

All values

Show a separate stat for every row. If you select this option, then you can also limit the number of rows to display.

- **Limit** – The maximum number of rows to display. Default is 5,000.
- **Fields** – Select the fields to display in the panel.

Gauge

Adjust how the gauge is displayed.

- **Show threshold labels** – Controls if threshold values are shown.
- **Show threshold markers** – Controls if a threshold band is shown outside the inner gauge value band.

Text size

Adjust the sizes of the gauge text.

- **Title** – Enter a numeric value for the gauge title size.

- **Value** – Enter a numeric value for the gauge value size.

Geomap panel

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The Geomap panel visualization allows you to view and customize the world map using geospatial data. You can configure various overlay styles and map view settings to easily focus on the important location-based characteristics of the data.

Map View

The map view controls the initial view of the map when the dashboard loads.

Initial View

The initial view configures how the GeoMap panel renders when the panel is first loaded.

- **View** sets the center for the map when the panel first loads.
  - **Fit to data** fits the map view based on the data extents of Map layers and updates when data changes.
    - **Data** option allows selection of extent based on data from “All layers”, a single “Layer”, or the “Last value” from a selected layer.
    - **Layer** can be selected if fitting data from a single “Layer” or the “Last value” of a layer.
    - **Padding** sets padding in relative percent beyond data extent (not available when looking at “Last value” only).
    - **Max Zoom** sets the maximum zoom level when fitting data.
  - **Coordinates** sets the map view based on:
    - **Latitude**

    - **Longitude**
- Default Views are also available including:
    - **(0°, 0°)**
    - **North America**
    - **South America**
    - **Europe**
    - **Africa**
    - **West Asia**
    - **South Asia**
    - **South-East Asia**
    - **East Asia**
    - **Australia**
    - **Oceania**
- **Zoom** sets the initial zoom level.

Map layers

The Geomap visualization supports showing multiple layers. Each layer determines how you visualize geospatial data on top of the base map.

Types

There are three map layer types to choose from in the Geomap visualization.

- Markers layer renders a marker at each data point.
- Heatmap layer visualizes a heatmap of the data.
- GeoJSON layer renders static data from a GeoJSON file.

There are also five layer types that are currently in alpha.

- Night / Day layer (Alpha) renders a night or day region.
- **Icon at last point (alpha)** renders an icon at the last data point.
- **Dynamic GeoJSON (alpha)** styles a GeoJSON file based on query results.
- **Route (alpha)** render data points as a route.

- Photos layer (Alpha) renders a photo at each data point.

Layer Controls

The layer controls allow you to create layers, change their name, reorder and delete layers.

- **Add layer** creates an additional, configurable data layer for the Geomap visualization. When you add a layer, you are prompted to select a layer type. You can change the layer type at any point during panel configuration. See the **Layer Types** section above for details on each layer type.
- The layer controls allow you to rename, delete, and reorder the layers of the panel.
  - **Edit layer name** (pencil icon) renames the layer.
  - **Trash Bin** deletes the layer.
  - **Reorder** (six dots/grab handle) allows you to change the layer order. Data on higher layers will appear above data on lower layers. The panel will update the layer order as you drag and drop to help simplify choosing a layer order.

You can add multiple layers of data to a single Geomap panel in order to create rich, detailed visualizations.

Location

The Geomap panel needs a source of geographical data. This data comes from a database query, and there are four mapping options for your data.

- **Auto** automatically searches for location data. Use this option when your query is based on one of the following names for data fields.
  - geohash: “geohash”
  - latitude: “latitude”, “lat”
  - longitude: “longitude”, “lng”, “lon”
  - lookup: “lookup”
- **Coords** specifies that your query holds coordinate data. You will get prompted to select numeric data fields for latitude and longitude from your database query.
- **Geohash** specifies that your query holds geohash data. You will be prompted to select a string data field for the geohash from your database query.

- **Lookup** specifies that your query holds location name data that needs to be mapped to a value. You will be prompted to select the lookup field from your database query and a gazetteer. The gazetteer is the directory that is used to map your queried data to a geographical point.

Markers layer

The markers layer allows you to display data points as different marker shapes such as circles, squares, triangles, stars, and more.

Markers have many customization options.

- **Marker Color** configures the color of the marker. The default `Single color` keeps all points a single color. There is an alternate option to have multiple colors depending on the data point values and the threshold set at the `Thresholds` section.
- **Marker Size** configures the size of the marker. The default is `Fixed size`, which makes all marker sizes the same regardless of the data points. However, there is also an option to scale the circles to the corresponding data points. `Min` and `Max` marker size has to be set such that the Marker layer can scale within this range.
- **Marker Shape** allows you to choose the shape, icon, or graphic to aid in providing additional visual context to your data. Choose from assets that are included with Grafana such as simple shapes or the Unicon library. You can also specify a URL containing an image asset. The image must be a scalable vector graphic (SVG).
- **Fill opacity** configures the transparency of each marker.

Heatmap layer

The heatmap layer clusters various data points to visualize locations with different densities. To add a heatmap layer:

Click on the dropdown menu under Data Layer and choose `Heatmap`.

Similar to `Markers`, you are prompted with options to determine which data points to visualize and how you want to visualize them.

- **Weight values** configure the intensity of the heatmap clusters. `Fixed value` keeps a constant weight value throughout all data points. This value should be in the range of 0~1. Similar to Markers, there is an alternate option in the dropdown to automatically scale the weight values depending on data values.

- **Radius** configures the size of the heatmap clusters.
- **Blur** configures the amount of blur on each cluster.

GeoJSON layer

The GeoJSON layer allows you to select and load a static GeoJSON file from the filesystem.

- **GeoJSON URL** provides a choice of GeoJSON files that ship with Grafana.
- **Default Style** controls which styles to apply when no rules above match.
  - **Color** configures the color of the default style
  - **Opacity** configures the default opacity
- **Style Rules** apply styles based on feature properties
  - **Rule** allows you to select a *feature*, *condition*, and *value* from the GeoJSON file in order to define a rule. The trash bin icon can be used to delete the current rule.
  - **Color** configures the color of the style for the current rule
  - **Opacity** configures the transparency level for the current rule
- **Add style rule** creates additional style rules.

CARTO layer

A CARTO layer is from CARTO Raster basemaps.

Options

- **Theme**

  Choose a theme, either a **Light** theme, **Dark** theme, or **Auto** theme.
- **Show labels** shows the Country details on top of the map.
- **Opacity** from 0 (transparent) to 1 (opaque)

XYZ tile layer

The XYZ tile layer is a map from a generic tile layer.

> **Note**
>
> For more information about generic tile layers, see Tiled Web Maps, and List of Open Street Map Tile Servers.

Options

- **URL template**

  > **Note**
  >
  > Set a valid tile server url, with {z}/{x}/{y} for example: `https://tile.openstreetmap.org/{z}/{x}/{y}.png`.

- **Attribution** sets the reference string for the layer if displayed in map controls
- **Opacity** from 0 (transparent) to 1 (opaque)

Open Street Map layer

A map from Open Street Map, a collaborative, free geographic world database.

Options

- **Opacity** from 0 (transparent) to 1 (opaque)

ArcGIS layer

An ArcGIS layer is a layer from an ESRI ArcGIS MapServer.

Options

- **Server Instance** to select from the following map types.
  - World Street Map
  - World Imagery
  - World Physical
  - Topographic
  - USA Topographic

- World Ocean
- Custom MapServer (see XYZ for formatting)
  - URL template
  - Attribution
- **Opacity** from 0 (transparent) to 1 (opaque)

Night / Day layer (Alpha)

The Night / Day layer displays night and day regions based on the current time range.

> **Note**
>
> For more information, seeExtensions for OpenLayers - DayNight.

Options

- **Show** toggles time source from panel time range
- **Night region color** picks color for night region
- **Display sun** toggles sun icon
- **Opacity** from 0 (transparent) to 1 (opaque)

Photos layer (Alpha)

The Photos layer renders a photo at each data point.

> **Note**
>
> For more information, see Extensions for OpenLayers - Image Photo Style.

Options

- **Image Source Field**

  Select a string field containing image data in either of the following formats:

  - **Image URLs**

  - **Base64 encoded** image binary (`...`)
- **Kind**

  select the frame style around the images

  - **Square**
  - **Circle**
  - **Anchored**
  - **Folio**
- **Crop** toggle if the images are cropped to fit
- **Shadow** toggle a box shadow behind the images
- **Border** set the border size around images
- **Border color** set the border color around images
- **Radius** set the overall size of images in pixels

Map Controls

The map controls interface contains the following options for map information and tool overlays.

Zoom

This section describes each of the zoom controls.

Show zoom control

Displays zoom controls in the upper left corner.

Mouse wheel zoom

Turns on or off using the mouse wheel for zooming in or out.

Show attribution

Displays attribution for basemap layers on the map.

Show scale

Displays scale information in the bottom left corner.

> **Note**
>
> Displays units in [m]/[km].

Show measure tools

Displays measure tools in the upper right corner. Measurements appear only when this control is open.

- **Click** to start measuring
- **Continue clicking** to continue measurement
- **Double-click** to end measurement

> **Note**
>
> When you change measurement type or units, the previous measurement is removed from the map.
> If the control is closed and then re-opened, the most recent measurement is displayed.
> A measurement can be modified by clicking and dragging on it.

Length

Get the spherical length of a geometry. This length is the sum of the great circle distances between coordinates. For multi-part geometries, the length is the sum of the length of each part. Geometries are assumed to be in 'EPSG:3857'.

You can choose the following units for length measurements:

- **Metric (m/km)**
- **Feet (ft)**
- **Miles (mi)**
- **Nautical miles (nmi)**

Area

Get the spherical area of a geometry. This area is calculated assuming that polygon edges are segments of great circles on a sphere. Geometries are assumed to be in 'EPSG:3857'.

You can choose the following units for area measurements:

- **Square Meters (m^2)**
- **Square Kilometers (km^2)**
- **Square Feet (ft^2)**
- **Square Miles (mi^2)**
- **Acres (acre)**
- **Hectare (ha)**

Show debug

Displays debug information in the upper right corner of the map. This can be useful for debugging or validating a data source.

- **Zoom** displays the current zoom level of the map.
- **Center** displays the current **longitude**, and **latitude** of the map center.

Tooltip

- **None** displays tooltips only when a data point is clicked.
- **Details** displays tooltips when a pointer hovers over a data point.

Graph panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A graph panel can render as a line, a path of dots, or a series of bars. This type of graph is versatile enough to display almost any time-series data.

Data and field options

When using graph visualizations, you can apply the following options:

- Transform data
- Alerts. This is the only type of visualization that allows you to set alerts. For more information, see Alerts in Grafana version 9.
- Configure thresholds

Display options

To refine your visualization, use the following settings:

- **Bars** – Display values as a bar chart.
- **Lines** – Display values as a line graph.
- **Line width** – Specify the width of the line for a series. The default is 1.
- **Staircase** – Draw adjacent points as staircase.
- **Area fill** – Specify the amount of color fill for a series. The default is 1; 0 is none.
- **Fill gradient** – Specify the degree of gradient on the area fill. The default is 0, which is no gradient; 10 is a steep gradient.
- **Points** – Display points for values.
- **Point radius** – Control how large the points are.
- **Alert thresholds** – Display alert thresholds and Regions on the panel.

Stacking and null value

- **Stack** – Each series is stacked on top of another.
- **Percent** – Each series is drawn as a percentage of the total of all series. This option is available when **Stack** is selected.
- **Null value** – Specify how null values are displayed. *This is an important setting.* See the note below.

- **connected** – If there is a gap in the series, meaning one or more null values, the line will skip the gap and connect to the next non-null value.
- **null** If there is a gap in the series, meaning a null value, the line in the graph will be broken and show the gap. This is the default setting.
- **null as zero** – If there is a gap in the series, meaning a null value, it will be displayed as a zero value in the graph panel.

> **Important**
>
> If you are monitoring a server's CPU load and the load reaches 100 percent, the server will lock up, and the agent sending statistics will not be able to collect the load statistic. This leads to a gap in the metrics, and using the default *null* setting means that Amazon Managed Grafana will show the gaps and indicate that something is wrong. If this is set to *connected*, it will be easy to miss this signal.

Hover tooltip

Use these settings to change the appearance of the tooltip that appears when you pause over the graph visualization.

- **Mode** – Determines how many series the hover tooltip shows.
  - **All series** – The hover tooltip shows all series in the graph. In the series list in the tooltip, the Grafana workspace highlights the series that you are pausing on in bold.
  - **Single** – The hover tooltip shows only a single series, the one that you are pausing on in the graph.
- **Sort order** – Sorts the order of series in the hover tooltip if you have selected **All series** mode. When you pause on a graph, Amazon Managed Grafana displays the values associated with the lines. Generally, users are most interested in the highest or lowest values. Sorting these values can make it much easier to find the data that you want.
  - **None** – The order of the series in the tooltip is determined by the sort order in your query. For example, you can sort the series alphabetically by series name.
  - **Increasing** – The series in the hover tooltip are sorted by value and in increasing order, with the lowest value at the top of the list.
  - **Decreasing** – The series in the hover tooltip are sorted by value and in decreasing order, with the highest value at the top of the list.

Series overrides

Series overrides allow a series in a graph panel to be rendered differently from the others. You can customize display options on a per-series basis or by using regex rules. For example, one series can have a thicker line width to make it stand out or be moved to the right Y-axis.

You can add multiple series overrides.

To add a series override

1. Choose **Add series override**.
2. In **Alias or regex**, type or select a series. Choose the field to see a list of available series.

   For example, `/Network.*/` would match two series named `Network out` and `Network in`.
3. Choose **+** and then select a style to apply to the series. You can add multiple styles to each entry.

- **Bars** – Show series as a bar graph.
- **Lines** – Show series as a line graph.
- **Line fill** – Show a line graph with area fill.
- **Fill gradient** – Specify the area fill gradient amount.
- **Line width** – Set line width.
- **Null point mode** – Use this option to ignore null values or replace them with zero. This is important if you want to ignore gaps in your data.
- **Fill below to** – Fill the area between two series.
- **Staircase line** – Show series as a staircase line.
- **Dashes** – Show a line with dashes.
- **Hidden Series** – Hide the series.
- **Dash Length** – Set the length of dashes in the line.
- **Dash Space** – Set the length of the spaces between the dashes in the line.
- **Points** – Show series as separate points.
- **Point Radius** – Set the radius for point rendering.
- **Stack** – Set the stack group for the series.
- **Color** – Set the series color.

- **Y-axis** – Set the series y-axis.
- **Z-index** – Set the series z-index (rendering order). This option is important when you are overlaying different styles, such as bar charts and area charts.
- Transform – Transform value to negative to render below the y-axis.
- **Legend** – Control whether a series is shown in the legend.
- **Hide in tooltip** – Control whether a series is shown in a graph tooltip.

Axes

Use these options to control the display of axes in the visualization.

Left Y/Right Y

Options are identical for both y-axes.

- **Show** – Choose to show or hide the axis.
- **Unit** – Choose the display unit for the y value.
- **Scale** – Choose the scale to use for the y value: **linear**, or **logarithmic**. The default is **linear**.
- **Y-Min** – The minimum y value. The default is **auto**.
- **Y-Max** – The maximum Y value. The default is **auto**.
- **Decimals** – Define how many decimals are displayed for the y value. The default is **auto**.
- **Label** – Specify the y axis label. The default is “",

Y-Axes

- **Align** – Align left and right y-axes by value. The default is unchecked/false.
- **Level** – Enter the value to use for alignment of left and right y-axes, starting from Y=0. The default is 0. This option is available when **Align** is selected.

X-Axis

- **Show** – Choose to show or hide the axis.
- **Mode** – The display mode completely changes the visualization of the graph panel. It’s like three panels in one. The main mode is the time series mode with time on the x-axis. The other two

modes are a basic bar chart mode with series on the x-axis instead of time and a histogram mode.

- **Time** (default) – The x-axis represents time and the data is grouped by time (for example, by hour, or by minute).
- **Series** – The data is grouped by series, and not by time. The y-axis still represents the value.
  - **Value** – This is the aggregation type to use for the values. The default is **total** (summing the values together).
- **Histogram** – This option converts the graph into a histogram. A histogram is a kind of bar chart that groups numbers into ranges, often called buckets or bins. Taller bars show that more data falls in that range.

  For more information about histograms, see Introduction to histograms and heatmaps.
  - **Buckets** – Sets the number of buckets to group the values by. If left empty, Amazon Managed Grafana tries to calculate a suitable number of buckets.
  - **X-Min** – Filters out values from the histogram that are less than this minimum limit.
  - **X-Max** – Filters out values that are greater than this maximum limit.

Legend

Use these settings to refine how the legend appears in your visualization.

Options

- **Show** – Clear to hide the legend. The default is selected (true).
- **As Table** – Select to display the legend in the table. The default is checked (true).
- **To the right** – Select to display the legend to the right.
- Width – Enter the minimum width for the legend in pixels. This option is available when **To the right** is selected.

Values

Additional values can be shown alongside the legend names.

- **Min** – The minimum value returned from the metric query.
- **Max** – The maximum value returned from the metric query.
- **Avg** – The average value returned from the metric query.

- **Current** – The last value returned from the metric query.
- **Total** – The sum of all values returned from the metric query.
- **Decimals** – How many decimals are displayed for legend values and graph hover tooltips.

Amazon Managed Grafana calculates the legend values on the client side. These legend values depend on the type of aggregation or point consolidation that your metric query is using. All the above legend values cannot be correct at the same time.

For example, if you plot a rate such requests/second, which is probably using average as an aggregator, the Total in the legend will not represent the total number of requests. It is just the sum of all data points received by Amazon Managed Grafana.

Hide series

Hide series when all values of a series from a metric query are of a specific value.

- **With only nulls** – Value=null (default unchecked)
- **With only zeroes** – Value=zero (default unchecked)

Time regions

You can highlight specific time regions on the graph to make it easier to see, for example, weekends, business hours, and off-work hours. All configured time regions refer to UTC time.

Heatmap panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The Heatmap panel visualization allows you to view histograms over time. For more information about histograms, refer to Introduction to histograms and heatmaps.

Calculate from data

This setting determines if the data is already a calculated heatmap (from the data source/ transformer), or one that should be calculated in the panel.

X Bucket

This setting determines how the X-axis is split into buckets. You can specify a time interval in the **Size** input. For example, a time range of 1h makes the cells 1-hour wide on the X-axis.

Y Bucket

This setting determines how the Y-axis is split into buckets.

Y Bucket scale

Select one of the following Y-axis value scales:

- **linear** – Linear scale.
- **log (base 2)** – Logarithmic scale with base 2.
- **log (base 10)** – Logarithmic scale with base 10.

Y Axes

Defines how the Y axis is displayed

Placement

- **Left** – On the left
- **Right** – On the right
- **Hidden** – Hidden

Unit

Unit configuration

Decimals

This setting determines decimal configuration.

Min/Max value

This setting configures the axis range.

Reverse

When selected, the axis appears in reverse order.

Colors

The color spectrum controls the mapping between value count (in each bucket) and the color assigned to each bucket. The leftmost color on the spectrum represents the minimum count and the color on the right most side represents the maximum count. Some color schemes are automatically inverted when using the light theme.

You can also change the color mode to Opacity. In this case, the color will not change but the amount of opacity will change with the bucket count

- **Mode**
  - **Scheme** – Bucket value represented by cell color.
    - **Scheme** – If the mode is **scheme**, then select a color scheme.
  - **opacity** – Bucket value represented by cell opacity. Opaque cell means maximum value.
    - **Color** – Cell base color.
    - **Scale** – Scale for mapping bucket values to the opacity.
      - **linear** – Linear scale. Bucket value maps linearly to the opacity.
      - **sqrt** – Power scale. Cell opacity calculated as `value ^ k`, where k is a configured **Exponent** value. If exponent is less than 1, you will get a logarithmic scale. If exponent is greater than 1, you will get an exponential scale. In case of 1, scale will be the same as linear.
    - **Exponent** – value of the exponent, greater than `0`.

Start/end color from value

By default, Grafana calculates cell colors based on minimum and maximum bucket values. With Min and Max you can overwrite those values. Consider a bucket value as a Z-axis and Min and Max as Z-Min and Z-Max, respectively.

- **Start** – Minimum value using for cell color calculation. If the bucket value is less than Min, then it is mapped to the “minimum” color. The series min value is the default value.

- **End** – Maximum value using for cell color calculation. If the bucket value is greater than Max, then it is mapped to the “maximum” color. The series max value is the default value.

Cell display

Use cell display settings to refine the visualization of the cells in your heatmap.

Additional display options

Tooltip

- **Show tooltip** – Show heatmap tooltip.
- **Show Histogram** – Show a Y-axis histogram on the tooltip. A histogram represents the distribution of the bucket values for a specific timestamp.

Legend

Choose whether you want to display the heatmap legend on the visualization.

Exemplars

Set the color used to show exemplar data.

Histogram panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The histogram visualization calculates the distribution of values and presents them as a bar chart. The Y-axis and the height of each bar represent the count of values that fall into each bracket while the X-axis represents the value range.

Histogram visualization supports time series and any table results with one or more numerical fields.

Supported formats

Histogram visualization supports time series and any table results with one or more numerical fields.

Display options

Use these options to refine your visualizations:

Bucket size

The size of the buckets. Leave this empty for automatic bucket sizing (~10% of the full range).

Bucket offset

If the first bucket should not start at zero. A non-zero offset shifts the aggregation window. For example, 5-sized buckets that are 0–5, 5–10, 10–15 with a default 0 offset would become 2–7, 7–12, 12–17 with an offset of 2; offsets of 0, 5, or 10, in this case, would effectively do nothing. Typically, this option would be used with an explicitly defined bucket size rather than automatic. For this setting to affect, the offset amount should be greater than 0 and less than the bucket size; values outside this range will have the same effect as values within this range.

Combine series

This will merge all series and fields into a combined histogram.

Line width controls line width of the bars.

Fill opacity controls the fill opacity of the bars.

Gradient mode sets the mode of the gradient fill. Fill gradient is based on the line color. To change the color, use the standard color scheme field option. Gradient appearance is influenced by the Fill opacity setting.

- **None** – No gradient fill, this is the default setting.
- **Opacity** – Transparency of the gradient is calculated based on the values on the Y-axis. Opacity of the fill is increasing with the values on the Y-axis.

- **Hue** – Gradient color is generated based on the hue of the line color.

Tooltip mode When you hover your cursor over the graph, Grafana can display tooltips. Choose how tooltips behave:

- **Single** – The hover tooltip shows only the series that you are hovering over.
- **All** – The hover tooltip shows all the series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden** – Do not display the tooltip.

> **Note**
>
> Use an override to hide individual series from the tooltip.

Legend options

When the legend option is enabled, it can show either the value mappings or the threshold brackets. To show the value mappings in the legend, it's important that the Color scheme option under standard options is set to Single color or Classic palette. To see the threshold brackets in the legend, set the Color scheme to From thresholds.

Legend mode Use these settings to refine how the legend appears in your visualization.

- **List** – Displays the legend as a list. This is a default display mode of the legend.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement Choose where to place the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend Values

Choose which of the standard calculations to show in the legend. You can have more than one. For more information, see Calculation types.

Legend calculations

Choose which calculations to show in the legend. You can select more than one.

Logs panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The logs panel visualization shows log lines from data sources that support logs, such as Elastic, Influx, and Loki. Typically, you would use this panel next to a graph panel to display the log output of a related process.

The logs panel shows the result of queries that were entered on the **Query** tab. The results of multiple queries are merged and sorted by time. You can scroll inside the panel if the data source returns more lines than can be displayed.

To limit the number of lines rendered, you can use the **Max data points** setting in the **Query options**. If it is not set, the data source will usually enforce a default limit.

Log level

For logs where a **level** label is specified, we use the value of the label to determine the log level and update color accordingly. If the log doesn't have a level label specified, we try to find out if its content matches any of the supported expressions (see below for more information). The log level is always determined by the first match. In case Grafana is not able to determine a log level, it will be visualized with **unknown** log level. For more information, see Logs visualization.

Log details

Each log row has an extendable area with its labels and detected fields, for more robust interaction. Each field or label has a stats icon to display statistics in relation to all displayed logs.

Data links

By using data links, you can turn any part of a log message into an internal or external link. The created link is visible as a button in the **Links** section inside the **Log details** view.

Display options

Use the following settings to refine your visualization:

- **Time** – Show or hide the time column. This is the timestamp associated with the log line as reported from the data source.
- **Unique labels** – Show or hide the unique labels column, which shows only non-common labels.
- **Common labels** – Show or hide the common labels
- **Wrap lines** – Toggle line wrapping.
- **Prettify JSON** – Set this to `true` to pretty print all JSON logs. This setting does not affect logs in any format other than JSON.
- **Enable log details** – Toggle option to see the log details view for each log row. The default setting is `true`.
- **Order** – Display results in descending or ascending time order. The default is **Descending**, showing the newest logs first. Set to **Ascending** to show the oldest log lines first.

News panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This panel displays an RSS feed. By default, it displays articles from the Grafana Labs blog.

Enter the URL of an RSS in the **Display** section. This panel type does not accept any other queries.

Note

RSS feeds are loaded by the Grafana front end without a proxy. As a result, only RSS feeds that are configured with the appropriate CORS headers will load. If the RSS feed you're trying to display fails to load, consider re-hosting the RSS feed or creating your own proxy.

Node graph panel

This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The node graph panel visualizes directed graphs or networks. It uses directed force layout to effectively position the nodes so it can help with displaying complex infrastructure maps, hierarchies, or run diagrams.

Data requirements

The node graph panel requires a specific shape of the data to be able to display its nodes and edges. Not every data source or query can be visualized in this panel.

The Node graph visualization consists of *nodes* and *edges*.

- A *node* is displayed as a circle. A node might represent an application, a service, or anything else that is relevant from an application perspective.
- An *edge* is displayed as a line that connects two nodes. The connection might be a request, an operation, or some other relationship between the two nodes.

Both nodes and edges can have associated metadata or statistics. The data source defines what information and values is shown, so different data sources can show different type of values or not show some values.

Nodes

Usually, nodes show two statistical values inside the node and two identifiers just below the node, usually name and type. Nodes can also show another set of values as a color circle around the node, with sections of different color represents different values that should add up to 1. For example, you can have the percentage of errors represented by red portion of the circle.

Additional details can be displayed in a context menu, which is displayed when you choose the node. There also can be additional links in the context menu that can target either other parts of the Grafana workspace or any external link.

> **Note**
>
> Node graph can show only 1,500 nodes. If this limit is crossed, a warning is visible in the upper right corner, and some nodes will be hidden. You can expand hidden parts of the graph by clicking on the **Hidden nodes** markers in the graph.

Edges

Edges can also show statistics when you hover over the edge. Similar to nodes, you can open a context menu with additional details and links by choosing the edge.

The first data source supporting this visualization is the AWS X-Ray data source for its service map feature. For more information, see Connect to an AWS X-Ray data source.

Navigating the node graph

You can pan within the node graph by choosing outside of any node or edge and dragging the pointer.

You can zoom by using the buttons on the upper left corner of the node graph.

Hidden nodes

The number of nodes shown at a given time is limited to maintain a reasonable performance. Nodes that are outside this limit are hidden behind clickable markers that show an approximate number of hidden nodes that are connected to that edge. You can choose the marker to expand the graph around that node.

Grid view

You can switch to the grid view to have a better overview of the most interesting nodes in the graph. Grid view shows nodes in a grid without edges and can be sorted by stats shown inside the node or by stats represented by the a colored border of the nodes.

To sort the nodes, choose the stats inside the legend. The marker next to the stat name shows which stat is currently used for sorting and sorting direction.

Choose the node and then the **Show in Graph layout** option to switch back to graph layout with focus on the selected node, to show it in context of the full graph.

Data API

This visualization needs a specific shape of the data to be returned from the data source in order to correctly display it.

Node Graph at minimum requires a data frame describing the edges of the graph. By default, node graph will compute the nodes and any stats based on this data frame. Optionally a second data frame describing the nodes can be sent in case there is need to show more node specific metadata. You have to set `frame.meta.preferredVisualisationType = 'nodeGraph'` on both data frames or name them `nodes` and `edges` respectively for the node graph to render.

Edges data from structure

Required fields:

Field name	Type	Description
id	string	Unique identifier of the edge.
source	string	Id of the source node.
target	string	Id of the target.

Optional fields:

Field name	Type	Description
mainstat	string/number	First stat shown in the overlay when hovering over the edge. It can be a string showing

Field name	Type	Description
		the value as is or it can be a number. If it is a number, any unit associated with that field is also shown
secondarystat	string/number	Same as mainStat, but shown right under it.
detail__*	string/number	Any field prefixed with `detail__` will be shown in the header of context menu when clicked on the edge. Use `config.displayName` for a more human readable label.

Nodes data from structure

Required fields:

Field name	Type	Description
id	string	Unique identifier of the node. This ID is referenced by edge in its source and target field.

Optional fields:

Field name	Type	Description
title	string	Name of the node visible in just under the node.
subtitle	string	Additional, name, type or other identifier shown under the title.

Field name	Type	Description
mainstat	string/number	First stat shown inside the node itself. It can either be a string showing the value as is or a number. If it is a number, any unit associated with that field is also shown.
secondarystat	string/number	Same as mainStat, but shown under it inside the node.
arc__*	number	Any field prefixed with `arc__` will be used to create the color circle around the node. All values in these fields should add up to 1. You can specify color using `config.color.fixedColor` .
detail__*	string/number	Any field prefixed with `detail__` will be shown in the header of context menu when clicked on the node. Use `config.displayName` for more human readable label.

Pie chart panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The pie chart displays reduced series, or values in a series, from one or more queries, as they relate to each other, in the form of slices of a pie. The arc length, area and central angle of a slice are all proportional to the slices value, as it relates to the sum of all values. This type of chart is best used when you want a quick comparison of a small set of values in an aesthetically pleasing form.

Value options

Use the following options to refine the value in your visualization.

Show

Choose how much information to show.

- **Calculate** – Reduces each value to a single value per series.
- **All values** – Displays every value from a single series.

Calculation

Select a calculation to reduce each series when Calculate has been selected. For information about available calculations, refer to Calculation types.

Limit

When displaying every value from a single series, this limits the number of values displayed.

Fields

Select which field or fields to display in the visualization. Each field name is available on the list, or you can select one of the following options:

- **Numeric fields** – All fields with numerical values.
- **All fields** – All fields that are not removed by transformations.
- **Time** – All fields with time values.

Pie chart options

Use these options to refine how your visualization looks.

Pie chart type

Select the pie chart display style. Can be either:

- **Pie** – A standard pie chart
- **Donut** – A pie chart with a hole in the middle

Labels

Select labels to display on the pie chart. You can select more than one.

- **Name** – The series or field name.
- **Percent** – The percentage of the whole.
- **Value** – The raw numerical value.

Labels are displayed in white over the body of the chart by default. You can select darker chart colors to make them more visible. Long names or numbers might be clipped.

Tooltip mode

When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.

- **Single** – The hover tooltip shows only a single series, the one that you are hovering over on the visualization.
- **All** – The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden** – Do not display the tooltip when you interact with the visualization.

Use an override to hide individual series from the tooltip.

Legend mode

Use these settings to define how the legend appears in your visualization. For more information about the legend, refer to Configure a legend.

- **List** – Displays the legend as a list. This is a default display mode of the legend.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement

Choose where to display the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend values

Choose which of the standard calculations to show in the legend. You can have more than one.

Select values to display in the legend. You can select more than one.

- **Percent** – The percentage of the whole.
- **Value** – The raw numerical value.

For more information about the legend, refer to Configure a legend.

Plotly panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The Plotly panel renders charts using Plotly, an open source javascript graphing library.

The **Data**, **Layout** and **Config** fields match the common parameters described in the Plotly documentation. They must be in JSON format.

Data provided by the datasource can be transformed via a user-defined script before to be injected in the Plotly chart. The script includes 2 arguments.

- `data` – Data returned by the data source.
- `variables` – An object that contains Grafana variables in the current dashboard (user variables and these few global variables: `__from`, `__to`, `__interval`, and `__interval_ms`).

The script must return an object with one or more of the following properties: `data`, `layout`, `config` and `frames`. The following is an example.

```
let x  = data.series[0].fields[0].values.buffer
let y  = data.series[0].fields[1].values.buffer
let serie = {
    x : x,
    y : y,
    name : variables.project //where project is the name of a Grafana's variable
}

return {
    data : [serie],
    config : {
    displayModeBar: false
    }
}
```

Object returned by the script and JSON provided in the *Data*, *Layout* and *Config* fields will be merged (deep merge).

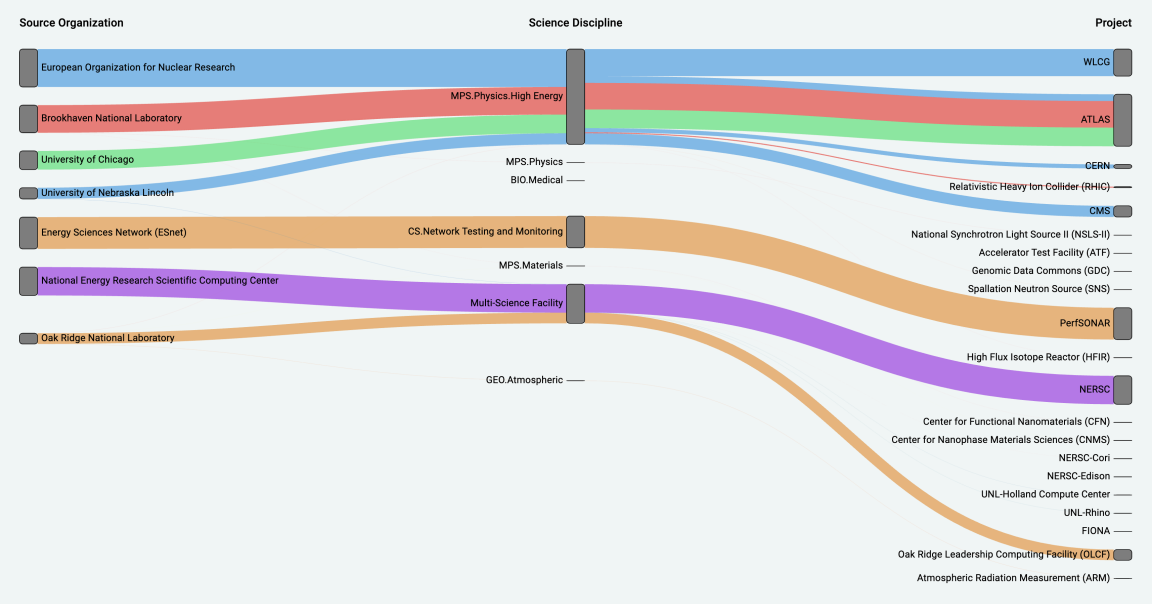
If no script is provided, the panel will use only *Data*, *Layout* and *Config* fields.

Sankey panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The Sankey panel shows Sankey diagrams, which are good for visualizing flow data, with the width of the flow being proportional to the selected metric. The following image shows a Sankey diagram with two groups of source and destinations.



How it works

The sankey panel requires at least 2 columns of data, a source and destination for the flows. Your query should group your data into at least two groups. The panel will draw links from the first column of data points, to the last in order of the query. The thickness of the links will be proportionate to the value as assigned by the metric in the query.

Customizing

- **Links** – There are currently two options for link color: multi or single. It is multi-colored by default. To choose a single color for the links, toggle the **Single Link color only** option and choose your color from Grafana's color picker.
- **Nodes** – You can change the color of the rectangular nodes by changing the **Node color** option
- **Node Width** – The width of the nodes can be adjusted with the **Node Width** slider or entering a number in the input box. This number must be an integer.
- **Node Padding** – The vertical padding between nodes can be adjusted with the **Node Padding** slider or entering a number in the input box. This number must be an integer. If your links are too thin, try adjusting this number

- **Headers** – The column headers can be changed by using a **Display Name** override in the editor panel. They will be the same color you choose for **Text color**
- **Sankey Layout** – The layout of the Sankey links can be adjusted slightly using the **Layout iteration** slider. This number must be an integer and is the number of relaxation iterations used to generate the layout.

Scatter panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The scatter panel shows an X/Y scatter plot for table data with a simpler interface than other graphing panels. Unlike the graph panel, the scatter panel does not require the data to be in a time series. The scatter panel requires a table formatted dataset with two or more numeric columns of data.

One of these can be assigned to the X axis. One or more can be assigned to a series of Y axis values and the resulting data plotted as a series of dots. Each series can optionally also show a regression line using one of a number of statistical best fits.

Creating a scatter panel

The following procedure describes how to create a scatter plot using the scatter panel. For this example, we will assume that there is data, as in the following table called `HEIGHT` with three columns of numerical values, `Age`, `Boys`, and `Girls`, showing the average height of boys and girls by age.

Age	Boy's Height	Girl's Height
5	109.7	109.5

Age	Boy's Height	Girl's Height
6	115.6	115.4
7	121.1	120.8
8	126.3	126
9	131.3	131.3
10	136.2	137.1
11	141.2	143.2
12	147	148.7
13	153.6	152.6
14	159.9	155.1
15	164.4	156.7
16	167.3	157.6
17	169	158
18	170	158.3
19	170.8	158.6

To create a scatter plot with the scatter panel

1. In your Grafana dashboard, choose **Add Panel**.
2. For the Query, write a query that will return the data needed. In this case, you would use a query such as `SELECT * FROM HEIGHT`.
3. Select the **Scatter** visualization.

This will create a scatter plot, using the first column as the X axis, and the other numeric columns as Y axes.

Configuration options

The scatter panel provides the following four custom configuration options.

- **X Axis** – You can choose which field to use as the X axis, as well as extents and title and display information for the axis.
- **Y Axis** – You can choose which fields to display on the Y axis, including display options for each field, and extents and title information for the axis. You can also choose to display a regression line for each field. See the following information for more details on regression line configuration.
- **Legend** – You can turn a legend for the panel on or off, as well as choose the size of the text in the legend.
- **Display** – You can set other options for the chart, including grid color and border style.

Regression line configuration

Each Y axis dataset can display a line, in addition to the individual dots. There are five options for the line type.

- **None** – Do not display a regression line.
- **Simple** – Display a regression line that connects the dataset points.
- **Linear** – Display a straight line, using the least-squares, best-fit method.
- **Exponential** – Display an exponential best-fit regression line.
- **Power** – Display a power best-fit regression line.

Stat panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The Stat panel visualization shows a one large stat value with an optional graph sparkline. You can control the background or value color using thresholds.

By default, the Stat panel displays one of the following:

- Just the value for a single series or field.
- Both the value and name for multiple series or fields.

You can use the **Text mode** to control whether the text is displayed or not.

Automatic layout adjustment

The panel automatically adjusts the layout depending on available width and height in the dashboard. It automatically hides the graph (sparkline) if the panel becomes too small.

Value options

Use the following options to refine how your visualization displays the value:

Show

Choose how Grafana displays your data.

Calculate

Show a calculated value based on all rows.

- **Calculation** – Select a reducer function that Grafana will use to reduce many fields to a single value. For a list of available calculations, see standard calculations.
- **Fields** – Select the fields display in the panel.

All values

Show a separate stat for every row. If you select this option, then you can also limit the number of rows to display.

- **Limit** – The maximum number of rows to display. Default is 5,000.
- **Fields** – Select the fields display in the panel.

Stat styles

Style your visualization.

Orientation

Choose a stacking direction.

- **Auto** – Grafana selects what it thinks is the best orientation.
- **Horizontal** – Bars stretch horizontally, left to right.
- **Vertical** – Bars stretch vertically, top to bottom.

Text mode

You can use the Text mode option to control what text the panel renders. If the value is not important, only the name and color is, then change the **Text mode** to **Name**. The value will still be used to determine color and is displayed in a tooltip.

- **Auto** – If the data contains multiple series or fields, show both name and value.
- **Value** – Show only value, never name. Name is displayed in the hover tooltip instead.
- **Value and name** – Always show value and name.
- **Name** – Show name instead of value. Value is displayed in the hover tooltip.
- **None** – Show nothing (empty). Name and value are displayed in the hover tooltip.

Color mode

Select a color mode.

- **Value** – Colors only the value and graph area.
- **Background** – Colors the background as well.

Graph mode

Select a graph and sparkline mode.

- **None** – Hides the graph and only shows the value.
- **Area** – Shows the area graph below the value. This requires that your query returns a time column.

Text alignment

Choose an alignment mode.

- **Auto** – If only a single value is shown (no repeat), then the value is centered. If multiple series or rows are shown, then the value is left-aligned.
- **Center** – Stat value is centered.

Text size

Adjust the sizes of the gauge text.

- **Title** – Enter a numeric value for the gauge title size.
- **Value** – Enter a numeric value for the gauge value size.

State timeline panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The state timeline panel visualization shows discrete state changes over time. Each field or series is rendered as its unique horizontal band. State regions can either be rendered with or without values. This panel works well with string or boolean states but can also be used with time series. When used with time series, the thresholds are used to turn the numerical values into discrete state regions.

State timeline options

Use these options to refine your visualizations:

Merge equal consecutive values

Controls whether Grafana merges identical values if they are next to each other.

Show values

Controls whether values are rendered inside the state regions. Auto will render values if there is sufficient space.

Align values

Controls value alignment inside state regions.

Row height

Controls space between rows. 1 = no space = 0.5 = 50% space.

Line width

Controls the line width of state regions.

Fill opacity

Controls the opacity of state regions.

Value mappings

To assign colors to boolean or string values, use Configure value mappings.

Time series data with thresholds

The panel can be used with time series data as well. In this case, the thresholds are used to turn the time series into discrete colored state regions.

Legend options

When the legend option is enabled, it can show either the value mappings or the threshold brackets. To show the value mappings in the legend, it's important that the Color scheme option under Standard options is set to Single color or Classic palette. To see the threshold brackets in the legend, set the Color scheme to From thresholds.

Legend mode Use these settings to refine how the legend appears in your visualization.

- **List** – Displays the legend as a list. This is a default display mode of the legend.
- **Table** – Displays the legend as a table.

- **Hidden** – Hides the legend.

Legend placement Choose where to place the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend values

Choose which of the standard calculations to show in the legend. You can have more than one.

Status history panel

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The Status history visualization shows periodic states over time. Each field or series is rendered as a horizontal row. Boxes are rendered and centered around each value.

Status history visualization works with string, boolean, and numerical fields or time series. A time field is required. You can use value mappings to color strings or assign text values to numerical ranges.

Display options

Use these options to refine your visualizations:

Show values

Controls whether values are rendered inside the value boxes. Auto will render values if there is sufficient space.

Column width controls the width of boxes. 1=max and 0=Min width.

Line width controls line width of state regions.

Fill opacity controls the fill opacity of state regions.

Value mappings

To assign colors to boolean or string values, use Configure value mappings.

Time series data with thresholds

The panel can be used with time series data as well. In this case, the thresholds are used to color the boxes. You can also use gradient color schemes to color values.

Legend options

When the legend option is enabled, it can show either the value mappings or the threshold brackets. To show the value mappings in the legend, it's important that the Color scheme option under Standard options is set to Single color or Classic palette. To see the threshold brackets in the legend set the Color scheme to From thresholds.

Legend mode Use these settings to refine how the legend appears in your visualization.

- **List** – Displays the legend as a list. This is a default display mode of the legend.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement Choose where to place the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend values

Choose which of the standard calculations to show in the legend. You can have more than one.

Table panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The table panel visualization is very flexible, supporting multiple modes for time series and for tables, annotation, and raw JSON data. This panel also provides date formatting, value formatting, and coloring options.

Sort column

Click a column title to change the sort order from default to descending to ascending. Each time you click, the sort order changes to the next option in the cycle. You can only sort by one column at a time.

Table options

Show header

Show or hide column names imported from your data source.

Column width

By default, Grafana automatically calculates the column width based on the table size and the minimum column width. This field option can override the setting and define the width for all columns in pixels.

For example, if you enter `100` in the field, then when you click outside the field, all the columns will be set to 100 pixels wide.

Minimum column width

By default, the minimum width of the table column is 150 pixels. This field option can override that default and will define the new minimum column width for the table panel in pixels.

For example, if you enter 75 in the field, then when you click outside the field, all the columns will scale to no smaller than 75 pixels wide.

For small-screen devices, such as smartphones or tablets, reduce the default `150` pixel value to `50` to allow table based panels to render correctly in dashboards.

Column alignment

Choose how Grafana should align cell contents:

- Auto (default)
- Left
- Center
- Right

Cell type

By default, Grafana automatically chooses display settings. You can override the settings by choosing one of the following options to set the default for all fields. Additional configuration is available for some cell types.

> **Note**
>
> If you set these in the **Field** tab, then the type will apply to all fields, including the time field. You can set them in the **Override** tab to apply the change to one or more fields.

Color text

If thresholds are set, then the field text is displayed in the appropriate threshold color.

Color background (gradient or solid)

If thresholds are set, then the field background is displayed in the appropriate threshold color.

Gauge

Cells can be displayed as a graphical gauge, with several different presentation types.

Basic

The basic mode will show a simple gauge with the threshold levels defining the color of gauge.

Gradient

The threshold levels define a gradient.

LCD

The gauge is split up in small cells that are lit or unlit.

JSON view

Shows value formatted as code. If a value is an object the JSON view allowing browsing the JSON object will appear on hover.

Cell value inspect

Enables value inspection from table cell. The raw value is presented in a modal window.

> **Note**
>
> Cell value inspection is only available when cell display mode is set to Auto, Color text, Color background or JSON View.

Column filter

You can temporarily change how column data is displayed. For example, you can order values from highest to lowest or hide specific values. For more information, refer to Filter table columns, below.

Pagination

Use this option to enable or disable pagination. It is a front-end option that does not affect queries. When enabled, the page size automatically adjusts to the height of the table.

Filter table columns

If you turn on the **Column filter**, then you can filter table options.

To turn on column filtering

1. In Grafana, navigate to the dashboard with the table with the columns that you want to filter.
2. On the table panel you want to filter, open the panel editor.
3. Choose the **Field** tab.
4. In **Table** options, turn on the **Column filter** option.

A filter icon appears next to each column title.

Filter column values

To filter column values, choose the filter (funnel) icon next to a column title. Grafana displays the filter options for that column.

Choose the check box next to the values that you want to display. Enter text in the search field at the top to show those values in the display so that you can select them rather than scroll to find them.

Clear column filters

Columns with filters applied have a blue funnel displayed next to the title.

To remove the filter, choose the blue funnel icon and then select Clear filter.

Table footer

You can use the table footer to show calculations on fields.

After you enable the table footer, you can select the **Calculation**, and then the **Fields** that you want to calculate.

The system applies the calculation to all numeric fields if you do not select a field.

Count rows

If you want to show the number of rows in the dataset instead of the number of values in the selected fields, select the **Count** calculation and enable **Count rows**.

Text panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The text panel enables you to directly include text or HTML in your dashboards. This can be used to add contextual information and descriptions or embed complex HTML.

Mode

Mode determines how embedded content appears. It has the following options

- **Markdown** – This option formats the content as markdown.
- **HTML** – This setting renders the content as sanitized HTML.
- **Code** – This setting renders content inside a read-only code editor. Select an appropriate language to apply syntax highlighting to the embedded text.

Variables

Variables in the content will be expanded for display.

Time series panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The time series panel can render a time series as a line, a path of dots, or a series of bars. This type of graph is versatile enough to display almost any time-series data.

Note

You can migrate Graph panel visualizations to Time series visualizations. To migrate, on the **Panel** tab, choose **Time series visualization**. Grafana transfers all applicable settings.

Topics

- Tooltip options
- Legend options
- Graph style options

- Axis options
- Color options

Tooltip options

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

When you hover your cursor over the graph, Grafana can display tooltips. Choose how tooltips behave:

- **Single** – The hover tooltip shows only the series that you are hovering over.
- **All** – The hover tooltip shows all the series in the graph. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden** – Do not display the tooltip.

Legend options

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Legend mode – Choose how the legend appears.

- **List** – Displays the legend as a list. This is the default.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement – Choose where to display the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend calculations

Choose which calculations to show in the legend. For more information, see Calculation types.

Graph style options

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Graph style

Use this option to define how to display your time series data. You can use overrides to combine multiple styles in the same graph. There are three style options. Some of the other style options only apply to certain graph styles.

- **Lines** – Display the time series as a line on a graph.
- **Bars** – Display the time series as a series of bars on a graph, one for each data point.
- **Points** – Display the time series as dots on a graph, one for each data point.

Bar alignment

For bar graphs, , sets the position of the bar, relative to where the point would be drawn on the graph. Because a bar has a width, it can be placed before, after, or centered on the point. The choices for this option are **Before**, **Center**, or **After**.

Line width

Sets the thickness of the line for Line graphs, or the thickness of the outline for each bar in a bar graph.

Fill opacity

Sets the opacity of a fill color. Fills are used, for example, to show the area under the line in a line graph, or as the color of bars in a bar graph.

Gradient mode

Gradient mode specifies the gradient fill, which is based on the series color. To change the color, use the standard color scheme field option. For more information, see Color scheme. The gradient mode options are:

- **None** – No gradient fill.
- **Opacity** – An opacity gradient where the opacity of the fill increases as the Y-axis values increase.
- **Hue** – A gradient that is based on the hue of the series color.
- **Scheme** – A color gradient defined by your color scheme. This setting can be used by the fill and the line. For more information, see Color options.

The gradient appearance is also modified by the **Fill opacity** setting.

Show points

You can configure your visualization to add points to line or bar graphs. You can choose **Always**, **Never**, or **Auto**. When using **Auto**, Grafana determines whether to show points based on the density of the data. If the density of the data is low enough, points are shown.

Point size

Sets the size of drawn points, from 1 to 40 pixels in diameter.

Line interpolation

Choose how Grafana interpolates the series line. The choices are **Linear**, **Smooth**, **Step before**, and **Step after**.

Line style

Set the style of the line. To change the color, use the standard color scheme field option.

Line style appearance is influenced by the settings for **Line width** and **Fill opacity**.

The choices for line style are **Solid**, **Dash**, and **Dots**.

Connect null values

Choose how null values (gaps in the data) appear on the graph. Null values can be connected to form a continuous line or, optionally, set a threshold above which gaps in the data should no longer be connected. You can choose to **Never** connect data points with gaps, **Always** connect data points with gaps, or set a **Threshold** at which gaps in the data should no longer be connected.

Stack series

Stacking allows Grafana to display series on top of each other. Be cautious when using stacking in the visualization as it can easily create misleading graphs. To read more about why stacking might not be the best approach, refer to The Issue with Stacking.

The options for stacking are:

- **Off** – Turns off series stacking.
- **Normal** – Stacks series on top of each other.
- **100%** – Stack by percentage, where all series together add up to 100%.

Stack series in groups

You can override the stacking behavior to stack series in groups. For more information about creating an override, see Configure field overrides. When creating the override, give the name of the stacking group you want the series to be part of.

Fill below to

The **Fill below to** option fills the area between two series. This options is only available as a series or field override. Using this option you can fill the area between two series, rather than from the series line down to 0. For example, if you had two series called *Max* and *Min*, you could select the

Max series and override it to **Fill below to** the **Min** series.. This would fill only the area between the two series lines.

Axis options

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Options under the axis category change how the X and Y axes are rendered. Some options do not take effect until you click outside of the field option box you are editing. You can also or press `Enter`.

Placement

Select the placement of the Y-axis. The options are:

- **Auto** – Grafana automatically assigns the Y-axis to the series. When there are two or more series with different units, Grafana assigns the left axis to the first unit, and the right axis to the units that follow.
- **Left** – Display all Y-axes on the left side.
- **Right** – Display all Y-axes on the right side.
- **Hidden** – Hide all Y-axes.

To assign axes for each field or series, add field overrides.

Label

Set a Y-axis text label. If you have more than one Y-axis, then you can assign different labels using an override.

Width

Set a fixed width of the axis. By default, Grafana dynamically calculates the width of an axis.

By setting the width of the axis, data with different axes types can share the same display proportions. This setting makes it easier for you to compare more than one graph's worth of data because the axes are not shifted or stretched within visual proximity to each other.

Soft min and soft max

Set a **Soft min** or **soft max** option for better control of Y-axis limits. By default, Grafana sets the range for the Y-axis automatically based on the dataset.

Soft min and soft max settings allow visibility into small changes when big changes aren't present. Hard min or max derived from standard min and max field options can prevent intermittent spikes from flattening useful detail by clipping the spikes past a specific point.

To define hard limits of the Y-axis, You can set standard min/max options. For more information, refer to Configure standard options.

Scale

Set how the Y-axis scales. The choices are **Linear** or **Logarithmic**. If you choose logarithmic, you can further choose between base 2 or base 10 logarithmic scales.

Transform

You can override a series to apply a transform to values on a graph (without affecting the underlying values or the values in tooltips, context menus, or legends). You have two transform options:

- **Negative Y transform** – Flip the results to negative values on the Y axis.
- **Constant** – Show the first value as a constant line.

Color options

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

By default, the graph uses the standard Color scheme option to assign series colors. You can also use the legend to open the color picker by clicking the legend series color icon. Setting color this way automatically creates an override rule that set's a specific color for a specific series.

The following are additional options that you can use to override series color defaults.

Classic Palette

The most common setup is to use the **Classic palette** for graphs. This scheme automatically assigns a color for each field or series based on its order. If the order of a field changes in your query, the color also changes. You can manually configure a color for a specific field using an override rule.

Single color

Use this mode to specify a color. You can also click the colored line icon next to each series in the Legend to open the color picker. This automatically creates a new override that sets the color scheme to single color and the selected color.

By value color schemes

If you select a by value color scheme like **From thresholds (by value)** or **Green-Yellow-Red (by value)**, the **Color series by** option appears. This option controls which value (Last, Min, Max) to use to assign the series its color.

Scheme gradient mode

The **Gradient mode** option located under the **Graph styles** has a mode named **Scheme**. When you enable **Scheme**, the line or bar receives a gradient color defined from the selected **Color scheme**.

From thresholds

If the **Color scheme** is set to **From thresholds (by value)** and **Gradient mode** is set to **Scheme**, then the line or bar color changes as they cross the defined thresholds. You will see only the exact colors chosen in the scheme.

Gradient color schemes

Using a gradient color scheme *without* setting the **Gradient mode** to **Scheme**, means that the colors chosen will form a gradient between the colors chosen, as the values in the series move between the thresholds set.

Traces panel (Beta)

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

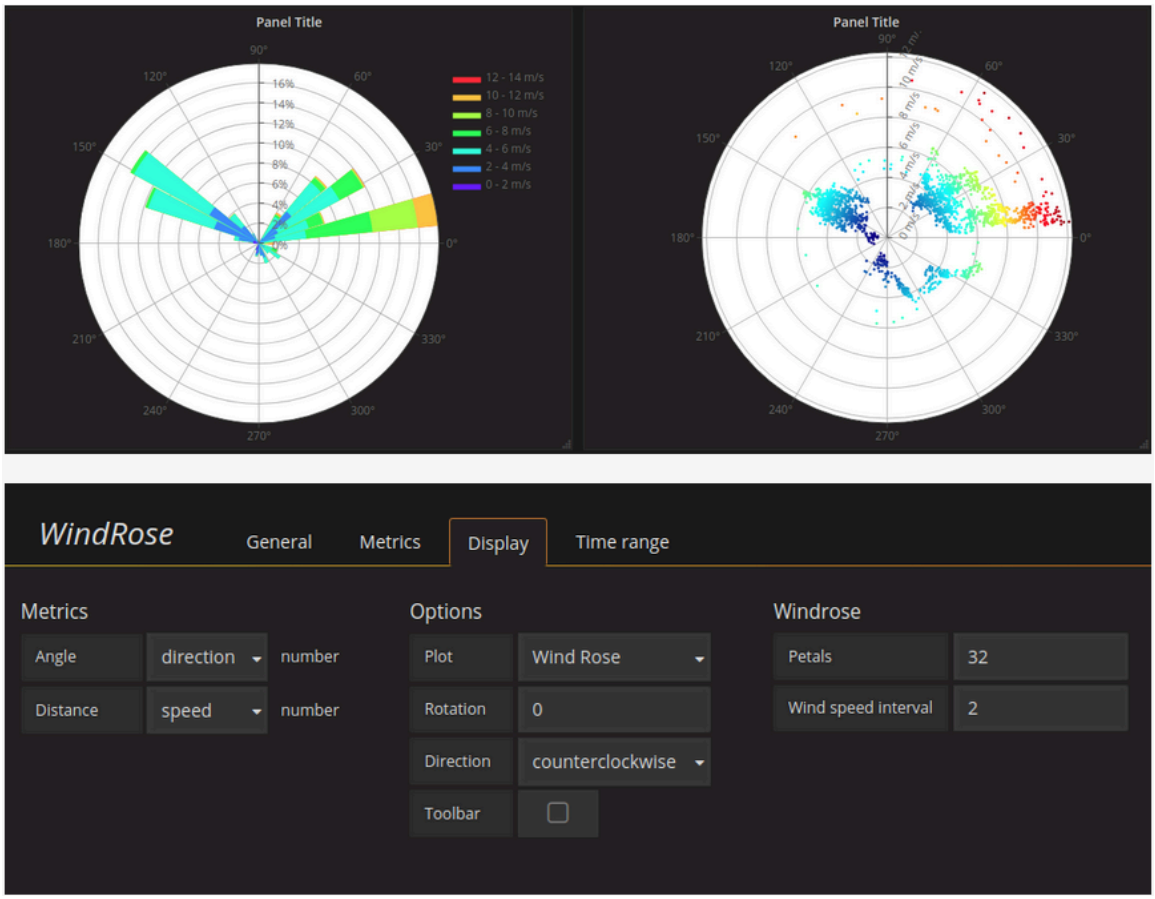
Traces are a visualization that enables you to track and log a request as it traverses the services in your infrastructure.

For more information about traces, see Tracing in Explore.

WindRose

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The WindRose panel receives raw time series data converts the data and maps it in a WindRose chart.



Options

The WindRose panel supports the following options:

- Axis frequency
- Axis style (degrees or compass)
- Scale (linear, square, log)

Explore in Grafana version 9

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana's dashboard UI provides functionality to build dashboards for visualization. *Explore* strips away the dashboard and panel options so that you can focus on the query. It helps you iterate until you have a working query and then you can build a dashboard from the query.

Note

If you just want to explore your data and do not want to create a dashboard, then Explore makes this much easier. If your data source supports graph and table data, then Explore shows the results both as a graph and a table. This allows you to see trends in the data and more details at the same time.

Start exploring

Note

In order to access Explore, you must have an editor or an administrator role.

To access Explore

1. In your Grafana workspace, choose the Explore menu item from the left menu bar.

   An empty Explore tab opens.

   Alternately, to start with an existing query in a panel, choose the Explore option from the Panel menu. This opens an Explore tab with the query from the panel and allows you to tweak or iterate in the query outside of your dashboard.
2. Choose your data source from the dropdown in the top left. Prometheus has a custom Explore implementation, the other data sources use their standard query editor.
3. In the query field, write your query to explore your data. There are three buttons beside the query field, a clear button (X), an add query button (+) and the remove query button (-). Just like the normal query editor, you can add and remove multiple queries.

   For more details about queries, see Query and transform data.

Split and compare

The split view provides an easy way to compare graphs and tables side-by-side or to look at related data together on one page.

Top open a split view

1. In the Explore view, choose the **Split** button to duplicate the current query and split the page into two side-by-side queries.

   > **Note**
   >
   > It is possible to select another data source for the new query which for example, allows you to compare the same query for two different servers or to compare the staging environment to the production environment.

   In split view, timepickers for both panels can be linked (if you change one, the other gets changed as well) by selecting a time-sync button attached to one of the timepickers. Linking timepickers keeps the start and the end times of the split view queries in sync. It ensures that you're looking at the same time interval in both split panels.
2. To close the newly created query, click on the Close Split button.

Share shortened link

The Share shortened link capability allows you to create smaller and simpler URLs of the format /goto/:uid instead of using longer URLs with query parameters. To create a shortened link to the query results, select the **Share** option in the Explore toolbar. A shortened link that is never used will automatically get deleted after seven (7) days.

Query management in Explore

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

To help with debugging queries, Explore allows you to investigate query requests and responses, as well as query statistics, via the Query inspector. This functionality is similar to the panel inspector tasks Inspect query performance and Inspect query request and response data.

Query history

Query history is a list of queries that you used in Explore. The history is stored in the Grafana database and it is not shared with other users. The retention period for queries in history is two weeks. Queries older than two weeks are automatically deleted. To open and interact with your history, select the **Query history** button in Explore.

Note

Starred (favorited) queries are not subject to the two weeks retention period and they are not deleted.

View query history

Query history lets you view the history of your querying. For each individual query, you can:

- Run a query.
- Create and/or edit a comment.
- Copy a query to the clipboard.
- Copy a shortened link with the query to the clipboard.
- Star (favorite) a query.

Manage favorite queries

All queries that have been starred in the Query history tab are displayed in the Starred list. This allows you to access your favorite queries faster and to reuse these queries without typing them from scratch.

Sorting query history

By default, query history shows you the most recent queries. You can sort your history by date or by data source name in ascending or descending order.

To sort your query history

1. Select the **Sort queries by** field.
2. Select one of the following options:
   - Newest first
   - Oldest first

Filtering query history

You can filter your query history in Query history and Starred tab to a specific data source.

Filtering history to a data source

1. Select the **Filter queries for specific data source(s)** field.
2. Select the data source for which you would like to filter your history. You can select multiple data sources.

In the **Query history** tab it is also possible to filter queries by date using the slider:

- Use the vertical slider to filter queries by date.
- Adjust the start date by dragging the top handle.
- Adjust the end date by dragging the top handle.

Searching in query history

You can search in your history across queries and your comments. Search is possible for queries in the Query history tab and Starred tab.

To search in query history

1. Select the **Search queries** field.
2. Enter the term you are searching for into search field.

Query history settings

You can customize the query history in the Settings tab. Options are described in the table below.

Setting	Default value
Change the default active tab	Query history tab

> **Note**
>
> Query history settings are global, and applied to both panels in split mode.

Prometheus-specific Features

Explore features a custom querying experience for Prometheus. When a query is run, it actually runs two queries, a normal Prometheus query for the graph and an *Instant Query* for the table. An Instant Query returns the last value for each time series which shows a good summary of the data shown in the graph.

Metrics explorer

On the left side of the query field, choose **Metrics** to open the Metric Explorer. This shows a hierarchical menu with metrics grouped by their prefix. For example, all Alertmanager metrics are grouped under the `alertmanager` prefix. This is a good starting point if you just want to explore which metrics are available.

Query field

The Query field supports autocomplete for metric names and functions, comparable to the standard Prometheus query editor. You can press the Enter key to create a new line and Shift +Enter to run a query.

The autocomplete menu can be triggered by pressing Ctrl+Space. The Autocomplete menu contains a new History section with a list of recently run queries.

Suggestions can appear under the query field - select on them to update your query with the suggested change.

- For counters (monotonically increasing metrics), a rate function will be suggested.
- For buckets, a histogram function will be suggested.
- For recording rules, possible to expand the rules.

Table filters

Select the filter button in the **label** column of a Table panel to add filters to the query expression. You can add filters for multiple queries as well - the filter is added for all the queries.

Logs in Explore

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Along with metrics, Explore allows you to investigate your logs in the following data sources:

- OpenSearch
- InfluxDB
- Loki

During an infrastructure monitoring and incident response, you can dig deeper into the metrics and logs to find the cause. Explore also allows you to correlate metrics and logs by viewing them side-by-side. This creates a new debugging workflow.

1. Receive an alert.
2. Drill down and examine metrics.
3. Drill down again and search logs related to the metric and time interval (and in the future, distributed traces).

Logs visualization

Results of log queries are shown as histograms in the graph and individual logs are explained in the following sections.

If the data source supports a full range log volume histogram, the graph with log distribution for all entered log queries is shown automatically. This feature is currently supported by OpenSearch and Loki data sources.

> **Note**
>
> In Loki, this full range log volume histogram is rendered by metric query which can be expensive depending on time range queried. This query can be particularly challenging for smaller Loki installations to process. To mitigate this, we recommend using a proxy like nginx in front of Loki to set a custom timeout (e.g. 10 seconds) for these queries. Log volume histogram queries can be identified by looking for queries with the HTTP header `X-Query-Tags` with value `Source=logvolhist`; these headers are added by Grafana to all log volume histogram queries.

If the data source does not support loading full range log volume histogram, the logs model computes a time series based on the log row counts bucketed by an automatically calculated time interval, and the first log row's timestamp then anchors the start of the histogram from the result. The end of the time series is anchored to the time picker's **To** range.

Log level

For logs where a level label is specified, Grafana uses the value of the label to determine the log level and update color accordingly. If the log doesn't have a level label specified, it tries to find out if its content matches any of the supported expressions (see below for more information). The log level is always determined by the first match. In case Grafana is not able to determine a log level, it will be visualized with an unknown log level.

> **Tip**
>
> If you use Loki data source and the `level` is in your log-line, use parsers (JSON, logfmt, regex,..) to extract the level information into a level label that is used to determine log level. This will allow the histogram to show the various log levels in separate bars.

Supported log levels and mapping of log level abbreviation and expressions:

Supported expressions	Log level	Color
emerg	critical	purple
fatal	critical	purple
alert	critical	purple
crit	critical	purple
critical	critical	purple
err	error	red
eror	error	red
error	error	red
warn	warning	yellow
warning	warning	yellow
info	info	green
information	info	green
notice	info	green
dbug	debug	blue
debug	debug	blue
trace	trace	light blue
*	unknown	grey

Logs navigation

The logs navigation interface, next to the log lines, can be used to request more logs. You can do this by selecting the **Older logs** button on the bottom of navigation. When you hit the line limit and you want to see more logs, you can use this to fetch more logs. Each request is displayed in the navigation as a separate page. Every page shows the from and to timestamp of the incoming log lines. You can see previous results by clicking on the page to view. Explore caches the last five requests run from the logs navigation, so it does not re-run the same query when clicking on those pages.

Visualization options

You can customize how logs are displayed and select which columns are shown.

Time

Shows or hides the time column. This is the timestamp associated with the log line as reported from the data source.

Unique labels

Shows or hides the unique labels column that includes only non-common labels. All common labels are displayed above.

Wrap lines

Set this to True if you want the display to use line wrapping. If set to False, it will result in horizontal scrolling.

Prettify JSON

Set this to `true` to pretty print all JSON logs. This setting does not affect logs in any format other than JSON.

Deduplication

Log data can be very repetitive and Explore can help by hiding duplicate log lines. There are a few different deduplication algorithms that you can use:

- **Exact** – Exact matches are done on the whole line except for date fields.
- **Numbers** – Matches on the line after stripping out numbers such as durations, IP addresses, and so on.

- **Signature** – The most aggressive deduplication, this strips all letters and numbers and matches on the remaining whitespace and punctuation.

Flip results order

You can change the order of received logs from the default descending order (newest first) to ascending order (oldest first).

Labels and detected fields

Each log row has an extendable area with its labels and detected fields, for more robust interaction. For all labels we have added the ability to filter for (positive filter) and filter out (negative filter) selected labels. Each field or label also has a stats icon to display statistics in relation to all displayed logs.

Escaping newlines

Explore automatically detects some incorrectly escaped sequences in log lines, such as newlines (`\n`, `\r`) or tabs (`\t`). When it detects such sequences, Explore provides an “Escape newlines” option.

Explore can automatically fix incorrectly escaped sequences that it has detected

To automatically fix escape sequences

1. Select **Escape newlines** to replace the sequences.
2. Manually review the replacements to confirm their correctness.

Explore replaces these sequences. When it does so, the option will change from **Escape newlines** to **Remove escaping**. Evaluate the changes as the parsing might not be accurate based on the input received. You can revert the replacements by selecting **Remove escaping**.

Data links

By using data links, you can turn any part of a log message into an internal or external link. The created link is visible as a button in the **Links** section inside the **Log details** view.

Toggle field visibility

Expand a log line and click the eye icon to show or hide fields.

Loki-specific features

Loki is the open source log aggregation system from Grafana Labs. Loki is designed to be cost effective, as it does not index the contents of the logs, but rather a set of labels for each log stream. The logs from Loki are queried in a similar way to querying with label selectors in Prometheus. It uses labels to group log streams which can be made to match up with your Prometheus labels. For more information about Grafana Loki, you can see the Grafana Loki Github.

For more information, see Loki on how to query for log data.

Switch from metrics to logs

If you switch from a Prometheus query to a logs query (you can do a split first to have your metrics and logs side by side) then it will keep the labels from your query that exist in the logs and use those to query the log streams. For example, if you have the following Prometheus query:

```
grafana_alerting_active_alerts{job="grafana"}
```

after switching to the Logs data source, it will change to:

```
{job="grafana"}
```

This will return a chunk of logs in the selected time range that can be searched.

Logs sample

If the selected data source implements logs sample, and supports both log and metric queries, then for metric queries you will be able to automatically see samples of log lines that contributed to visualized metrics. This feature is currently supported by Loki data sources.

Tracing in Explore

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Explore allows you to visualize traces from tracing data sources.

The following data sources are supported.

- Jaeger
- Tempo
- AWS X-Ray
- Zipkin

For information on how to configure queries for the data sources listed above, refer to the documentation for specific data source.

Trace View explanation

This section explains the elements of the Trace View dashboard.

Header

The header of the trace view has the following elements

- Header title: Shows the name of the root span and trace ID.
- Search: Highlights spans containing the searched text.
- Metadata: Various metadata about the trace.

Minimap

Shows condensed view or the trace timeline. Drag your pointer over the minimap to zoom into smaller time range. Zooming will also update the main timeline, so it is easy to see shorter spans. Hovering over the minimap, when zoomed, will show Reset Selection button which resets the zoom.

Timeline

Shows list of spans within the trace. Each span row consists of these components:

- Expand children button: Expands or collapses all the children spans of selected span.
- Service name: Name of the service logged the span.
- Operation name: Name of the operation that this span represents.
- Span duration bar: Visual representation of the operation duration within the trace.

Span details

Clicking anywhere on the span row shows span details, including the following.

- Operation name
- Span metadata
- Tags: Any tags associated with this span.
- Process metadata: Metadata about the process that logged this span.
- Logs: List of logs logged by this span and associated key values. In case of Zipkin logs section shows Zipkin annotations.

Node graph

You can optionally expand the node graph for the displayed trace. Depending on the data source, this can show spans of the trace as nodes in the graph, or add some additional context, including the service graph based on the current trace.

Trace to logs

You can navigate from a span in a trace view directly to logs relevant for that span. This is available for Tempo, Jaeger, and Zipkin data sources. Refer to their relevant documentation for instructions on how to configure each data source.

Click the document icon to open a split view in Explore with the configured data source and query relevant logs for the span.

Service Graph view

The Service Graph view visualizes the span metrics (traces data for rates, error rates, and durations (RED)) and service graphs. Once the requirements are set up, this pre-configured view is immediately available.

For more information, see Tempo data source page. You can also see the service graph view page in the *Tempo documentation*.

Inspector in Explore

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The inspector helps you understand and troubleshoot your queries. You can inspect the raw data, export that data to a comma-separated values (CSV) file, export log results in TXT format, and view query requests.

Inspector UI

The inspector has the following tabs:

- **Stats tab** – Shows how long your query takes and how much it returns.
- **Query tab** – Shows you the requests to the server sent when Grafana queries the data source.
- **JSON tab** – Allows you to view and copy the data JSON and data frame structure JSON.
- **Data tab** – Shows the raw data returned by the query.
- **Error tab** – Shows the error. Only visible when query returns error.

Inspector tasks

You can perform a variety of tasks in the Explore inspector.

Open the Inspector

After you run the query you would like to inspect, select the **Inspector** button.

The inspector pane opens on the bottom of the screen.

Inspect raw query results

You can view raw query results that is the data returned by the query in a table.

In the **Inspector** tab, click the **Data** tab.

For multiple queries or for queries multiple nodes, there are additional options.

- **Show data frame:** Select the result set data you want to view.

- **Series joined by time:** View the raw data from all of your queries at once, one result set per column. You can click a column heading to sort the data.

Download raw query results as CSV

After you are viewing the raw query results, you can generate a CSV file of the results. You will get a CSV file of the result you see, so be sure to refine the results to get the results you want before generating the CSV file.

To generate the CSV file, select **Download CSV** in the **Inspector** tab.

In order to download a CSV file specifically formatted for Excel, expand **Data options** and then turn on the **Download for Excel** toggle before you select the **Download CSV** option.

Download log results as TXT

You can generate a TXT file of the logs you are currently viewing, by selecting **Download logs** in the **Inspector**tab.

Download trace results

Based on the data source type, Grafana can generate a JSON file for the trace results in one of the supported formats: Jaeger, Zipkin, or OTLP formats.

To download the traces, select **Download traces** on the **Inspector**tab.

Inspect query performance

The **Stats** tab displays statistics that tell you how long your query takes, how many queries you send, and the number of rows returned. This information can help you troubleshoot your queries, especially if any of the numbers are unexpectedly high or low.

Statistics are read-only.

View JSON model

You can explore and export data as well as data frame JSON models.

To view the JSON model

1. In the Inspector panel, click the **JSON** tab.

2. From the **Select source** dropdown, choose one of the following options:
   - **Data** – Displays a JSON object representing the data that was returned to Explore.
   - **DataFrame structure** – Displays the raw result set.
3. You can expand or collapse portions of the JSON to view separate sections. You can also select the **Copy to clipboard** option to copy JSON body and paste it into another application.

View raw request and response to data source

As you are working with Explore and the Inspector tab, you can view the raw request and response data that you are generating with a query. In the Inspector, select the **Query** tab and choose **Refresh** to see the raw data.

Grafana sends the query to the server and displays the result. You can drill down on specific portions of the query, expand or collapse all of it, or copy the data to the clipboard to use in other applications.

Alerts in Grafana version 9

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana alerting provides you with robust and actionable alerts that help you learn about problems in the systems moments after they occur, minimizing disruption to your services.

Amazon Managed Grafana includes access to an updated alerting system, *Grafana alerting*, that centralizes alerting information in a single, searchable view. It includes the following features:

- Create and manage Grafana alerts in a centralized view.
- Create and manage Cortex and Loki managed alerts through a single interface.

- View alerting information from Prometheus, Amazon Managed Service for Prometheus, and other Alertmanager compatible data sources.

When you create your Amazon Managed Grafana workspace, you have the choice of using Grafana alerting, or the Classic dashboard alerts. This section covers Grafana alerting.

> **Note**
>
> If you created your workspace with the Classic alerts enabled, and want to switch to Grafana alerting, you can switch between the two alerting systems..

Grafana alerting limitations

- The Grafana alerting system can retrieve rules from all available Amazon Managed Service for Prometheus, Prometheus, Loki, and Alertmanager data sources. It might not be able to fetch rules from other supported data sources.
- Alert rules defined in Grafana, rather than in Prometheus, send multiple notifications to your contact point. If you are using native Grafana alerts, we recommend that you stay on classic dashboard alerting and not enable the new Grafana alerting feature. If you would like to view Alerts defined in your Prometheus data source, then we recommend you enable Grafana Alerting, which sends only a single notification for alerts created in Prometheus Alertmanager.

> **Note**
>
> This limitation is no longer a limitation in Amazon Managed Grafana workspaces that support Grafana v10.4 and later.

Topics

- Overview
- Exploring alerting
- Set up Alerting
- Migrating classic dashboard alerts to Grafana alerting
- Manage your alert rules
- Manage your alert notifications

Overview

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The following gives you an overview of how Grafana Alerting works and introduces you to some of the key concepts that work together and form the core of its flexible and powerful alerting engine.

1. **Data source**

   Connects to data to be used by alerting. This data is often time-series data, for alerts, and shows the details of a system to be monitored and analyzed. For more information, see data sources.

2. **Alert rules**

   Set evaluation criteria that determines whether an alert instance will fire. An alert rule consists of one or more queries and expressions to pull data from the datasource, a condition describing what constitutes the need for an alert, the frequency of evaluation, and optionally, the duration over which the condition must be met for an alert to fire.

   Grafana managed alerts support multi-dimensional alerting, which means that each alert rule can create multiple alert instances. This is exceptionally powerful if you are observing multiple series in a single expression.

3. **Labels**

   Match an alert rule and its instances to notification policies and silences. They can also be used to group your alerts by severity.

4. **Notification policies**

   Set where, when, and how the alerts get routed to notify your team when the alert fires. Each notification policy specifies a set of label matchers to indicate which alerts they are responsible for. A notification policy has a contact point assigned to it that consists of one or more notifiers.

5. **Contact points**

   Define how your contacts are notified when an alert fires. We support a multitude of ChatOps tools to ensure the alerts come to your team.

Features

One page for all alerts

A single Grafana Alerting page consolidates both Grafana-managed alerts and alerts that reside in your Prometheus-compatible data source in one single place.

Multi-dimensional alerts

Alert rules can create multiple individual alert instances per alert rule, known as multi-dimensional alerts, giving you the power and flexibility to gain visibility into your entire system with just a single alert.

Routing alerts

Route each alert instance to a specific contact point based on labels you define. Notification policies are the set of rules for where, when, and how the alerts are routed to contact points.

Silencing alerts

Silences allow you to stop receiving persistent notifications from one or more alerting rules. You can also partially pause an alert based on certain criteria. Silences have their own dedicated section for better organization and visibility, so that you can scan your paused alert rules without cluttering the main alerting view.

Mute timings

With mute timings, you can specify a time interval when you don't want new notifications to be generated or sent. You can also freeze alert notifications for recurring periods of time, such as during a maintenance period.

Exploring alerting

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Whether you're starting or expanding your implementation of Grafana Alerting, learn more about the key concepts and available features that help you create, manage, and take action on your alerts and improve your team's ability to resolve issues quickly.

First of all, let's look at the different alert rule types that Grafana Alerting offers.

Alert rule types

Grafana-managed rules

Grafana-managed rules are the most flexible alert rule type. They allow you to create alerts that can act on data from any of our supported data sources. In addition to supporting multiple data sources, you can also add expressions to transform your data and set alert conditions. This is the only type of rule that allows alerting from multiple data sources in a single rule definition.

Mimir and Loki rules

To create Mimir or Loki alerts you must have a compatible Prometheus or Loki data source. You can check if your data source supports rule creation via Grafana by testing the data source and observing if the ruler API is supported.

Recording rules

Recording rules are only available for compatible Prometheus or Loki data sources. A recording rule allows you to pre-compute frequently needed or computationally expensive expressions and save their result as a new set of time series. This is useful if you want to run alerts on aggregated data or if you have dashboards that query computationally expensive expressions repeatedly.

Key concepts and features

The following table includes a list of key concepts, features and their definitions, designed to help you make the most of Grafana Alerting.

Key concept or feature	Definition
Data sources for Alerting	Select data sources you want to query and visualize metrics, logs and traces from.
Provisioning for Alerting	Manage your alerting resources and provision them into your Grafana system using file provisioning or Terraform.
Alertmanager	Manages the routing and grouping of alert instances.
Alert rule	A set of evaluation criteria for when an alert rule should fire. An alert rule consists of one or more queries and expressions, a condition, the frequency of evaluation, and the duration over which the condition is met. An alert rule can produce multiple alert instances.
Alert instance	An alert instance is an instance of an alert rule. A single-dimensional alert rule has one alert instance. A multidimensional alert rule has one or more alert instances. A single alert rule that matches to multiple results, such as CPU against 10 VMs, is counted as multiple (in this case 10) alert instances. This number can vary over time. For example, an alert rule that monitors CPU usage for all VMs in a system has more alert instances as VMs are added. For more information about alert-instance quotas, see Quota reached errors.
Alert group	The Alertmanager groups alert instances by default using the labels for the root notificat ion policy. This controls de-duplication and groups of alert instances, which are sent to contact points.

Key concept or feature	Definition
Contact point	Define how your contacts are notified when an alert rule fires.
Message templating	Create reusable custom templates and use them in contact points.
Notification policy	Set of rules for where, when, and how the alerts are grouped and routed to contact points.
Labels and label matchers	Labels uniquely identify alert rules. They link alert rules to notification policies and silences, determining which policy should handle them and which alert rules should be silenced.
Silences	Stop notifications from one or more alert instances. The difference between a silence and a mute timing is that a silence only lasts for only a specified window of time whereas a mute timing is meant to be recurring on a schedule. Uses label matchers to silence alert instances.
Mute timings	Specify a time interval when you don't want new notifications to be generated or sent. You can also freeze alert notifications for recurring periods of time, such as during a maintenance period. Must be linked to an existing notificat ion policy.

Data sources

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

There are a number of data sources that are compatible with Grafana Alerting. Each data source is supported by a plugin. You can use one of the built-in data sources listed below.

These are the data sources that are compatible with and supported by Amazon Managed Grafana.

- Connect to an Alertmanager data source
- Connect to an Amazon CloudWatch data source
- Connect to an Amazon OpenSearch Service data source
- Connect to an AWS IoT SiteWise data source
- Connect to an AWS IoT TwinMaker data source
- Connect to Amazon Managed Service for Prometheus and open-source Prometheus data sources
- Connect to an Amazon Timestream data source
- Connect to an Amazon Athena data source
- Connect to an Amazon Redshift data source
- Connect to an AWS X-Ray data source
- Connect to an Azure Monitor data source
- Connect to a Google Cloud Monitoring data source
- Connect to a Graphite data source
- Connect to an InfluxDB data source
- Connect to a Loki data source
- Connect to a Microsoft SQL Server data source
- Connect to a MySQL data source
- Connect to an OpenTSDB data source
- Connect to a PostgreSQL data source
- Connect to a Jaeger data source
- Connect to a Zipkin data source

- Connect to a Tempo data source
- Configure a TestData data source for testing

About alert rules

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

An alerting rule is a set of evaluation criteria that determines whether an alert instance will fire. The rule consists of one or more queries and expressions, a condition, the frequency of evaluation, and optionally, the duration over which the condition is met.

While queries and expressions select the data set to evaluate, a condition sets the threshold that an alert must meet or exceed to create an alert.

An interval specifies how frequently an alerting rule is evaluated. Duration, when configured, indicates how long a condition must be met. The alert rules can also define alerting behavior in the absence of data.

Topics

- Alert rule types
- Alert instances
- Namespaces and groups
- Notification templating

Alert rule types

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana supports several alert rule types. The following sections will explain their merits and demerits and help you choose the right alert type for your use case.

Grafana managed rules

Grafana-managed rules are the most flexible alert rule type. They allow you to create alerts that can act on data from any of your existing data sources.

In addition to supporting any data source, you can add expressions to transform your data and express alert conditions.

Mimir, Loki and Cortex rules

To create Mimir, Loki or Cortex alerts you must have a compatible Prometheus data source. You can check if your data source is compatible by testing the data source and checking the details if the ruler API is supported.

Recording rules

Recording rules are only available for compatible Prometheus data sources like Mimir, Loki and Cortex.

A recording rule allows you to save an expression's result to a new set of time series. This is useful if you want to run alerts on aggregated data or if you have dashboards that query the same expression repeatedly.

Read more about recording rules in Prometheus.

Alert instances

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana managed alerts support multi-dimensional alerting. Each alert rule can create multiple alert instances. This is powerful if you are observing multiple series in a single expression.

Consider the following PromQL expression:

```
sum by(cpu) (
  rate(node_cpu_seconds_total{mode!="idle"}[1m])
)
```

A rule using this expression will create as many alert instances as the amount of CPUs observed during evaluation, allowing a single rule to report the status of each CPU.

Namespaces and groups

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Alerts can be organized using Folders for Grafana-managed rules and namespaces for Mimir, Loki, or Prometheus rules and group names.

Namespaces

When creating Grafana-managed rules, the folder can be used to perform access control and grant or deny access to all rules within a specific folder.

Groups

All rules within a group are evaluated at the same **interval**.

Alert rules and recording rules within a group will always be evaluated **sequentially**, meaning no rules will be evaluated at the same time and in order of appearance.

> **Tip**
>
> If you want rules to be evaluated concurrently and with different intervals, consider storing them in different groups.

Notification templating

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Notifications sent via contact points are built using notification templates. Grafana's default templates are based on the Go templating system where some fields are evaluated as text, while others are evaluated as HTML (which can affect escaping).

The default template default_template.go is a useful reference for custom templates.

Since most of the contact point fields can be templated, you can create reusable custom templates and use them in multiple contact points. To learn about custom notifications using templates, see Customize notifications.

Nested templates

You can embed templates within other templates.

For example, you can define a template fragment using the `define` keyword.

```
{{ define "mytemplate" }}
  {{ len .Alerts.Firing }} firing. {{ len .Alerts.Resolved }} resolved.
```

```
{{ end }}
```

You can then embed custom templates within this fragment using the `template` keyword. For example:

```
Alert summary:
{{ template "mytemplate" . }}
```

You can use any of the following built-in template options to embed custom templates.

Name	Notes
`default.title`	Displays high-level status information.
`default.message`	Provides a formatted summary of firing and resolved alerts.
`teams.default.message`	Similar to `default.messsage` , formatted for Microsoft Teams.

HTML in notification templates

HTML in alerting notification templates is escaped. We do not support rendering of HTML in the resulting notification.

Some notifiers support alternative methods of changing the look and feel of the resulting notification. For example, Grafana installs the base template for alerting emails to `<grafana-install-dir>/public/emails/ng_alert_notification.html`. You can edit this file to change the appearance of all alerting emails.

Alerting on numeric data

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This topic describes how Grafana handles alerting on numeric rather than time series data.

Among certain data sources numeric data that is not time series can be directly alerted on, or passed into Server Side Expressions (SSE). This allows for more processing and resulting efficiency within the data source, and it can also simplify alert rules. When alerting on numeric data instead of time series data, there is no need to reduce each labeled time series into a single number. Instead labeled numbers are returned to Grafana instead.

Tabular Data

This feature is supported with backend data sources that query tabular data:

- SQL data sources such as MySQL, Postgres, MSSQL, and Oracle.
- The Azure Kusto based services: Azure Monitor (Logs), Azure Monitor (Azure Resource Graph), and Azure Data Explorer.

A query with Grafana managed alerts or SSE is considered numeric with these data sources, if:

- The “Format AS” option is set to “Table” in the data source query.
- The table response returned to Grafana from the query includes only one numeric (e.g. int, double, float) column, and optionally additional string columns.

If there are string columns, then those columns become labels. The name of column becomes the label name, and the value for each row becomes the value of the corresponding label. If multiple rows are returned, then each row should be uniquely identified their labels.

Example

For a MySQL table called “DiskSpace”:

Time	Host	Disk	PercentFree
2021-June-7	web1	/etc	3
2021-June-7	web2	/var	4

Time	Host	Disk	PercentFree
2021-June-7	web3	/var	8
...	...	...	...

You can query the data filtering on time, but without returning the time series to Grafana. For example, an alert that would trigger per Host, Disk when there is less than 5% free space:

```
SELECT Host , Disk , CASE WHEN PercentFree  < 5.0 THEN PercentFree  ELSE 0 END FROM (
   SELECT
      Host,
      Disk,
      Avg(PercentFree)
   FROM DiskSpace
   Group By
      Host,
      Disk
   Where __timeFilter(Time)
```

This query returns the following Table response to Grafana:

Host	Disk	PercentFree
web1	/etc	3
web2	/var	4
web3	/var	0

When this query is used as the **condition** in an alert rule, then the non-zero will be alerting. As a result, three alert instances are produced:

Labels	Status
{Host=web1,disk=/etc}	Alerting
{Host=web2,disk=/var}	Alerting

Labels	Status
{Host=web3,disk=/var}	Normal

Labels and annotations

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Labels and annotations contain information about an alert. Both labels and annotations have the same structure: a set of named values; however their intended uses are different. An example of label, or the equivalent annotation, might be `alertname="test"`.

The main difference between a label and an annotation is that labels are used to differentiate an alert from all other alerts, while annotations are used to add additional information to an existing alert.

For example, consider two high CPU alerts: one for `server1` and another for `server2`. In such an example, we might have a label called `server` where the first alert has the label `server="server1"` and the second alert has the label `server="server2"`. However, we might also want to add a description to each alert such as `"The CPU usage for server1 is above 75%."`, where `server1` and `75%` are replaced with the name and CPU usage of the server (please refer to the documentation on Templating labels and annotations for how to do this). This kind of description would be more suitable as an annotation.

Labels

Labels contain information that identifies an alert. An example of a label might be `server=server1`. Each alert can have more than one label, and the complete set of labels for an alert is called its label set. It is this label set that identifies the alert.

For example, an alert might have the label set `{alertname="High CPU usage",server="server1"}` while another alert might have the label set `{alertname="High CPU usage",server="server2"}`. These are two separate alerts because although their `alertname` labels are the same, their `server` labels are different.

The label set for an alert is a combination of the labels from the datasource, custom labels from the alert rule, and a number of reserved labels such as `alertname`.

Custom Labels

Custom labels are additional labels from the alert rule. Like annotations, custom labels must have a name, and their value can contain a combination of text and template code that is evaluated when an alert is fired. Documentation on how to template custom labels can be found here.

When using custom labels with templates, it is important to make sure that the label value does not change between consecutive evaluations of the alert rule as this will end up creating large numbers of distinct alerts. However, it is OK for the template to produce different label values for different alerts. For example, do not put the value of the query in a custom label as this will end up creating a new set of alerts each time the value changes. Instead use annotations.

It is also important to make sure that the label set for an alert does not have two or more labels with the same name. If a custom label has the same name as a label from the datasource then it will replace that label. However, should a custom label have the same name as a reserved label then the custom label will be omitted from the alert.

Annotations

Annotations are named pairs that add additional information to existing alerts. There are a number of suggested annotations in Grafana such as `description`, `summary`, `runbook_url`, `dashboardUId` and `panelId`. Like custom labels, annotations must have a name, and their value can contain a combination of text and template code that is evaluated when an alert is fired. If an annotation contains template code, the template is evaluated once when the alert is fired. It is not re-evaluated, even when the alert is resolved. Documentation on how to template annotations can be found here.

How label matching works

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use labels and label matchers to link alert rules to notification policies and silences. This allows for a very flexible way to manage your alert instances, specify which policy should handle them, and which alerts to silence.

A label matchers consists of 3 distinct parts, the **label**, the **value** and the **operator**.

- The **Label** field is the name of the label to match. It must exactly match the label name.
- The **Value** field matches against the corresponding value for the specified **Label** name. How it matches depends on the **Operator** value.
- The **Operator** field is the operator to match against the label value. The available operators are:

Operator	Description
=	Select labels that are exactly equal to the value.
!=	Select labels that are not equal to the value.
=~	Select labels that regex-match the value.
!~	Select labels that do not regex-match the value.

If you are using multiple label matchers, they are combined using the AND logical operator. This means that all matchers must match in order to link a rule to a policy.

Example scenario

If you define the following set of labels for your alert:

```
{ foo=bar, baz=qux, id=12 }
```

then:

- A label matcher defined as `foo=bar` matches this alert rule.
- A label matcher defined as `foo!=bar` does *not* match this alert rule.
- A label matcher defined as `id=~[0-9]+` matches this alert rule.
- A label matcher defined as `baz!~[0-9]+` matches this alert rule.
- Two label matchers defined as `foo=bar` and `id=~[0-9]+` match this alert rule.

Labels in Grafana Alerting

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This topic explains why labels are a fundamental component of alerting.

- The complete set of labels for an alert is what uniquely identifies an alert within Grafana alerts.
- The Alertmanager uses labels to match alerts for silences and alert groups in notification policies.
- The alerting UI shows labels for every alert instance generated during evaluation of that rule.
- Contact points can access labels to dynamically generate notifications that contain information specific to the alert that is resulting in a notification.
- You can add labels to an alerting rule. Labels are manually configurable, use template functions, and can reference other labels. Labels added to an alerting rule take precedence in the event of a collision between labels (except in the case of Grafana reserved labels, see below for more information).

External Alertmanager Compatibility

Grafana's built-in Alertmanager supports both Unicode label keys and values. If you are using an external Prometheus Alertmanager, label keys must be compatible with their data model. This means that label keys must only contain **ASCII letters**, **numbers**, as well as **underscores** and match the regex `[a-zA-Z_][a-zA-Z0-9_]*`. Any invalid characters will be removed or replaced by the Grafana alerting engine before being sent to the external Alertmanager according to the following rules:

- `Whitespace` will be removed.
- `ASCII characters` will be replaced with _.
- `All other characters` will be replaced with their lower-case hex representation. If this is the first character it will be prefixed with _.

> **Note**
>
> If multiple label keys are sanitized to the same value, the duplicates will have a short hash of the original label appended as a suffix.

Grafana reserved labels

> **Note**
>
> Labels prefixed with `grafana_` are reserved by Grafana for special use. If a manually configured label is added beginning with `grafana_` it can be overwritten in case of collision.

Grafana reserved labels can be used in the same way as manually configured labels. The current list of available reserved labels are:

Label	Description
grafana_folder	Title of the folder containing the alert.

Templating labels and annotations

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

In Grafana, you template labels and annotations just like you would in Prometheus. If you have used Prometheus before then you should be familiar with the `$labels` and `$value` variables, which contain the labels and value of the alert. You can use the same variables in Grafana, even if the alert does not use a Prometheus datasource. If you haven't used Prometheus before then don't worry as each of these variables, and how to template them, will be explained as you follow the rest of this page.

Go's templating language

Templates for labels and annotations are written in Go's templating language, text/template.

Opening and closing tags

In text/template, templates start with `{{` and end with `}}` irrespective of whether the template prints a variable or runs control structures such as if statements. This is different from other templating languages such as Jinja where printing a variable uses `{{` and `}}` and control structures use `{%` and `%}`.

Print

To print the value of something use `{{` and `}}`. You can print the the result of a function or the value of a variable. For example, to print the `$labels` variable you would write the following:

```
{{ $labels }}
```

Iterate over labels

To iterate over each label in `$labels` you can use a `range`. Here `$k` refers to the name and `$v` refers to the value of the current label. For example, if your query returned a label `instance=test` then `$k` would be `instance` and `$v` would be `test`.

```
{{ range $k, $v := $labels }}
{{ $k }}={{ $v }}
{{ end }}
```

The labels, value and values variables

The labels variable

The `$labels` variable contains the labels from the query. For example, a query that checks if an instance is down might return an instance label with the name of the instance that is down. For example, suppose you have an alert rule that fires when one of your instances has been down for more than 5 minutes. You want to add a summary to the alert that tells you which instance is down. With the `$labels` variable, you can create a summary that prints the instance label in the summary:

```
Instance {{ $labels.instance }} has been down for more than 5 minutes
```

Labels with dots

If the label you want to print contains a dot (full stop or period) in its name using the same dot in the template will not work:

```
Instance {{ $labels.instance.name }} has been down for more than 5 minutes
```

This is because the template is attempting to use a non-existing field called `name` in `$labels.instance`. You should instead use the `index` function, which prints the label `instance.name` in the `$labels` variable:

```
Instance {{ index $labels "instance.name" }} has been down for more than 5 minutes
```

The value variable

The `$value` variable works different from Prometheus. In Prometheus `$value` is a floating point number containing the value of the expression, but in Grafana it is a string containing the labels and values of all Threshold, Reduce and Math expressions, and Classic Conditions for this alert rule.

It does not contain the results of queries, as these can return anywhere from 10s to 10,000s of rows or metrics.

If you were to use the `$value` variable in the summary of an alert:

```
{{ $labels.service }} has over 5% of responses with 5xx errors: {{ $value }})
```

The summary might look something like the following:

```
api has an over 5% of responses with 5xx errors: [ var='B' labels={service=api}
 value=6.789 ]
```

Here `var='B'` refers to the expression with the RefID B. In Grafana, all queries and expressions are identified by a RefID that identifies each query and expression in an alert rule. Similarly `labels={service=api}` refers to the labels, and `value=6.789` refers to the value.

You might have observed that there is no RefID A. That is because in most alert rules the RefID A refers to a query, and since queries can return many rows or time series they are not included in `$value`.

The values variable

If the `$value` variable contains more information than you need, you can instead print the labels and value of individual expressions using `$values`. Unlike `$value`, the `$values` variable is a table of objects containing the labels and floating point values of each expression, indexed by their RefID.

If you were to print the value of the expression with RefID B in the summary of the alert:

```
{{ $labels.service }} has over 5% of responses with 5xx errors: {{ $values.B }}%
```

The summary will contain just the value:

```
api has an over 5% of responses with 5xx errors: 6.789%
```

However, while `{{ $values.B }}` prints the number 6.789, it is actually a string as you are printing the object that contains both the labels and value for RefID B, not the floating point value of B. To use the floating point value of RefID B you must use the `Value` field from `$values.B`. If you were to humanize the floating point value in the summary of an alert:

```
{{ $labels.service }} has over 5% of responses with 5xx errors: {{ humanize
 $values.B.Value }}%
```

No data, runtime errors and timeouts

If the query in your alert rule returns no data, or fails because of a datasource error or timeout, then any Threshold, Reduce or Math expressions that use that query will also return no data or an error. When this happens these expression will be absent from `$values`. It is good practice to check that a RefID is present before using it as otherwise your template will break should your query return no data or an error. You can do this using an if statement:

```
{{ if $values.B }}{{ $labels.service }} has over 5% of responses with 5xx errors:
 {{ humanizePercentage $values.B.Value }}{{ end }}
```

Classic Conditions

If the rule uses Classic Conditions instead of Threshold, Reduce and Math expressions, then the `$values` variable is indexed by both the Ref ID and position of the condition in the Classic Condition. For example, if you have a Classic Condition with RefID B containing two conditions, then `$values` will contain two conditions `B0` and `B1`.

```
The first condition is {{ $values.B0 }}, and the second condition is {{ $values.B1 }}
```

Functions

The following functions are also available when expanding labels and annotations:

args

The `args` function translates a list of objects to a map with keys arg0, arg1 etc. This is intended to allow multiple arguments to be passed to templates.

Example

```
{{define "x"}}{{.arg0}} {{.arg1}}{{end}}{{template "x" (args 1 "2")}}
```

```
1 2
```

externalURL

The `externalURL` function returns the external URL of the Grafana server as configured in the ini file(s).

Example

```
{{ externalURL }}
```

```
https://example.com/grafana
```

graphLink

The `graphLink` function returns the path to the graphical view in Explore in Grafana version 9 for the given expression and data source.

Example

```
{{ graphLink "{\"expr\": \"up\", \"datasource\": \"gdev-prometheus\"}" }}
```

```
/explore?left=["now-1h","now","gdev-prometheus",{"datasource":"gdev-
prometheus","expr":"up","instant":false,"range":true}]
```

humanize

The `humanize` function humanizes decimal numbers.

Example

```
{{ humanize 1000.0 }}
```

```
1k
```

humanize1024

The `humanize1024` works similar to `humanize` but uses 1024 as the base rather than 1000.

Example

```
{{ humanize1024 1024.0 }}
```

```
1ki
```

humanizeDuration

The `humanizeDuration` function humanizes a duration in seconds.

Example

```
{{ humanizeDuration 60.0 }}
```

```
1m 0s
```

humanizePercentage

The `humanizePercentage` function humanizes a ratio value to a percentage.

Example

```
{{ humanizePercentage 0.2 }}
```

```
20%
```

humanizeTimestamp

The `humanizeTimestamp` function humanizes a Unix timestamp.

Example

```
{{ humanizeTimestamp 1577836800.0 }}
```

```
2020-01-01 00:00:00 +0000 UTC
```

match

The `match` function matches the text against a regular expression pattern.

Example

```
{{ match "a.*" "abc" }}
```

```
true
```

pathPrefix

The `pathPrefix` function returns the path of the Grafana server as configured in the ini file(s).

Example

```
{{ pathPrefix }}
```

```
/grafana
```

tableLink

The `tableLink` function returns the path to the tabular view in Explore in Grafana version 9 for the given expression and data source.

Example

```
{{ tableLink "{\"expr\": \"up\", \"datasource\": \"gdev-prometheus\"}" }}
```

```
/explore?left=["now-1h","now","gdev-prometheus",{"datasource":"gdev-
prometheus","expr":"up","instant":true,"range":false}]
```

title

The `title` function capitalizes the first character of each word.

Example

```
{{ title "hello, world!" }}
```

```
Hello, World!
```

toLower

The `toLower` function returns all text in lowercase.

Example

```
{{ toLower "Hello, world!" }}
```

```
hello, world!
```

toUpper

The `toUpper` function returns all text in uppercase.

Example

```
{{ toUpper "Hello, world!" }}
```

```
HELLO, WORLD!
```

reReplaceAll

The `reReplaceAll` function replaces text matching the regular expression.

Example

```
{{ reReplaceAll "localhost:(.*)" "example.com:$1" "localhost:8080" }}
```

```
example.com:8080
```

State and health of alerting rules



This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The state and health of alerting rules help you understand several key status indicators about your alerts.

There are three key components: *alert rule state*, *alert instance state*, and *alert rule health*. Although related, each component conveys subtly different information.

Alert rule state

An alert rule can be in either of the following states:

State	Description
Normal	None of the time series returned by the evaluation engine is in a `Pending` or `Firing` state.
Pending	At least one time series returned by the evaluation engine is `Pending`.
Firing	At least one time series returned by the evaluation engine is `Firing`.

> **Note**
>
> Alerts will transition first to `pending` and then `firing`, thus it will take at least two evaluation cycles before an alert is fired.

Alert instance state

An alert instance can be in either of the following states:

State	Description
Normal	The state of an alert that is neither firing nor pending, everything is working correctly.
Pending	The state of an alert that has been active for less than the configured threshold duration.

State	Description
Alerting	The state of an alert that has been active for longer than the configured threshold duration.
NoData	No data has been received for the configured time window.
Error	The error that occurred when attempting to evaluate an alerting rule.

Alert rule health

An alert rule can have one the following health statuses:

State	Description
Ok	No error when evaluating an alerting rule.
Error	An error occurred when evaluating an alerting rule.
NoData	The absence of data in at least one time series returned during a rule evaluation.

Special alerts for `NoData` and `Error`

When evaluation of an alerting rule produces state `NoData` or `Error`, Grafana Alerting will generate alert instances that have the following additional labels:

Label	Description
alertname	Either `DatasourceNoData` or `DatasourceError` depending on the state.
datasource_uid	The UID of the data source that caused the state.

You can handle these alerts the same way as regular alerts by adding a silence, route to a contact point, and so on.

Contact points

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use contact points to define how your contacts are notified when an alert rule fires. A contact point can have one or more contact point types, for example, email, Slack, webhook, and so on. When an alert rule fires, a notification is sent to all contact point types listed for a contact point. Contact points can be configured for the Grafana Alertmanager as well as external alertmanagers.

You can also use notification templating to customize notification messages for contact point types.

Supported contact point types

The following table lists the contact point types supported by Grafana.

Name	Type
Amazon SNS	`sns`
OpsGenie	`opsgenie`
Pager Duty	`pagerduty`
Slack	`slack`
VictorOps	`victorops`

For more information about contact points, see Working with contact points and Customize notifications.

Notifications

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana uses Alertmanagers to send notifications for firing and resolved alerts. Grafana has its own Alertmanager, referred to as “Grafana” in the user interface, but also supports sending notifications from other Alertmanagers too, such as the Prometheus Alertmanager. The Grafana Alertmanager uses notification policies and contact points to configure how and where a notification is sent; how often a notification should be sent; and whether alerts should all be sent in the same notification, sent in grouped notifications based on a set of labels, or as separate notifications.

Notification policies

Notification policies control when and where notifications are sent. A notification policy can choose to send all alerts together in the same notification, send alerts in grouped notifications based on a set of labels, or send alerts as separate notifications. You can configure each notification policy to control how often notifications should be sent as well as having one or more mute timings to inhibit notifications at certain times of the day and on certain days of the week.

Notification policies are organized in a tree structure where at the root of the tree there is a notification policy called the root policy. There can be only one root policy and the root policy cannot be deleted.

Specific routing policies are children of the root policy and can be used to match either all alerts or a subset of alerts based on a set of matching labels. A notification policy matches an alert when its matching labels match the labels in the alert.

A specific routing policy can have its own child policies, called nested policies, which allow for additional matching of alerts. An example of a specific routing policy could be sending

infrastructure alerts to the Ops team; while a child policy might send high priority alerts to Pagerduty and low priority alerts to Slack.

All alerts, irrespective of their labels, match the root policy. However, when the root policy receives an alert it looks at each specific routing policy and sends the alert to the first specific routing policy that matches the alert. If the specific routing policy has further child policies, then it can attempt to the match the alert against one of its nested policies. If no nested policies match the alert then the specific routing policy is the matching policy. If there are no specific routing policies, or no specific routing policies match the alert, then the root policy is the matching policy.

Contact points

Contact points contain the configuration for sending notifications. A contact point is a list of integrations, each of which sends a notification to a particular email address, service or URL. Contact points can have multiple integrations of the same kind, or a combination of integrations of different kinds. For example, a contact point could contain a Pager Duty integration; a Pager Duty and Slack integration; or a Pager Duty integration, a Slack integration, and two Amazon SNS integrations. You can also configure a contact point with no integrations; in which case no notifications are sent.

A contact point cannot send notifications until it has been added to a notification policy. A notification policy can only send alerts to one contact point, but a contact point can be added to a number of notification policies at the same time. When an alert matches a notification policy, the alert is sent to the contact point in that notification policy, which then sends a notification to each integration in its configuration.

> **Note**
>
> For information about supported integrations for contact points, see Contact points.

Templating notifications

You can customize notifications with templates. For example, templates can be used to change the title and message of notifications sent to Slack.

Templates are not limited to an individual integration or contact point, but instead can be used in a number of integrations in the same contact point and even integrations across different contact points. For example, a Grafana user can create a template called `custom_subject_or_title`

and use it for both templating subjects in Pager Duty and titles of Slack messages without having to create two separate templates.

All notifications templates are written in Go's templating language, and are in the Contact points tab on the Alerting page.

Silences

You can use silences to mute notifications from one or more firing rules. Silences do not stop alerts from firing or being resolved, or hide firing alerts in the user interface. A silence lasts as long as its duration which can be configured in minutes, hours, days, months or years.

Set up Alerting

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Configure the features and integrations that you need to create and manage your alerts.

Topics

- Add an external Alertmanager
- Provisioning Grafana Alerting resources

Add an external Alertmanager

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Set up Grafana to use an external Alertmanager as a single Alertmanager to receive all of your alerts. This external Alertmanager can then be configured and administered from within Grafana itself.

Once you have added the Alertmanager, you can use the Grafana Alerting UI to manage silences, contact points, and notification policies. A dropdown option in these pages allows you to switch between alertmanagers.

Note

Starting with Grafana 9.2, the URL configuration of external alertmanagers from the Admin tab on the Alerting page is deprecated. It will be removed in a future release.

External alertmanagers should now be configured as data sources using Grafana Configuration from the main Grafana navigation menu. This enables you to manage the contact points and notification policies of external alertmanagers from within Grafana and also encrypts HTTP basic authentication credentials that were previously visible when configuring external alertmanagers by URL.

To add an external Alertmanager, complete the following steps.

1. Click Configuration and then Data sources.
2. Search for Alertmanager.
3. Choose your Implementation and fill out the fields on the page, as required.

   If you are provisioning your data source, set the flag `handleGrafanaManagedAlerts` in the `jsonData` field to `true` to send Grafana-managed alerts to this Alertmanager.

   Note

   Prometheus, Grafana Mimir, and Cortex implementations of Alertmanager are supported. For Prometheus, contact points and notification policies are read-only in the Grafana Alerting UI.
4. Click Save & test.

Provisioning Grafana Alerting resources

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Alerting infrastructure is often complex, with many pieces of the pipeline that often live in different places. Scaling this across multiple teams and organizations is an especially challenging task. Grafana Alerting provisioning makes this process easier by enabling you to create, manage, and maintain your alerting data in a way that best suits your organization.

There are two options to choose from:

1. Provision your alerting resources using the Alerting Provisioning HTTP API.

   Note

   Typically, you cannot edit API-provisioned alert rules from the Grafana UI.
   In order to enable editing, add the x-disable-provenance header to the following requests when creating or editing your alert rules in the API:

   ```
   POST /api/v1/provisioning/alert-rules
   PUT /api/v1/provisioning/alert-rules/{UID}
   ```

2. Provision your alerting resources using Terraform.

Note

Currently, provisioning for Grafana Alerting supports alert rules, contact points, mute timings, and templates. Provisioned alerting resources using file provisioning or Terraform can only be edited in the source that created them and not from within Grafana or any

other source. For example, if you provision your alerting resources using files from disk, you cannot edit the data in Terraform or from within Grafana.

Topics

- Create and manage alerting resources using Terraform
- Viewing provisioned alerting resources in Grafana

Create and manage alerting resources using Terraform

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use Terraform's Grafana Provider to manage your alerting resources and provision them into your Grafana system. Terraform provider support for Grafana Alerting makes it easy to create, manage, and maintain your entire Grafana Alerting stack as code.

For more information on managing your alerting resources using Terraform, refer to the Grafana Provider documentation in the Terraform documentation.

Complete the following tasks to create and manage your alerting resources using Terraform.

1. Create an API key for provisioning.
2. Configure the Terraform provider.
3. Define your alerting resources in Terraform.
4. Run `terraform apply` to provision your alerting resources.

Prerequisites

- Ensure you have the grafana/grafana Terraform provider 1.27.0 or higher.

- Ensure you are using Grafana 9.1 or higher. If you created your Amazon Managed Grafana instance with Grafana version 9, this will be true.

Create an API key for provisioning

You can create a normal Grafana API key to authenticate Terraform with Grafana. Most existing tooling using API keys should automatically work with the new Grafana Alerting support. For information specifically about creating keys for use with Terraform, see Using Terraform for Amazon Managed Grafana automation.

To create an API key for provisioning

1. Create a new service account for your CI pipeline.
2. Assign the role “Access the alert rules Provisioning API.”
3. Create a new service account token.
4. Name and save the token for use in Terraform.

Alternatively, you can use basic authentication. To view all the supported authentication formats, see Grafana authentication in the Terraform documentation.

Configure the Terraform provider

Grafana Alerting support is included as part of the Grafana Terraform provider.

The following is an example you can use to configure the Terraform provider.

```
terraform {
    required_providers {
        grafana = {
            source = "grafana/grafana"
            version = ">= 1.28.2"
        }
    }
}

provider "grafana" {
    url = <YOUR_GRAFANA_URL>
    auth = <YOUR_GRAFANA_API_KEY>
}
```

Provision contact points and templates

Contact points connect an alerting stack to the outside world. They tell Grafana how to connect to your external systems and where to deliver notifications. There are over fifteen different integrations to choose from. This example uses a Slack contact point.

To provision contact points and templates

1. Copy this code block into a .tf file on your local machine. Replace *<slack-webhook-url>* with your Slack webhook URL (or other contact

   This example creates a contact point that sends alert notifications to Slack.

```
resource "grafana_contact_point" "my_slack_contact_point" {
    name = "Send to My Slack Channel"

    slack {
        url = <slack-webhook-url>
        text = <<EOT
{{ len .Alerts.Firing }} alerts are firing!

Alert summaries:
{{ range .Alerts.Firing }}
{{ template "Alert Instance Template" . }}
{{ end }}
EOT
    }
}
```

2. Enter text for your notification in the text field.

   The `text` field supports Go-style templating. This enables you to manage your Grafana Alerting notification templates directly in Terraform.

3. Run the command `terraform apply`.

4. Go to the Grafana UI and check the details of your contact point.

   You cannot edit resources provisioned via Terraform from the UI. This ensures that your alerting stack always stays in sync with your code.

5. Click **Test** to verify that the contact point works correctly.

Note

You can re-use the same templates across many contact points. In the example above, a shared template ie embedded using the statement `{{ template "Alert Instance Template" . }}`
This fragment can then be managed separately in Terraform:

```
resource "grafana_message_template" "my_alert_template" {
    name = "Alert Instance Template"

    template = <<EOT
{{ define "Alert Instance Template" }}
Firing: {{ .Labels.alertname }}
Silence: {{ .SilenceURL }}
{{ end }}
EOT
}
```

Provision notification policies and routing

Notification policies tell Grafana how to route alert instances, as opposed to where. They connect firing alerts to your previously defined contact points using a system of labels and matchers.

To provision notification policies and routing

1. Copy this code block into a .tf file on your local machine.

   In this example, the alerts are grouped by `alertname`, which means that any notifications coming from alerts which share the same name, are grouped into the same Slack message.

   If you want to route specific notifications differently, you can add sub-policies. Sub-policies allow you to apply routing to different alerts based on label matching. In this example, we apply a mute timing to all alerts with the label a=b.

```
resource "grafana_notification_policy" "my_policy" {
    group_by = ["alertname"]
    contact_point = grafana_contact_point.my_slack_contact_point.name

    group_wait = "45s"
    group_interval = "6m"
```

```
    repeat_interval = "3h"

    policy {
        matcher {
            label = "a"
            match = "="
            value = "b"
        }
        group_by = ["..."]
        contact_point = grafana_contact_point.a_different_contact_point.name
        mute_timings = [grafana_mute_timing.my_mute_timing.name]

        policy {
            matcher {
                label = "sublabel"
                match = "="
                value = "subvalue"
            }
            contact_point = grafana_contact_point.a_third_contact_point.name
            group_by = ["..."]
        }
    }
}
```

2. In the mute_timings field, link a mute timing to your notification policy.
3. Run the command `terraform apply`.
4. Go to the Grafana UI and check the details of your notification policy.

> **Note**
>
> You cannot edit resources provisioned from Terraform from the UI. This ensures that your alerting stack always stays in sync with your code.

5. Click **Test** to verify that the notification point is working correctly.

Provision mute timings

Mute timings provide the ability to mute alert notifications for defined time periods.

To provision mute timings

1. Copy this code block into a .tf file on your local machine.

   In this example, alert notifications are muted on weekends.

```
resource "grafana_mute_timing" "my_mute_timing" {
    name = "My Mute Timing"

    intervals {
        times {
          start = "04:56"
          end = "14:17"
        }
        weekdays = ["saturday", "sunday", "tuesday:thursday"]
        months = ["january:march", "12"]
        years = ["2025:2027"]
    }
}
```

2. Run the command `terraform apply`.
3. Go to the Grafana UI and check the details of your mute timing.
4. Reference your newly created mute timing in a notification policy using the `mute_timings` field. This will apply your mute timing to some or all of your notifications.

   > **Note**
   >
   > You cannot edit resources provisioned from Terraform from the UI. This ensures that your alerting stack always stays in sync with your code.

5. Click **Test** to verify that the mute timing is working correctly.

Provision alert rules

Alert rules enable you to alert against any Grafana data source. This can be a data source that you already have configured, or you can define your data sources in Terraform alongside your alert rules.

To provision alert rules

1. Create a data source to query and a folder to store your rules in.

   In this example, the Configure a TestData data source for testing data source is used.

   Alerts can be defined against any backend datasource in Grafana.

```
resource "grafana_data_source" "testdata_datasource" {
    name = "TestData"
    type = "testdata"
}

resource "grafana_folder" "rule_folder" {
    title = "My Rule Folder"
}
```

2. Define an alert rule.

   For more information on alert rules, refer to how to create Grafana-managed alerts.

3. Create a rule group containing one or more rules.

   In this example, the `grafana_rule_group` resource group is used.

```
resource "grafana_rule_group" "my_rule_group" {
    name = "My Alert Rules"
    folder_uid = grafana_folder.rule_folder.uid
    interval_seconds = 60
    org_id = 1

    rule {
        name = "My Random Walk Alert"
        condition = "C"
        for = "0s"

        // Query the datasource.
        data {
            ref_id = "A"
            relative_time_range {
                from = 600
                to = 0
            }
            datasource_uid = grafana_data_source.testdata_datasource.uid
```

```
            // `model` is a JSON blob that sends datasource-specific data.
            // It's different for every datasource. The alert's query is defined
 here.
            model = jsonencode({
                intervalMs = 1000
                maxDataPoints = 43200
                refId = "A"
            })
        }

        // The query was configured to obtain data from the last 60 seconds. Let's
 alert on the average value of that series using a Reduce stage.
        data {
            datasource_uid = "__expr__"
            // You can also create a rule in the UI, then GET that rule to obtain
 the JSON.
            // This can be helpful when using more complex reduce expressions.
            model = <<EOT
{"conditions":[{"evaluator":{"params":[0,0],"type":"gt"},"operator":
{"type":"and"},"query":{"params":["A"]},"reducer":{"params":
[],"type":"last"},"type":"avg"}],"datasource":
{"name":"Expression","type":"__expr__","uid":"__expr__"},"expression":"A","hide":false,"int
EOT
            ref_id = "B"
            relative_time_range {
                from = 0
                to = 0
            }
        }

        // Now, let's use a math expression as our threshold.
        // We want to alert when the value of stage "B" above exceeds 70.
        data {
            datasource_uid = "__expr__"
            ref_id = "C"
            relative_time_range {
                from = 0
                to = 0
            }
            model = jsonencode({
                expression = "$B > 70"
                type = "math"
                refId = "C"
            })
```

```
        }
    }
}
```

4. Go to the Grafana UI and check your alert rule.

   You can see whether the alert rule is firing. You can also see a visualization of each of the alert rule's query stages.

   When the alert fires, Grafana routes a notification through the policy you defined.

   For example, if you chose Slack as a contact point, Grafana's embedded Alertmanager automatically posts a message to Slack.

Viewing provisioned alerting resources in Grafana

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can verify that your alerting resources were created in Grafana.

To view your provisioned resources in Grafana

1. Open your Grafana instance.
2. Navigate to Alerting.
3. Click an alerting resource folder, for example, Alert rules.

   Provisioned resources are labeled **Provisioned**, so that it is clear that they were not created manually.

Note

You cannot edit provisioned resources from Grafana. You can only change the resource properties by changing the provisioning file and restarting Grafana or carrying out a hot reload. This prevents changes being made to the resource that would be overwritten if a file is provisioned again or a hot reload is carried out.

Migrating classic dashboard alerts to Grafana alerting

This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Workspaces that choose not to use Grafana alerting, use the classic dashboard alerting. To switch to the new Grafana alerting, you must opt in to the feature.

You can configure your Amazon Managed Grafana instance to use Grafana alerting using the AWS Management Console, the AWS CLI, or the Amazon Managed Grafana API. For details about how to configure Amazon Managed Grafana, including turning Grafana alerting on or off, see Configure a Amazon Managed Grafana workspace.

Note

When using Grafana alerting, alert rules defined in Grafana, rather than in Prometheus, send multiple notifications to your contact point. If you are using native Grafana alerts, we recommend that you stay on classic dashboard alerting and not enable the new Grafana alerting feature. If you would like to view Alerts defined in your Prometheus data source, then we recommend you enable Grafana Alerting, which sends only a single notification for alerts created in Prometheus Alertmanager.

This limitation has been removed in Amazon Managed Grafana workspaces that support Grafana v10.4 and later.

Migrating to Grafana alerting system

When Grafana alerting is turned on, existing classic dashboard alerts migrate in a format compatible with the Grafana alerting. In the Alerting page of your Grafana instance, you can view the migrated alerts alongside new alerts. With Grafana alerting, your Grafana-managed alert rules send multiple notifications rather than a single alert when they are matched.

Read and write access to classic dashboard alerts and Grafana alerts are governed by the permissions of the folders storing them. During migration, classic dashboard alert permissions are matched to the new rules permissions as follows:

- If the original alert's dashboard has permissions, migration creates a folder named with this format `Migrated {"dashboardUid": "UID", "panelId": 1, "alertId": 1}` to match permissions of the original dashboard (including the inherited permissions from the folder).
- If there are no dashboard permissions and the dashboard is under a folder, then the rule is linked to this folder and inherits its permissions.
- If there are no dashboard permissions and the dashboard is under the General folder, then the rule is linked to the General Alerting folder, and the rule inherits the default permissions.

Note

Since there is no `Keep Last State` option for `NoData` in Grafana alerting, this option becomes `NoData` during the classic rules migration. Option `Keep Last State` for `Error` handling is migrated to a new option `Error`. To match the behavior of the `Keep Last State`, in both cases, during the migration Amazon Managed Grafana automatically creates a silence for each alert rule with a duration of one year.

Notification channels are migrated to an Alertmanager configuration with the appropriate routes and receivers. Default notification channels are added as contact points to the default route. Notification channels not associated with any Dashboard alert go to the `autogen-unlinked-channel-recv` route.

Limitations

- Grafana alerting system can retrieve rules from all available Prometheus, Loki, and Alertmanager data sources. It might not be able to fetch alerting rules from other supported data sources.
- Migrating back and forth between Grafana alerts and the classic dashboard alerting can result in data loss for features supported in one system, but not the other.

> **Note**
>
> If you migrate back to the classic dashboard alerting, you lose all changes made to alerting configuration made while you had Grafana alerting enabled, including any new alert rules that were created.

Manage your alert rules

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

An alert rule is a set of evaluation criteria that determines whether an alert will fire. The alert rule consists of one or more queries and expressions, a condition, the frequency of evaluation, and optionally, the duration over which the condition is met.

While queries and expressions select the data set to evaluate, a condition sets the threshold that an alert must meet or exceed to create an alert. An interval specifies how frequently an alert rule is evaluated. Duration, when configured, indicates how long a condition must be met. Alert rules can also define alerting behavior in the absence of data.

> **Note**
>
> Grafana managed alert rules can only be edited or deleted by users with Edit permissions for the folder storing the rules.
> Alert rules for an external Grafana Mimir or Loki instance can be edited or deleted by users with Editor or Admin roles.

Topics

- Creating Grafana managed alert rules
- Creating Grafana Mimir or Loki managed alert rules
- Creating Grafana Mimir or Loki managed recording rules
- Grafana Mimir or Loki rule groups and namespaces
- View and edit alerting rules

Creating Grafana managed alert rules

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Grafana allows you to create alerting rules that query one or more data sources, reduce or transform the results and compare them to each other or to fixed thresholds. When these are run, Grafana sends notifications to the contact point.

To add a Grafana managed rule

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page listing existing alerts.
2. Choose **New alert rule**.

3. In **Step 1**, add the rule name, type and storage location, as follows:

   - In **Rule name**, add a descriptive name. This name is displayed in the alert rules list. It is also the `alertname` label for every alert instance that is created from this rule.
   - From the **Rule type** dropdown, select **Grafana managed alert**.
   - From the **Folder** dropdown, select the folder where you want to store the rule. If you do not select a folder, the rule is stored in the `General` folder. To create a folder, select the dropdown and enter a new folder name.

4. In **Step 2**, add the queries and expressions to evaluate.

   - Keep the default name or hover over and choose the edit icon to change the name.
   - For queries, select a data source from the dropdown.
   - Add one or more queries or expressions.
   - For each expression, select either **Classic condition** to create a single alert rule, or choose from **Math**, **Reduce**, **Resample** options to generate separate alerts for each series. For details on these options, see Single and multidimensional rules.
   - Choose **Run queries** to verify that the query is successful.

5. In **Step 3**, add conditions.

   - From the **Condition** dropdown, select the query or expression to initiate the alert rule.
   - For **Evaluate every**, specify the frequency of evaluation. Must be a multiple of 10 seconds. For example, `1m`, `30s`.
   - For **Evaluate for**, specify the duration for which the condition must be true before an alert is initiated.

     > **Note**
     >
     > After a condition is breached, the alert goes into `Pending` state. If the condition remains breached for the duration specified, the alert transitions to the `Firing` state. If it is no longer met, it reverts to the `Normal` state.

   - In **Configure no data and error handling**, configure alerting behavior in the absence of data. use the guidelines in Handling no data or error cases.
   - Choose **Preview alerts** to check the result of running the query at this moment. Preview excludes no data and error handling conditions.

6. In **Step 4**, add additional metadata associated with the rule.
   - Add a description and summary to customize alert messages. Use the guidelines in Labels and annotations.
   - Add Runbook URL, panel, dashboard, and alert IDs.
   - Add custom labels.
7. Choose **Save** to save the rule or **Save and exit** to save the rule and go back to the **Alerting** page.

After you have created your rule, you can create a notification for your rule. For more information about notifications, see Manage your alert notifications.

Single and multidimensional rules

For Grafana managed alert rules, you can create a rule with a classic condition or you can create a multidimensional rule.

Single dimensional rule (classic condition)

Use a classic condition expression to create a rule that initiates a single alert when its condition is met. For a query that returns multiple series, Grafana does not track the alert state of each series. As a result, Grafana sends only a single alert even when alert conditions are met for multiple series.

For more information about how to format expressions, see Expressions in the *Grafana documentation*.

Multidimensional rule

To generate a separate alert instance for each series returned in the query, create a multidimensional rule.

> **Note**
>
> Each alert instance generated by a multi-dimensional rule counts toward your total quota of alerts. Rules are not evaluated when you reach your quota of alerts. For more information about quotas for multi-dimensional rules, see Quota reached errors.

To create multiple instances from a single rule, use `Math`, `Reduce`, or `Resample` expressions to create a multidimensional rule. For example, you can:

- Add a `Reduce` expression for each query to aggregate values in the selected time range into a single value. (Not needed for rules using numeric data).
- Add a `Math` expression with the condition for the rule. This is not needed in case a query or a reduce expression already returns 0 if rule should not initiate an alert, or a positive number if it should initiate an alert.

  Some examples:

  - `$B > 70` if it should initiate an alert in case value of B query/expression is more than 70.
  - `$B < $C * 100` in case it should initiate an alert if value of B is less than value of C multiplied by 100. If queries being compared have multiple series in their results, series from different queries are matched if they have the same labels, or one is a subset of the other.

> **Note**
>
> Grafana does not support alert queries with template variables. More information is available at the community page Template variables are not supported in alert queries while setting up Alert.

Performance considerations for multidimensional rules

Each alert instance counts toward the alert quota. Multidimensional rules that create more instances than can be accommodated within the alert quota are not evaluated and return a quota error. For more information, see Quota reached errors.

Multidimensional alerts can have a high impact on the performance of your Grafana workspace, as well as on the performance of your data sources as Grafana queries them to evaluate your alert rules. The following considerations can be helpful as you are trying to optimize the performance of your monitoring system.

- **Frequency of rule evaluation** – The **Evaluate Every** property of an alert rule controls the frequency of rule evaluation. We recommend using the lowest acceptable evaluation frequency.
- **Result set cardinality** – The number of alert instances you create with a rule affects its performance. Suppose you are monitoring API response errors for every API path, on every VM in your fleet. This set has a cardinality of the number of paths multiplied by the number of VMs.

You can reduce the cardinality of the result set, for example, by monitoring total errors per VM instead of per path per VM.

- **Complexity of the query** – Queries that data sources can process and respond to quickly consume fewer resources. Although this consideration is less important than the other considerations listed above, if you have reduced those as much as possible, looking at individual query performance could make a difference. You should also be aware of the performance impact that evaluating these rules has on your data sources. Alerting queries are often the vast majority of queries handled by monitoring databases, so the same load factors that affect the Grafana instance affect them as well.

Quota reached errors

There is a quota for the number of alert instances you can have within a single workspace. When you reach that number, you can no longer create new alert rules in that workspace. With multidimensional alerts, the number of alert instances can vary over time.

The following are important to remember when working with alert instances.

- If you create only single-dimensional rules, each rule is a single alert instance. You can create the same number of rules in a single workspace as your alert-instance quota, and no more.
- Multidimensional rules create multiple alert instances, however, the number is not known until they are evaluated. For example, if you create an alert rule that tracks the CPU usage of your Amazon EC2 instances, there might be 50 EC2 instances when you create it (and therefore 50 alert instances), but if you add 10 more EC2 instances a week later, the next evaluation has 60 alert instances.

  The number of alert instances is evaluated when you create a multidimensional alert, and you can't create one that immediately puts you over your alert instance quota. Because the number of alert instances can change, your quota is checked each time that your rules are evaluated.
- At rule evaluation time, if a rule causes you to go beyond your quota for alert instances, that rule is not evaluated until an update is made to the alert rule that brings the total count of alert instances below the service quota. When this happens, you receive an alert notification letting you know that your quota has been reached (the notification uses the notification policy for the rule being evaluated). The notification includes an `Error` annotation with the value `QuotaReachedError`.
- A rule that causes a `QuotaReachedError` stops being evaluated. Evaluation is only resumed when an update is made and the evaluation after the update does not itself cause a

`QuotaReachedError`. A rule that is not being evaluated shows the **Quota reached** error in the Grafana console.

- You can lower the number of alert instances by removing alert rules, or by editing multidimensional alerts to have fewer alert instances (for example, by having one alert on errors per VM, rather than one alert on error per API in a VM).
- To resume evaluations, update the alert and save it. You can update it to lower the number of alert instances, or if you have made other changes to lower the number of alert instances, you can save it with no changes. If it can be resumed, it is. If it causes another `QuotaReachedError`, you are not able to save it.
- When an alert is saved and resumes evaluation without going over the alerts quota, the **Quota reached** error can continue to show in the Grafana console for some time (up to its evaluation interval), however, the alert rule evaluation does start and alerts are sent if the rule threshold is met.
- For details on the alerts quota, as well as other quotas, see Amazon Managed Grafana service quotas.

Handling no data or error cases

Choose options for how to handle alerting behavior in the absence of data or when there are errors.

The options for handling no data are listed in the following table.

No Data option	Behavior
No Data	Create an alert `DatasourceNoData` with the name and UID of the alert rule, and UID of the data source that returned no data as labels.
Alerting	Set alert rule state to `Alerting`.
OK	Set alert rule state to `Normal`.

The options for handling error cases are listed in the following table.

Error or timeout option	Behavior
Alerting	Set alert rule state to `Alerting`
OK	Set alert rule state to `Normal`
Error	Create an alert `DatasourceError` with the name and UID of the alert rule, and UID of the data source that returned no data as labels.

Creating Grafana Mimir or Loki managed alert rules

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Using Grafana, you can create alerting rules for an external Grafana Mimir or Loki instance.

Note

Grafana Mimir can connect to Amazon Managed Service for Prometheus and Prometheus data sources.

Prerequisites

- Verify that you have write permissions to the Prometheus data source. Otherwise, you are not able to create or update Cortex managed alerting rules.
- For Grafana Mimir and Loki data sources, enable the ruler API by configuring their respective services.

- **Loki** – The `local` rule storage type, default for the Loki data source, supports only viewing of rules. To edit rules, configure one of the other storage types.
- **Grafana Mimir** – Use the legacy `/api/prom` prefix, not `/prometheus`. The Prometheus data source supports both Grafana Mimir and Prometheus, and Grafana expects that both the Query API and Ruler API are under the same URL. You cannot provide a separate URL for the Ruler API.

> **Note**
>
> If you do not want to manage alerting rules for a particular Loki or Prometheus data source, go to its settings and clear the **Manage alerts via Alerting UI** checkbox.

To add a Grafana Mimir or Loki managed alerting rule

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page listing existing alerts.
2. Choose **Create alert rule**.
3. In **Step 1**, choose the rule type, and details, as follows:
   - Choose **Mimir or Loki alert**.
   - In **Rule name**, add a descriptive name. This name is displayed in the alert rules list. It is also the `alertname` label for every alert instance that is created from this rule.
   - From the **Select data source** dropdown, select a Prometheus, or Loki data source.
   - From the **Namespace** dropdown, select an existing rule namespace. Otherwise, choose **Add new** and enter a name to create one. Namespaces can contain one or more rule groups and only have an organizational purpose. For more information, see Cortex or Loki rule groups and namespaces.
   - From the **Group** dropdown, select an existing group within the selected namespace. Otherwise, choose **Add new** and enter a name to create one. Newly created rules are appended to the end of the group. Rules within a group run sequentially at a regular interval, with the same evaluation time.
4. In **Step 2**, add the query to evaluate.

The value can be a PromQL or LogQL expression. The rule initiates an alert if the evaluation result has at least one series with a value that is greater than 0. An alert is created for each series.

5. In **Step 3**, specify the alert evaluation interval.

   In the **For** text box of the condition, specify the duration for which the condition must be true before the alert is initiated. If you specify `5m`, the conditions must be true for five minutes before the alert is initiated.

   > **Note**
   >
   > After a condition is met, the alert goes into `Pending` state. If the condition remains active for the duration specified, the alert transitions to the `Firing` state. If it is no longer met, it reverts to the `Normal` state.

6. In **Step 4**, add additional metadata associated with the rule.

   - Add a description and summary to customize alert messages. Use the guidelines in Labels and annotations.
   - Add Runbook URL, panel, dashboard, and alert IDs.
   - Add custom labels.

7. Choose **Preview alerts** to evaluate the rule and see what alerts it would produce. It displays a list of alerts with state and value of each one.
8. Choose **Save** to save the rule or **Save and exit** to save the rule and go back to the **Alerting** page.

After you have created your rule, you can create a notification for your rule. For more information about notifications, see Manage your alert notifications.

Creating Grafana Mimir or Loki managed recording rules

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can create and manage recording rules for an external Grafana Mimir or Loki instance. Recording rules calculate frequently needed expressions or computationally expensive expressions in advance and save the result as a new set of time series. Querying this new time series is faster, especially for dashboards since they query the same expression every time the dashboards refresh.

Prerequisites

For Grafana Mimir and Loki data sources, enable the ruler API by configuring their respective services.

- **Loki** – The `local` rule storage type, default for the Loki data source, supports only viewing of rules. To edit rules, configure one of the other storage types.
- **Grafana Mimir** – When configuring a data source to point to Grafana Mimir, use the legacy `/api/prom` prefix, not `/prometheus`. The Prometheus data source supports both Grafana Mimir and Prometheus, and Grafana expects that both the Query API and Ruler API are under the same URL. You cannot provide a separate URL for the Ruler API.

Note

If you do not want to manage alerting rules for a particular Loki or Prometheus data source, go to its settings and clear the **Manage alerts via Alerting UI** check box.

To add a Grafana Mimir or Loki managed recording rule

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page listing existing alerts.
2. Choose **Create alert rule**.
3. In **Step 1**, add the rule type, rule name, and storage location, as follows.
   - Select the **Mimir or Loki recording rule** option.

- In **Rule name**, add a descriptive name. This name is displayed in the alert rules list. It is also the `alertname` label for every alert instance that is created from this rule.
- From the **Select data source** dropdown, select a Prometheus, or Loki data source.
- From the **Namespace** dropdown, select an existing rule namespace. Otherwise, choose **Add new** and enter a name to create one. Namespaces can contain one or more rule groups and only have an organizational purpose. For more information, see Cortex or Loki rule groups and namespaces.
- From the **Group** dropdown, select an existing group within the selected namespace. Otherwise, choose **Add new** and enter a name to create one. Newly created rules are appended to the end of the group. Rules within a group run sequentially at a regular interval, with the same evaluation time.

4. In **Step 2**, add the query to evaluate.

   The value can be a PromQL or LogQL expression. The rule initiates an alert if the evaluation result has at least one series with a value that is greater than 0. An alert is created for each series.

5. In **Step 3**, add additional metadata associated with the rule.

   - Add a description and summary to customize alert messages. Use the guidelines in Annotations and labels for alerting rules.
   - Add Runbook URL, panel, dashboard, and alert IDs.
   - Add custom labels.

6. Choose **Save** to save the rule or **Save and exit** to save the rule and go back to the **Alerting** page.

Grafana Mimir or Loki rule groups and namespaces

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You can organize your rules. Rules are created within rule groups, and rule groups are organized into namespaces. The rules within a rule group are run sequentially at a regular interval. The default interval is one minute. You can rename Grafana Mimir or Loki namespaces and rule groups, and edit rule group evaluation intervals.

To edit a rule group or namespace

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page.
2. Navigate to a rule within the rule group or namespace you want to edit.
3. Choose the **Edit** (pen) icon.
4. Make changes to the rule group or namespace.

   > **Note**
   >
   > For namespaces, you can only edit the name. For rule groups, you change the name, or the evaluation interval for rules in the group. For example, you can choose `1m` to have the rules be evaluated once per minute, or `30s` to evaluate once every 30 seconds.

5. Choose **Save changes**.

View and edit alerting rules

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

The **Alerting** page lists alerting rules. By default, rules are grouped by types of data sources. The **Grafana** section lists rules managed by Grafana, and the **Cortex/Loki** section lists rules for Prometheus compatible data sources. You can view alerting rules for Prometheus compatible data sources but you cannot edit them.

The Mimir/Cortex/Loki rules section lists all rules for Mimir, Cortex, or Loki data sources. Cloud alert rules are also listed in this section.

When managing large volumes of alerts, you can use extended alert rule search capabilities to filter on folders, evaluation groups, and rules. Additionally, you can filter alert rules by their properties like labels, state, type, and health.

> **Note**
>
> You can view query definitions for provisioned alerts, but you cannot edit them. Being able to view them allows you to verify that your queries and rule definitions are correct without going back to your provisioning repository for rule definitions.

View alerting rules

Using Grafana alerts, you can view all of your alerts in one page.

To view alerting details

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page. By default, rules are displayed in groups by data source type. You can also view by the current state of each alert (these are described in more detail in the following text).
2. In **View as**, you can toggle between the group and state views by choosing the option you prefer.
3. Choose the arrow next to a row to view more details for that row. The details for a rule include the rule labels, annotations, data sources, and queries, as well as a list of alert instances resulting from the rule.

> **Note**
>
> For more information about understanding the details of your alerts, see State and health of alerting rules.

Group view

Group view shows Grafana alert rules grouped by folder and Loki or Prometheus alert rules grouped by `namespace` + `group`. This is the default rule list view, intended for managing rules.

You can expand each group to view a list of rules in this group. Expand a rule further to view its details. You can also expand action buttons and alerts resulting from the rule to view their details.

State view

State view shows alert rules grouped by state. Use this view to get an overview of which rules are in what state. Each rule can be expanded to view its details. Action buttons and any alerts generated by this rule, and each alert can be further expanded to view its details.

Filter alerting rules

You can filter the alerting rules that appear on the **Alerting** page in several ways.

- You can filter to display the rules that query a specific data source by choosing **Select data sources**, then selecting a data source to filter to.
- You can filter by labels by choosing search criteria in **Search by label**. For example, you could type `environment=production,region=~US|EU,severity!=warning` to filter on production warnings in the US and EU.
- You can filter to display the rules in a specific state by choosing **Filter alerts by state**, and then selecting the state you want to view.

Edit or delete alerting rules

Grafana managed alerting rules can only be edited or deleted by users with Edit permissions for the folder storing the rules. Alerting rules for an external Mimir or Loki instance can be edited or deleted by users with Editor or Admin roles.

To edit or delete a rule

1. Expand a rule until you can see the rule controls for **View**, **Edit**, and **Delete**.
2. Choose **Edit** to open the create rule page. Make updates in the same way that you create a rule. For details, see the instructions in Creating Grafana managed alert rules or Creating Grafana Mimir or Loki managed alert rules.
3. Optionally, choose **Delete** to delete a rule.

Export alert rules

You can export rules to YAML or JSON in the Grafana workspace, by choosing **Export**. It will give you the option to define a new rule, then export it. You can create a rule using the UI and then export it for use in the provisioning API or terraform scripts.

> **Note**
>
> This is supported in both the Grafana workspace and the provisioning interface.

Manage your alert notifications

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Choosing how, when, and where to send your alert notifications is an important part of setting up your alerting system. These decisions will have a direct impact on your ability to resolve issues quickly and not miss anything important.

As a first step, define your *contact points*; where to send your alert notifications to. A contact point can be a set of destinations for matching notifications. Add notification templates to contact points for reuse and consistent messaging in your notifications.

Next, create a *notification policy*, which is a set of rules for where, when and how your alerts are routed to contact points. In a notification policy, you define where to send your alert notifications by choosing one of the contact points you created. Add mute timings to your notification policy. A *mute timing* is a recurring interval of time during which you don't want any notifications to be sent out.

When an alert rule is evaluated, the alert ruler sends alert instances to the Alertmanager — one alert rule can trigger multiple individual *alert instances*.

The Alertmanager receives these alert instances and then handles mute timings, groups alerts, and sends notifications to your contact points as defined in the notification policy.

Topics

- Alertmanager
- Working with contact points
- Working with notification policies
- Customize notifications
- Silencing alert notifications for Prometheus data sources
- Mute timings
- View and filter by alert groups
- View notification errors

Alertmanager

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Alertmanager enables you to quickly and efficiently manage and respond to alerts. It receives alerts, handles mutings, inhibition, grouping, and routing by sending notifications out via your channel of choice, for example, email or Slack.

In Grafana, you can use the Grafana Alertmanager, or an external Alertmanager. You can also run multiple alertmanagers; your decision depends on your set up and where your alerts are being generated.

Grafana Alertmanager

Grafana Alertmanager is an internal Alertmanager that is pre-configured and available for selection by default if you run Grafana on-premise or open-source.

The Grafana Alertmanager can receive alerts from Grafana, but it cannot receive alerts from outside Grafana, for example, from Mimir or Loki.

> **Note**
>
> Inhibition rules are not supported in the Grafana Alertmanager.

External Alertmanager

If you want to use a single alertmanager to receive all your Grafana, Loki, Mimir, and Prometheus alerts, you can set up Grafana to use an external Alertmanager. This external Alertmanager can be configured and administered from within Grafana itself.

Here are two examples of when you might want to configure your own external alertmanager and send your alerts there instead of the Grafana Alertmanager:

1. You already have alertmanagers on-premise in your own Cloud infrastructure that you have set up and still want to use, because you have other alert generators, such as Prometheus.
2. You want to use both Prometheus on-premise and hosted Grafana to send alerts to the same alertmanager that runs in your Cloud infrastructure.

Alertmanagers are visible from the dropdown menu on the Alerting Contact Points, and Notification Policies pages.

If you are provisioning your data source, set the flag `handleGrafanaManagedAlerts` in the `jsonData` field to `true` to send Grafana-managed alerts to this Alertmanager.

Working with contact points

> This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Use contact points to define how your contacts are notified when an alert is initiated. A contact point can have one or more contact point integrations, for example, Amazon Simple Notification Service or Slack. When an alert is initiated, a notification is sent to all contact point integrations listed for a contact point. Optionally, use notification templates to customize the notification messages for the contact point types.

> **Note**
>
> You can create and edit contact points for Grafana managed alerts. Contact points for Alertmanager alerts are read-only.

Working with contact points

The following procedures detail how to add, edit, test, and delete contact points.

To add a contact point

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page.
2. Choose **Contact points**, then **Add contact point**.
3. From the **Alertmanager** dropdown, select an Alertmanager. The Grafana Alertmanager is selected by default.
4. Enter a **Name** for the contact point.
5. From **Contact point integration**, choose a type, and the mandatory fields based on that type. For example, if you choose Slack, enter the Slack channels and users who should be contacted.
6. If available for the contact point you selected, choose any desired **Optional settings** to specify additional settings.
7. Under **Notification settings**, optionally select **Disable resolved message** if you do not want to be notified when an alert resolves.
8. If your contact point needs more contact points types, you can choose **Add contact point integration** and repeat the steps for each contact point type needed.
9. Choose **Save contact point** to save your changes.

To edit a contact point

1. Choose **Contact points** to see a list of existing contact points.

2. Select the contact point to edit, then choose the **Edit** icon (pen).
3. Make any necessary changes, and then choose **Save contact point** to save your changes.

After your contact point is created, you can send a test notification to verify that it is configured properly.

To send a test notification

1. Choose **Contact points** to open the list of existing contact points.
2. Select the contact point to test, then choose the **Edit** icon (pen).
3. Select the **Test** icon (paper airplane).
4. Choose whether to send a predefined test notification or choose **Custom** to add your own custom annotations and labels in the test notification.
5. Choose **Send test notification** to test the alert with the given contact points.

You can delete contact points that are not in use by a notification policy.

To delete a contact point

1. Choose **Contact points** to open the list of existing contact points.
2. Select the contact point to delete, then choose the **Delete** icon (trash can).
3. In the confirmation dialog box, choose **Yes, delete**.

> **Note**
>
> If the contact point is in use by a notification policy, you must delete the notification policy or edit it to use a different contact point before deleting the contact point.

List of supported notifiers

Name	Type
Amazon SNS	`sns`
OpsGenie	`opsgenie`

Name	Type
Pager Duty	`pagerduty`
Slack	`slack`
VictorOps	`victorops`

Working with notification policies

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Notification policies determine how alerts are routed to contact points. Policies have a tree structure, where each policy can have one or more child policies. Each policy, except for the root policy, can also match specific alert labels. Each alert is evaluated by the root policy and then by each child policy. If you enable the `Continue matching subsequent sibling nodes` option for a specific policy, then evaluation continues even after one or more matches. A parent policy's configuration settings and contact point information govern the behavior of an alert that does not match any of the child policies. A root policy governs any alert that does not match a specific policy.

Note

You can create and edit notification policies for Grafana managed alerts. Notification policies for Alertmanager alerts are read-only.

Grouping notifications

Grouping categorizes alert notifications of similar nature into a single funnel. This allows you to control alert notifications during larger outages when many parts of a system fail at once causing a high number of alerts to initiate simultaneously.

Grouping example

Suppose you have 100 services connected to a database in different environments. These services are differentiated by the label `env=environmentname`. An alert rule is in place to monitor whether your services can reach the database. The alert rule creates alerts named `alertname=DatabaseUnreachable`.

If a network partition occurs, where half of your services can no longer reach the database, 50 different alerts are initiated. For this situation, you want to receive a single-page notification (as opposed to 50) with a list of the environments that are affected.

You can configure grouping to be `group_by: [alertname]` (not using the `env` label, which is different for each service). With this configuration in place, Grafana sends a single compact notification that has all the affected environments for this alert rule.

Special Groups

Grafana has two special groups. The default group, `group_by: null` groups *all* alerts together into a single group. You can also use a special label named `...` to group alerts by all labels, effectively disabling grouping, and sending each alert into its own group.

Working with notifications

The following procedures show you how to create and manage notification policies.

To edit the root notification policy

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page.
2. Choose **Notification policies**.
3. From the **Alertmanager** dropdown, select the Alertmanager you want to edit.
4. In the **Root policy** section, choose the **Edit** icon (pen).
5. In **Default contact point**, update the contact point where notifications should be sent for rules when alert rules do not match any specific policy.
6. In **Group by**, choose the labels (or special groups) to group alerts by.

7. In **Timing options**, select from the following options.

   - **Group wait** – Time to wait to buffer alerts of the same group before sending an initial notification. The default is 30 seconds.
   - **Group interval** – Minimum time interval between two notifications for a group. The default is 5 minutes.
   - **Repeat interval** – Minimum time interval before resending a notification if no new alerts were added to the group. The default is 4 hours.
8. Choose **Save** to save your changes.

To add a new, top-level specific policy

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page.
2. Choose **Notification policies**.
3. From the **Alertmanager** dropdown, select the Alertmanager you want to edit.
4. In the **Specific routing** section, choose **New specific policy**.
5. In the **Matching labels** section, add one or more matching alert labels. More information about label matching is later in this topic.
6. In **Contact point**, add the contact point to send notifications to if the alert matches this specific policy. Nested policies override this contact point.
7. Optionally, enable **Continue matching subsequent sibling nodes** to continue matching sibling policies even after the alert matched the current policy. When this policy is enabled, you can get more than one notification for the same alert.
8. Optionally select **Override grouping** to specify a grouping different from the root policy.
9. Optionally select **Override general timings** to override the timing options in the group notification policy.
10. Choose **Save policy** to save your changes.

To add a nested policy

1. Expand the specific policy you want to create a nested policy under.
2. Choose **Add nested policy**, then add the details (as when adding a top-level specific policy).
3. Choose **Save policy** to save your changes.

To edit a specific policy

1. From the **Alerting** page, choose **Notification policies** to open the page that listing existing policies.
2. Select the policy that you want to edit, then choose the **Edit** icon (pen).
3. Make any changes (as when adding a top-level specific policy).
4. Choose **Save policy**.

Searching for policies

You can search within the tree of policies by *Label matchers* or *contact points*.

- To search by contact point, enter a partial or full name of a contact point in the **Search by contact point** field.
- To search by label, enter a valid label matcher in the **Search by label** field. Multiple matchers can be entered, separated by a comma. For example, a valid matcher input could be `severity=high, region=~EMEA|NA`.

> **Note**
>
> When searching by label, all matched policies will be exact matches. Partial matches and regex-style matches are not supported.

How label matching works

A policy matches an alert if the alert's labels match all the *Matching Labels* specified on the policy.

- **Label** – The name of the label to match. It must exactly match the label name of the alert.
- **Operator** – The operator used to compare the label value with the matching label value. The available operators are:
  - = Select labels whose value exactly matches the provided string.
  - != Select labels whose value does not match the provided string.
  - =~ Select labels whose value match the regex interpreted value of the provided string (the provided string is interpreted as a regular expression.
  - != Select labels that do not match the provided regular expression.

- **Value** – The value to match the label value to. It can match as a string or as a regular expression, depending on the operator chosen.

Customize notifications

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Customize your notifications with notifications templates.

You can use notification templates to change the title, message, and format of the message in your notifications.

Notification templates are not tied to specific contact point integrations, such as email or Slack. However, you can choose to create separate notification templates for different contact point integrations.

You can use notification templates to:

- Add, remove, or re-order information in the notification including the summary, description, labels and annotations, values, and links
- Format text in bold and italic, and add or remove line breaks

You cannot use notification templates to:

- Change the design of notifications in instant messaging services such as Slack and Microsoft Teams

Topics

- Using Go's templating language

- Create notification templates
- Template reference

Using Go's templating language

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

You write notification templates in Go's templating language, text/template.

This section provides an overview of Go's templating language and writing templates in text/template.

Dot

In text/template there is a special cursor called dot, and is written as `.`. You can think of this cursor as a variable whose value changes depending where in the template it is used. For example, at the start of a notification template `.` refers to the `ExtendedData` object, which contains a number of fields including `Alerts`, `Status`, `GroupLabels`, `CommonLabels`, `CommonAnnotations` and `ExternalURL`. However, dot might refer to something else when used in a `range` over a list, when used inside a `with`, or when writing feature templates to be used in other templates. You can see examples of this in Create notification templates, and all data and functions in the Template reference.

Opening and closing tags

In text/template, templates start with `{{` and end with `}}` irrespective of whether the template prints a variable or runs control structures such as if statements. This is different from other templating languages such as Jinja where printing a variable uses `{{` and `}}` and control structures use `{%` and `%}`.

Print

To print the value of something use {{ and }}. You can print the value of dot, a field of dot, the result of a function, and the value of a variable. For example, to print the Alerts field where dot refers to ExtendedData you would write the following:

```
{{ .Alerts }}
```

Iterate over alerts

To print just the labels of each alert, rather than all information about the alert, you can use a range to iterate the alerts in ExtendedData:

```
{{ range .Alerts }}
{{ .Labels }}
{{ end }}
```

Inside the range dot no longer refers to ExtendedData, but to an Alert. You can use {{ .Labels }} to print the labels of each alert. This works because {{ range .Alerts }} changes dot to refer to the current alert in the list of alerts. When the range is finished dot is reset to the value it had before the start of the range, which in this example is ExtendedData:

```
{{ range .Alerts }}
{{ .Labels }}
{{ end }}
{{/* does not work, .Labels does not exist here */}}
{{ .Labels }}
{{/* works, cursor was reset */}}
{{ .Status }}
```

Iterate over annotations and labels

Let's write a template to print the labels of each alert in the format The name of the label is $name, and the value is $value, where $name and $value contain the name and value of each label.

Like in the previous example, use a range to iterate over the alerts in .Alerts such that dot refers to the current alert in the list of alerts, and then use a second range on the sorted labels so dot is updated a second time to refer to the current label. Inside the second range use .Name and .Value to print the name and value of each label:

```
{{ range .Alerts }}
{{ range .Labels.SortedPairs }}
The name of the label is {{ .Name }}, and the value is {{ .Value }}
{{ end }}
{{ range .Annotations.SortedPairs }}
The name of the annotation is {{ .Name }}, and the value is {{ .Value }}
{{ end }}
{{ end }}
```

If statements

You can use if statements in templates. For example, to print `There are no alerts` if there are no alerts in `.Alerts` you would write the following:

```
{{ if .Alerts }}
There are alerts
{{ else }}
There are no alerts
{{ end }}
```

With

With is similar to if statements, however unlike if statements, `with` updates dot to refer to the value of the with:

```
{{ with .Alerts }}
There are {{ len . }} alert(s)
{{ else }}
There are no alerts
{{ end }}
```

Variables

Variables in text/template must be created within the template. For example, to create a variable called `$variable` with the current value of dot you would write the following:

```
{{ $variable := . }}
```

You can use `$variable` inside a range or `with` and it will refer to the value of dot at the time the variable was defined, not the current value of dot.

For example, you cannot write a template that use `{{ .Labels }}` in the second range because here dot refers to the current label, not the current alert:

```
{{ range .Alerts }}
{{ range .Labels.SortedPairs }}
{{ .Name }} = {{ .Value }}
{{/* does not work because in the second range . is a label not an alert */}}
There are {{ len .Labels }}
{{ end }}
{{ end }}
```

You can fix this by defining a variable called `$alert` in the first range and before the second range:

```
{{ range .Alerts }}
{{ $alert := . }}
{{ range .Labels.SortedPairs }}
{{ .Name }} = {{ .Value }}
{{/* works because $alert refers to the value of dot inside the first range */}}
There are {{ len $alert.Labels }}
{{ end }}
{{ end }}
```

Range with index

You can get the index of each alert within a range by defining index and value variables at the start of the range:

```
{{ $num_alerts := len .Alerts }}
{{ range $index, $alert := .Alerts }}
This is alert {{ $index }} out of {{ $num_alerts }}
{{ end }}
```

Define templates

You can define templates that can be used within other templates, using `define` and the name of the template in double quotes. You should not define templates with the same name as other templates, including default templates such as `__subject`, `__text_values_list`, `__text_alert_list`, `default.title` and `default.message`. Where a template has been created with the same name as a default template, or a template in another notification template, Grafana might use either template. Grafana does not prevent, or show an error message, when there are two or more templates with the same name.

```
{{ define "print_labels" }}
{{ end }}
```

Embed templates

You can embed a defined template within your template using `template`, the name of the template in double quotes, and the cursor that should be passed to the template:

```
{{ template "print_labels" . }}
```

Pass data to templates

Within a template dot refers to the value that is passed to the template.

For example, if a template is passed a list of firing alerts then dot refers to that list of firing alerts:

```
{{ template "print_alerts" .Alerts }}
```

If the template is passed the sorted labels for an alert then dot refers to the list of sorted labels:

```
{{ template "print_labels" .SortedLabels }}
```

This is useful when writing reusable templates. For example, to print all alerts you might write the following:

```
{{ template "print_alerts" .Alerts }}
```

Then to print just the firing alerts you could write this:

```
{{ template "print_alerts" .Alerts.Firing }}
```

This works because both `.Alerts` and `.Alerts.Firing` are lists of alerts.

```
{{ define "print_alerts" }}
{{ range . }}
{{ template "print_labels" .SortedLabels }}
{{ end }}
```

```
{{ end }}
```

Comments

You can add comments with {{/* and */}}:

```
{{/* This is a comment */}}
```

To prevent comments from adding line breaks use:

```
{{- /* This is a comment with no leading or trailing line breaks */ -}}
```

Indentation

You can use indentation, both tabs and spaces, and line breaks, to make templates more readable:

```
{{ range .Alerts }}
  {{ range .Labels.SortedPairs }}
    {{ .Name }} = {{ .Value }}
  {{ end }}
{{ end }}
```

However, indentation in the template will also be present in the text. Next we will see how to remove it.

Remove spaces and line breaks

In text/template use {{- and -}} to remove leading and trailing spaces and line breaks.

For example, when using indentation and line breaks to make a template more readable:

```
{{ range .Alerts }}
  {{ range .Labels.SortedPairs }}
    {{ .Name }} = {{ .Value }}
  {{ end }}
{{ end }}
```

The indentation and line breaks will also be present in the text:

```
    alertname = "Test"
```

```
    grafana_folder = "Test alerts"
```

You can remove the indentation and line breaks from the text changing `}}` to `-}}` at the start of each range:

```
{{ range .Alerts -}}
  {{ range .Labels.SortedPairs -}}
    {{ .Name }} = {{ .Value }}
  {{ end }}
{{ end }}
```

The indentation and line breaks in the template are now absent from the text:

```
alertname = "Test"
grafana_folder = "Test alerts"
```

Create notification templates

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Create reusable notification templates to send to your contact points.

You can add one or more templates to your notification template.

Your notification template name must be unique. You cannot have two templates with the same name in the same notification template or in different notification templates. Avoid defining templates with the same name as default templates, such as: `__subject`, `__text_values_list`, `__text_alert_list`, `default.title` and `default.message`.

In the Contact points tab, you can see a list of your notification templates.

Creating notification templates

To create a notification template

1. Click **Add template**.
2. Choose a name for the notification template, such as `email.subject`.
3. Write the content of the template in the content field.

   For example:

   ```
   {{ if .Alerts.Firing -}}
      {{ len .Alerts.Firing }} firing alerts
      {{ end }}
      {{ if .Alerts.Resolved -}}
      {{ len .Alerts.Resolved }} resolved alerts
      {{ end }}
   ```

4. Click Save.

   `{{ define "email.subject" }}` (where `email.subject` is the name of your template) and `{{ end }}` is automatically added to the start and end of the content.

To create a notification template that contains more than one template:

1. Click **Add Template**.
2. Enter a name for the overall notification template. For example, `email`.
3. Write each template in the Content field, including `{{ define "`*`name-of-template`*`" }}` and `{{ end }}` at the start and end of each template. You can use descriptive names for each of the templates in the notification template, for example, `email.subject` or `email.message`. In this case, do not reuse the name of the notification template you entered above.

   The following sections show detailed examples for templates you might create.
4. Click Save.

Creating a template for the subject of an email

Create a template for the subject of an email that contains the number of firing and resolved alerts, as in this example:

```
1 firing alerts, 0 resolved alerts
```

To create a template for the subject of an email

1. Create a template called `email.subject` with the following content:

   ```
   {{ define "email.subject" }}
   {{ len .Alerts.Firing }} firing alerts, {{ len .Alerts.Resolved }} resolved alerts
   {{ end }}
   ```

2. Use the template when creating your contact point integration by putting it into the **Subject** field with the `template` keyword.

   ```
   {{ template "email.subject" . }}
   ```

Creating a template for the message of an email

Create a template for the message of an email that contains a summary of all firing and resolved alerts, as in this example:

```
There are 2 firing alerts, and 1 resolved alerts

Firing alerts:

- alertname=Test 1 grafana_folder=GrafanaCloud has value(s) B=1
- alertname=Test 2 grafana_folder=GrafanaCloud has value(s) B=2

Resolved alerts:

- alertname=Test 3 grafana_folder=GrafanaCloud has value(s) B=0
```

To create a template for the message of an email

1. Create a notification template called `email` with two templates in the content: `email.message_alert` and `email.message`.

The `email.message_alert` template is used to print the labels and values for each firing and resolved alert while the `email.message` template contains the structure of the email.

```
{{- define "email.message_alert" -}}
{{- range .Labels.SortedPairs }}{{ .Name }}={{ .Value }} {{ end }} has value(s)
{{- range $k, $v := .Values }} {{ $k }}={{ $v }}{{ end }}
{{- end -}}

{{ define "email.message" }}
There are {{ len .Alerts.Firing }} firing alerts, and {{ len .Alerts.Resolved }}
 resolved alerts

{{ if .Alerts.Firing -}}
Firing alerts:
{{- range .Alerts.Firing }}
- {{ template "email.message_alert" . }}
{{- end }}
{{- end }}

{{ if .Alerts.Resolved -}}
Resolved alerts:
{{- range .Alerts.Resolved }}
- {{ template "email.message_alert" . }}
{{- end }}
{{- end }}

{{ end }}
```

2. Use the template when creating your contact point integration by putting it into the **Text Body** field with the `template` keyword.

```
{{ template "email.message" . }}
```

Creating a template for the title of a Slack message

Create a template for the title of a Slack message that contains the number of firing and resolved alerts, as in the following example:

```
1 firing alerts, 0 resolved alerts
```

To create a template for the title of a Slack message

1. Create a template called `slack.title` with the following content:

```
{{ define "slack.title" }}
{{ len .Alerts.Firing }} firing alerts, {{ len .Alerts.Resolved }} resolved alerts
{{ end }}
```

2. Use the template when creating your contact point integration by putting it into the **Title** field with the `template` keyword.

```
{{ template "slack.title" . }}
```

Creating a template for the content of a Slack message

Create a template for the content of a Slack message that contains a description of all firing and resolved alerts, including their labels, annotations, and Dashboard URL:

```
1 firing alerts:

[firing] Test1
Labels:
- alertname: Test1
- grafana_folder: GrafanaCloud
Annotations:
- description: This is a test alert
Go to dashboard: https://example.com/d/dlhdLqF4z?orgId=1

1 resolved alerts:

[firing] Test2
Labels:
- alertname: Test2
- grafana_folder: GrafanaCloud
Annotations:
- description: This is another test alert
Go to dashboard: https://example.com/d/dlhdLqF4z?orgId=1
```

To create a template for the content of a Slack message

1. Create a template called `slack` with two templates in the content: `slack.print_alert` and `slack.message`.

   The `slack.print_alert` template is used to print the labels, annotations, and DashboardURL while the `slack.message` template contains the structure of the notification.

```
{{ define "slack.print_alert" -}}
[{{.Status}}] {{ .Labels.alertname }}
Labels:
{{ range .Labels.SortedPairs -}}
- {{ .Name }}: {{ .Value }}
{{ end -}}
{{ if .Annotations -}}
Annotations:
{{ range .Annotations.SortedPairs -}}
- {{ .Name }}: {{ .Value }}
{{ end -}}
{{ end -}}
{{ if .DashboardURL -}}
  Go to dashboard: {{ .DashboardURL }}
{{- end }}
{{- end }}

{{ define "slack.message" -}}
{{ if .Alerts.Firing -}}
{{ len .Alerts.Firing }} firing alerts:
{{ range .Alerts.Firing }}
{{ template "slack.print_alert" . }}
{{ end -}}
{{ end }}
{{ if .Alerts.Resolved -}}
{{ len .Alerts.Resolved }} resolved alerts:
{{ range .Alerts.Resolved }}
{{ template "slack.print_alert" .}}
{{ end -}}
{{ end }}
{{- end }}
```

2. Use the template when creating your contact point integration by putting it into the **Text Body** field with the `template` keyword.

```
{{ template "slack.message" . }}
```

Template both email and Slack with shared templates

Instead of creating separate notification templates for each contact point, such as email and Slack, you can share the same template.

For example, if you want to send an email with this subject and Slack message with this title `1 firing alerts, 0 resolved alerts`, you can create a shared template.

To create a shared template

1. Create a template called `common.subject_title` with the following content:

```
{{ define "common.subject_title" }}
{{ len .Alerts.Firing }} firing alerts, {{ len .Alerts.Resolved }} resolved alerts
{{ end }}
```

2. For email, run the template from the subject field in your email contact point integration:

```
{{ template "common.subject_title" . }}
```

3. For Slack, run the template from the title field in your Slack contact point integration:

```
{{ template "common.subject_title" . }}
```

Using notification templates

Use templates in contact points to customize your notifications.

To use a template when creating a contact point

1. From the **Alerting** menu, choose **Contact points** to see a list of existing contact points.
2. Choose **Add contact point**. Alternately, you can edit an existing contact point by choosing the **Edit** icon (pen) next to the contact point you wish to edit.
3. Enter the templates you wish to use in one or more field, such as **Message** or **Subject**. To enter a template, use the form `{{ template "template_name" . }}`, replacing *`template_name`* with the name of the template you want to use.

4. Click **Save contact point**.

Template reference

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

This section provides reference information for creating your templates.

Template data

The following data is passed to message templates.

Name	Type	Notes
`Receiver`	string	Name of the contact point that the notification is being sent to.
`Status`	string	firing if at least one alert is firing, otherwise resolved.
`Alerts`	Alert	List of alert objects that are included in this notification (see below).
`GroupLabels`	KeyValue	Labels these alerts were grouped by.
`CommonLabels`	KeyValue	Labels common to all the alerts included in this notification.

Name	Type	Notes
`CommonAnnotations`	KeyValue	Annotations common to all the alerts included in this notification.
`ExternalURL`	string	Back link to the Grafana that sent the notification. If using external Alertmanager, back link to this Alertmanager.

The `Alerts` type exposes two functions for filtering the alerts returned.

- `Alerts.Firing` – Returns a list of firing alerts.
- `Alerts.Resolved` – Returns a list of resolved alerts.

Alert (type)

The alert type contains the following data.

Name	Type	Notes
Status	string	`firing` or `resolved`.
Labels	KeyValue	A set of labels attached to the alert.
Annotations	KeyValue	A set of annotations attached to the alert.
Values	KeyValue	The values of all expressions, including Classic Conditions
StartsAt	time.Time	Time the alert started firing.
EndsAt	time.Time	Only set if the end time of an alert is known. Otherwise set to a configurable timeout

Name	Type	Notes
		period from the time since the last alert was received.
GeneratorURL	string	A back link to Grafana or external Alertmanager.
SilenceURL	string	A link to silence the alert (with labels for this alert pre-filled). Only for Grafana managed alerts.
DashboardURL	string	Link to grafana dashboard, if alert rule belongs to one. Only for Grafana managed alerts.
PanelURL	string	Link to grafana dashboard panel, if alert rule belongs to one. Only for Grafana managed alerts.
Fingerprint	string	Fingerprint that can be used to identify the alert.
ValueString	string	A string that contains the labels and value of each reduced expression in the alert.

ExtendedData

The ExtendedData object contains the following properties.

Name	Kind	Description	Example
Receiver	`string`	The name of the contact point sending the notification.	`{{ .Receiver }}`
Status	`string`	The status is `firing` if at least one alert is firing, otherwise `resolved`.	`{{ .Status }}`
Alerts	`[]Alert`	List of all firing and resolved alerts in this notification.	`There are {{ len .Alerts }} alerts`
Firing alerts	`[]Alert`	List of all firing alerts in this notification.	`There are {{ len .Alerts.Firing }} firing alerts`
Resolved alerts	`[]Alert`	List of all resolved alerts in this notification.	`There are {{ len .Alerts.Resolved }} resolved alerts`
GroupLabels	`KeyValue`	The labels that group these alerts int his notification.	`{{ .GroupLabels }}`
CommonLabels	`KeyValue`	The labels common to all alerts in this notification.	`{{ .CommonLabels }}`
CommonAnnotations	`KeyValue`	The annotations common to all alerts in this notification.	`{{ .CommonAnnotations }}`

Name	Kind	Description	Example
ExternalURL	`string`	A link to the Grafana workspace or Alertmanager that sent this notification.	`{{ .ExternalURL }}`

KeyValue type

The `KeyValue` type is a set of key/value string pairs that represent labels and annotations.

In addition to direct access of the data stored as a `KeyValue`, there are also methods for sorting, removing, and transforming the data.

Name	Arguments	Returns	Notes	Example
SortedPairs		Sorted list of key and value string pairs		`{{ .Annotations.SortedPairs }}`
Remove	[]string	KeyValue	Returns a copy of the Key/Value map without the given keys.	`{{ .Annotations.Remove "summary" }}`
Names		[]string	List of label names	`{{ .Names }}`
Values		[]string	List of label values	`{{ .Values }}`

Time

Time is from the Go `time` package. You can print a time in a number of different formats. For example, to print the time that an alert fired in the format `Monday, 1st January 2022 at 10:00AM`, you write the following template:

```
{{ .StartsAt.Format "Monday, 2 January 2006 at 3:04PM" }}
```

You can find a reference for Go's time format here.

Template functions

Using template functions you can process labels and annotations to generate dynamic notifications. The following functions are available.

Name	Argument type	Return type	Description
humanize	number or string	string	Converts a number to a more readable format, using metric prefixes.
humanize1024	number or string	string	Like humanize, but uses 1024 as the base rather than 1000.
humanizeDuration	number or string	string	Converts a duration in seconds to a more readable format.
humanizePercentage	number or string	string	Converts a ratio value to a fraction of 100.
humanizeTimestamp	number or string	string	Converts a Unix timestamp in seconds to a more readable format.
title	string	string	strings.Title, capitalizes first character of each word.
toUpper	string	string	strings.ToUpper, converts all characters to upper case.

Name	Argument type	Return type	Description
`toLower`	string	string	strings.ToLower, converts all characters to lower case.
`match`	pattern, text	Boolean	regexp.MatchString Tests for an unanchored regexp match.
`reReplaceAll`	pattern, replacement, text	string	Regexp.ReplaceAllString Regexp substitution, unanchored.
`graphLink`	string - JSON Object with `expr` and `datasource` fields	string	Returns the path to graphical view in Explore for the given expression and data source.
`tableLink`	string - JSON Object with `expr` and `datasource` fields	string	Returns the path to tabular view in Explore for the given expression and data source.
`args`	[]interface{}	map[string]interface{}	Converts a list of objects to a map with keys, for example, arg0, arg1. Use this function to pass multiple arguments to templates.

Name	Argument type	Return type	Description
`externalURL`	nothing	string	Returns a string representing the external URL.
`pathPrefix`	nothing	string	Returns the path of the external URL.

The following table shows examples of using each function.

Function	TemplateString	Input	Expected
humanize	{ humanize $value }	1234567.0	1.235M
humanize1024	{ humanize1024 $value }	1048576.0	1Mi
humanizeDuration	{ humanizeDuration $value }	899.99	14m 59s
humanizePercentage	{ humanizeP ercentage $value }	0.1234567	12.35%
humanizeTimestamp	{ humanizeT imestamp $value }	1435065584.128	2015-06-23 13:19:44.128 +0000 UTC
title	{ $value | title }	aa bB CC	Aa Bb Cc
toUpper	{ $value | toUpper }	aa bB CC	AA BB CC
toLower	{ $value | toLower }	aa bB CC	aa bb cc
match	{ match "a+" $labels.i nstance }	aa	true
reReplaceAll	{{ reReplaceAll "localhost:(.*)"	localhost:3000	my.domain:3000

Function	TemplateString	Input	Expected
	"my.domain:$1" $labels.instance }}		
graphLink	{{ graphLink "{\"expr \": \"up\", \"datasou rce\": \"gdev-pr ometheus\"}" }}		/explore?left=["no w-1h","now","gdev- prometheus",{"data source":"gdev-prom etheus","expr":"up ","instant":false, "range":true}]
tableLink	{{ tableLink "{\"expr \":\"up\", \"datasou rce\":\"gdev-prome theus\"}" }}		/explore?left=["no w-1h","now","gdev- prometheus",{"data source":"gdev-prom etheus","expr":"up ","instant":true," range":false}]
args	{{define "x"}}{{.arg0}} {{.arg1}}{{end}}{{ template "x" (args 1 "2")}}		1 2
externalURL	{ externalURL }		http://localhost/p ath/prefix
pathPrefix	{ pathPrefix }		/path/prefix

Silencing alert notifications for Prometheus data sources

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

For external Alert manager data sources (including Amazon Managed Service for Prometheus), you can suppress alert notifications with a *silence*. A silence only stops notifications from being created: Silences do not prevent alert rules from being evaluated, and they do not stop alerting instances from being shown in the user interface. When you silence an alert, you specify a window of time for it to be suppressed.

You can configure silences for an external Alertmanager data source.

Note

To suppress alert notifications at regular time intervals, or for other data sources, (for example, during regular maintenance periods), use Mute timings rather than silences.

To add a silence

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page.
2. Choose **Silences** to open a page listing existing Working with contact points.
3. Choose the external Alertmanager from the **Alertmanager** dropdown.
4. Select **Add Silence**.
5. Select the start and end date in **Silence start and end** to indicate when the silence should go into effect and when it should end.

   As an alternative to setting an end time, in **Duration**, specify how long the silence is enforced. This automatically updates the end time in the **Silence start and end** field.
6. In the **Name** and **Value** fields, enter one or more *Matching Labels*. Matchers determine which rules the silence applies to. Label matching is discussed in more detail following this procedure.
7. Optionally, add a **Comment**, or modify the **Creator** to set the owner of the silence.
8. Choose **Create** to create the silence.

You can edit an existing silence by choosing the **Edit** icon (pen).

Label matching for alert suppression

When you create a silence, you create a set of *matching labels* as part of the silence. This is a set of rules about labels that must match for the alert to be suppressed. The matching labels consist of three parts:

- **Label** – The name of the label to match. It must exactly match the label name of the alert.
- **Operator** – The operator used to compare the label value with the matching label value. The available operators are:
  - = Select labels whose value exactly matches the provided string.
  - != Select labels whose value does not match the provided string.
  - =~ Select labels whose value match the regex interpreted value of the provided string (the provided string is interpreted as a regular expression).
  - != Select labels that do not match the provided regular expression.
- **Value** – The value to match the label value to. It can match as a string or as a regular expression, depending on the operator chosen.

A silence ends at the indicated end date, but you can manually end the suppression at any time.

To end a silence manually

1. In the **Alerting** page, choose **Silences** to view the list of existing silences.
2. Select the silence that you want to end, and choose **Unsilence**. This ends the alert suppression.

> **Note**
>
> Unsilencing ends the alert suppression, as if the end time was set for the current time. Silences that have ended (automatically or manually) are retained and listed for five days. You cannot remove a silence from the list manually.

Creating a link to the silence creation form

You can create a URL to the silence creation form with details already filled in. Operators can use this to suppress an alarm quickly during an operational event.

When creating a link to a silence form, use a `matchers` query parameter to specify the matching labels, and a `comment` query parameter to specify a comment. The `matchers` parameter requires one or more values in the form `[label][operator][value]`, separated by commas.

Example URL

To link to a silence form, with matching labels `severity=critical` and `cluster!~europe-.*`, with a comment that says `Silencing critical EU alerts`, use a URL like the following. Replace *`mygrafana`* with the hostname of your Grafana instance.

```
https://mygrafana/alerting/silence/new?matchers=severity%3Dcritical%2Ccluster!~europe-
*&comment=Silence%20critical%20EU%20alert
```

To link to a new silence page for an external Alertmanager, add an `alertmanager` query parameter with the Alertmanage data source name, such as `alertmanager=myAlertmanagerdatasource`.

Mute timings

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

A mute timing is a recurring interval of time when no new notifications for a policy are generated or sent. Use them to prevent alerts from firing a specific and reoccurring period, for example, a regular maintenance period.

Similar to silences, mute timings do not prevent alert rules from being evaluated, nor do they stop alert instances from being shown in the user interface. They only prevent notifications from being created.

You can configure Grafana managed mute timings as well as mute timings for an external Alertmanager data source.

Mute timings compared to silences

The following table highlights the differences between mute timings and silences.

Mute timing	Silence
Uses time interval definitions that can reoccur.	Has a fixed start and end time.
Is created and then added to notification policies.	Uses labels to match against an alert to determine whether to silence or not.
Works with Grafana alerting and external Alertmanagers.	Works only with external Alertmanagers.

To create a mute timing

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page.
2. Choose **Notification policies**.
3. From the **Alertmanager** dropdown, select the Alertmanager you want to edit.
4. In the **Mute timings** section, choose the **Add mute timing** button.
5. Choose the time interval for which you want the mute timing to apply.
6. Choose **Submit** to create the mute timing.

To add a mute timing to a notification policy

1. Select the notification policy you would like to add the mute timing to, and choose the **Edit** button.
2. From the **Mute timings** dropdown, select the mute timings you would like to add to the policy.

   Choose the **Save policy** button.

Time intervals

A time interval is a definition for a range of time. If an alert is initiated during this interval it is suppressed. Ranges are supported using `:` (for example, `monday:thursday`). A mute timing can

contain multiple time intervals. A time interval consists of multiple fields (details in the following list), all of which must match in order to suppress the alerts. For example, if you specify days of the week `monday:friday` and time range from 8:00-9:00, then alerts are suppressed from 8–9, Monday through Friday, but not, for example, 8–9 on Saturday.

- **Time range** – The time of day to suppress notifications. Consists of two sub-fields, **Start time** and **End time**. An example time is `14:30`. Time is in 24 hour notation, in UTC.
- **Days of the week** – The days of the week. Can be a single day, such as `monday`, a range, such as `monday:friday`, or a comma-separate list of days, such as `monday, tuesday, wednesday`.
- **Months** – The months to select. You can specify months with numeric designations, or with the full month name, for example `1` or `january` both specify January. You can specify a single month, a range of months, or a comma-separated list of months.
- **Days of the month** – The dates within a month. Values can range from 1-31. Negative values specify days of the month in reverse order, so `-1` represents the last day of the month. Days of the month can be specified as a single day, a range of days, or a comma-separate list of days.
- **Years** – The year or years for the interval. For example, `2023:2025`.

Each of these elements can be a list, and at least one item in the element must be satisfied to be a match. So if you set years to `2023:2025, 2027`, then it would be true during 2023, 2024, 2025, and 2027 (but not 2026).

If a field is left blank, any moment of time will match the field. A moment of time must match all fields to match a complete time interval.

If you want to specify an exact duration, specify all the options needed for that duration. For example, if you wanted to create a time interval for the first Monday of the month, for March, June, September, and December, between the hours of 12:00 and 24:00 UTC, your time interval specification could be:

- Time range:
  - Start time: `12:00`
  - End time: `24:00`
- Days of the week: `monday`
- Months: `3, 6, 9, 12`
- Days of the month: `1:7`

View and filter by alert groups

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

Alert groups show grouped alerts from an Alertmanager instance. By default, alert rules are grouped by the label keys for the root policy in notification policies. Grouping common alert rules into a single alert group prevents duplicate alert rules from being fired.

You can view alert groups and also filter for alert rules that match specific criteria.

To view alert groups

1. In the Grafana menu, click the **Alerting** (bell) icon to open the Alerting page listing existing alerts.
2. Click **Alert groups** to open the page listing existing groups.
3. From the **Alertmanager** dropdown, select an external Alertmanager as your data source.
4. From **custom group by** dropdown, select a combination of labels to view a grouping other than the default. This is useful for debugging and verifying your grouping of notification policies.

If an alert does not contain labels specified either in the grouping of the root policy or the custom grouping, then the alert is added to a catch all group with a header of `No grouping`.

To filter by label

- In **Search**, enter an existing label to view alerts matching the label.

  For example, `environment=production,region=~US|EU,severity!=warning`.

To filter by state

- In **States**, select from Active, Suppressed, or Unprocessed states to view alerts matching your selected state. All other alerts are hidden.

View notification errors

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 9.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 8.x, see Working in Grafana version 8.

View notification errors and understand why they failed to be sent or were not received.

Note

This feature is only supported for Grafana Alertmanager.

To view notification errors

1. In the Grafana menu, click the **Alerting** (bell) icon to open the Alerting page listing existing alerts.
2. Choose **Contact points** to see a list of the existing contact points.

   If any contact points are failing, a message at the right-hand corner of the screen alerts the user to the fact that there are errors and how many.
3. Click on a contact point to view the details of errors for that contact point.

   Error details are displayed if you hover over the Error icon.

   If a contact point has more than one integration, you see all errors for each of the integrations listed.

4. In the Health column, check the status of the notification.

   This can be either OK, No attempts, or Error.

Working in Grafana version 8

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

When you create your Grafana workspace, you have the option of which version of Grafana to use. The following topics describe using a Grafana workspace that uses version 8 of Grafana.

Topics

- Panels
- Dashboards
- Explore
- Linking
- Templates and variables
- Grafana alerting

Panels

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The panel is the basic visualization building block in a Grafana server. A panel is a visual representation of one or more queries except a few special-use panels. The queries display data over time. This can range from temperature fluctuations to current server status to a list of logs or alerts.

Each panel has a query editor specific to the data source selected in the panel. The query editor allows you to extract a visualization to display on the panel.

To display data, you must have at least one data source added to your workspace. For more information about data sources, see Connect to data sources.

There are a wide variety of styling and formatting options for each panel. Panels can be dragged, rearranged, and resized.

Topics

- Adding a panel
- Deleting a panel
- Queries
- Recorded queries
- Transformations
- Field options and overrides
- Panel editor
- Library panels
- Visualizations

Adding a panel

This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

You can use panels to show your data in visual form. This topic walks you through the basic steps to build a panel.

To add a panel to a dashboard

1. Choose the dashboard that you want to add a panel to.
2. Choose the **Add panel** icon.
3. Choose **Add new panel**.

   The Grafana workspace creates an empty graph panel with your default data source selected.
4. While not required, we recommend that you add a helpful title and description to your panel. You can optionally use variables that you have defined in either field, but not global variables. For more information, see Templates and variables.

   - **Panel title** – Text entered in this field is displayed at the top of your panel in the panel editor and in the dashboard.
   - **Description** – Text entered in this field is displayed in a tooltip in the upper-left corner of the panel. Write a description of the panel and the data that you are displaying.
5. Write a query for the panel. To display a visualization, each panel needs at least one query. You write queries on the **Query** tab of the panel editor. For more information, see Queries.

   a. Choose a data source. In the first line of the **Query** tab, choose the dropdown list to see all available data sources. This list includes all data sources that you added. For more information about data sources, see Connect to data sources.

   b. Write or construct a query in the query language of your data source. Options will vary. See your specific data source documentation for specific guidelines.
6. In the **Visualization** section of the **Panel** tab, choose a visualization type. The Grafana workspace displays a preview of your query results with that visualization applied.
7. We recommend that you add a note to describe your changes before you choose **Save**. Notes are helpful if you need to revert the dashboard to a previous version.
8. To save the dashboard, choose **Save** in the upper-right corner of the screen.

Deleting a panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

To delete a panel in Grafana, complete the following steps:

- Choose the drop-down next to the panel title and select **Remove panel**.

or

- Hover your pointer over the panel and use the keyboard shortcut sequence "p r".

Queries

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Grafana workspace panels use *Queries* to communicate with data sources to get data for the visualization. A query is a question written in the query language that is used by the data source. If the query is properly formed, the data source responds. In the panel data source options, you can adjust how often the query is sent to the data source and how many data points are collected.

Grafana workspaces supports up to 26 queries per panel.

Query editors

Query editors are forms that help you write queries. Depending on your data source, the query editor might provide automatic completion, metric names, or variable suggestion.

Because of differences between query languages, data sources might have query editors that look different.

Query syntax

Data sources have different query languages and syntaxes to ask for the data. Here are two query examples.

PostgreSQL

```
SELECT hostname FROM host WHERE region IN($region)
```

PromQL

```
query_result(max_over_time(<metric>[${__range_s}s]) != <state>)
```

For more information about writing a query for your data source, see the documentation for that data source. Data sources are listed in Connect to data sources.

Query tab UI

The **Query** tab consists of the following elements:

- Data source selector
- Query options
- Query inspector button
- Query editor list

Data source selector

The data source selector is a dropdown list. Choose it to select a data source that you have added. When you create a panel, Amazon Managed Grafana automatically selects your default data source. For more information about data sources, see Connect to data sources.

In addition to the data sources that you have configured in your Grafana workspace, three special data sources are available.

- **TestDataDB** – A built-in data source that generates random walk data. The Grafana data source is useful for testing visualizations and running experiments.
- **Mixed** – A data source for querying multiple data sources in the same panel. When this data source is selected, you can select a data source for every new query that you add.
  - The first query will use the data source that was selected before you selected **Mixed**.
  - You cannot change an existing query to use the Mixed data source.
- **Dashboard** –A data source for using a result set from another panel in the same dashboard.

Query options

To see settings for your selected data source, choose **Query options** next to the data source selector. Changes you make here affect only the queries made in this panel.

Amazon Managed Grafana sets defaults that are shown in dark gray text. Changes are displayed in white text. To return a field to the default setting, delete the white text from the field.

You can use the following panel data source query options:

- **Max data points** – If the data source supports it, sets the maximum numbers of data points for each series returned. If the query returns more data points than the max data points setting, the data source consolidates them (reduces the number of points returned by aggregating them together by average or max or other function).

  There are two main reasons for limiting the number of points: performance and smoothing the line. The default value is the width (or number of pixels) of the graph, which avoids having more data points than the graph panel can display.

  With streaming data, the max data points value is used for the rolling buffer. (Streaming is a continuous flow of data, and buffering is a way to divide the stream into chunks).
- **Min interval** – Sets a minimum limit for the automatically calculated interval, typically the minimum scrape interval. If a data point is saved every 15 seconds, you don't need to have an interval lower than that. Another use case is to set it to a higher minimum than the scrape interval to get more coarse-grained, well-functioning queries.
- **Interval** – A time span that you can use when aggregating or grouping data points by time.

Amazon Managed Grafana automatically calculates an appropriate interval that can be used as a variable in templated queries. The variable is either in seconds: `$__interval`; or in milliseconds: `$__interval_ms`. It is typically used in aggregation functions like `sum` or `average`. For example, this is a Prometheus query using the interval variable: `rate(http_requests_total[$__interval])`.

This automatic interval is calculated based on the width of the graph. If the user zooms out a lot, the interval becomes greater resulting in a more coarse-grained aggregation. If the user zooms in, the interval decreases resulting in a more fine-grained aggregation.

For more information, see Global variables.

- **Relative time** – Override of the relative time range for individual panels causing them to be different from what is selected in the dashboard time picker in the top-right corner of the dashboard. This enables you to show metrics from different time periods or days on the same dashboard.
- **Time shift** – Provides another way to override the time range for individual panels. This function works only with relative time ranges, and you can adjust the time range.

  For example, you can shift the time range for the panel to be 2 hours earlier than the dashboard time picker. For more information, see Time range controls.
- **Cache timeout** – (This field is visible only if it is available in your data source.) Overrides the default cache timeout if your time series store has a query cache. It is specified as a numeric value in seconds.

Query inspector button

You can choose **Query inspector** to open the **Query** tab of the panel inspector. On the **Query** tab, you can see the query request sent by the panel and the response.

Choose **Refresh** to see the full text of the request sent by this panel to the server.

> **Note**
>
> You need to add at least one query before the Query inspector can return results.

For more information about the panel inspector, see Inspect a panel.

Query editor list

In the UI, queries are organized in collapsible query rows. Each query row contains a query editor and is identified with a letter (A, B, C, and so on).

Sharing query results between panels

With Amazon Managed Grafana, you can use the query result from one panel for any other panel in the dashboard. Sharing query results across panels reduces the number of queries made to your data source, which can improve the performance of your dashboard.

The Dashboard data source lets you select a panel in your dashboard that contains the queries that you want to share the results for. Instead of sending a separate query for each panel, Amazon Managed Grafana sends one query, and other panels use the query results to construct visualizations.

This strategy can drastically reduce the number of queries being made when, for example, you have several panels visualizing the same data.

To share data source queries with another panel

1. Create a dashboard. For more information, see Creating a dashboard.
2. Add a panel. For more information, see Adding a panel.
3. Change the title to **`Source panel`**. You'll use this panel as a source for the other panels. Define the query or queries that will be shared. If you don't have a data source available at the moment, you can use the **Grafana** data source, which returns a random time series that you can use for testing.
4. Add a second panel, and then select the **Dashboard** data source in the query editor.
5. In **Use results from panel list**, select the first panel that you created.

All queries defined in the source panel are now available to the new panel. Queries made in the source panel can be shared with multiple panels.

To go to a panel where a query is defined, choose that query.

Recorded queries

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

You can see trends over time by taking a snapshot of a data point on a set interval using recorded queries. This can give you insight into historic trends.

For the plugins that do not return time series, it might be useful to plot historical data. For example, you might want to query **ServiceNow** to see a history of request response times but it can only return current point-in-time metrics.

How recorded queries work

Recorded queries only work with backend data source plugins. For more information, refer to Backend data source plugin. You can record three types of queries:

- single row and column - A query that returns a single row and column.
- row count - A query that returns meaningful rows to be counted.
- expression - Any expression. To learn more about creating and using expressions, see Expressions.

After a recorded query is created or enabled, it immediately creates a snapshot and continues to create snapshots at the set interval. The recorded query stops taking snapshots when it is disabled, deleted, or when Grafana is not running. Data points are gathered in the backend by running the recorded query and forwarding each result to a remote write enabled Prometheus instance.

Note

You must configure a Prometheus data source and associate it with a Remote write target before recorded queries can be used.

Create a recorded query

To create a recorded query, complete the following steps:

1. Find or create a query you want to record on a dashboard in an edit panel. The query must only return one row and column. If it returns more, you can still record the number of results returned using the *count* option. The query's data source must be a backend data source. Expressions can be used to aggregate data from a time series query. To learn more about creating and using expressions, refer to Expressions.
2. Choose the **record query** menu located in the query editor.
3. Enter recorded query information. All fields are required unless otherwise indicated.
   - Name - Name of the recorded query.
   - Description - (optional) Describe the recorded query as you want it to appear in the recorded query list.
   - Interval - The interval at which the snapshot will be taken. The interval starts when you create the recorded query and stops if you pause or delete the recorded query. For more information on pausing and deleting recorded queries, refer to Managing recorded queries.
   - Range - The relative time range of the query. If you select a range of 30m and an interval of 1h the query will take a snapshot every hour of the past 30 minutes.
   - Count query results - If you want to count the rows returned from your query toggle this option on. If this option is off, your query must return one row with one value.
4. Test your recorded query by choosing the test recorded query button.
5. Choose **Start recording query**.

Adding a recorded query

You can add existing recorded queries to panels in a dashboard. For each recorded query that you add, a Prometheus query is created: `generated_recorded_query_name{id="generated_id", name="recorded query name"}`. The created query from Prometheus returns all the recorded query's gathered snapshots.

1. Navigate to a panel in a dashboard where you wish to add a recorded query.
2. Choose the **+ Recorded query** menu.
3. If you want to filter recorded queries by data source, select a data source from the filter by data source drop down menu.
4. Choose **Add** menu on your recorded query to add it to the panel.

After adding your recorded query to the panel, the panel data source will become `-- Mixed --`. Your recorded query is represented by a `Prometheus` query with a name label matching your recorded query name. Refer to Prometheus to learn more about the Prometheus data source.

If after adding a recorded query, a query with a `-- Mixed --` data source instead of Prometheus data source appears, this could mean that a `Prometheus` remote write target was not set up for recorded queries. Refer to Remote write target to set up a remote write point.

Using a recorded query

To use a recorded query, create one and add it to a dashboard. After that, it can be managed in **Preferences** from the **Recorded queries** tab.

Managing recorded queries

Recorded queries can be paused or activated and deleted from the Recorded queries tab in Preferences. Deleting a recorded query will remove it from Grafana, but the information that was gathered in Prometheus will still be there. Pausing a recorded query will no longer gather new data points until it is resumed.

Remote write target

The remote write target is the **Prometheus** data source that recorded query data points are written to. You will need a Prometheus with remote write enabled and you will need to create a data source for this Prometheus.

To edit the remote write target choose **Edit Remote Write Target** in the console menu on the **Recorded Queries** tab in **Preferences**. Select the **Prometheus** data source that has remote write enabled and enter the remote write path.

Transformations

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Transformations process the result set before it's passed to the visualization. You access transformations in the **Transform** tab of the Amazon Managed Grafana panel editor.

You can use transformations to rename fields, join separate time series together, do math across queries, and more. If you have large dashboards or heavy queries, being able to reuse the query result from one panel to another panel can provide a huge performance gain.

> **Note**
>
> Transformations sometimes result in data that cannot be graphed. When that happens, Amazon Managed Grafana displays a suggestion on the visualization. Choose the suggestion to switch to table visualization. This often helps you better understand what the transformation is doing to your data.

Amazon Managed Grafana applies transformations in the sequence that they are listed on the screen. Every transformation creates a new result set that is passed to the next transformation in the pipeline.

The order can make a huge difference in how your results look. For example, if you use reduce transformation to condense all the results of one column to a single value, you can apply transformations only to that single value.

Prerequisites

Before you apply transformations, all of the following must be true:

- You have entered a query and returned data from a data source. For more information about queries, see Queries.
- You have applied a visualization that supports queries, such as one of the following visualizations:
  - Bar gauge
  - Gauge
  - Graph
  - Heatmap
  - Logs
  - Stat
  - Table

Applying a transformation

Transformations are available from the **Transform** tab in the bottom pane of the panel editor, next to the **Queries** tab.

To apply a transformation

1. On the panel that you want to add transformations to, choose the panel title, and then choose **Edit**.
2. Choose the **Transform** tab.
3. Select a transformation.

   In the transformation row that appears, you can configure the transformation options.
4. To apply another transformation, choose **Add transformation**. Keep in mind that the next transformation acts on the result set returned by the previous transformation.

If you have trouble, choose the bug icon to debug your transformations.

To remove a transformation, choose the trash can icon.

Transformation types and options

Grafana workspaces include the following transformations.

Topics

- Reduce
- Merge
- Filter by name
- Filter data by query
- Organize fields
- Join by field (outer join)
- Add field from calculation
- Labels to fields
- Group By
- Group By
- Series to rows

- Filter data by value
- Debug transformations

Reduce

Apply a **Reduce** transformation when you want to simplify your results down to one value. Reduce basically removes the time component. If visualized as a table, it reduces a column down to one row (value).

In the **Calculations** field, enter one or more calculation types. Choose to see a list of calculation choices. For information about available calculations, see Calculations list.

After you select at least one calculation, Amazon Managed Grafana displays one value using the calculation you selected. If you select more than one calculation, more than one value is displayed.

Merge

Use this transformation to combine the results from multiple queries into one single result. This is helpful when using the table panel visualization. Values that can be merged are combined into the same row. Values can be merged if the shared fields contain the same data.

In the following example, two queries return table data. The data are visualized as two separate tables before applying the transformation.

Query A

Time	Job	Uptime
2020-07-07 11:34:20	node	25260122
2020-07-07 11:24:20	postgre	123001233

Query B

Time	Job	Errors
2020-07-07 11:34:20	node	15

Time	Job	Errors
2020-07-07 11:24:20	postgre	5

Here is the result after applying the **Merge** transformation.

Time	Job	Errors	Uptime
2020-07-07 11:34:20	node	15	25260122
2020-07-07 11:24:20	postgre	5	123001233

Filter by name

Use this transformation to remove portions of the query results.

Amazon Managed Grafana displays the **Identifier** field, followed by the fields returned by your query.

You can apply filters in one of two ways:

- Enter a regex expression.
- Choose a field to toggle filtering on that field. Filtered fields are displayed with dark gray text, unfiltered fields have white text.

Filter data by query

Use this transformation in panels that have multiple queries, if you want to hide one or more of the queries.

Amazon Managed Grafana displays the query identification letters in dark gray text. To toggle filtering, choose a query identifier. If the query letter is white, the results are displayed. If the query letter is dark, the results are hidden.

Organize fields

Use this transformation to rename, reorder, or hide fields returned by the query.

> **Note**
>
> This transformation works only in panels that have a single query. If your panel has multiple queries, you must either apply a **Join by field (outer join)** transformation or remove the extra queries.

Amazon Managed Grafana displays a list of fields returned by the query. You can make any of the following changes:

- Change the field order by pausing over a field. The cursor turns into a hand, and then you can drag the field to its new place.
- Hide or show a field by choosing the eye icon next to the field name.
- Rename fields by typing a new name in the **Rename** box.

Join by field (outer join)

Use this transformation to join multiple time series from a result set by field.

This transformation is useful if you want to combine queries so that you can calculate results from the fields.

Add field from calculation

Use this transformation to add a new field calculated from two other fields. Each transformation allows you to add one new field.

- **Mode** – Select a mode:
  - **Reduce row** – Apply selected calculation on each row of selected fields independently.
  - **Binary option** – Apply a basic math operation (`sum`, `multiply`, and so on) on values in a single row from two selected fields.
- **Field name** – Select the names of fields that you want to use in the calculation for the new field.
- **Calculation -** Select a calculation to use when Amazon Managed Grafana creates the new field. Choose the field to see a list of calculation choices. For information about available calculations, see Calculations list.
- **Alias** – (Optional) Enter the name of your new field. If you leave this blank, the field will be named to match the calculation.

- **Replace all fields** – (Optional) Use this option if you want to hide all other fields and display only your calculated field in the visualization.

Labels to fields

> **Note**
>
> To apply this transformation, your query needs to return labeled fields.

When you select this transformation, Amazon Managed Grafana automatically transforms all labeled data into fields.

For example, consider a query result of two time series.

1: labels Server=Server A, Datacenter=EU 2: labels Server=Server B, Datacenter=EU

This transformation would result in the following table.

Time	Server	Datacenter	Value
2020-07-07 11:34:20	Server A	EU	1
2020-07-07 11:34:20	Server B	EU	2

Value field name; If you selected `Server` as the **Value field name**, you would get one field for every value of the `Server` label.

Time	Datacenter	Server A	Server B
2020-07-07 11:34:20	EU	1	2

Group By

This transformation sorts each frame by the configured field. When `reverse` is checked, the values are returned in the opposite order.

Group By

This transformation groups the data by a specified field (column) value and processes calculations on each group. The available calculations are the same as for the Reduce transformation.

Here's an example of original data.

Time	Server ID	CPU Temperature	Server Status
2020-07-07 11:34:20	server 1	80	Shutdown
2020-07-07 11:34:20	server 3	62	OK
2020-07-07 10:32:20	server 2	90	Overload
2020-07-07 10:31:22	server 3	55	OK
2020-07-07 09:30:57	server 3	62	Rebooting
2020-07-07 09:30:05	server 2	88	OK
2020-07-07 09:28:06	server 1	80	OK
2020-07-07 09:25:05	server 2	88	OK
2020-07-07 09:23:07	server 1	86	OK

This transformation takes two steps. First, you specify one or multiple fields to group the data by. This will group all the same values of those fields together, as if you sorted them. For instance, if you **Group By** the `Server ID` field, it will group the data this way:

Time	Server ID	CPU Temperature	Server Status
2020-07-07 11:34:20	**server 1**	80	Shutdown
2020-07-07 09:28:06	**server 1**	80	OK
2020-07-07 09:23:07	**server 1**	86	OK

```
2020-07-07 10:32:20 | server 2 | 90 | Overload
2020-07-07 09:30:05 | server 2 | 88 | OK
2020-07-07 09:25:05 | server 2 | 88 | OK

2020-07-07 11:34:20 | server 3 | 62 | OK
2020-07-07 10:31:22 | server 3 | 55 | OK
2020-07-07 09:30:57 | server 3 | 62 | Rebooting
```

All rows with the same value of `Server ID` are grouped together.

After choosing which field you want to group your data by, you can add various calculations on the other fields, and the calculation will be applied on each group of rows. For instance, you might want to calculate the average `CPU temperature` for each of those servers. You can add the *mean* calculation applied on the `CPU Temperature` field to get the following.

Server ID	CPU Temperature (mean)
server 1	82
server 2	88.6
server 3	59.6

And you can add more than one of those calculations. For instance, you can use the following calculations.

- For field `Time`, you can calculate the *Last* value, to know when the last data point was received for each server.
- For field `Server Status`, you can calculate the *Last* value to know the last state value for each server.
- For field `Temperature`, you can also calculate the *Last* value to know the latest monitored temperature for each server.

The Group By transformation produces the following results.

Server ID	CPU Temperature (mean)	CPU Temperature (last)	Time (last)	Server Status (last)
server 1	82	80	2020-07-07 11:34:20	Shutdown
server 2	88.6	90	2020-07-07 10:32:20	Overload
server 3	59.6	62	2020-07-07 11:34:20	OK

Using this transformation, you can extract some key information out of your time series and display it in a convenient way.

Series to rows

Use this transformation to combine the results from multiple time series data queries into one single result. This is helpful when using the table panel visualization.

The result from this transformation will contain three columns: `Time`, `Metric`, and `Value`. The `Metric` column is added so that you can see which query the metric originates from. Customize this value by defining `Label` on the source query.

In the example below, two queries return time series data. It is visualized as two separate tables before the transformation is applied.

Query A

Time	Temperature
2020-07-07 11:34:20	25
2020-07-07 10:31:22	22
2020-07-07 09:30:05	19

Query B

Time	Humidity
2020-07-07 11:34:20	24
2020-07-07 10:32:20	29
2020-07-07 09:30:57	33

Applying the `Series to rows` transformation produces the following results.

Time	Metric	Value
2020-07-07 11:34:20	Temperature	25
2020-07-07 11:34:20	Humidity	22
2020-07-07 10:32:20	Humidity	29
2020-07-07 10:31:22	Temperature	22
2020-07-07 09:30:57	Humidity	33
2020-07-07 09:30:05	Temperature	19

Filter data by value

This transformation allows you to filter your data directly in the Grafana workspace and remove some data points from your query result. You have the option to include or exclude data that match one or more conditions you define. The conditions are applied on a selected field.

This transformation is useful if your data source does not natively filter by values. You might also use this to narrow values to display if you are using a shared query.

The available conditions for all fields are as follows:

- **Regex** – Match a regex expression.
- **Is Null** – Match if the value is null.
- **Is Not Null** – Match if the value is not null.

- **Equal** – Match if the value is equal to the specified value.
- **Different** – Match if the value is different than the specified value.

The available conditions for number fields are as follows:

- **Greater** – Match if the value is greater than the specified value.
- **Lower** – Match if the value is lower than the specified value.
- **Greater or equal** – Match if the value is greater than or equal to the specified value.
- **Lower or equal** – Match if the value is lower than or equal to the specified value.
- **Range** – Match a range between a specified minimum and maximum. The minimum and maximum are included in the range.

You can add more than one condition to the filter. When you have more than one condition, you can choose if you want the include or exclude action to be applied on rows that Match all or any of the conditions you added.

Conditions that are not valid or incompletely configured are ignored.

Debug transformations

To see the input and the output result sets of the transformation, choose the bug icon on the right side of the transformation row.

Amazon Managed Grafana displays the transformation debug view below the transformation row.

Field options and overrides

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

This section explains what field options and field overrides in Amazon Managed Grafana are and how to use them.

The data model used in Grafana workspaces, the data frame, is a columnar-oriented table structure that unifies both time series and table query results. Each column within this structure is called a *field*. A field can represent a single time series or table column.

Field options allow you to change how the data is displayed in your visualizations. Options and overrides that you apply do not change the data, they change how Amazon Managed Grafana displays the data.

Field options

Field options, both standard and custom, can be found on the **Field** tab in the panel editor. Changes that are made on this tab apply to all fields (that is, series and columns). For example, if you change the unit to percentage, all fields with numeric values are displayed in percentages. Learn how to apply a field option in Configure all fields.

Field overrides

Field overrides can be added on the **Overrides** tab in the panel editor. There you can add the same options as you find on the **Field** tab, but they are applied only to specific fields. Learn how to apply an override in Configure specific fields.

Available field options and overrides

Field option types are common to both field options and field overrides. The only difference is whether the change will apply to all fields (applied on the **Field** tab) or to a subset of fields (applied on the **Overrides** tab).

- Standard field options apply to all panel visualizations that allow transformations.
- Table field options apply only to table panel visualizations.

Configure all fields

To change how all fields display data, you can change an option on the **Field** tab. On the **Overrides** tab, you can then override the field options for specific fields. For more information, see Configure specific fields.

For example, you can change the number of decimal places shown in all fields by changing the **Decimals** option. For more information about options, see Standard field options and Table field options.

Change a field option

You can change as many options as you want to.

To change a field option

1. Choose the panel that you want to edit, choose the panel title, and then choose **Edit**.
2. Choose the **Field** tab.
3. Find the option that you want to change. You can define the following:
   - Standard field options, which apply to all panel visualizations that allow transformations.
   - Table field options, which only apply to table panel visualizations.
4. Add options by adding values in the fields. To return options to default values, delete the white text in the fields.
5. When you have finished making edits to the dashboard, choose **Save**.

Field option example

Let's assume that the result set is a data frame that consists of two fields: time and temperature.

time	temperature
2020-01-02 03:04:00	45.0
2020-01-02 03:05:00	47.0
2020-01-02 03:06:00	48.0

Each field (column) of this structure can have field options applied in a way that alter the way its values are displayed. For example, you can set the Unit to Temperature > Celsius, resulting in the following table.

time	temperature
2020-01-02 03:04:00	45.0 °C
2020-01-02 03:05:00	47.0 °C
2020-01-02 03:06:00	48.0 °C

The decimal place doesn't mean anything in this case. You can change the Decimals from `auto` to zero (`0`), resulting in the following table.

time	temperature
2020-01-02 03:04:00	45 °C
2020-01-02 03:05:00	47 °C
2020-01-02 03:06:00	48 °C

Configure specific fields

You can use overrides to change the settings for one or more fields. Field options for overrides are exactly the same as the field options that are available in a particular visualization. The only difference is that you choose which fields to apply them to.

For example, you can change the number of decimal places shown in all numeric fields or columns by changing the **Decimals** option for **Fields with type** that matches **Numeric**. For more information about options, see Standard field options, which apply to all panel visualizations that allow transformations, and Table field options, which apply only to table panel visualizations.

Add a field override

You can override as many field options as you want to.

To add a field override

1. Choose the panel that you want to edit, choose the panel title, and then choose **Edit**.
2. Choose the **Overrides** tab.

3. Choose **Add an override for**.
4. Select the fields to which you want to apply an override rule.

   - **Fields with name** – Use this to select a field from the list of all available fields. Properties that you add to a rule with this selector are applied only to this single field.
   - **Fields with name matching regex** – Use this to specify fields to override with a regular expression. Properties that you add to a rule by using this selector are applied to all fields where the field names match the regex.
   - **Fields with type** – Use this to select fields by type, such as string, numeric, and so on. Properties that you add to a rule with this selector are applied to all fields that match the selected type.
5. Choose **Add override property**.
6. Select the field option that you want to apply.
7. Enter options by adding values in the fields. To return options to default values, delete the white text in the fields.
8. Continue to add overrides to this field by choosing **Add override property**, or you can choose **Add override** and select a different field to add overrides to.
9. When finished, choose **Save**.

Delete a field override

1. Choose the Overrides tab that contains the override that you want to delete.
2. Choose the trash can icon next to the override.

Field override example

Let's assume that our result set is a data frame that consists of four fields: time, high temp, low temp, and humidity.

time	high temp	low temp	humidity
2020-01-02 03:04:00	45.0	30.0	67
2020-01-02 03:05:00	47.0	34.0	68
2020-01-02 03:06:00	48.0	31.0	68

Let's apply the field options from the Field option example to apply the Celsius unit and get rid of the decimal place. This results in the following table.

time	high temp	low temp	humidity
2020-01-02 03:04:00	45 °C	30 °C	67 °C
2020-01-02 03:05:00	47 °C	34 °C	68 °C
2020-01-02 03:06:00	48 °C	31 °C	68 °C

The temperature fields look good, but the humidity is nonsensical. You can fix this by applying a field option override to the humidity field and changing the unit to Misc > percent (0-100). This results in a table that makes a lot more sense.

time	high temp	low temp	humidity
2020-01-02 03:04:00	45 °C	30 °C	67%
2020-01-02 03:05:00	47 °C	34 °C	68%
2020-01-02 03:06:00	48 °C	31 °C	68%

Standard field options

This section explains the available field options. They are listed in alphabetical order.

You can apply standard field options to most built-in Grafana workspace panels. Some older panels and community panels that have not updated to the new panel and data model will be missing either all or some of these field options.

Most field options will not affect the visualization until you choose outside of the field option box you are editing or press Enter.

For more information about applying these options, see the following sections:

- Configure all fields
- Configure specific fields

Decimals

This option sets the number of decimals to include when rendering a value. Leave empty for Amazon Managed Grafana to use the number of decimals provided by the data source.

To change this setting, enter a number in the field.

Data links

This option controls the URL to which a value or visualization links. For more information and instructions, see Data links.

Display name

This option sets the display title of all fields. You can use variables in the field title. For more information about templating and template variables, see Templates and variables.

When multiple stats, fields, or series are shown, this field controls the title in each stat. You can use expressions such as `${__field.name}` to use only the series name or the field name in the title.

Given a field with a name of Temp, and labels of {"Loc"="PBI", "Sensor"="3"}

Expression syntax	Example	Renders to	Explanation
`${__field.displayname}`	Same as syntax	`Temp {Loc="PBI", Sensor="}`	Displays the field name and labels in {} if they are present. If there is only one label key in the response, then for the label portion, Amazon Managed Grafana displays the value of the label without the enclosing braces.
`${__field.name}`	Same as syntax	`Temp`	Displays the name of the field (without labels).
`${__field.labels}`	Same as syntax	`Loc="PBI", Sensor="`	Displays the labels without the name.

Expression syntax	Exam	Renders to	Explanation
`${__fi .label }`	`${__ .labe oc}`	PBI	Displays the value of the specified label key.
`${__fi .label _value`	Same as Synta	PBI, 3	Displays the values of the labels separated by a comma (without label keys).

If the value is an empty string after rendering the expression for a particular field, the default display method is used.

Max

This option sets the maximum value used in percentage threshold calculations. For automatic calculation based on all series and fields, leave this setting blank.

Min

This option sets the minimum value used in percentage threshold calculations. For automatic calculation based on all series and fields, leave this setting blank.

No value

Enter what Amazon Managed Grafana should display if the field value is empty or null.

Unit

This option specifies the unit that a field should use. Choose the **Unit** field, then drill down until you find the unit that you want. The unit that you select is applied to all fields except time.

Custom units

You can also use the unit dropdown list to specify custom units, custom prefix or suffix, and date/time formats.

To select a custom unit, enter the unit and select the last `Custom: xxx` option in the dropdown list.

- `suffix:<suffix>` for custom unit that should go after value.
- `time:<format>` for custom date/time formats; for example, `time:YYYY-MM-DD`. For the format syntax and options, see Display.
- `si:<base scale><unit characters>` for custom SI units; for example, `si: mF`. This option is a bit more advanced because you can specify both a unit and the source data scale. For example, if your source data is represented as milli (thousands of) unit, prefix the unit with that SI scale character.
- `count:<unit>` for a custom count unit.
- `currency:<unit>` for custom a currency unit.

You can also paste a native emoji in the unit picker and pick it as a custom unit.

String units

Amazon Managed Grafana can sometimes parse strings and show them as numbers. To make Amazon Managed Grafana show the original string, create a field override and add a unit property with the `string` unit.

Color scheme

The field color option defines how Amazon Managed Grafana colors series or fields. There are multiple modes here that work differently, and their utility depends largely on the currently selected visualization.

Continuous color modes use the percentage of a value relative to min and max to interpolate a color.

- **Single color** – Specific color set by using the color picker. This is most useful from an override rule.
- **From thresholds** – Color derived from the matching threshold. This is useful for gauges, stat, and table visualizations.

Color by series

Amazon Managed Grafana includes color schemes that define color by series. This is useful for graphs and pie charts, for example.

Color by value

Amazon Managed Grafana also includes continuous (gradient) color schemes. This is useful for visualizations that color individual values; for example, stat panels and the table panel.

Thresholds

You can use thresholds to change the color of a field based on the value. For more information and instructions, see Thresholds.

Value mapping

You can use this option to set rules that translate a field value or range of values into explicit text. You can add more than one value mapping.

- **Mapping type** – Choose an option.
  - **Value** – Enter a value. If the field value is greater than or equal to the value, the **Text** is displayed.
  - **From** and **To** – Enter a range. If the field value is between or equal to the values in the range, the **Text** is displayed.
- **Text** – Text that is displayed if the conditions are met in a field. This field accepts variables.

Panel editor

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

This topic describes the parts of the Amazon Managed Grafana panel editor, and it includes links to where you can find more information.

Opening the panel editor

There are several ways to access the panel editor, also called the **Edit Panel** screen, *edit mode*, or *panel edit mode*.

- Choose the **Add panel** icon at the top of the screen, and then choose **Add new panel**. The new panel opens in the panel editor. For more information about how to add a panel, see Adding a panel.
- Choose the title of an existing panel, and then choose **Edit**. The panel opens in edit mode.
- Choose anywhere on an existing panel, and then press **e** on your keyboard. The panel opens in edit mode.

Resizing panel editor sections

Drag to resize sections of the panel editor. If the side pane becomes too narrow, the **Panel**, **Field**, and **Overrides** tabs change to a dropdown list.

Parts of the panel editor

This section describes the parts of the panel editor screen, with information about fields, options, or tasks associated with each part.

Header

The header section lists the name of the dashboard that the panel is in and some dashboard commands. You can also choose the **Go back** arrow to return to the dashboard.

On the right side of the header are the following options:

- **Dashboard settings (gear) icon** – Choose to access the dashboard settings.
- **Discard** Choose to discard all changes that you have made to the panel since you last saved the dashboard.
- **Save** – Choose to saves the dashboard, including all changes that you have made in the panel editor.
- **Apply** – Choose to apply changes that you made and then close the panel editor, returning to the dashboard. Also save the dashboard to persist the applied changes.

Visualization preview

The visualization preview section contains viewing options, time range controls, the visualization preview, and (if applicable) the panel title, axes, and legend.

- **Fill** – The visualization preview fills the available space in the preview part. If you change the width of the side pane or height of the bottom pane, the visualization adapts to fill whatever space is available.
- **Fit** – The visualization preview fills in the available space, but it preserves the aspect ratio of the panel.
- **Exact** – The visualization preview has the exact size as the size on the dashboard. If not enough space is available, the visualization scales down, preserving the aspect ratio.
- **Time range controls** – For more information, see Time range controls. .

Data section (lowest pane)

The data section contains tabs where you enter queries, transform your data, and create alert rules (if applicable).

- **Query tab** – Select your data source and enter queries here. For more information, see Queries.
- **Transform tab** – Apply data transformations. For more information, see Transformations.
- **Alert tab** – Write alert rules. For more information, see Grafana alerting.

Panel and field options (side pane)

This section contains tabs where you control almost every aspect of how your data is visualized. Not all tabs are available for each visualization.

Features on these tabs are documented in the following topics:

- Adding a panel
- Visualizations
- Field options and overrides
- Panel links and Data links, which help you connect your visualization to other resources

Library panels

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Library panels allow users to create reusable panels where any changes made to one instance of the library panel are reflected on every dashboard affecting all other instances where the panel is used. These panels can be saved in folders alongside Dashboards and streamline reuse of panels across multiple dashboards.

Create a library panel

Note

When you create library panels, the panel on the source dashboard is converted to a library panel as well. You will need to save the original dashboard once a panel is converted.

To create a library panel

1. Create a panel as you normally would. You can also use an existing panel.
2. Choose the title of the panel and choose **Edit**.
3. In the panel display options side pane, choose the down arrow option to bring changes to the visualization.
4. Choose **Library panels**, and choose **Create new library panel**.
5. Enter a name for the library panel, and select the folder to save it in.
6. Choose **Create library panel** and then save the dashboard.

You can also create library panels by using the **Share** option for any panel.

Once created, you can modify the library panel using any dashboard on which it appears. Once the library panel changes are saved, all instances of the library panel will reflect these modifications.

Add a library panel

To add a library panel to a dashboard

1. Pause on the + option on the left menu, then choose **Create**.
2. Choose **Add a panel from the panel library**.
3. Filter the list of library panels to find the panel that you want.
4. Choose that panel and add it to the dashboard.

Unlink a library panel

If you have a library panel on your dashboard that you want to modify without affecting all other instances of the library panel, you can unlink the library panel.

To unlink a library panel from a dashboard

1. Pause on **Dashboard** on the left menu, then choose **Library panels**.
2. Select a library panel. You will see a list of all the dashboards where the library panel is used.
3. Select the panel that you want to unlink and update.
4. Choose the title of the panel and choose **Edit**.
5. Choose **Unlink**.

Delete a library panel

Before you delete a library panel, verify that it is no longer in use on any dashboard.

To delete a library panel

1. Pause on **Dashboard** on the left menu, then choose **Library panels**.
2. Select a library panel. You will see a list of all the dashboards where the library panel is used.
3. Select the panel that you want to delete.
4. Choose the delete icon next to the panel name.

Parts of the panel editor

This section describes the parts of the panel editor screen, with information about fields, options, or tasks associated with each part.

Header

The header section lists the name of the dashboard that the panel is in and some dashboard commands. You can also choose the **Go back** arrow to return to the dashboard.

On the right side of the header are the following options:

- **Dashboard settings (gear) icon** – Choose to access the dashboard settings.
- **Discard** Choose to discard all changes that you have made to the panel since you last saved the dashboard.
- **Save** – Choose to save the dashboard including all changes that you have made in the panel editor.
- **Apply** – Choose to apply changes that you made and then close the panel editor, returning to the dashboard. Also save the dashboard to persist the applied changes.

Visualization preview

The visualization preview section contains viewing options, time range controls, the visualization preview, and (if applicable) the panel title, axes, and legend.

- **Fill** – The visualization preview fills the available space in the preview part. If you change the width of the side pane or height of the bottom pane, the visualization adapts to fill whatever space is available.
- **Fit** – The visualization preview fills in the available space, but it preserves the aspect ratio of the panel.
- **Exact** – The visualization preview has the exact size as the size on the dashboard. If not enough space is available, the visualization scales down, preserving the aspect ratio.
- **Time range controls** – For more information, see Time range controls. .

Data section (lowest pane)

The data section contains tabs where you enter queries, transform your data, and create alert rules (if applicable).

- **Query tab** – Select your data source and enter queries here. For more information, see Queries.
- **Transform tab** – Apply data transformations. For more information, see Transformations.
- **Alert tab** – Write alert rules. For more information, see Grafana alerting.

Panel and field options (side pane)

This section contains tabs where you control almost every aspect of how your data is visualized. Not all tabs are available for each visualization.

Features on these tabs are documented in the following topics:

- Adding a panel
- Visualizations
- Field options and overrides
- Panel links and Data links, which help you connect your visualization to other resources

Visualizations

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Amazon Managed Grafana offers a variety of visualizations to suit different use cases. The following sections list the visualizations that are available in Amazon Managed Grafana and their display settings.

Topics

- Alert list panel
- Bar chart panel
- Bar gauge panel

- Clock panel
- Dashboard list panel
- Gauge panel
- Geomap panel
- Graph panel
- Heatmap
- Histogram panel
- Logs panel
- News panel
- Node graph panel (Beta)
- Pie chart panel
- Plotly panel
- Sankey panel
- Scatter panel
- Stat panel
- State timeline panel
- Status history panel
- Table panel
- Text panel
- Time series panel
- Thresholds
- WindRose
- Inspect a panel
- Calculations list

Alert list panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The alert list panel displays your dashboards alerts. You can configure the list to show current state or recent state changes. For more information about alerts, see Grafana alerting.

Use these settings to refine your visualization.

Options

- **Show** – Choose whether the panel should display the current alert state or recent alert state changes.
- **Max Items** – Set the maximum number of alerts to list.
- **Sort order** – Select how to order the alerts displayed.
  - **Alphabetical (asc)** – Alphabetical order
  - **Alphabetical (desc)** – Reverse alphabetical order
  - **Importance** – By importance according to the following values, with 1 being the highest:
    - alerting: 1
    - no_data: 2
    - pending: 3
    - ok: 4
    - paused: 5
- **Alerts from this dashboard** – Show alerts only from the dashboard that the alert list is in.

Filter

Use the following options to filter the alerts to match the query, folder, or tags that you choose:

- **Alert name** – Enter an alert name query.
- **Dashboard title** – Enter a dashboard title query.
- **Folder** – Select a folder. Only alerts from dashboards in the selected folder will be displayed.
- **Dashboard tags -** Select one or more tags. Only alerts from dashboards with one or more of the tags will be displayed.

State filter

Choose which alert states to display in this panel.

- Ok
- Paused
- No data
- Execution error
- Alerting
- Pending

Bar chart panel



This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

This panel visualization allows you to graph categorical data.

Supported data formats

Only one data frame is supported and it needs to have at least one string field that will be used as the category for an X or Y axis and one or more numerical fields. The following is an example of data formats:

Browser	Market share
Chrome	50
Internet Explorer	17.5

If you have more than one numerical field, the panel shows grouped bars.

Visualizing time series or multiple result sets

If you have multiple time series or tables, you first need to join them using a join, or reduce transform. For example, if you have multiple time series and you want to compare their last and max value, add the Reduce transform and specify Max and Last as options under Calculations.

Bar chart options

Use these options to refine your visualizations:

Orientation

- **Auto** – Grafana decides the bar orientation based on the panel dimensions.
- **Horizontal** – Makes the X axis the category axis.
- **Vertical** – Makes the Y axis the category axis.

Show values

Controls whether values are shown on top of or to the left of bars.

- **Auto** – Values are shown if there is space.
- **Always** – Always show values.
- **Never** – Never show values.

Group width controls the width of groups. 0=min and 1=max width.

Bar width controls the width of bars. 0=min and 1=max width.

Line width controls line width of the bars.

Fill opacity controls the fill opacity of the bars.

Gradient mode sets the mode of the gradient fill. Fill gradient is based on the line color. To change the color, use the standard color scheme field option. Gradient appearance is influenced by the Fill opacity setting.

- **None** – no gradient fill, this is the default setting.

- **Opacity** – Transparency of the gradient is calculated based on the values on the y-axis. Opacity of the fill is increasing with the values on the Y-axis.
- **Hue** – Gradient color is generated based on the hue of the line color.

Tooltip mode – When you hover your cursor over the visualization, Grafana can display tooltips. Choose how tooltips behave.

- **Single** – The hover tooltip shows only a single series, the one that you are hovering over on the visualization.
- **All** – The hover tooltip shows all series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden** – Do not display the tooltip when you interact with the visualization.

> **Note**
>
> Use an override to hide individual series from the tooltip.

Legend mode – Use these settings to refine how the legend appears in your visualization.

- **List** – Displays the legend as a list. This is a default display mode of the legend.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement – Choose where to place the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend calculations – Choose which of the standard calculations to show in the legend. You can have more than one.

Text size – Enter a value to change the size of the text on your bar chart.

Axis – Use the following field settings to refine how your axes display. Some field options will not affect the visualization until you click outside of the field option box you are editing or press Enter.

- **Placement** – Sets the placement of the Y-axis.
- **Auto** – Grafana automatically assigns Y-axis to the series. When there are two or more series with different units, then Grafana assigns the left axis to the first unit and right to the following units.
- **Left** – Display all Y-axes on the left side.
- **Right** – Display all Y-axes on the right side.
- **Hidden** – Hide all Y-axes.
- **Label** – Set a Y-axis text label. If you have more than one Y-axis, then you can assign different labels with an override.
- **Width** – Set a fixed width of the axis. By default, Grafana dynamically calculates the width of an axis.

  By setting the width of the axis, data with different axes types can share the same display proportions. This makes it easier to compare more than one graph's worth of data because the axes are not shifted or stretched within visual proximity of each other.
- **Soft min and soft max** – Set a soft min and soft max option for better control of Y-axis limits. By default, Grafana sets the range for the Y-axis automatically based on the dataset.

  Soft min and soft max settings can prevent blips from turning into mountains when the data is mostly flat, and hard min or max derived from standard min and max field options can prevent intermittent spikes from flattening useful detail by clipping the spikes past a defined point.

  You can set standard min/max options to define hard limits of the Y-axis.

Bar gauge panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The bar gauge simplifies your data by reducing every field to a single value. You choose how Amazon Managed Grafana calculates the reduction.

This panel can show one or more bar gauges depending on how many series, rows, or columns your query returns.

Data and field options

With Bar gauge visualizations, you can apply the following options:

- Transformations
- Field options and overrides
- Thresholds

Display options

Use the following options to refine your visualization:

- **Show** – Choose how Amazon Managed Grafana displays your data.
  - **Calculate** – Show a calculated value based on all rows. For a list of available calculations, see Calculations list.
  - **All values** – Show a separate stat for every row. If you select this option, you can also select a **Limit**, or the maximum number of rows to display.
- **Value** – Select a reducer function that Amazon Managed Grafana will use to reduce many fields to a single value. Choose the **Value** list to see functions and brief descriptions.
- **Orientation** – Choose a stacking direction.
  - **Auto** – Amazon Managed Grafana selects what the orientation that it thinks fits best.
  - **Horizontal** – Bars stretch horizontally, left to right.
  - **Vertical** – Bars stretch vertically, top to bottom.
- **Display mode** – Choose a display mode.
  - **Gradient** – Choose a threshold level to define a gradient.
  - **Retro LCD** – Display the gauge split into small cells that are lit or unlit.
  - **Basic** – Use a single color based on the matching threshold.
- **Show unfilled area** – Select this option if you want to render the unfilled region of the bars as dark gray. This option is not available for the Retro LCD display mode.

Clock panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The clock panel shows the current time or a countdown. It updates every second.

- **Mode** – The default is **time**. If you choose **countdown**, set the **Countdown Deadline** to start the countdown.
- **12 or 24 hour** – The options for showing the time are 12-hour format and 24-hour format.
- **Timezone** – The time zones are supplied by the moment timezone library. The default is the time zone on your computer.
- **Countdown Deadline** – Specify the time and date to count down to, if you have set **Mode** to **countdown**.
- **Countdown End Text** – Specify the text to show when the countdown ends.
- **Date/Time formatting options** – Customize the font size, weight, and date/time formatting. If you are showing a countdown and don't want to see the seconds ticking down, change the time format to `hh:mm` for the 24-hour clock or `h:mm A` for the 12-hour clock. For a complete list of options, see Display.
- **Bg Color** – Select a background color for the clock.

Dashboard list panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The dashboard list panel displays dynamic links to other dashboards. The list can be configured to use starred dashboards, recently viewed dashboards, a search query, and dashboard tags.

On each dashboard load, this panel queries the dashboard list, always providing the most up-to-date results.

Options

Use the following options to refine your visualization:

- **Starred** – Display starred dashboards in alphabetical order.
- **Recently viewed** – Display recently viewed dashboards in alphabetical order.
- **Search** – Display dashboards by search query or tags. This option requires you to enter at least one value in **Query** or **Tags**.
- **Show headings** – Show the chosen list selection (Starred, Recently viewed, Search) as a heading.
- **Max items** – Set the maximum number of items to list per section. For example, at the default value of 10, if choose to display starred and recently viewed dashboards, the panel displays up to 20 total dashboards, 10 in each section.

Search

The following options apply only if the **Search** option is selected.

- **Query** – Enter the query that you want to search. Queries are not case sensitive, and partial values are accepted.
- **Folder** – Select the dashboard folders that you want to display.
- **Tags** – Enter the tags that you want to search. Note that existing tags will not appear as you type, and tags *are* case sensitive.

Note

When multiple tags and strings appear, the dashboard list displays those matching *all* conditions.

Gauge panel

This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Gauge is a single-value panel that can repeat a gauge for every series, column, or row.

Data and field options

Gauge visualizations allow you to apply the following options:

- Transformations
- Field options and overrides
- Thresholds

Display options

To refine your visualization, use the following options:

- **Show** – Choose how Amazon Managed Grafana displays your data.
  - **Calculate** – Show a calculated **Value** based on all rows. For a list of available calculations, see Calculations list.
  - **All values** – Show a separate stat for every row. If you select this option, you can also select a **Limit**, or the maximum number of rows to display.
- **Orientation** – Choose a stacking direction.

- **Auto** – Amazon Managed Grafana selects what it thinks is the best orientation.
- **Horizontal** – Bars stretch horizontally, left to right.
- **Vertical** – Bars stretch vertically, top to bottom.

- **Show threshold labels** – Choose whether to show threshold values.
- **Show threshold markers** – Choose whether to show a threshold band outside the inner gauge value band.

Geomap panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The **Geomap** panel visualization enables you to view and customize the world map using geospatial data. To easily focus on the important location-based characteristics of the data, you can configure various overlay styles and map view settings.

Data layer

The data layer in the Geomap plugin determines how you visualize geospatial data on top of the base map.

Location

The **Geomap** panel needs a source of geographical data. This data comes from a database query, and there are four mapping options for your data.

- **Auto** automatically searches for location data. Use this option when your query is based on one of the following names for data fields.
  - geohash: “geohash”
  - latitude: “latitude”, “lat”

  - longitude: “longitude”, “lng”, “lon”
  - lookup: “lookup”
- **Coords** specifies that your query holds coordinate data. You will get prompted to select numeric data fields for latitude and longitude from your database query.
- **Geohash** specifies that your query holds geohash data. You will get prompted to select a string data field for the geohash from your database query.
- **Lookup** specifies that your query holds location name data that needs to be mapped to a value. You will get prompted to select the lookup field from your database query and a `gazetteer`. The `gazetteer` is the directory that is used to map your queried data to a geographical point.

Markers layer

The **Markers** layer allows you to display data points as different marker shapes such as circle, squares, triangles, stars, and more.

- **Marker Color** configures the color of the marker. The default Fixed size keeps all points a single color. There is an alternate option to have multiple colors depending on the data point values and the threshold set at the **Thresholds** section.
- **Marker Size** configures the size of the marker. Default is `Fixed size`, making all marker size the same regardless of the data points. However, there is also an option to scale the circles to the corresponding data points. `Min` and `Max` marker size has to be set such that the Marker layer can scale within these ranges.
- **Marker Shape** provides you with the flexibility to visualize the data points differently.
  - Circle
  - Square
  - Triangle
  - Cross
  - X
- **Fill opacity** configures the transparency of each marker.

Heatmap layer

The **Heatmap** layer clusters various data points to visualize locations with different densities. To add a heatmap layer, under **Data Layer**, choose **Heatmap**.

Similar to **Markers**, you are prompted with various options to determine which data points to visualize and how.

- **Weight values** configures the intensity of the heatmap clusters. Fixed value keeps a constant weight value throughout all data points. This value should be in the range of 0~1. Similar to **Markers**, there is an alternate option in the dropdown to automatically scale the weight values depending on data values.
- **Radius** configures the size of the heatmap clusters.
- **Blur** configures the amount of blur on each cluster.

Graph panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

A graph panel can render as a line, a path of dots, or a series of bars. This type of graph is versatile enough to display almost any time-series data.

Data and field options

When using graph visualizations, you can apply the following options:

- Transformations
- Alerts. This is the only type of visualization that allows you to set alerts. For more information, see Grafana alerting.
- Thresholds

Display options

To refine your visualization, use the following settings:

- **Bars** – Display values as a bar chart.
- **Lines** – Display values as a line graph.
- **Line width** – Specify the width of the line for a series. The default is 1.
- **Staircase** – Draw adjacent points as staircase.
- **Area fill** – Specify the amount of color fill for a series. The default is 1; 0 is none.
- **Fill gradient** – Specify the degree of gradient on the area fill. The default is 0, which is no gradient; 10 is a steep gradient.
- **Points** – Display points for values.
- **Point radius** – Control how large the points are.
- **Alert thresholds** – Display alert thresholds and Regions on the panel.

Stacking and null value

- **Stack** – Each series is stacked on top of another.
- **Percent** – Each series is drawn as a percentage of the total of all series. This option is available when **Stack** is selected.
- **Null value** – Specify how null values are displayed. *This is an important setting.* See the note below.
  - **connected** – If there is a gap in the series, meaning one or more null values, the line will skip the gap and connect to the next non-null value.
  - **null** If there is a gap in the series, meaning a null value, the line in the graph will be broken and show the gap. This is the default setting.
  - **null as zero** – If there is a gap in the series, meaning a null value, it will be displayed as a zero value in the graph panel.

> **Important**
>
> If you are monitoring a server's CPU load and the load reaches 100 percent, the server will lock up, and the agent sending statistics will not be able to collect the load statistic. This leads to a gap in the metrics, and using the default *null* setting means that Amazon Managed Grafana will show the gaps and indicate that something is wrong. If this is set to *connected*, it will be easy to miss this signal.

Hover tooltip

Use these settings to change the appearance of the tooltip that appears when you pause over the graph visualization.

- **Mode** – Determines how many series the hover tooltip shows.
  - **All series** – The hover tooltip shows all series in the graph. In the series list in the tooltip, the Grafana workspace highlights the series that you are pausing on in bold.
  - **Single** – The hover tooltip shows only a single series, the one that you are pausing on in the graph.
- **Sort order** – Sorts the order of series in the hover tooltip if you have selected **All series** mode. When you pause on a graph, Amazon Managed Grafana displays the values associated with the lines. Generally, users are most interested in the highest or lowest values. Sorting these values can make it much easier to find the data that you want.
  - **None** – The order of the series in the tooltip is determined by the sort order in your query. For example, you can sort the series alphabetically by series name.
  - **Increasing** – The series in the hover tooltip are sorted by value and in increasing order, with the lowest value at the top of the list.
  - **Decreasing** – The series in the hover tooltip are sorted by value and in decreasing order, with the highest value at the top of the list.

Series overrides

Series overrides allow a series in a graph panel to be rendered differently from the others. You can customize display options on a per-series basis or by using regex rules. For example, one series can have a thicker line width to make it stand out or be moved to the right Y-axis.

You can add multiple series overrides.

To add a series override

1. Choose **Add series override**.
2. In **Alias or regex**, type or select a series. Choose the field to see a list of available series.

   For example, `/Network.*/` would match two series named `Network out` and `Network in`.
3. Choose **+** and then select a style to apply to the series. You can add multiple styles to each entry.

- **Bars** – Show series as a bar graph.
- **Lines** – Show series as a line graph.
- **Line fill** – Show a line graph with area fill.
- **Fill gradient** – Specify the area fill gradient amount.
- **Line width** – Set line width.
- **Null point mode** – Use this option to ignore null values or replace them with zero. This is important if you want to ignore gaps in your data.
- **Fill below to** – Fill the area between two series.
- **Staircase line** – Show series as a staircase line.
- **Dashes** – Show a line with dashes.
- **Hidden Series** – Hide the series.
- **Dash Length** – Set the length of dashes in the line.
- **Dash Space** – Set the length of the spaces between the dashes in the line.
- **Points** – Show series as separate points.
- **Point Radius** – Set the radius for point rendering.
- **Stack** – Set the stack group for the series.
- **Color** – Set the series color.
- **Y-axis** – Set the series y-axis.
- **Z-index** – Set the series z-index (rendering order). This option is important when you are overlaying different styles, such as bar charts and area charts.
- Transform – Transform value to negative to render below the y-axis.
- **Legend** – Control whether a series is shown in the legend.
- **Hide in tooltip** – Control whether a series is shown in a graph tooltip.

Axes

Use these options to control the display of axes in the visualization.

Left Y/Right Y

Options are identical for both y-axes.

- **Show** – Choose to show or hide the axis.
- **Unit** – Choose the display unit for the y value.
- **Scale** – Choose the scale to use for the y value: **linear**, or **logarithmic**. The default is **linear**.
- **Y-Min** – The minimum y value. The default is **auto**.
- **Y-Max** – The maximum Y value. The default is **auto**.
- **Decimals** – Define how many decimals are displayed for the y value. The default is **auto**.
- **Label** – Specify the y axis label. The default is “",

Y-Axes

- **Align** – Align left and right y-axes by value. The default is unchecked/false.
- **Level** – Enter the value to use for alignment of left and right y-axes, starting from Y=0. The default is 0. This option is available when **Align** is selected.

X-Axis

- **Show** – Choose to show or hide the axis.
- **Mode** – The display mode completely changes the visualization of the graph panel. It’s like three panels in one. The main mode is the time series mode with time on the x-axis. The other two modes are a basic bar chart mode with series on the x-axis instead of time and a histogram mode.
  - **Time** (default) – The x-axis represents time and the data is grouped by time (for example, by hour, or by minute).
  - **Series** – The data is grouped by series, and not by time. The y-axis still represents the value.
    - **Value** – This is the aggregation type to use for the values. The default is **total** (summing the values together).
  - **Histogram** – This option converts the graph into a histogram. A histogram is a kind of bar chart that groups numbers into ranges, often called buckets or bins. Taller bars show that more data falls in that range.

    For more information about histograms, see Introduction to histograms and heatmaps.
    - **Buckets** – Sets the number of buckets to group the values by. If left empty, Amazon Managed Grafana tries to calculate a suitable number of buckets.
    - **X-Min** – Filters out values from the histogram that are less than this minimum limit.

- **X-Max** – Filters out values that are greater than this maximum limit.

Legend

Use these settings to refine how the legend appears in your visualization.

Options

- **Show** – Clear to hide the legend. The default is selected (true).
- **As Table** – Select to display the legend in the table. The default is checked (true).
- **To the right** – Select to display the legend to the right.
- Width – Enter the minimum width for the legend in pixels. This option is available when **To the right** is selected.

Values

Additional values can be shown alongside the legend names.

- **Min** – The minimum value returned from the metric query.
- **Max** – The maximum value returned from the metric query.
- **Avg** – The average value returned from the metric query.
- **Current** – The last value returned from the metric query.
- **Total** – The sum of all values returned from the metric query.
- **Decimals** – How many decimals are displayed for legend values and graph hover tooltips.

Amazon Managed Grafana calculates the legend values on the client side. These legend values depend on the type of aggregation or point consolidation that your metric query is using. All the above legend values cannot be correct at the same time.

For example, if you plot a rate such requests/second, which is probably using average as an aggregator, the Total in the legend will not represent the total number of requests. It is just the sum of all data points received by Amazon Managed Grafana.

Hide series

Hide series when all values of a series from a metric query are of a specific value.

- **With only nulls** – Value=null (default unchecked)
- **With only zeroes** – Value=zero (default unchecked)

Time regions

You can highlight specific time regions on the graph to make it easier to see, for example, weekends, business hours, and off-work hours. All configured time regions refer to UTC time.

Heatmap

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The heatmap panel visualization allows you to view histograms over time. For more information about histograms, see Introduction to histograms and heatmaps.

Axes options

Use these settings to adjust how axes are displayed in your visualization.

Y Axis

- **Unit** – The display unit for the y-axis value
- **Scale** – The scale to use for the y-axis value
  - **linear** – Linear scale
  - **log (base 2)** – Logarithmic scale with base 2
  - **log (base 10)** – Logarithmic scale with base 10
  - **log (base 32)** – Logarithmic scale with base 32
  - **log (base 1024)** – Logarithmic scale with base 1024
- **Y-Min** – The minimum y value (default is auto)
- **Y-Max** – The maximum y value (default is auto)

- **Decimals** – Number of decimals to render y-axis values with (default is auto)

Buckets

> **Note**
>
> If the data format is **Time series buckets**, this section is not be available.

- **Y Axis Buckets** – The number of buckets that the y-axis will be split into.
- **Size** – The size of each y-axis bucket (visible only if **Scale** is *linear*). This option has priority over **Y Axis Buckets**.
- **Split Factor** – (Only visible if **Scale** is *log (base 2)* or greater). By default, Amazon Managed Grafana splits y values by log base. With this option, you can split each default bucket into the specified number of buckets.
- **X Axis Buckets** – The number of buckets that the x-axis will be split into.
- **Size** – The size of each x-axis bucket. Number or time interval (10s, 5m, 1h, etc). Supported intervals: ms, s, m, h, d, w, M, y. This option has priority over **X Axis Buckets**.

Bucket bound

When Data format is Time series buckets, the data source returns series with names representing bucket bound. But depending on the data source, a bound can be upper or lower. You can use this option to adjust a bound type. If **Auto** is set, a bound option is chosen based on panels' data source type.

Bucket size

The Bucket count and size options are used by Amazon Managed Grafana to calculate how big each cell in the heatmap is. You can define the bucket size either by count (the first input box) or by specifying a size interval. For the y-axis, the size interval is just a value. For the X-bucket, you can specify a time interval in the **Size** input. For example, you can set the time range to 1h. This will make the cells 1h wide on the x-axis.

Data format

Choose an option in the **Format** list.

- **Time series** – Amazon Managed Grafana does the bucketing by going through all time series values. The bucket sizes and intervals are set in the **Buckets** options.
- **Time series buckets** – Each time series already represents a y-axis bucket. The time series name (alias) must be a numeric value representing the upper or lower interval for the bucket. The Grafana workspace does no bucketing, so the bucket size options are hidden.

Display options

Use these settings to refine your visualization.

Colors

The color spectrum controls the mapping between value count (in each bucket) and the color assigned to each bucket. The color on the far-left side of the spectrum represents the minimum count, and the color on the far-right side represents the maximum count. Some color schemes are automatically inverted when using the light theme.

You can also change the color mode to **Opacity**. In this case, the color will not change, but the amount of opacity will change with the bucket count.

- **Mode**
  - **Opacity** – Bucket value represented by cell opacity. An opaque cell means the maximum value.
    - **Color** – Cell base color.
    - **Scale** – Scale for mapping bucket values to the opacity.
      - **linear** – Linear scale. Bucket value maps linearly to the opacity.
      - **sqrt** – Power scale. Cell opacity calculated as `value ^ k`, where k is a configured **Exponent** value. If the exponent is less than 1, you will get a logarithmic scale. If the exponent is greater than 1, you will get an exponential scale. In the case of 1, the scale will be the same as linear.
    - **Exponent** – value of the exponent, greater than `0`.
  - **spectrum** – Bucket value represented by cell color.
    - **Scheme** – If the mode is **spectrum**, select a color scheme.

Color scale

By default, Amazon Managed Grafana calculates cell colors based on minimum and maximum buckets values. With **Min** and **Max**, you can overwrite those values. Think of a bucket value as a z-axis and Min and Max as Z-Min and Z-Max respectively.

- **Min** – Minimum value used for cell color calculation. If the bucket value is less than Min, it is mapped to the minimum color. The default is `series min value`.
- **Max** – Maximum value used for cell color calculation. If the bucket value is greater than Max, it is mapped to the maximum color. The default is `series max value`.

Legend

Choose whether to display the heatmap legend on the visualization or not.

Buckets

- **Hide zero** – Do not draw cells that have zero values.
- **Space** – Set the space between cells in pixels. Default is 1 pixel.
- **Round** – Set the cell roundness in pixels. Default is 0.

Tooltip

- **Show tooltip** – Show the heatmap tooltip.
- **Histogram** – Show the y-axis histogram on the tooltip. The histogram represents distribution of the bucket values for the specific timestamp.
- **Decimals** – Set the number of decimals to render bucket value with (default is auto).

Histogram panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The histogram visualization calculates the distribution of values and presents them as a bar chart. The Y-axis and the height of each bar represent the count of values that fall into each bracket while the X-axis represents the value range.

Histogram visualization supports time series and any table results with one or more numerical fields.

Display options

Use these options to refine your visualizations:

Bucket size

The size of the buckets. Leave this empty for automatic bucket sizing (~10% of the full range).

Bucket offset

If the first bucket should not start at zero. A non-zero offset shifts the aggregation window. For example, 5-sized buckets that are 0–5, 5–10, 10–15 with a default 0 offset would become 2–7, 7–12, 12–17 with an offset of 2; offsets of 0, 5, or 10, in this case, would effectively do nothing. Typically, this option would be used with an explicitly defined bucket size rather than automatic. For this setting to affect, the offset amount should be greater than 0 and less than the bucket size; values outside this range will have the same effect as values within this range.

Combine series

This will merge all series and fields into a combined histogram.

Line width controls line width of the bars.

Fill opacity controls the fill opacity of the bars.

Gradient mode sets the mode of the gradient fill. Fill gradient is based on the line color. To change the color, use the standard color scheme field option. Gradient appearance is influenced by the Fill opacity setting.

- **None** – No gradient fill, this is the default setting.
- **Opacity** – Transparency of the gradient is calculated based o n the values on the Y-axis. Opacity of the fill is increasing with the values on the Y-axis.
- **Hue** – Gradient color is generated based on the hue of the line color.

Tooltip mode When you hover your cursor over the graph, Grafana can display tooltips. Choose how tooltips behave:

- **Single** – The hover tooltip shows only the series that you are hovering over.

- **All** – The hover tooltip shows all the series in the visualization. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden** – Do not display the tooltip.

> **Note**
>
> Use an override to hide individual series from the tooltip.

Legend options

When the legend option is enabled, it can show either the value mappings or the threshold brackets. To show the value mappings in the legend, it's important that the Color scheme option under standard options is set to Single color or Classic palette. To see the threshold brackets in the legend, set the Color scheme to From thresholds.

Legend mode Use these settings to refine how the legend appears in your visualization.

- **List** – Displays the legend as a list. This is a default display mode of the legend.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement Choose where to place the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend calculations

Choose which calculations to show in the legend. For more information, see Calculations list.

Logs panel

> This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The logs panel visualization shows log lines from data sources that support logs, such as Elastic, Influx, and Loki. Typically, you would use this panel next to a graph panel to display the log output of a related process.

The logs panel shows the result of queries that were entered on the **Query** tab. The results of multiple queries are merged and sorted by time. You can scroll inside the panel if the data source returns more lines than can be displayed.

To limit the number of lines rendered, you can use the **Max data points** setting in the **Query options**. If it is not set, the data source will usually enforce a default limit.

Display options

Use the following settings to refine your visualization:

- **Time** – Show or hide the time column. This is the timestamp associated with the log line as reported from the data source.
- **Unique labels** – Show or hide the unique labels column, which shows only non-common labels.
- **Wrap lines** – Toggle line wrapping.
- **Order** – Display results in descending or ascending time order. The default is **Descending**, showing the newest logs first. Set to **Ascending** to show the oldest log lines first.

News panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

This panel displays an RSS feed. By default, it displays articles from the Grafana Labs blog.

In the **Display** section, in the **URL** field, enter the URL of an RSS feed. This panel type does not accept any other queries.

Node graph panel (Beta)



This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The node graph panel visualizes directed graphs or networks. It uses directed force layout to effectively position the nodes so it can help with displaying complex infrastructure maps, hierarchies, or execution diagrams.

Data requirements

The node graph panel requires a specific shape of the data to be able to display its nodes and edges. Not every data source or query can be visualized in this panel.

The Node graph visualization consists of *nodes* and *edges*.

- A *node* is displayed as a circle. A node might represent an application, a service, or anything else that is relevant from an application perspective.
- An *edge* is displayed as a line that connects two nodes. The connection might be a request, an execution, or some other relationship between the two nodes.

Nodes

Usually, nodes show two statistical values inside the node and two identifiers just below the node, usually name and type. Nodes can also show another set of values as a color circle around the node, with sections of different color represents different values that should add up to 1. For example, you can have the percentage of errors represented by red portion of the circle.

Additional details can be displayed in a context menu, which is displayed when you choose the node. There also can be additional links in the context menu that can target either other parts of the Grafana workspace or any external link.

Edges

Edges can also show statistics when you hover over the edge. Similar to nodes, you can open a context menu with additional details and links by choosing the edge.

The first data source supporting this visualization is the AWS X-Ray data source for its service map feature. For more information, see Connect to an AWS X-Ray data source.

Additional details can be displayed in a context menu ,which is displayed when you choose the node. There also can be additional links in the context menu that can target either other parts of the Grafana workspace or any external link.

Navigating the node graph

You can pan within the node graph by choosing outside of any node or edge and dragging the mouse.

You can zoom by using the buttons on the upper left corner of the node graph.

Pie chart panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The pie chart displays reduced series, or values in a series, from one or more queries, as they relate to each other, as slices of a pie. The arc length, area, and central angle of a slice are all proportional to the slices value, as it relates to the sum of all values. This type of chart is best used when you want a quick comparison of a small set of values in an aesthetically pleasing form.

Pie chart visualizations allow you to apply the following options:

- Transformations.
- Field options and overrides.
- Thresholds.

Options

You can use the following options to refine your visualization.

- **Show** – Choose how much information to show. **Calculate** reduces each value to a single value per series. **All values** displays every value from a single series.
- **Calculation** – Select a calculation to reduce each series when **Calculate** has been selected. For information about available calculations, see Calculations list.
- **Limit** – When displaying every value from a single series, this limits the number of values displayed.
- **Fields** – Select which fields to display in the visualization.
  - **Numeric fields** – All fields with numeric values.
  - **All fields** – All fields that are not removed by transformations.
  - **Time** – All fields with time values.

Labels

Select labels to display on the pie chart. You can select more than one.

- **Name** – The series or field name.
- **Percent** – The percentage of the whole.
- **Value** – The raw numerical value.

Labels are displayed in white over the body of the chart. You might need to select darker chart colors to make them more visible. Long names or numbers might be clipped.

Legend placement and values

Choose where to display the legend.

- **Bottom** – Below the chart.
- **Right** – To the right of the chart.

You can select more than one value to display in the legend. **Percent** is the percentage of the whole and **Value** is the raw numerical value.

Plotly panel

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The Plotly panel renders charts using Plotly, an open source javascript graphing library.

The **Data**, **Layout** and **Config** fields match the common parameters described in the Plotly documentation. They must be in JSON format.

Data provided by the datasource can be transformed via a user-defined script before to be injected in the Plotly chart. The script includes 2 arguments.

- `data` – Data returned by the data source.
- `variables` – An object that contains Grafana variables in the current dashboard (user variables and these few global variables: `__from`, `__to`, `__interval`, and `__interval_ms`).

The script must return an object with one or more of the following properties: `data`, `layout`, `config` and `frames`. The following is an example.

```
let x = data.series[0].fields[0].values;
let y = data.series[0].fields[1].values;
let series = {
    x: x,
    y: y,
    name: variables.name, // where 'name' is the name of a Grafana dashboard variable
};

return {
    data: [series],
```

```
    config: {
        displayModeBar: false,
    },
};
```

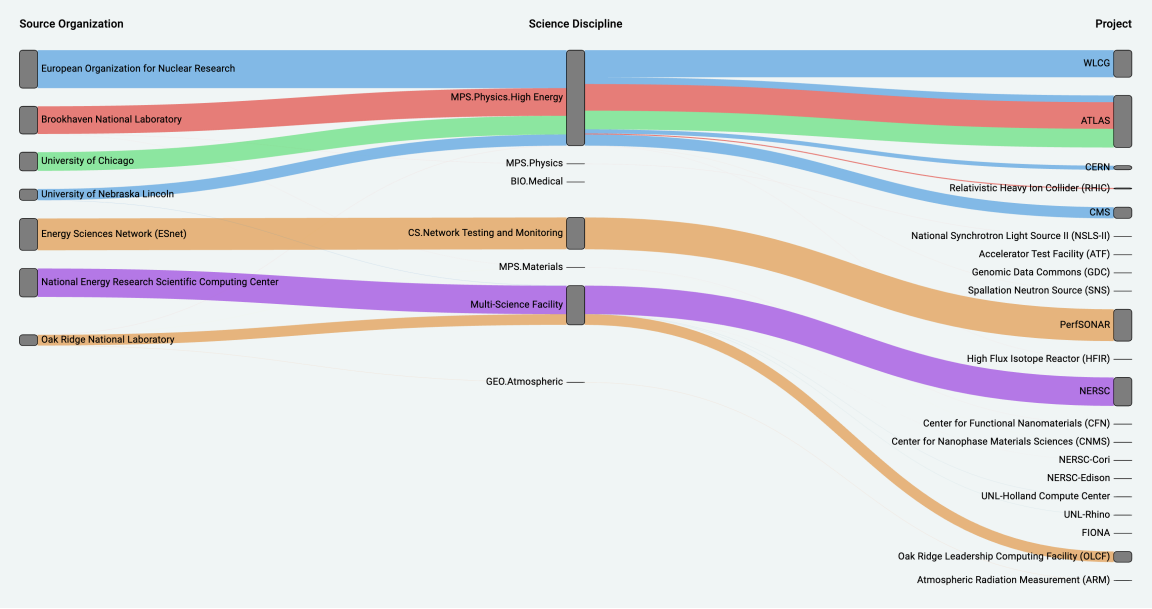
Object returned by the script and JSON provided in the *Data*, *Layout* and *Config* fields will be merged (deep merge).

If no script is provided, the panel will use only *Data*, *Layout* and *Config* fields.

Sankey panel

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The Sankey panel shows Sankey diagrams, which are good for visualizing flow data, with the width of the flow being proportional to the selected metric. The following image shows a Sankey diagram with two groups of source and destinations.



How it works

The sankey panel requires at least 2 columns of data, a source and destination for the flows. Your query should group your data into at least two groups. The panel will draw links from the first column of data points, to the last in order of the query. The thickness of the links will be proportionate to the value as assigned by the metric in the query.

Customizing

- **Links** – There are currently two options for link color: multi or single. It is multi-colored by default. To choose a single color for the links, toggle the **Single Link color only** option and choose your color from Grafana's color picker.
- **Nodes** – You can change the color of the rectangular nodes by changing the **Node color** option
- **Node Width** – The width of the nodes can be adjusted with the **Node Width** slider or entering a number in the input box. This number must be an integer.
- **Node Padding** – The vertical padding between nodes can be adjusted with the **Node Padding** slider or entering a number in the input box. This number must be an integer. If your links are too thin, try adjusting this number
- **Headers** – The column headers can be changed by using a **Display Name** override in the editor panel. They will be the same color you choose for **Text color**
- **Sankey Layout** – The layout of the Sankey links can be adjusted slightly using the **Layout iteration** slider. This number must be an integer and is the number of relaxation iterations used to generate the layout.

Scatter panel



This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The scatter panel shows an X/Y scatter plot for table data with a simpler interface than other graphing panels. Unlike the graph panel, the scatter panel does not require the data to be in a time series. The scatter panel requires a table formatted dataset with two or more numeric columns of data.

One of these can be assigned to the X axis. One or more can be assigned to a series of Y axis values and the resulting data plotted as a series of dots. Each series can optionally also show a regression line using one of a number of statistical best fits.

Creating a scatter panel

The following procedure describes how to create a scatter plot using the scatter panel. For this example, we will assume that there is data, as in the following table called `HEIGHT` with three columns of numerical values, `Age`, `Boys`, and `Girls`, showing the average height of boys and girls by age.

Age	Boy's Height	Girl's Height
5	109.7	109.5
6	115.6	115.4
7	121.1	120.8
8	126.3	126
9	131.3	131.3
10	136.2	137.1
11	141.2	143.2
12	147	148.7
13	153.6	152.6
14	159.9	155.1
15	164.4	156.7
16	167.3	157.6

Age	Boy's Height	Girl's Height
17	169	158
18	170	158.3
19	170.8	158.6

To create a scatter plot with the scatter panel

1. In your Grafana dashboard, choose **Add Panel**. For more details about adding panels, see Adding a panel.
2. For the Query, write a query that will return the data needed. In this case, you would use a query such as `SELECT * FROM HEIGHT`.
3. Select the **Scatter** visualization.

This will create a scatter plot, using the first column as the X axis, and the other numeric columns as Y axes.

Configuration options

The scatter panel provides the following four custom configuration options.

- **X Axis** – You can choose which field to use as the X axis, as well as extents and title and display information for the axis.
- **Y Axis** – You can choose which fields to display on the Y axis, including display options for each field, and extents and title information for the axis. You can also choose to display a regression line for each field. See the following information for more details on regression line configuration.
- **Legend** – You can turn a legend for the panel on or off, as well as choose the size of the text in the legend.
- **Display** – You can set other options for the chart, including grid color and border style.

Regression line configuration

Each Y axis dataset can display a line, in addition to the individual dots. There are five options for the line type.

- **None** – Do not display a regression line.
- **Simple** – Display a regression line that connects the dataset points.
- **Linear** – Display a straight line, using the least-squares, best-fit method.
- **Exponential** – Display an exponential best-fit regression line.
- **Power** – Display a power best-fit regression line.

Stat panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The stat panel shows a one large stat value with an optional graph sparkline. You can control the background or value color by using thresholds.

By default, the stat panel shows one of the following displays:

- Only the value for a single series or field.
- Both the value and name for multiple series or fields.

You can use the **Text mode** option to control whether the text is displayed or not.

Data and field options

Stat visualizations allow you to apply the following options:

- Transformations.
- Field options and overrides.
- Thresholds.

Automatic layout adjustment

The panel automatically adjusts the layout depending on available width and height in the dashboard. It automatically hides the graph (sparkline) if the panel becomes too small.

Display options

Use the following options to refine your visualization:

- **Show** – Choose how Amazon Managed Grafana displays your data.
  - **Calculate** – Show a calculated value based on all rows.
    - **Calculation** – Select a calculation to apply. For information about available calculations, see Calculations list.
  - **All values** – Show a separate stat for each row.
    - **Limit** – Specify the maximum number of rows to display.
- **Fields** – Select a field name or a field type (including **All fields** or **Numeric fields**) to include in this panel.
- **Value** – Select a reducer function that Amazon Managed Grafana will use to reduce many fields to a single value. Choose the **Value** list to see functions and brief descriptions.
- **Orientation** – Choose a stacking direction.
  - **Auto** – Amazon Managed Grafana selects what it thinks is the best orientation.
  - **Horizontal** – Bars stretch horizontally, left to right.
  - **Vertical** – Bars stretch vertically, top to bottom.
- **Text mode** – You can use the **Text mode** option to control what text the panel displays. If only the name and color are important, and the value is not, change the **Text mode** to **Name**. The value is still used to determine color and is displayed in a tooltip.
  - **Auto** – If the data contains multiple series or fields, show both the name and the value.
  - **Value** – Show only the value, never the name. The name is displayed in tooltip.
  - **Value and name** – Always show the value and the name.
  - **Name** – Show the name instead of the value. The value is displayed in the tooltip.
  - **None** – Show nothing (empty). The name and the value are displayed in the tooltip.
- **Color mode** – Choose a color mode.
  - **Value** – Colors only the value and graph area.
  - **Background** – Colors the background as well.
- **Graph mode** – Choose a graph mode.

- **None** – Hides the graph and shows only the value.
- **Area** – Shows the area graph below the value. This option requires that your query returns a time column.

- **Alignment mode** – Choose an alignment mode.
  - **Auto** – If only a single value is shown (no repeat), the value is centered. If multiple series or rows are shown, the value is left-aligned.
  - **Center** – Stat value is centered.

State timeline panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The state timeline panel visualization shows discrete state changes over time. Each field or series is rendered as its unique horizontal band. State regions can either be rendered with or without values. This panel works well with string or boolean states but can also be used with time series. When used with time series, the thresholds are used to turn the numerical values into discrete state regions.

State timeline options

Use these options to refine your visualizations:

Merge equal consecutive values

Controls whether Grafana merges identical values if they are next to each other.

Show values

Controls whether values are rendered inside the state regions. Auto will render values if there is sufficient space.

Align values

Controls value alignment inside state regions.

Row height

Controls space between rows. 1 = no space = 0.5 = 50% space.

Line width

Controls the line width of state regions.

Fill opacity

Controls the opacity of state regions.

Value mappings

To assign colors to boolean or string values, use Value mapping.

Time series data with thresholds

The panel can be used with time series data as well. In this case, the thresholds are used to turn the time series into discrete colored state regions.

Legend options

When the legend option is enabled, it can show either the value mappings or the threshold brackets. To show the value mappings in the legend, it's important that the Color scheme option under Standard options is set to Single color or Classic palette. To see the threshold brackets in the legend, set the Color scheme to From thresholds.

Legend mode Use these settings to refine how the legend appears in your visualization.

- **List** – Displays the legend as a list. This is a default display mode of the legend.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement Choose where to place the legend.

- **Bottom** – Below the graph.

- **Right** – To the right of the graph.

Status history panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The Status history visualization shows periodic states over time. Each field or series is rendered as a horizontal row. Boxes are rendered and centered around each value.

Status history visualization works with string, boolean, and numerical fields or time series. A time field is required. You can use value mappings to color strings or assign text values to numerical ranges.

Display options

Use these options to refine your visualizations:

Show values

Controls whether values are rendered inside the value boxes. Auto will render values if there is sufficient space.

Column width controls the width of boxes. 1=max and 0=Min width.

Line width controls line width of state regions.

Fill opacity controls the fill opacity of state regions.

Value mappings

To assign colors to boolean or string values, use Value mapping.

Time series data with thresholds

The panel can be used with time series data as well. In this case, the thresholds are used to color the boxes. You can also use gradient color schemes to color values.

Legend options

When the legend option is enabled, it can show either the value mappings or the threshold brackets. To show the value mappings in the legend, it's important that the Color scheme option under Standard options is set to Single color or Classic palette. To see the threshold brackets in the legend set the Color scheme to From thresholds.

Legend mode Use these settings to refine how the legend appears in your visualization.

- **List** – Displays the legend as a list. This is a default display mode of the legend.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Legend placement Choose where to place the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Table panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The table panel supports multiple modes for time series and for tables, annotation, and raw JSON data. This panel also provides date formatting, value formatting, and coloring options.

Data and field options

With table visualizations you can apply the following options:

- Transformations.
- Field options and overrides.
- Thresholds.

Display options

- **Show header** – Show or hide column names imported from your data source.
- **Sort ascending/descending** – Choose a column title to change the sort order from the default to descending to ascending. Each time you choose, the sort order changes to the next option in the cycle. You can sort by only one column at a time.
- Table field options – Change field options such as column width, alignment, and cell display mode.
- Filter table columns – Temporarily change how column data is displayed. For example, you can order values from highest to lowest or hide specific values.

Annotation support

Annotations are not currently supported in the new table panel.

Table field options

This section explains all available table field options. The options are listed in the same order as in Amazon Managed Grafana. Options listed in this topic apply only to table panel visualizations.

Most field options will not affect the visualization until you choose outside of the field option box that you are editing or press Enter.

For more information about applying these options, see Configure all fields and Configure specific fields.

Column alignment

Choose how Amazon Managed Grafana should align cell contents:

- Auto (default)

- Left
- Center
- Right

Column width

By default, Amazon Managed Grafana automatically calculates the column width based on the cell contents. In this field option, you can override the setting and define the width for all columns in pixels.

For example, if you enter `100` in the field, all the columns will be set to 100 pixels wide when you choose outside the field.

Cell display mode

By default, Amazon Managed Grafana automatically chooses display settings. You can override the settings by choosing one of the following options to change all fields.

> **Note**
>
> If you set these in the **Field** tab, the display modes apply to all fields, including the time field. Many options work best if you set them in the **Override** tab.

Color text

If thresholds are set, the field text is displayed in the appropriate threshold color.

Color background

If thresholds are set, the field background is displayed in the appropriate threshold color.

Gradient gauge

The threshold levels define a gradient.

LCD gauge

The gauge is split up in small cells that are lit or unlit.

JSON view

The value is shown formatted as code. If a value is an object, the JSON view that enables you to browse the JSON object appears when you pause on the value.

Column filter

Filter table columns

If you turn on the **Column filter** in table options, you can filter table options. For more information, see Table field options.

Turn on column filtering

1. In Amazon Managed Grafana, choose the dashboard that shows the table with the columns that you want to filter.
2. On the table panel that you want to filter, Opening the panel editor.
3. Choose the **Field** tab.
4. In **Table** options, turn on the **Column filter** option.

A filter icon appears next to each column title.

Filter column values

To filter column values, choose the filter (funnel) icon next to a column title. The Grafana workspace displays the filter options for that column.

Select the check boxes next to the values that you want to display. Enter text in the search field at the top to show those values in the display so that you can select them rather than scroll to find them.

Clear column filters

Columns with filters applied have a blue funnel displayed next to the title.

To remove the filter, choose the blue funnel icon, and then choose **Clear filter**.

Text panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

You can use the text panel to make information and description panels for your dashboards.

In **Mode**, select whether you want to use markdown or HTML to style your text, then enter content in the box below. The Grafana workspace includes a title and paragraph to help you get started, or you can paste content from another editor.

Time series panel

This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The time series panel can render a time series as a line, a path of dots, or a series of bars. This type of graph is versatile enough to display almost any time-series data.

Note

You can migrate Graph panel visualizations to Time series visualizations. To migrate, on the **Panel** tab, choose **Time series visualization**. Grafana transfers all applicable settings.

Time series visualizations enable you to apply the following options:

- Transformations
- Field options and overrides

- Thresholds

You can also use field options to create different types of graphs or adjust your axes.

Use these settings to refine your visualization.

Tooltip mode

When you hover your cursor over the graph, Grafana can display tooltips. Choose how tooltips behave:

- **Single** – The hover tooltip shows only the series that you are hovering over.
- **All** – The hover tooltip shows all the series in the graph. Grafana highlights the series that you are hovering over in bold in the series list in the tooltip.
- **Hidden** – Do not display the tooltip.

Legend mode and placement

Choose how the legend appears.

- **List** – Displays the legend as a list. This is the default.
- **Table** – Displays the legend as a table.
- **Hidden** – Hides the legend.

Choose where to display the legend.

- **Bottom** – Below the graph.
- **Right** – To the right of the graph.

Legend calculations

Choose which calculations to show in the legend. For more information, see Calculations list.

Graph time series as lines

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

This section explains how to use Time series field options to visualize time series data as lines and illustrates what the options do.

Create the panel

1. Create a panel, selecting the **Time series** visualization. For more information, see Adding a panel.
2. In the **Panel editor**, choose **Field**.
3. In **Style**, choose **Lines**.

Style the lines

There are a variety of options for styling the lines.

- **Line interpolation** – Choose how Grafana interpolates the series line. The choices are **Linear**, **Smooth**, **Step before**, and **Step after**.
- **Line width** – Set the line thickness between 0 and 10 pixels.
- **Fill opacity** – Set the opacity of the series fill, from 0 to 100 percent.
- **Gradient mode** – Set the mode of the gradient fill. Fill gradient is based on the line color. To change the color, use the standard color scheme field option.

  Gradient appearance is influenced by the setting for **Fill opacity**.

  The choices for gradient fill are **None**, **Opacity**, and **Hue**. With **Opacity**, Transparency of the gradient is calculated based on the values on the y-axis. Opacity of the fill is increasing with the values on the y-axis. With **Hue**, the gradient color is generated based on the hue of the line color.
- **Line style** – Set the style of the line. To change the color, use the standard color scheme field option.

  Line style appearance is influenced by the settings for **Line width** and **Fill opacity**.

The choices for line style are **Solid**, **Dash**, and **Dots**.

- **Null values** – Choose how gaps in the data are displayed. Null values can be connected to form a continuous line or, optionally, set a threshold above which gaps in the data should no longer be connected. You can choose to **Never** connect data points with gaps, **Always** connect data points with gaps, or set a **Threshold** at which gaps in the data should no longer be connected.
- **Show points** – Choose when the points should be shown on the graph. The choices are **Auto**, **Always**, and **Never**.

Fill below to

This option is available only in the overrides tab.

To fill the area between two series

1. Select the fields to fill below.
2. In **Add override property**, choose **Fill below to**.
3. Select the series that you want the fill to stop at.

Graph time series as bars

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

This section explains how to use Time series field options to visualize time series data as bars and illustrates what the options do.

Create the panel

1. Create a panel, selecting the **Time series** visualization. For more information, see Adding a panel.

2. In the **Panel editor**, choose **Field**.
3. In **Style**, choose **Bars**.

Style the bars

There are a variety of options for styling the bars.

- **Bar alignment** – Set the position of the bar relative to a data point. The choices are **Before**, **Center**, and **After**.
- **Line width** – Set the thickness of the bar outlines between 0 and 10 pixels.
- **Fill opacity** – Set the opacity of the bar fill, from 0 to 100 percent.
- **Gradient mode** – Set the mode of the gradient fill. Fill gradient is based on the line color. To change the color, use the standard color scheme field option.

  Gradient appearance is influenced by the setting for **Fill opacity**.

  The choices for gradient fill are **None**, **Opacity**, and **Hue**. With **Opacity**, Transparency of the gradient is calculated based on the values on the y-axis. Opacity of the fill is increasing with the values on the y-axis. With **Hue**, the gradient color is generated based on the hue of the line color.
- **Show points** – Choose when the points should be shown on the graph. The choices are **Auto**, **Always** and **Never**.

Graph time series as points

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

This section explains how to use Time series field options to visualize time series data as points and illustrates what the options do.

Create the panel

1. Create a panel, selecting the **Time series** visualization. For more information, see Adding a panel.
2. In the **Panel editor**, choose **Field**.
3. In **Style**, choose **Points**.

Style the points

When you graph as points, you can choose the point size.

- **Point size** – Choose the point size, between 1 and 40 pixels in diameter.

Change axis display

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

This section explains how to use Time series field options to control the display of axes in the visualization and illustrates what the axis options do.

There are a variety of options for the axes.

- **Y-axis placement** – Set the placement of the y-axis. The choices are **Left**, **Right**, and **Hidden**.
- **Y-axis label** – Set a text label for the y-axis. If you have more than one y-axis, you can use the **Override** tab to assign them different labels.
- **Width** – Set the fixed width of the axis. By default, the Grafana workspace dynamically calculates the axis width. By setting the width of the axis, data whose axes types are different can share the same display proportions. This makes it easier to compare more than one graph's worth of data because the axes are not shifted or stretched within visual proximity of each other.

- **Soft min and soft max** – Set a **Soft min** or **Soft max** for better control of y-axis limits. By default, the Grafana workspace sets the range for the y-axis automatically based on the data.

  Soft min or **Soft max** settings can prevent blips from appearing as mountains when the data is mostly flat, and hard min or max derived from standard min and max field options can prevent intermittent spikes from flattening useful detail by clipping the spikes past a defined point.
- **Scale** – Set the scale to use for y-axis values. The choices are **Linear** and **Logarithmic**.

Graph stacked time series

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

This section explains how to use Time series panel field options to control the stacking of the series and illustrates what the stacking options do. Stacking allows Grafana to display series on top of each other. Be cautious when using stacking in the visualization as it can easily create misleading graphs. You can read more on why stacking might be not the best approach here: The Issue with Stacking.

Stack series in groups

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The stacking group option is only available as an override.

To stack series in the same group

1. In the Overrides section, create a field override for the **Stack series** option.
2. Choose the **Normal** stacking mode.
3. Name the stacking group that you want the series to appear in. The stacking group name option is available only when creating an override.

Thresholds

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Thresholds set the color of either the value text or the background depending on conditions that you define.

You can define thresholds one of two ways:

- **Absolute** thresholds are defined based on a number; for example, 80 on a scale of 1–150.
- **Percentage** thresholds are defined relative to minimum or maximum; for example, 80 percent.

You can apply thresholds to the following visualizations:

- Bar gauge panel
- Gauge panel
- Graph panel
- Stat panel
- Table panel

Default thresholds

On visualizations that support it, Amazon Managed Grafana sets the following default threshold value: 80 = red; Base = green; Mode = Absolute.

The **Base** value represents minus infinity. It is generally the *good* color.

Adding a threshold

You can add as many thresholds to a panel as you want. The Grafana workspace automatically sorts thresholds from highest value to lowest.

> **Note**
>
> These instructions apply only to the stat, gauge, bar gauge, and table visualizations.

1. Choose the panel that you want to add a threshold to.
2. Choose the **Field** tab.
3. Choose **Add threshold**.

   Amazon Managed Grafana adds a threshold with suggested numerical and color values.
4. Accept the recommendations or edit the new threshold.

   - **Edit color** – Choose the color dot that you want to change, and then select a new color.
   - **Edit number** – Choose the number that you want to change, and then enter a new number.
   - **Thresholds mode** – Choose the mode to change it for all thresholds on this panel.
5. Choose **Save** to save the changes in the dashboard.

Adding a threshold to a graph panel

In the graph panel visualization, you can use thresholds to add arbitrary lines or sections to the graph to make it easier to see when the graph crosses a particular threshold.

1. Choose the graph panel that you want to add a threshold to.
2. On the **Panel** tab, choose **Thresholds**.
3. Choose **Add threshold**.
4. Fill in as many fields as you want. Only the **T1** fields are required.

- **T1** – Both values are required to display a threshold.
  - **lt** or **gt** – Select **lt** for less than or **gt** for greater than to indicate what the threshold applies to.
  - **Value** – Enter a threshold value. The Grafana workspace draws a threshold line along the y-axis at that value.
- **Color** – Choose a condition that corresponds to a color, or define your own color.
  - **custom** – You define the fill color and line color.
  - **critical** – Fill and line color are red.
  - **warning** – Fill and line color are yellow.
  - **ok** – Fill and line color are green.
- **Fill** – Choose whether the threshold fill is displayed.
- **Line** – Choose whether the threshold line is displayed.
- **Y-Axis** – Choose **left** or **right**.

5. Choose **Save** to save the changes in the dashboard.

Deleting a threshold

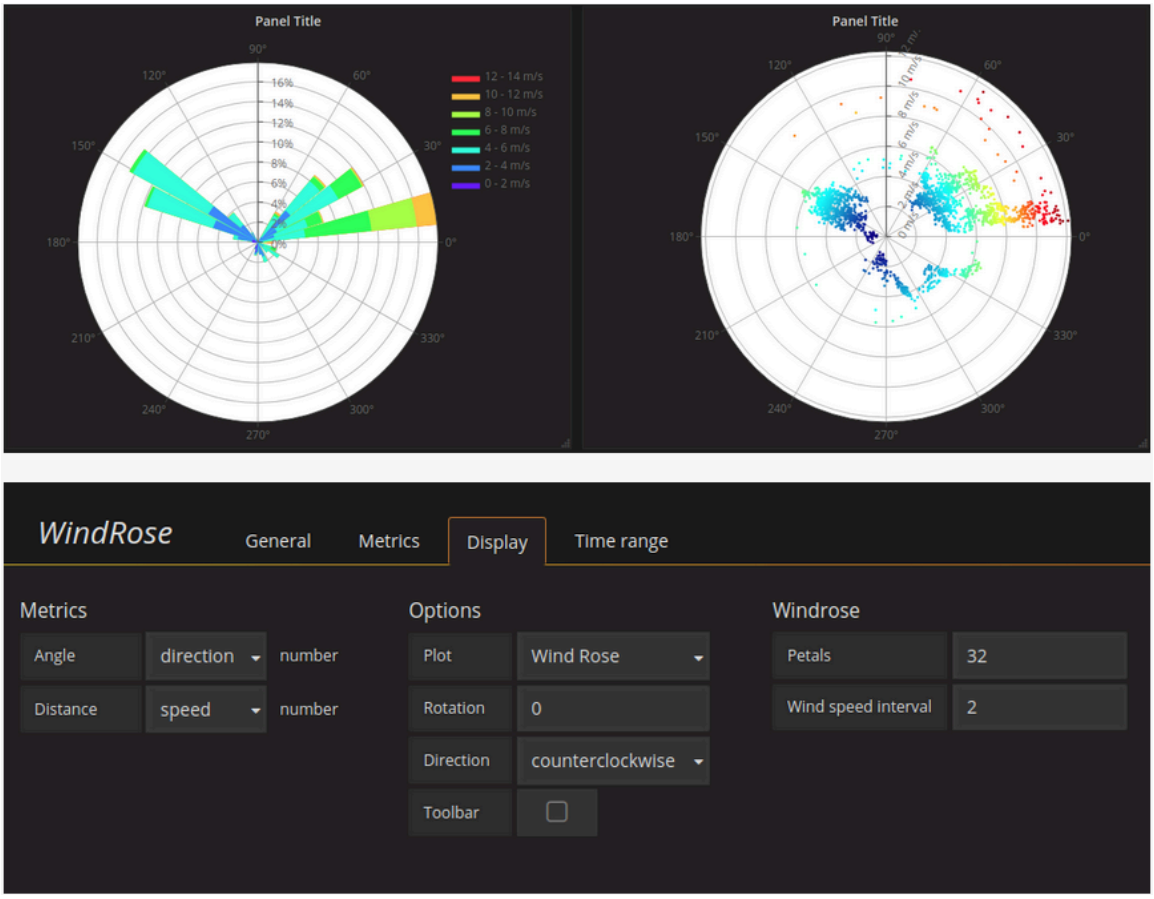
1. Choose the panel that you want to remove a threshold from.
2. Choose the **Field** tab. (Or, for a graph panel, choose the **Panel** tab.)
3. Choose the trash can icon next to the threshold that you want to remove.
4. Choose **Save** to save the changes in the dashboard.

WindRose



This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The WindRose panel receives raw time series data converts the data and maps it in a WindRose chart.



Options

The WindRose panel supports the following options:

- Axis frequency
- Axis style (degrees or compass)
- Scale (linear, square, log)

Inspect a panel

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

The panel inspector helps you understand and troubleshoot your panels. You can inspect the raw data for any Grafana workspace panel, export that data to a comma-separated values (CSV) file, view query requests, and export panel and data JSON.

Panel inspector UI

The panel inspector displays **Inspect: <NameOfPanelBeingInspected>** at the top of the pane. Choose the arrow in the upper right corner to expand or reduce the pane.

The panel inspector consists of four tabs:

- **Data tab** – Shows the raw data returned by the query with transformations applied. Field options, such as overrides and value mappings, are not applied by default.
- **Stats tab** – Shows how long your query takes and how much it returns.
- **JSON tab** – Allows you to view and copy the panel JSON, panel data JSON, and data frame structure JSON. This is useful if you are provisioning or administering Amazon Managed Grafana.
- **Query tab** – Shows you the requests to the server sent when Amazon Managed Grafana queries the data source.

Note

Not all panel types include all four tabs. For example, dashboard list panels do not have raw data to inspect, so they do not display the Stats, Data, or Query tabs.

Panel inspector tasks

In the panel inspector, you can inspect panels, inspect and download raw query results, inspect query performance, view panel JSON models, and view the raw request and response to the data source.

Open the panel inspector

You can inspect any panel that you can view.

1. In the Grafana workspace console, choose the dashboard that contains the panel that you want to inspect.
2. Choose the title of the panel that you want to inspect, and then choose **Inspect**. Or pause over the panel title, and then press **i**.

The panel inspector pane opens on the right side of the screen.

Inspect raw query results

View raw query results in a table. These are the data returned by the query with transformations applied and before the panel applies field options or field option overrides.

1. Open the panel inspector, and then choose the **Data** tab. Or in the panel menu, choose **Inspect**, **Data**.
2. If your panel contains multiple queries or queries multiple nodes, you have additional options.

- **Select result** – Choose which result set data you want to view.
- **Transform data**
  - **Join by time** – View raw data from all your queries at the same time, with one result set per column. Choose a column heading to reorder the data.

  View raw query results in a table with field options and option overrides applied.

  1. Open the **Data** tab in the panel inspector.
  2. Above the table, choose on **Data display options**.
  3. Choose the **Apply field configuration** toggle button.

Download raw query results as a CSV file

Amazon Managed Grafana generates a CSV file in your default browser download location. You can open it in the viewer of your choice.

1. Open the panel inspector.
2. Inspect the raw query results as described above. Adjust settings until you see the raw data that you want to export.
3. Choose **Download CSV**.

To download a CSV file formatted for Excel, expand the **Data options** panel and turn on the **Download for Excel** option before you choose **Download CSV**.

Inspect query performance

The **Stats** tab displays statistics that tell you how long your query takes, how many queries you send, and the number of rows returned. This information can help you troubleshoot your queries, especially if any of the numbers are unexpectedly high or low.

1. Open the panel inspector.
2. Choose the **Stats** tab.

Statistics are displayed in read-only format.

View panel JSON models

Explore and export panel, panel data, and data frame JSON models.

1. Open the panel inspector, and then choose the **JSON** tab. Or, in the panel menu, choose **Inspect**, **Panel JSON**.
2. In **Select source**, choose one of the following options:
   - **Panel JSON** – Displays a JSON object representing the panel.
   - **Panel data** – Displays a JSON object representing the data that was passed to the panel.
   - **DataFrame structure** – Displays the raw result set with your transformations, field configuration, and overrides applied.
3. You can expand or collapse portions of the JSON to explore it, or you can choose **Copy to clipboard** and paste the JSON in another application.

View raw request and response to data source

1. Open the panel inspector, and then choose the **Query** tab. Or, in the panel menu, choose **Inspect**, **Query**.
2. Choose **Refresh**.

Amazon Managed Grafana sends a query to the server to collect information, and then it displays the result. You can drill down on specific portions of the query, expand or collapse all of it, or copy the data to the clipboard to use in other applications.

Calculations list

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

This topic lists and defines the calculations used in Amazon Managed Grafana.

Among other places, these calculations are used in the **Transform** tab and the bar gauge, gauge, and stat visualizations.

Calculation	Description
All nulls	True when all values are null
All zeros	True when all values are 0
Change count	Number of times the field's value changes
Count	Number of values in a field
Delta	Cumulative change in value
Difference	Difference between first and last value of a field
Distinct count	Number of unique values in a field
First (not null)	First, not null value in a field

Calculation	Description
Max	Maximum value of a field
Mean	Mean value of all values in a field
Min	Minimum value of a field
Min (above zero)	Minimum, positive value of a field
Range	Difference between maximum and minimum values of a field
Step	Minimal interval between values of a field
Total	Sum of all values in a field

Dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

A *dashboard* is a set of one or more panels organized and arranged into one or more rows. Amazon Managed Grafana ships with a variety of panels. Amazon Managed Grafana makes it easy to construct the right queries and customize the display properties so that you can create the dashboard you need. Each panel can interact with data from any configured data source.

Manage dashboards

To control the time period for the dashboard, you can use the Time range controls in the upper right of the dashboard.

Dashboards can use templates and variables to make them more dynamic and interactive. For more information, see Templates and variables.

Dashboards can use Annotations to display event data across panels. This can help correlate the time series data in the panel with other events.

Dashboards can be shared easily in a variety of ways. For more information, see Sharing a dashboard.

Dashboards can be tagged, and the dashboard picker provides quick, searchable access to all dashboards in a particular organization.

Rows

A *row* is a logical divider within a dashboard. It is used to group panels together.

Rows are always 12 *units* wide. These units are automatically scaled based on the horizontal resolution of your browser. You can control the relative width of panels within a row by setting their specific width.

Amazon Managed Grafana uses a unit abstraction to optimize appearance on all screen sizes.

> **Note**
>
> With MaxDataPoint functionality, Amazon Managed Grafana can display the required number of data points, regardless of resolution or time range.

To collapse a row, choose the row title. If you save a dashboard with a row collapsed, the dashboard is saved in that state, and those graphs do not load until you expand the row.

Use the repeating rows functionality to dynamically create or remove entire rows of panels, based on the template variables selected.

Annotations

> This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Annotations provide a way to mark points on the graph with rich events. When you pause on an annotation, you can see an event description and event tags. The text field can include links to other systems for more detail.

Native annotations

Amazon Managed Grafana comes with a native annotation store and the ability to add annotation events directly from the graph panel.

Adding annotations

To add an annotation, press **Ctrl** or **Cmd** and choose where you want to add the annotation. To make the annotation searchable from other dashboards, add tags to it.

Adding region annotations

To create an annotation for a region, press **Ctrl** or **Cmd** while you choose the region.

Built-in query

After you add an annotation, it remains visible. This is because a built-in annotation query exists on all dashboards. This annotation query fetches all annotation events that originate from the current dashboard and displays them on the panel where they were created. This includes alert state history annotations. You can stop annotations from being fetched and displayed by choosing the **Dashboard settings** (gear) icon, choosing **Annotations**, and then modifying the query named `Annotations & Alerts (Built-in)`.

When you copy a dashboard using the **Save As** feature, the new dashboard has a new dashboard ID, so annotations created on the source dashboard are not visible on the copy. If the source dashboard annotations have tags to filter by, you can show the annotations on the copy by adding a new **Annotation Query** and filtering by the tags.

Query by tag

You can create new annotation queries that fetch annotations from the native annotation store by using the `-- Grafana --` data source and setting **Filter by** to **Tags**. Specify at least one tag.

For example, create an annotation query named `outages`, and specify a tag named `outage`. This query will show all annotations that you create (from any dashboard or via API) that have the `outage` tag.

By default, if you add multiple tags in the annotation query, Amazon Managed Grafana will show only annotations that have all the tags you supplied. To show annotations that contain at least one of the tags you supplied, turn on **Match any**.

In Amazon Managed Grafana, it's possible to use template variables in the tag query. For example, if you have a dashboard showing stats for different services and a template variable that controls which services to show, you can use the same template variable in your annotation query to show annotations for only those services.

Querying other data sources

Annotation events are fetched by using annotation queries. To add a new annotation query to a dashboard, choose the **Dashboard settings** (gear) icon, choose `Annotations`, and then choose **New**.

Specify a name for the annotation query. This name is displayed by the check box for showing or hiding annotation events for this query. For example, you might have two annotation queries named `Deploys` and `Outages`. You can select or clear the check boxes to specify which annotations to show.

Annotation query details

The annotation query options are different for each data source.

- Annotations using Graphite query editor
- Annotations using OpenSearch data source
- Annotations using Prometheus
- Annotations using MySQL
- Annotations using PostgreSQL

Dashboard folders

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.

For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Folders are a way to organize and group dashboards. This is useful if you have a lot of dashboards or if multiple teams use the same Grafana workspace.

Creating a folder

To create a folder, do one of the following:

- On the side menu, under the + icon, choose the **Create Folder** link.
- On the **Manage Dashboards** page, choose the **Create Folder** button.

On the **Create Folder** page, enter a unique name for the folder, and then choose **Create**.

When saving a dashboard, you can either choose an existing folder or create a new folder.

Manage dashboards

On the **Manage Dashboards** page, you can perform a variety of tasks:

- Create a folder.
- Create a dashboard.
- Move dashboards into folders.
- Delete multiple dashboards.
- Navigate to a folder page (where you can set permissions for a folder or its dashboards).

Dashboard Folder page

To open a Dashboard Folder page, choose the cog icon that appears when you pause on a folder in the dashboard list in the search results or on the Manage Dashboards page.

The Dashboard Folder page is similar to the Manage Dashboards page. On the Dashboard Folder page, you can perform the following tasks:

- Move or delete dashboards in a folder.
- Rename a folder (on the **Settings** tab).
- Set permissions for the folder (inherited by dashboards in the folder).

Permissions

Permissions can be assigned to a folder and inherited by the dashboards that it contains. An Access Control List (ACL) is used where **Organization Role**, **Team** and Individual **User** can be assigned permissions. For more information, see Dashboard and folder permissions.

Playlist

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

A playlist is a list of dashboards that are displayed in a sequence. You can use a playlist to build situational awareness or to present your metrics to your team or visitors.

Amazon Managed Grafana automatically scales dashboards to any resolution, including big screens.

You can access the **Playlist** feature from the side menu, in the **Dashboards** submenu.

Creating a playlist

A playlist presents dashboards in a sequence, with a set order and a time interval between dashboards.

1. To access the **Playlist** feature, pause on the side menu.
2. Choose **Playlists**.
3. Choose **New playlist**.
4. In the **Name** text box, enter a name for your playlist.

5. In the **Interval** text box, enter a time interval.

   The time interval is the amount of time for Amazon Managed Grafana to stay on a particular dashboard before advancing to the next one on the playlist.
6. Next to each dashboard that you want to add to your playlist, choose **Add to playlist**.
7. Choose **Create**.

Editing a playlist

You can edit playlists while creating them or after saving them.

1. To access the Playlist feature, pause on the side menu.
2. Choose **Playlists**.
3. Choose the playlist that you want to edit.

Editing the name of a playlist

1. Choose the **Name** text box.
2. Edit the name.
3. Choose **Save** to save your changes.

Editing the interval of a playlist

1. Choose the **Interval** text box.
2. Edit the interval.
3. Choose **Save** to save your changes.

Adding a dashboard to a playlist

1. Next to the dashboard that you want to add, choose **Add to playlist**.
2. Choose **Save** to save your changes.

Searching for a dashboard to add

1. Under **Add dashboards**, choose the **Search dashboards by name** text box.

2. Enter a name or regular expression.
3. If needed, filter your results by starred status or tags. By default, your starred dashboards appear as options to add to the playlist.
4. Choose **Save**to save your changes.

Rearranging dashboard order

1. Next to the dashboard that you want to move, choose the up or down arrow.
2. Choose **Save** to save your changes.

Removing a dashboard

1. Choose the x icon to remove a dashboard from the playlist.
2. Choose **Save** to save your changes.

Deleting a playlist

1. Choose **Playlists**.
2. Next to the playlist that you want to delete, choose the x icon.

Saving a playlist

You can save a playlist to add it to your **Playlists** page, where you can start it. Be sure to add all the dashboards that you want to appear in your playlist before you save it.

1. To access the **Playlist** feature, pause on the side menu.
2. Choose **Playlists**.
3. Choose the playlist.
4. Edit the playlist.

   Ensure that your playlist has a **Name**, **Interval**, and at least one **Dashboard** added to it.
5. Choose **Save**.

Starting a playlist

You can start a playlist in five different view modes. The mode determines how the menus and navigation bar are displayed on the dashboards.

By default, each dashboard is displayed for the amount of time entered in the **Interval** field, which can be set while creating or editing a playlist. After you start a playlist, you can control it by using the navbar at the top of your screen.

1. On the **Dashboards** menu, choose **Playlists**.
2. Next to the playlist that you want to start, choose **Start playlist**.
3. In the dropdown list, choose one of the following display modes:

   - **Normal mode**
     - The side menu remains visible.
     - The navbar, row, and panel controls appear at the top of the screen.
   - **TV mode**
     - The side menu is hidden or removed.
     - The navbar, row, and panel controls appear at the top of the screen.
     - TV mode is turned on automatically after 1 minute of user inactivity.
     - You can turn TV mode on manually by using the **d v** sequence shortcut, or by appending the parameter `?inactive` to the dashboard URL.
     - You can disable TV mode with any mouse movement or keyboard action.
   - **TV mode (with auto fit panels)**
     - The side menu is hidden or removed.
     - The navbar, row, and panel controls appear at the top of the screen.
     - Dashboard panels automatically adjust to optimize space on screen.
   - **Kiosk mode**
     - The side menu, navbar, row, and panel controls are completely hidden or removed from view.
     - You can turn Kiosk mode on manually by using the **d v** sequence shortcut after the playlist has started.
     - You can turn off Kiosk mode manually by using the same shortcut.
   - **Kiosk mode (with auto fit panels):**

- The side menu, navbar, row, and panel controls are completely hidden or removed from view.
- Dashboard panels automatically adjust to optimize space on screen.

Controlling a playlist

You can control a playlist in **Normal** or **TV** mode after it has started by using the navigation bar at the top of your screen.

Button	Result
Next (double right arrow)	Advances to the next dashboard.
Back (left arrow)	Returns to the previous dashboard.
Stop (square)	Ends the playlist, and exits to the current dashboard.
Cycle view mode (monitor icon)	Changes the display of the dashboards to different view modes.
Time range	Displays data within a time range. It can be set to display the last 5 minutes up to 5 years ago, or a custom time range, using the dropdown arrow.
Refresh (circle arrow)	Reloads the dashboard to display the current data. It can be set to reload automatically, from every 5 seconds to 1 day, by using the dropdown arrow.

To stop the playlist from your keyboard, press **Esc**.

Sharing a playlist in a view mode

You can share a playlist by copying the URL in the view mode that you want and pasting the URL to your destination.

1. From the **Dashboards** menu, choose **Playlists**.
2. Next to the playlist that you want to share, choose **Start playlist**, and then choose the view mode that you want.
3. To copy the URL to your clipboard, choose **Copy Link Address**.

   For example, the URL for a playlist on the Grafana Play site in Kiosk mode could be `https://play.grafana.org/d/000000010/annotations?orgId=1&kiosk`
4. Paste the URL to your destination.

Dashboard search

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Dashboards can be searched by the dashboard name, filtered by one (or many) tags, or filtered by starred status. The dashboard search is accessed through the dashboard picker, available in the dashboard top navigation bar. The dashboard search can also be opened by using the shortcut **F**.

When using only a keyboard, you can use the keyboard arrow keys to navigate the results, and press **Enter** to open the dashboard that you want.

Finding by dashboard name

Type any part of the dashboard name in the search bar. As you type, search returns results for any partial string match in real time.

Dashboard search is the following:

- Real time
- Not case sensitive

- Functional across stored and file-based dashboards

Filtering by tags

Tags are a helpful way to organize your dashboards, especially as the number of dashboards grows. Tags can be added and managed in the dashboard **Settings**.

To filter the dashboard list by tag, choose any tag that appears in the right column. You can further filter the list by choosing additional tags.

To see a list of all available tags, choose the **Filter by tags** dropdown menu. When you select a tag, the dashboard search is instantly filtered.

When using only a keyboard, press **Tab** to focus on the tags link, press the down arrow key to find a tag, and press **Enter** to select the tag.

> **Note**
>
> When multiple tags are selected, Amazon Managed Grafana shows dashboards that include all the tags.

Sharing a dashboard

> This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

To share a dashboard, choose **Share dashboard** (the share icon) in the top navigation bar. This opens the **Share** dialog box, where you can get a link to the current dashboard with the current selected time range and template variables. If you have made changes to the dashboard, be sure to save those changes before you copy the link.

Dashboard snapshot

A dashboard snapshot is an instant way to share an interactive dashboard publicly. When creating the snapshot, Amazon Managed Grafana strips sensitive data such as queries (metric, template, and annotation) and panel links, leaving only the visible metric data and series names embedded in your dashboard. Dashboard snapshots can be accessed by anyone who has the link and can reach the URL.

Publish snapshots

You can publish snapshots to your local instance.

Sharing a panel

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Choose a panel title to open the panel menu, and then choose **Share** in the panel menu to open the **Share Panel** dialog box. You can copy the link, which will take you to exactly this panel with the current time range and selected template variables.

Time range controls

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Amazon Managed Grafana provides several ways to manage the time ranges of the data that are being visualized, both at the dashboard level and at the panel level.

This topic describes supported time units and relative ranges, the common time controls, dashboard-wide time settings, and panel-specific time settings.

> **Note**
>
> To have time controls, your data must include a time column. See the documentation for your specific data source for more information about including a time column.

Time units and relative ranges

The following time units are supported:

- `s` (seconds)
- `m` (minutes)
- `h` (hours),
- `d` (days)
- `w` (weeks)
- `M` (months)
- `y` (years)

Use the minus operator to step back in time, relative to now. To display the full period of the unit (such as day, week, or month), append `/<time unit>`.

Use the plus operator to step forward in time relative to now. You can use this feature to look at predicted data for the future.

Here are some examples:

Example relative range	From	To
Last 5 minutes	`now-5m`	now
The day so far	`now/d`	now

Example relative range	From	To
This week	`now/w`	`now/w`
Week to date	`now/w`	`now`
Previous month	`now-1M/M`	`now-1M/M`

Common time range controls

The dashboard and panel time controls have a common user interface, with the following options.

Current time range

The current time range, also called the *time picker*, shows the time range currently displayed in the dashboard or panel that you are viewing.

Pause on a field to see the exact timestamps in the range and their source, such as the local browser.

To change the time range, choose on the current time range. You can change the current time using a *relative time range*, such as the last 15 minutes, or an *absolute time range*, such as `2020-05-14 00:00:00 to 2020-05-15 23:59:59`.

Relative time range

Select the relative time range from the **Relative time ranges** list. Here are some examples of relative time ranges:

- Last 30 minutes
- Last 12 hours
- Last 7 days
- Last 2 years
- Yesterday
- Day before yesterday
- This day last week
- Today so far

- This week so far
- This month so far

Absolute time range

Set an absolute time range one of two ways:

- Enter values in the **From** and **To** fields. You can enter exact time values or relative values, such as `now-24h`, and then choose **Apply time range**.
- Choose the **From** or **To** field. Amazon Managed Grafana displays a calendar. Choose the day or days that you want to use as the current time range and then choose **Apply time range**.

Amazon Managed Grafana also displays recently used absolute ranges.

Zoom out (Cmd+Z or Ctrl+Z)

To view a larger time range in the dashboard or panel visualization, choose the **Time range zoom out** icon.

Zoom in (for graph visualizations only)

In the graph visuaization, drag to select the time range that you want to view.

Refresh dashboard

Choose the **Refresh dashboard** icon to run every query on the dashboard immediately and refresh the visualizations. Amazon Managed Grafana cancels any pending requests when a new refresh is started.

By default, Amazon Managed Grafana does not automatically refresh the dashboard. Queries run on their own schedule according to the panel settings. However, if you want to regularly refresh the dashboard, choose the down arrow next to the **Refresh dashboard** icon, and then select a refresh interval.

Dashboard time settings

Time settings are saved on a per-dashboard basis.

To access the dashboard time settings, choose the **Dashboard settings** (gear) icon at the top of the screen. The settings are in the **Time Options** section of the **General** tab.

- **Timezone** – The local time zone of the service or system that you are monitoring. This can be helpful when you monitor a system or service that operates across several time zones.
  - **Default** – The default selected time zone for the user profile, team, or organization. If no time zone is specified for the user profile, a team that the user is a member of, or the organization, Amazon Managed Grafana uses local browser time.
  - **Browser Time** The time zone that is configured for the browser that is being used. This is usually the time zone that is set on the computer.
  - **Coordinated Universal Time** – Standard ISO 8601 time zones, including UTC. For more information, see a list of time zones.
- **Auto-refresh** – Customizable options the relative time and auto-refresh settings. Entries are separated by commas and can be any valid time unit.
- **Now delay now-** – Time delay value that overrides the now value. Most commonly, this feature is used to avoid null values by accommodating known delays in data aggregation.
- **Hide time picker** – Option for not displaying the time picker.

Panel time overrides and time shift

In Query options, you can override the relative time range for individual panels, causing them to be different from what is selected in the dashboard time picker in the top navigation bar. You can show metrics from different time periods or days at the same time.

Controlling the time range using a URL

You can control the time range of a dashboard by providing the following query parameters in the dashboard URL:

- `from` – Defines the lower limit of the time range, specified in ms epoch or relative time. For more information, see Relative time range.
- `to` – Defines the upper limit of the time range, specified in ms epoch or relative time. For more information, see Relative time range.
- `time` and `time.window` – Define a time range from `time-time.window/2` to `time+time.window/2`. Both parameters should be specified in milliseconds. For example, `?time=1500000000000&time.window=10000` results in a 10-second time range from 1499999995000 to 1500000005000

Exporting and importing dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Amazon Managed Grafana Dashboards can easily be exported and imported, either from the UI or from the [HTTP API]

Exporting a dashboard

Dashboards are exported in Amazon Managed Grafana JSON format, and contain everything you need, including layout, variables, styles, data sources, and queries, to import the dashboard at a later time.

The export feature is accessed in the share window, which you open by choosing the share button in the dashboard menu.

Making a dashboard portable

When you export a dashboard for others to use, it's good to add template variables for values such as a metric prefix (use a constant variable) and a server name.

A template variable of the type `Constant` is automatically hidden in the dashboard. It is also added as a required input when the dashboard is imported. For more information about templating and template variables, see Templates and variables.

Importing a dashboard

To import a dashboard, choose the + icon in the side menu, and then choose **Import**.

You can upload a dashboard JSON file, paste a dashboard URL or paste dashboard JSON text directly into the text area.

In step 2 of the import process, you can change the name of the dashboard, specify the data source that you want the dashboard to use, and specify any metric prefixes (if the dashboard uses any).

Discover dashboards on Grafana.com

Find dashboards for common server applications at Grafana.com/dashboards.

Dashboard version history

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Whenever you save a version of your dashboard, a copy of that version is saved so that previous versions of your dashboard are not lost. A list of these versions is available by choosing **Dashboard settings** and then selecting **Versions** in the left side menu.

The dashboard version history feature lets you compare and restore to previously saved dashboard versions.

Comparing two dashboard versions

To compare two dashboard versions, select the two versions from the list that you want to compare. After you select two versions, choose **Compare versions** to open the diff view. By default, you'll see a textual summary of the changes, as in the following image.

To view the diff of the raw JSON that represents your dashboard, choose **View JSON Diff**.

To restore to the earlier version that you are comparing against, choose **Restore to version <x>**.

Restoring to a previously saved dashboard version

If you need to restore to a previously saved dashboard version, you can do so either by choosing the "Restore" button on the right of a row in the dashboard version list or by choosing **Restore to**

version <x> appearing in the diff view. After you choose to restore, a pop-up box will prompt you to confirm the restoration.

After restoring to a previous version, a new version will be created that containing the same exact data as the previous version, but with a different version number. This is indicated in the **Notes** column. This helps to ensure that your previous dashboard versions are not affected by the change.

Keyboard shortcuts

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Amazon Managed Grafana has a number of keyboard shortcuts available. To display all keyboard shortcuts available in your version of Amazon Managed Grafana, press **Shift + ?** on your keyboard.

Amazon Managed Grafana includes the following popular shortcuts:

- **Ctrl+S** saves the current dashboard.
- **Ctrl+F** opens the dashboard finder / search.
- **Ctrl+H** hides all controls (hiding controls works well for TV displays).
- **Escape** exits a graph when in full-screen or edit mode.

Dashboard JSON model

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

A dashboard in Amazon Managed Grafana is represented by a JSON object, which stores metadata of its dashboard. Dashboard metadata includes dashboard properties, metadata from panels, template variables, and panel queries.

To view the JSON of a dashboard

1. Open a dashboard.
2. On the top navigation bar, choose on **Manage dashboard**.
3. Select **View JSON** from the dropdown menu.

JSON fields

When a user creates a new dashboard, a new dashboard JSON object is initialized with the following fields.

Note

In the following JSON, `id` is shown as `null`, which is the default value assigned to it until a dashboard is saved. After you save a dashboard, an integer value is assigned to the `id` field.

```
{
  "id": null,
  "uid": "cLV5GDCkz",
  "title": "New dashboard",
  "tags": [],
  "style": "dark",
  "timezone": "browser",
  "editable": true,
  "hideControls": false,
  "graphTooltip": 1,
  "panels": [],
  "time": {
    "from": "now-6h",
    "to": "now"
  },
```

```
  "timepicker": {
    "time_options": [],
    "refresh_intervals": []
  },
  "templating": {
    "list": []
  },
  "annotations": {
    "list": []
  },
  "refresh": "5s",
  "schemaVersion": 17,
  "version": 0,
  "links": []
}
```

The following table provides usage details for each field in the dashboard JSON.

Name	Usage
id	The unique numeric identifier for the dashboard (generated by the database).
uid	The unique dashboard identifier that can be generated by anyone. The `uid` is a string of 8-40 characters.
title	The current title of the dashboard.
tags	The tags that are associated with the dashboard. In the JSON, the tags are an array of strings.
`style`	The theme of the dashboard (for example, `dark` or `light`).
`timezone`	The timezone of dashboard (`utc` or `browser`).
`editable`	Whether a dashboard can be edited.
`graphTooltip`	The tooltip style. • `0` for no shared crosshair or tooltip (default) • 1 for shared crosshair • 2 for shared crosshair and shared tooltip

Name	Usage
`time`	The time range for the dashboard (for example, last 6 hours, last 7 days).
`timepicker`	The timepicker metadata. For more information, see Time picker.
`templating`	The templating metadata. For more information, see Templates and variables.
`annotations`	The annotations metadata. For more information, see Annotations.
`refresh`	The auto-refresh interval.
`schemaVersion`	The version of the JSON schema (integer), which is incremented each time an Amazon Managed Grafana update changes the schema.
`version`	The version of the dashboard (integer), which is incremented each time the dashboard is updated.
`panels`	The panels array. For more information, see Panels.

Panels

Panels are the building blocks of a dashboard. It consists of data source queries, type of graphs, aliases, and other data. Panel JSON consists of an array of JSON objects, each representing a different panel. Most of the fields are common for all panels, but some fields depend on the panel type. The following example shows the panel JSON of a text panel.

```
"panels": [
  {
    "type": "text",
    "title": "Panel Title",
    "gridPos": {
      "x": 0,
      "y": 0,
      "w": 12,
      "h": 9
```

```
    },
    "id": 4,
    "mode": "markdown",
    "content": "# title"
  }
```

Panel size and position

The `gridPos` property describes the panel size and position in grid coordinates:

- `w` – 1-24. The width of the dashboard is divided into 24 columns.
- `h` – In grid height units. Each grid height unit represents 30 pixels.
- `x` – The x position. The x position uses the in same column unit as `w`.
- `y` – The y position. The y position uses the same grid height unit as `h`.

The grid has a negative gravity that moves panels up if there is empty space above a panel.

Time picker

The following example shows the `timepicker` options.

```
"timepicker": {
    "collapse": false,
    "enable": true,
    "notice": false,
    "now": true,
    "refresh_intervals": [
      "5s",
      "10s",
      "30s",
      "1m",
      "5m",
      "15m",
      "30m",
      "1h",
      "2h",
      "1d"
    ],
    "status": "Stable",
    "type": "timepicker"
```

```
  }
```

The following table provides usage details for `timepicker`.

Name	Usage
`collapse`	Whether `timepicker` is collapsed
`enable`	Whether `timepicker` is activated
`notice`	TODO
`now`	TODO
`refresh_intervals`	TODO
`status`	TODO
`type`	TODO

Templating

The `templating` field contains an array of template variables with their saved values and other metadata. The following example shows `templating` metadata.

```
"templating": {
   "enable": true,
   "list": [
     {
       "allFormat": "wildcard",
       "current": {
         "tags": [],
         "text": "prod",
         "value": "prod"
       },
       "datasource": null,
       "includeAll": true,
       "name": "env",
       "options": [
         {
```

```
            "selected": false,
            "text": "All",
            "value": "*"
          },
          {
            "selected": false,
            "text": "stage",
            "value": "stage"
          },
          {
            "selected": false,
            "text": "test",
            "value": "test"
          }
        ],
        "query": "tag_values(cpu.utilization.average,env)",
        "refresh": false,
        "type": "query"
      },
      {
        "allFormat": "wildcard",
        "current": {
          "text": "apache",
          "value": "apache"
        },
        "datasource": null,
        "includeAll": false,
        "multi": false,
        "multiFormat": "glob",
        "name": "app",
        "options": [
          {
            "selected": true,
            "text": "tomcat",
            "value": "tomcat"
          },
          {
            "selected": false,
            "text": "cassandra",
            "value": "cassandra"
          }
        ],
        "query": "tag_values(cpu.utilization.average,app)",
        "refresh": false,
```

```
        "regex": "",
        "type": "query"
      }
    ]
  }
```

The following table provides usage details for templating section.

Name	Usage
enable	Whether templating is activated.
list	An array of objects, each representing one template variable
allFormat	The format to use while fetching all values from the data source (for example, `wildcard`, `glob`, `regex`, and `pipe`).
current	Shows current selected variable text or value on the dashboard
data source	Shows the data source for the variables
includeAll	Whether the all value option is available
multi	Whether multiple values can be selected from variable value list
multiFormat	The format to use while fetching timeseries from the data source
name	The name of a variable
options	The array of variable text/value pairs available for selection on dashboard
query	The data source query that is used to fetch values for a variable
refresh	TODO
regex	TODO
type	The type of variable (`custom`, `query`, or `interval`)

Scripted dashboards

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

⚠ **Warning**

This feature is deprecated and will be removed in a future release.

If you have many metric names that change (for example, new servers) in a defined pattern, it can be time consuming to constantly create new dashboards.

With scripted dashboards, you can dynamically create your dashboards using JavaScript. In the Grafana install folder, under `public/dashboards/`, there is a file named `scripted.js`. This file contains an example of a scripted dashboard. You can access it by using the URL: `http://grafana_url/dashboard/script/scripted.js?rows=3&name=myName`

When you open `scripted.js`, you can see how it reads URL parameters from the ARGS variable and then adds rows and panels.

Example: scripted.js

```
var seriesName = 'argName';

if (!_.isUndefined(ARGS.name)) {
  seriesName = ARGS.name;
}

dashboard.panels.push({
  title: 'Events',
```

```
  type: 'graph',
  fill: 1,
  linewidth: 2,
  gridPos: {
    h: 10,
    w: 24,
    x: 0,
    y: 10,
  },
  targets: [
    {
      target: "randomWalk('" + seriesName + "')",
    },
    {
      target: "randomWalk('random walk2')",
    },
  ],
});

return dashboard;
```

More examples

You can find more examples in the `public/dashboards/` directory of your Grafana installation.

Explore

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

In a Grafana workspace, the dashboard UI provides tools for building dashboards for visualization. *Explore* strips away all the dashboard and panel options so that you can focus on the query. Iterate until you have a working query, and then plan and build a dashboard.

For infrastructure monitoring and incident response, you no longer need to switch to other tools to debug what went wrong. You can use Explore to dig deeper into your metrics and logs to find the cause.

Explore makes it easier to view your data without creating a dashboard. If your data source supports graph and table data, Explore shows the results both as a graph and as a table. This helps you to see trends in the data and more details at the same time.

Start exploring

> **Note**
>
> By default, users with the Viewer role cannot edit and do not have access to Explore.

The **Explore** icon on the left side menu opens an empty Explore tab.

To start with an existing query in a panel, choose the **Explore** option from the **Panel** menu. This opens an Explore tab that contains the query from the panel. You can then to tweak or iterate in the query outside of your dashboard.

Choose your data source from the dropdown list in the top left. Prometheus has a custom Explore implementation. The other data sources use their standard query editor.

In the query field, you can write your query and explore your data. There are three buttons beside the query field, a clear button (X), an add query button (+) and the remove query button (-). As in the panel query editor, you can add and remove multiple queries.

Splitting and comparing

The split view feature is a way to compare graphs and tables side-by-side or to look at related data together on one page. Choose **Split** to duplicate the current query and split the page into two side-by-side queries. You have the option of selecting a different data source for the new query. This gives you the opportunity to compare the same query for two different servers or to compare the staging environment to the production environment.

In split view, time pickers for both panels can be linked (if you change one, the other gets changed as well) by choosing one of the time-sync buttons attached to the time pickers. Linking the time pickers helps keep the start and end times of the split view queries in sync, so that you’re looking at the same time interval in both split panels.

You can close the newly created query by choosing **Close Split**.

Sharing a shortened link

Use the **Share shortened link** capability to create smaller and simpler URLs of the format `/goto/:uid` instead of sharing longer URLs that contain complex query parameters. You can create a shortened link by choosing the **Share** option in the Explore toolbar. Any shortened links that are never used are automatically deleted after 7 days.

Query history

Query history is a list of queries that you have used in Explore. The history is local to your browser and is not shared. To open and interact with your history, choose **Query history** in Explore.

Viewing the query history

FIn the query history, you can do the following:

- Run a query.
- Create or edit a comment.
- Copy a query to the clipboard.
- Copy a shortened link with the query to the clipboard.
- Star a query.

Managing favorite queries

All queries that have been starred in the Query history tab are displayed on the Starred tab. You can access your favorite queries faster and reuse those queries without retyping them from.

Sort query history

By default, query history shows you the most recent queries. You can sort your history by date or by data source name in ascending or descending order.

On the right side of the query history, in the dropdown list, choose one of the following options: field.

- Newest first
- Oldest first
- Data source A-Z

- Data source Z-A

> **Note**
>
> If you are in split view, the sorting mode applies only to the active panel.

Filter query history

On the **Query history** and **Starred** tabs, you can filter the query history by data source name.

1. Choose **Filter queries for specific data source(s)**.
2. Select the data source that you want to use to filter your history. You can select multiple data sources.

On the **Query history** tab, you can use the vertical slider to filter queries by date:

- Drag the lower handle to adjust the start date.
- Drag the upper handle to adjust the end date.

> **Note**
>
> If you are in split view, filters are applied only to the active panel.

Searching in the query history

You can search in your history across queries and your comments. Search is possible for queries in the **Query history** and **Starred** tabs.

1. Choose the **Search queries** field.
2. In the search field, enter your search term.

Query history settings

You can customize the query history in the **Settings** tab. The following table lists the available options.

Setting	Default value
Specify how long Grafana will save your query history.	1 week
Change the default active tab.	Query history tab
Show queries only for the data source that is currently active in Explore.	True
Clear query history.	(Choose **Clear query history** to permanently delete all stored queries.)

> **Note**
>
> Query history settings are global, and they are applied to both panels in split mode.

Prometheus-specific features

The first version of Explore features a custom querying experience for Prometheus. When you run a query, Grafana actually runs two queries: a normal Prometheus query for the graph and an Instant Query for the table. An Instant Query returns the last value for each time series, which shows a good summary of the data shown in the graph.

Metrics explorer

On the left side of the query field, choose **Metrics** to open the Metric Explorer. This shows a hierarchical menu with metrics grouped by their prefix. For example, all Alertmanager metrics are grouped under the `alertmanager` prefix. This is a good starting point for exploring which metrics are available.

Query field

The Query field supports automatic completion for metric names, function and works mostly the same way as the standard Prometheus query editor. Press **Enter** to run a query.

The Autocomplete menu can be accessed by pressing **Ctrl+Space**. The Autocomplete menu contains a new History section with a list of recently run queries.

Suggestions can appear under the Query field. Choose a suggestion to update your query with the suggested change.

- For counters (monotonically increasing metrics), a rate function is suggested.
- For buckets, a histogram function is suggested.
- For recording rules, possible to expand the rules.

Table filters

Choose the **Filter** button in the **label** column of a table panel to add filters to the query expression. You can add filters for multiple queries as well. The filter is added for all the queries.

Logs integration

You can also use Explore to investigate your logs with the following data sources:

- InfluxDB
- Elasticsearch

Logs visualization

Results of log queries are shown as histograms in the graph, and individual logs are displayed below. If the data source does not send histogram data for the requested time range, the logs model computes a time series based on the log row counts bucketed by an automatically calculated time interval. The start of the histogram is then anchored by the first log row's timestamp from the result. The end of the time series is anchored to the time picker's **To** range.

Log level

For logs where a **level** label is specified, Grafana uses the value of the label to determine the log level and update the color accordingly. If the log doesn't have a level label specified, Grafana parses the log to find out whether its content matches any of the supported expressions. The log level is always determined by the first match. If Grafana is not able to determine a log level, it will be visualized with the **unknown** log level. The following table lists log levels and the mapping of log level abbreviations and expressions.

Supported expressions	Log level	Color
emerg	critical	purple
fatal	critical	purple
alert	critical	purple
crit	critical	purple
critical	critical	purple
err	error	red
eror	error	red
error	error	red
warn	warning	yellow
warning	warning	yellow
info	info	green
information	info	green
notice	info	green
dbug	debug	blue
debug	debug	blue
trace	trace	light blue
*	unknown	grey

Visualization options

You can customize how logs are displayed and select which columns are shown.

Time

This option shows or hides the time column. This is the timestamp that is associated with the log line as reported from the data source.

Unique labels

This option shows or hides the unique labels column, which includes only non-common labels. All common labels are displayed above.

Wrap lines

To use line wrapping in the display, set this to **True**. Setting this option to **False** results in horizontal scrolling.

Deduping

Log data can be very repetitive. Explore can help by hiding duplicate log lines. You can choose from different deduplication algorithms:

- **Exact** – Exact matches are done on the whole line except for date fields.
- **Numbers** – Matches are done on the line after stripping out numbers such as durations, IP addresses, and so on.
- **Signature** – The most aggressive deduping, this strips all letters and numbers. Matches are done on the remaining whitespace and punctuation.

Flip results order

You can change the order of received logs from the default descending order (newest first) to ascending order (oldest first).

Labels and detected fields

Each log row has an extendable area with its labels and detected fields, for more robust interaction. For all labels, you can filter for (positive filter) and filter out (negative filter) selected labels. Each field or label also has a stats icon to display one-time statistics in relation to all displayed logs.

Toggle detected fields

If your logs are structured in JSON or logfmt, you can show or hide detected fields. Expand a log line, and then choose the eye icon to show or hide fields.

{{< docs-imagebox img="/img/docs/explore/parsed-fields-7-2.gif" max-width="800px" caption="Toggling detected fields in Explore" >}}

Tracing integration

You can visualize traces from tracing data sources in Explore. Data sources currently supported:

- Connect to a Jaeger data source
- Connect to a Tempo data source
- Connect to an AWS X-Ray data source
- Connect to a Zipkin data source

For information on using the query editor, see the documentation for the specific data source.

Header

The header includes the following items:

- Header title, which shows the name of the root span and trace ID
- Search, which highlights spans that contain the searched text
- Metadata about the trace

Minimap

Minimap shows a condensed view or the trace timeline. Drag your mouse over the minimap to zoom into a smaller time range. Zooming will also update the main timeline, so it's easy to see shorter spans. If you pause on the minimap, when zoomed, you can see the **Reset Selection** button, which resets the zoom.

Timeline

The timeline shows a list of spans within the trace. Each span row consists of the following components:

- **Expand children** button: Expands or collapses all the children spans of selected span
- Service name: Name of the service that logged the span
- Operation name: Name of the operation that this span represents
- Span duration bar: Visual representation of the operation duration within the trace

Choosing anywhere on the span row shows span details.

Span details

The span details include the following items:

- Operation name
- Span metadata
- Tags (any tags associated with this span)
- Process metadata (metadata about the process that logged this span)
- Logs: List of logs logged by this span and associated key values. In case of Zipkin logs section shows Zipkin annotations.

Navigating between Explore and a dashboard

To help accelerate workflows that involve regularly switching from Explore to a dashboard and vice-versa, we've added the ability to return to the origin dashboard after navigating to Explore from the panel's dropdown.

After you've navigated to Explore, you should notice a "Back" button in the Explore toolbar.

Simply choosing the button will return you to the origin dashboard, or, if you'd like to bring changes you make in Explore back to the dashboard, simply choose the arrow next to the button to reveal a "Return to panel with changes" menu item.

Query inspector

To help with debugging queries, Explore allows you to investigate query requests and responses, as well as query statistics, via the Query inspector. This functionality is similar to the panel inspector **Stats** tab and **Query** tab. For more information, see Inspect query performance and View raw request and response to data source.

Linking

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

You can use links to navigate between commonly used dashboards or to connect others to your visualizations. Links let you create shortcuts to other dashboards, panels, and even external websites.

Amazon Managed Grafana supports dashboard links, panel links, and data links. Dashboard links are displayed at the top of the dashboard. Panel links are accessible by choosing an icon on the top-left corner of the panel.

Which link should you use?

Start by examining how you're currently navigating between dashboards. If you're often jumping between a set of dashboards and struggling to find the same context in each, links can help optimize your workflow.

The next step is to figure out which link type is right for your workflow. Although all the link types in Grafana are used to create shortcuts to other dashboards or external websites, they work in different contexts.

- To add links that relate to most or all of the panels in the dashboard, use Dashboard links.
- To drill down into specific panels, use Panel links.
- To link to external sites, you can use either a dashboard link or a panel link.
- To drill down into a specific series, or even a single measurement, use Data links.

Controlling time range using the URL

You can control the time range of a panel or dashboard by providing the following query parameters in the dashboard URL:

- `from` defines lower limit of the time range, specified in ms epoch.
- `to` defines upper limit of the time range, specified in ms epoch.
- `time` and `time.window` define a time range from `time-time.window/2` to `time +time.window/2`. Both parameters should be specified in milliseconds. For example, ?

`time=1500000000000&time.window=10000` will result in a 10-second time range from 1499999995000 to 1500000005000

Dashboard links

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

When you create a dashboard link, you can include the time range and current template variables to directly jump to the same context in another dashboard. This helps to ensure that the person you send the link to is looking at the right data. For other types of links, see Data link variables.

After you add a dashboard link, it appears in the upper-right corner of your dashboard.

Adding links to dashboards

Add links to other dashboards at the top of your current dashboard.

1. While viewing the dashboard that you want to add a link to, choose the gear icon at the top of the screen to open **Dashboard settings**.
2. Choose **Links**, and then choose **Add Dashboard Link** or **New**.
3. In **Type**, select **dashboards**.
4. Select link options:

   - **With tags** – Enter tags to limit the linked dashboards to only the ones with the tags you enter. Otherwise, the Grafana workspace includes links to all other dashboards.
   - **As dropdown** – Select this option if you are linking to many dashboards, and add an optional title to the dropdown list. If this option is not selected, the Grafana workspace displays the dashboard links side by side across the top of your dashboard.
   - **Time range** – Select this option to include the dashboard time range in the link. When the user chooses the link, the linked dashboard opens with the indicated time range already set.

- **Variable values** – Select this option to include the template variables that are currently used as query parameters in the link. When the user chooses the link, any matching templates in the linked dashboard are set to the values from the link.
- **Open in new tab** – Select this option to open the dashboard link in a new tab or window.

5. Choose **Add**.

Adding a URL link to a dashboard

Add a link to a URL at the top of your current dashboard. You can link to any available URL, including dashboards, panels, or external sites. You can even control the time range to ensure that the user sees the specific data in the Grafana workspace.

1. While viewing the dashboard that you want to link to, choose the gear icon at the top of the screen to open **Dashboard settings**.
2. Choose **Links**, and then choose **Add Dashboard Link** or **New**.
3. In **Type**, select **link**.
4. Select link options:

   - **Url** – Enter the URL that you want to link to. Depending on the target, you might want to include field values.
   - **Title** – Enter the title that you want the link to display.
   - **Tooltip** – Enter the tooltip that you want the link to display when the user pauses over it.
   - **Icon** – Choose the icon that you want to display with the link.
   - **Time range** – Select this option to include the dashboard time range in the link. When the user chooses the link, the linked dashboard opens with the indicated time range already set.
     - `from` defines the lower limit of the time range, specified in ms epoch.
     - `to` defines the upper limit of the time range, specified in ms epoch.
     - `time` and `time.window` define a time range from `time-time.window/2` to `time +time.window/2`. Both parameters should be specified in milliseconds. For example, `?time=1500000000000&time.window=10000` results in a 10-second time range from 1499999995000 to 1500000005000.
   - **Variable values** – Select this option to include the template variables that are currently used as query parameters in the link. When the user chooses the link, any matching templates in the linked dashboard are set to the values from the link; for example, https://

play.grafana.org/d/000000074/alerting?var-app=backend&var-server=backend_01&var-server=backend_03&var-interval=1h

- **Open in new tab** – Select this option to open the dashboard link in a new tab or window.

5. Choose **Add**.

Updating a dashboard link

To change or update an existing dashboard link, use the following procedure.

1. In **Dashboard Settings**, on the **Links** tab, choose the existing link that you want to edit.
2. Change the settings, and then choose **Update**.

Duplicating a dashboard link

To duplicate an existing dashboard link, choose the duplicate icon next to the existing link that you want to duplicate.

Deleting a dashboard link

To delete an existing dashboard link, choose the trash can icon for the link that you want to delete.

Panel links

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Each panel can have its own set of links that are shown in the upper-left corner of the panel. You can link to any available URLs, including dashboards, panels, or external sites. You can even control the time range to ensure the user sees the specific data in the Grafana workspace.

Choose the icon in the top-left corner of a panel to see available panel links.

Adding a panel link

1. Pause on the panel that you want to add a link to, and then press **e**. Or choose the dropdown arrow next to the panel title, and then choose **Edit**.
2. On the **Panel** tab, scroll down to the **Links** section.
3. Expand **Links**, and then choose **Add link**.
4. Enter a **Title** for the link. The title will be displayed in the UI.
5. Enter the **URL** that you want to link to. You can include one of the template variables that are defined in the dashboard. Press **Ctrl+Space** or **Cmd+Space**, and then choose the **URL** field to see the available variables. When you add template variables to your panel link, the link sends the user to the right context, with the relevant variables already set. You can also use time variables

   - `from` defines the lower limit of the time range, specified in ms epoch.
   - `to` defines the upper limit of the time range, specified in ms epoch.
   - `time` and `time.window` define a time range from `time-time.window/2` to `time+time.window/2`. Both parameters should be specified in milliseconds. For example, `?time=1500000000000&time.window=10000` results in a 10-second time range from 1499999995000 to 1500000005000.
6. To open in a new tab, select **Open in a new tab**.
7. Choose **Save** to save changes and close the window.
8. Choose **Save** in the upper right to save your changes to the dashboard.

Updating a panel link

1. On the **Panel** tab, find the link that you want to make changes to.
2. Choose the **Edit** (pencil) icon to open the **Edit link** window.
3. Make any necessary changes.
4. Choose **Save** to save changes and close the window.
5. Choose **Save** in the upper right to save your changes to the dashboard.

Deleting a panel link

1. On the **Panel** tab, find the link that you want to delete.

2. Choose the **X** icon next to the link that you want to delete.
3. Choose **Save** in the upper right to save your changes to the dashboard.

Data links

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Data links provide more granular context to your links. You can create links that include the series name or even the value. For example, if your visualization shows four servers, you can add a data link to one or two of them.

The link itself is accessible in different ways, depending on the visualization. For the graph panel, you need to choose a data point or line. For a panel such as stat, gauge, or bar gauge, you can choose anywhere on the visualization to open the context menu.

You can use variables in data links to send people to a detailed dashboard with preserved data filters. For example, you can use variables to specify a time range, series, and variable selection. For more information, see Data link variables.

Typeahead suggestions

When you create or update a data link, press **Ctrl+Space** or **Cmd+Space**Cmd+Space on your keyboard to open the typeahead suggestions to more easily add variables to your URL.

Adding a data link

1. Pause on the panel that you want to add a link to, and then press **e**. Or choose the dropdown arrow next to the panel title, and then choose **Edit**.
2. On the **Field** tab, scroll down to the **Data links** section.
3. Expand **Data links**, and then choose **Add link**.

4. Enter a **Title** for the link. The title will be displayed in the UI.
5. Enter the **URL** that you want to link to.

   You can add one of the template variables that are defined in the dashboard. Choose the **URL** field, and then type **$**, or press Ctrl+Space or **Cmd+Space** to see a list of available variables. When you add template variables to your panel link, the link sends the user to the right context, with the relevant variables already set. For more information, see Data link variables.
6. To open in a new tab, select **Open in a new tab**.
7. Choose **Save** to save changes and close the window.
8. Choose **Save** in the upper right to save your changes to the dashboard.

Updating a data link

1. On the **Field** tab, find the link that you want to make changes to.
2. Choose the **Edit** (pencil) icon to open the **Edit link** window.
3. Make any necessary changes.
4. Choose **Save** to save changes and close the window.
5. Choose **Save** in the upper right to save your changes to the dashboard.

Deleting a data link

1. On the **Field** tab, find the link that you want to delete.
2. Choose the **X** icon next to the link that you want to delete.
3. Choose **Save** in the upper right to save your changes to the dashboard.

Data link variables

You can use variables in data links to see series fields, labels, and values. For more information about data links, see Data links.

To see a list of available variables, enter **$** in the data link **URL** field.

You can also use template variables in your data links URLs. For more information, see Templates and variables.

Time range panel variables

You can use the following variables to include the current time range in the data link URL:

- `__url_time_range` – The current dashboard's time range; for example, `?from=now-6h&to=now`
- `$__from and $__to` – For more information, see [Global variables]({{< relref "../variables/variable-types/global-variables.md#__from-and-__to" >}}).

Series variables

Series-specific variables are available under the `__series` namespace:

- `__series.name` – Adds the series name to the URL
- `__series.labels.<LABEL>` – Adds the label's value to the URL. If your label contains dots, use `__series.labels["<LABEL>"]` syntax.

Field variables

Field-specific variables are available under the `__field` namespace:

- `__field.name` – The name of the field

Value variables

Value-specific variables are available under the `__value` namespace:

- `__value.time` – The value's timestamp (Unix ms epoch) to the URL; for example, `?time=1560268814105`
- `__value.raw` – The raw value
- `__value.numeric` – The numeric representation of a value
- `__value.text` – The text representation of a value
- `__value.calc` – The calculation name if the value is result of calculation

Template variables

When linking to another dashboard that uses template variables, select variable values for whoever chooses the link.

Use `var-myvar=${myvar}`, where `myvar` is a name of the template variable that matches one in the current dashboard that you want to use.

To add all of the current dashboard's variables to the URL, use `__all_variables`.

Templates and variables

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

A variable is a placeholder for a value. You can use variables in metric queries and in panel titles. Variables give you the ability to create more interactive and dynamic dashboards. Instead of hardcoding things like server, application, and sensor names in your metric queries, you can use variables in their place.

Variables are displayed as dropdown lists at the top of the dashboard. When you change the value by using the dropdown list at the top of the dashboard, your panel's metric queries reflect the new value.

These can be especially useful for administrators who want to allow viewers to adjust visualizations quickly but do not want to give them full editing permissions. Grafana viewers can use variables.

By using variables and templates, you can single-source dashboards. If you have multiple identical data sources or servers, you can make one dashboard and use variables to change what you are viewing. This simplifies maintenance and upkeep.

For a list of supported variable types, and instructions for adding each type of variable, see Variable types

Templates

A *template* is any query that contains a variable.

For example, if you were administering a dashboard to monitor several servers, you could make a dashboard for each server. Or you could create one dashboard and use panels with template queries, as shown in the following example.

```
wmi_system_threads{instance=~"$server"}
```

Variable values are always synced to the URL by using the syntax `var-<varname>=value`.

Variable best practices

Variable dropdown lists are displayed in the order they are listed in the variable list in **Dashboard settings**.

Put the variables that you will change often at the top, so that they will be shown first, at the far left on the dashboard.

Variable syntax

Panel titles and metric queries can see variables by using two different syntaxes:

- `$varname` This syntax is easier to read, as in the following example: `apps.frontend.$server.requests.count`. However, you cannot use a variable in the middle of a word.
- `${var_name}` Use this syntax when you want to interpolate a variable in the middle of an expression.
- `${var_name:<format>}` This format gives you more control over how Grafana interpolates values. For more information, see Advanced variable format options.

Before queries are sent to your data source, the query is *interpolated*, meaning that the variable is replaced with its current value. During interpolation, the variable value might be *escaped* to conform to the syntax of the query language and where it is used. For example, a variable that is used in a regex expression in a Prometheus query will be regex-escaped. Read the data source–specific documentation topic for details on value escaping during interpolation.

For information about advanced syntax to override data source default formatting, see Advanced variable format options.

Variable types

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Grafana uses several types of variables.

Variable type	Description
Query	Query-generated list of values such as metric names, server names, sensor IDs, data centers, and so on. For more information, see Adding a query variable.
Custom	Define the variable options manually using a comma-separated list. For more information, see Adding a custom variable.
Text box	Display a text input field with an optional default value. For more information, see Adding a text box variable.
Constant	Define a hidden constant. For more informati on, see Adding a constant variable.
Data source	Quickly change the data source for an entire dashboard. For more information, see Adding a data source variable.

Variable type	Description
Interval	Interval variables represent time spans. For more information, see Adding an interval variable.
Ad hoc filters	Key/value filters that are automatically added to all metric queries for a data source (InfluxDB, Prometheus, and OpenSearch only). For more information, see Adding ad hoc filters.
Global variables	Built-in variables that can be used in expressio ns in the query editor. For more information, see Global variables.
Chained variables	Variable queries can contain other variables. For more information, see Chained variables.

Adding a query variable

Using query variables, you can write a data source query that returns a list of metric names, tag values, or keys. For example, a query variable might return a list of server names, sensor IDs, or data centers. The variable values change as they dynamically fetch options with a data source query.

Query expressions can contain references to other variables and, in effect, create linked variables. Grafana detects this and automatically refreshes a variable when one of its linked variables change.

Query expressions

Query expressions are different for each data source. For more information, see the documentation for your data source at Connect to data sources.

Entering general options

To enter general options for a query variable

1. Navigate to the dashboard that you want to make a variable for, and then choose the **Dashboard settings** (gear) icon at the top of the page.

2. On the **Variables** tab, choose **New**.
3. Enter a **Name** for your variable.
4. In the **Type** list, select **Query**.
5. (Optional) For **Label**, enter the display name of the variable dropdown list. If you don't enter a display name, the dropdown label will be the variable name.
6. Choose a **Hide** option:

   - **No selection (blank)** – The variable dropdown list displays the variable **Name** or **Label** value. This is the default.
   - **Label** – The variable dropdown list displays only the selected variable value and a down arrow.
   - **Variable** – No variable dropdown list is displayed on the dashboard.

Entering query options

To enter query options for a query variable

1. In the **Data source** list, select the target data source for the query. For more information about data sources, see Connect to data sources.
2. In the **Refresh** list, select when the variable should update options.

   - **Never** - Caches variable queries, and values are not updated. This is fine if the values never change, but problematic if they are dynamic and change a lot.
   - **On Dashboard Load** - Queries the data source every time the dashboard loads. This slows down dashboard loading, because the variable query must to be completed before dashboard can be initialized.
   - **On Time Range Change** - Queries the data source when the dashboard time range changes. Use this option only if your variable options query contains a time range filter or is dependent on the dashboard time range.
3. In the **Query** field, enter a query.

   - The query field varies according to your data source. Some data sources have custom query editors.
   - If you need more room in a single input field query editor, pause on the lines in the lower right corner of the field and drag downward to expand.

4. (Optional) In the **Regex** field, type a regex expression to filter or capture specific parts of the names returned by your data source query. For examples, see Filtering variables with regex.
5. In the **Sort** list, select the sort order for values to be displayed in the dropdown list. The default option, **Disabled**, means that the order of options returned by your data source query will be used.
6. (Optional) Enter **Selection Options**. For more information, see Entering variable selection options.
7. In **Preview of values**, the Grafana workspace displays a list of the current variable values. Review them to ensure they match what you expect.
8. Choose **Add** to add the variable to the dashboard.

Adding a custom variable

Use a *custom* variable for values that do not change. This might be numbers, strings, or even other variables.

For example, if you have server names or region names that do not change, you can create them as custom variables rather than query variables. Because they do not change, you might use them in chained variables rather than other query variables. That would reduce the number of queries Grafana must send when chained variables are updated. For more information about chained variables, see Chained variables.

Entering general options

To enter query options for a custom variable

1. Navigate to the dashboard you want to make a variable for and then choose the **Dashboard settings** (gear) icon at the top of the page.
2. On the **Variables** tab, choose **New**.
3. Enter a **Name** for your variable.
4. In the **Type** list, choose **Custom**.
5. (Optional) For **Label**, enter the display name of the variable dropdown list. If you don't enter a display name, the dropdown label will be the variable name.
6. Choose a **Hide** option:

   - **No selection (blank)** – The variable dropdown list displays the variable **Name** or **Label** value. This is the default.

- **Label** - The variable list dropdown displays only the selected variable value and a down arrow.
- **Variable** – No variable dropdown list is displayed on the dashboard.

Entering custom options

To enter custom options for a custom variable

1. In the **Values separated by comma** list, enter the values for this variable in a comma-separated list. You can include numbers, strings, other variables, or key-value pairs separated by a colon.
2. (Optional) Enter **Selection Options**. For more information, see Entering variable selection options.
3. In **Preview of values**, the Grafana workspace displays a list of the current variable values. Review them to ensure they match what you expect.
4. Choose **Add** to add the variable to the dashboard.

Adding a text box variable

Text box variables display a text input field with an optional default value. This is the most flexible variable, because you can enter any value. Use this type of variable if you have metrics with high cardinality or if you want to update multiple panels in a dashboard at the same time.

Entering general options

To enter general options for a text box variable

1. Navigate to the dashboard you want to make a variable for and then choose the **Dashboard settings** (gear) icon at the top of the page.
2. On the **Variables** tab, choose **New**.
3. Enter a **Name** for your variable.
4. In the **Type** list, select **Text box**.
5. (Optional) For **Label**, enter the display name of the variable dropdown list. If you don't enter a display name, then the dropdown label will be the variable name.
6. Choose a **Hide** option:

- **No selection (blank)** – The variable dropdown list displays the variable **Name** or **Label** value. This is the default.
- **Label** – The variable dropdown list displays only the selected variable value and a down arrow.
- **Variable** – No variable dropdown list is displayed on the dashboard.

Entering text options

To enter text options for a text box variable

1. (Optional) In the **Default value** field, select the default value for the variable. If you do not enter anything in this field, then Grafana displays an empty text box that you can type text into.
2. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
3. Choose **Add** to add the variable to the dashboard.

Adding a constant variable

To define a hidden constant, use *constant* variables. Constant variables are useful for metric path prefixes for dashboards you want to share. When you export a dashboard, constant variables are converted to import options.

Constant variables are not flexible. Each constant variable holds only one value. To update it, you must update the variable settings.

Constant variables are useful when you have complex values that you must include in queries but don't want to retype in every single query. For example, if you had a server path called `i-0b6a61efe2ab843gg`, you could replace it with a variable called `$path_gg`.

Entering general options

To enter general options for a constant variable

1. Navigate to the dashboard that you want to make a variable for and then choose the **Dashboard settings** (gear) icon at the top of the page.
2. On the **Variables** tab, choose **New**.

3. Enter a **Name** for your variable.
4. In the **Type** list, select **Constant**.
5. (Optional) For **Label**, enter the display name of the variable dropdown list. If you don't enter a display name, then the dropdown label will be the variable name.
6. Choose a **Hide** option:

   - **Variable** – No variable dropdown list is displayed on the dashboard. This is the default.
   - **No selection (blank)** – The variable dropdown list displays the variable **Name** or **Label** value.
   - **Label** – The variable dropdown list displays only the selected variable value and a down arrow.

Entering constant options

To enter constant options for a constant variable

1. In the **Value** field, enter the variable value. You can enter letters, numbers, and symbols. If you use advanced variable format options, you can even use wild cards. For more information, see Advanced variable format options.
2. In **Preview of values**, the Grafana workspace displays the current variable value. Review it to ensure it matches what you expect.
3. Choose **Add** to add the variable to the dashboard.

Adding a data source variable

To change the data source for an entire dashboard quickly, you can use *data source* variables. They are useful if you have multiple instances of a data source, perhaps in different environments.

Entering general options

To enter general options for a data source variable

1. Navigate to the dashboard you want to make a variable for, and then choose the **Dashboard settings** (gear) icon at the top of the page.
2. On the **Variables** tab, choose **New**.
3. Enter a **Name** for your variable.

4. In the **Type** list, select **Datasource**.
5. (Optional) For **Label**, enter the display name of the variable dropdown list. If you don't enter a display name, the dropdown label will be the variable name.
6. Choose a **Hide** option:
   - **No selection (blank)** – The variable dropdown list displays the variable **Name** or **Label** value. This is the default.
   - **Label** – The variable dropdown list displays only the selected variable value and a down arrow.
   - **Variable** – No variable dropdown list is displayed on the dashboard.

Entering data source options

To enter data source options for a data source variable

1. In the **Type** list, select the target data source for the variable. For more information about data sources, see Connect to data sources.
2. (Optional) For **Instance name filter**, enter a regex filter for which data source instances to choose from in the variable value drop-down list. Keep this field empty to display all instances.
3. (Optional) Enter **Selection Options**. For more information, see Entering variable selection options.
4. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
5. Choose **Add** to add the variable to the dashboard.

Adding an interval variable

Use an *interval* variable to represent time spans such as `1m`,`1h`, `1d`. You can think of them as a dashboard-wide group-by-time command. Interval variables change how the data is grouped in the visualization. You can also use the Auto option to return a set number of data points per time span.

You can use an interval variable as a parameter to group by time (for InfluxDB), date histogram interval (for OpenSearch), or as a summarize function parameter (for Graphite).

Entering general options

To enter general options for an interval variable

1. Navigate to the dashboard you want to make a variable for and then choose the **Dashboard settings** (gear) icon at the top of the page.
2. On the **Variables** tab, choose **New**.
3. Enter a **Name** for your variable.
4. In the **Type** list, select **Interval**.
5. (Optional) For **Label**, enter the display name of the variable dropdownlist . If you don't enter a display name, the dropdown label will be the variable name.
6. Choose a **Hide** option:

   - **No selection (blank)** – The variable dropdown list displays the variable **Name** or **Label** value. This is the default.
   - **Label** – The variable dropdown list displays only the selected variable value and a down arrow.
   - **Variable** – No variable dropdown list is displayed on the dashboard.

Entering interval options

To enter interval options for an interval variable

1. In the **Values** field, enter the time range intervals that you want to appear in the variable drop-down list. The following time units are supported: `s (seconds)`, `m (minutes)`, `h (hours)`, `d (days)`, `w (weeks)`, `M (months)`, and `y (years)`. You can also accept or edit the default values: `1m,10m,30m,1h,6h,12h,1d,7d,14d,30d`.
2. (Optional) Turn on **Auto Option** if you want to add the `auto` option to the list. Use this option to specify how many times the current time range should be divided to calculate the current `auto` time span. If you turn it on, then two more options appear:

   - **Step count** – Select the number of times that the current time range will be divided to calculate the value, similar to the **Max data points** query option. For example, if the current visible time range is 30 minutes, then the `auto` interval groups the data into 30 one-minute increments. The default value is 30 steps.

- **Min Interval** – The minimum threshold below which the step count intervals will not divide the time. To continue the 30-minute example, if the minimum interval is set to `2m`, Grafana groups the data into 15 2-minute increments.

3. In **Preview of values**, Grafana displays a list of the current variable values. Review them to ensure they match what you expect.
4. Choose **Add** to add the variable to the dashboard.

Interval variable examples

Example using the template variable `myinterval` in a Graphite function:

```
summarize($myinterval, sum, false)
```

A more complex Graphite example:

```
groupByNode(summarize(movingAverage(apps.$app.$server.counters.requests.count, 5),
 '$interval', 'sum', false), 2, 'sum')
```

Adding ad hoc filters

You can use one-time, or *ad hoc* filters to add key/value filters that are automatically added to all metric queries that use the specified data source. Unlike other variables, you do not use one-time filters in queries. Instead, you use them to write filters for existing queries.

> **Note**
>
> **Note:** One-time, or ad hoc, filter variables work only with InfluxDB, Prometheus, and OpenSearch data sources.

Entering general options

To enter general options for an ad hoc filter

1. Navigate to the dashboard that you want to make a variable for, and then choose the **Dashboard settings** (gear) icon at the top of the page.

2. On the **Variables** tab, choose **New**.
3. Enter a **Name** for your variable.
4. In the **Type** list, select **Ad hoc filters**.
5. (Optional) For **Label**, enter the display name of the variable dropdown list. If you don't enter a display name, the dropdown label will be the variable name.
6. Choose a **Hide** option:

   - **No selection (blank)** – The variable dropdown list displays the variable **Name** or **Label** value. This is the default.
   - **Label** – The variable dropdown list displays only the selected variable value and a down arrow.
   - **Variable** – No variable dropdown list is displayed on the dashboard.

Entering options

To enter options for an ad hoc filter

1. In the **Data source** list, select the target data source. For more information about data sources, see Connect to data sources.
2. Choose **Add** to add the variable to the dashboard.

Creating ad hoc filters

Ad hoc filters are one of the most complex and flexible variable options available. Instead of a regular list of variable options, this variable enables the construction of a dashboard-wide ad hoc query. Filters that you apply in this manner are applied to all panels on the dashboard.

Chained variables

Chained variables, also called *linked variables* or *nested variables*, are query variables with one or more other variables in their variable query. This section explains how chained variables work, and it provides links to example dashboards that use chained variables.

Chained variable queries are different for each data source, but the premise is the same for all. You can use chained variable queries in any data source that supports them.

You can build complex linked, templated dashboards, 5 or 10 levels deep. Technically, there is no limit to how deep or complex you can go, but the more links you have, the greater the query load.

Best practices and tips

The following practices will make your dashboards and variables easier to use.

Creating new chained variables

- Chaining variables creates parent-child dependencies. You can envision them as a ladder or a tree.
- The quickest way to create a new chained variable is to copy the variable that you want to base the new one on. In the variable list, choose the **Duplicate variable** icon to the right of the variable entry to create a copy. You can then add on to the query for the parent variable.
- New chained variables that you create this way appear at the bottom of the list. To give the list a logical order, drag the variable to a different position in the list.

Variable order

To change the order of variables in the dashboard variable list, choose the up and down arrows on the right side of each entry. The Grafana workspace lists variable dropdown lists left to right according to this list, displaying the variable at the top of the list on the far left.

- List variables that do not have dependencies at the top, before their child variables.
- Each variable should follow the one that it is dependent on.
- The UI doesn't indicate which variables have dependency relationships. List the variables in a logical order to make it clearer to end users (and yourself).

Complexity consideration

The more layers of dependency you have in variables, the longer it takes to update dashboards after you change variables.

For example, if you have a series of four linked variables (country, region, server, metric) and you change a root variable value (country), the Grafana workspace must run queries for all the dependent variables before it updates the visualizations in the dashboard.

Global variables

Grafana has global built-in variables that can be used in expressions in the query editor. This topic lists them in alphabetical order and defines them. These variables are useful in queries, dashboard links, panel links, and data links.

$__dashboard

This variable is the name of the current dashboard.

$__from and $__to

Grafana has two built in time range variables: `$__from` and `$__to`. They are currently always interpolated as epoch milliseconds by default but you can control date formatting.

Syntax	Example result	Description
`${__from}`	1594671549254	Unix millisecond epoch
`${__from:date}`	2020-07-13T20:19:09.254Z	No args, defaults to ISO 8601/RFC 3339
`${__from:date:iso}`	2020-07-13T20:19:09.254Z	ISO 8601/RFC 3339
`${__from:date:seconds}`	1594671549	Unix seconds epoch
`${__from:date:YYYY-MM}`	2020-07	Any custom data format. For more information, see Display.

The above syntax works with `${__to}` as well.

You can use this variable in URLs as well. For example, to send an end user to a dashboard that shows a time range from six hours ago until now, use the following URL: https://play.grafana.org/d/000000012/grafana-play-home?viewPanel=2&orgId=1?from=now-6h&to=now

$__interval

You can use the `$__interval` variable as a parameter to group by time (for InfluxDB, Myself, Postgres, MSSQL), Date histogram interval (for OpenSearch), or as a *summarize* function parameter (for Graphite).

The Grafana workspace automatically calculates an interval that can be used to group by time in queries. When there are more data points than can be shown on a graph, queries can be made more efficient by grouping by a larger interval. For example, it is more efficient to group by 1 day than by 10s when looking at 3 months of data. The graph will look the same, and the query will be faster. The `$__interval` is calculated by using the time range and the width of the graph (the number of pixels).

Approximate Calculation: `(from - to) / resolution`

For example, when the time range is 1 hour and the graph is full screen, the interval might be calculated to `2m`; points are grouped in 2-minute intervals. If the time range is 6 months and the graph is full screen, the interval might be `1d` (1 day); points are grouped by day.

In the InfluxDB data source, the legacy variable `$interval` is the same variable. Use `$__interval` instead.

The InfluxDB and OpenSearch data sources have `Group by time interval` fields that are used to hardcode the interval or to set the minimum limit for the `$__interval` variable by using the > syntax -> `>10m`.

$__interval_ms

This variable is the `$__interval` variable in milliseconds, not a time interval formatted string. For example, if the `$__interval` is `20m` then the `$__interval_ms` is `1200000`.

$__name

This variable is only available in the Singlestat panel and can be used in the prefix or suffix fields on the **Options** tab. The variable will be replaced with the series name or alias.

$__org

This variable is the ID of the current organization. The variable `${__org.name}` is the name of the current organization.

$__user

The variable `${__user.id}` is the ID of the current user. The variable `${__user.login}` is the login handle of the current user. The variable `${__user.email}` is the email for the current user.

$__range

This variable is currently supported only for Prometheus data sources. This variable represents the range for the current dashboard. It is calculated by `to - from`. It has millisecond and second representations called `$__range_ms` and `$__range_s`.

$timeFilter or $__timeFilter

The `$timeFilter` variable returns the currently selected time range as an expression. For example, the time range interval `Last 7 days` expression is `time > now() - 7d`.

This variable is used in several places, including:

- The WHERE clause for the InfluxDB data source. Grafana adds it automatically to InfluxDB queries when in **Query Editor** mode. You can add it manually in **Text Editor** mode: `WHERE $timeFilter`.
- Log Analytics queries in the Azure Monitor data source.
- SQL queries in MySQL, Postgres, and MSSQL.
- The `$__timeFilter` variable is used in the MySQL data source.

Other variable options

This section explains the other variable options that are available.

Entering variable selection options

You can use **Selection Options** to manage variable option selections. All selection options are optional, and they are off by default.

Multi-value

If you turn this option on, the variable dropdown list supports the selection of multiple options at the same time. For more information, see Formatting multi-value variables.

Include All option

The Grafana workspace adds an `All` option to the variable dropdown list. If an end user selects this option, all variable options are selected.

Custom all value

This option is visible only if the **Include All option** is selected.

To define the value of the `All` option, enter regex, glob, or Lucene syntax in the **Custom all value** field.

By default, the `All` value includes all options in combined expression. This can become very long and can have performance problems. Sometimes it can be better to specify a custom all value, like a wild card regex.

When you use custom regex, glob, or Lucene syntax in the **Custom all value** option, it is never escaped, so you must consider what is a valid value for your data source.

Advanced variable format options

The formatting of the variable interpolation depends on the data source, but there are some situations where you might want to change the default formatting.

For example, the default for the MySQL data source is to join multiple values as comma-separated with quotes: `'server01','server02'`. In some cases, you might want to have a comma-separated string without quotes: `server01,server02`. To do this, use the following advanced variable formatting options.

General syntax

Syntax: `${var_name:option}`

If any invalid formatting option is specified, `glob` is the default, or fallback, option.

CSV

Formats variables with multiple values as a comma-separated string.

```
servers = ['test1', 'test2']
```

```
String to interpolate: '${servers:csv}'
Interpolation result: 'test1,test2'
```

Distributed - OpenTSDB

Formats variables with multiple values in custom format for OpenTSDB.

```
servers = ['test1', 'test2']
String to interpolate: '${servers:distributed}'
Interpolation result: 'test1,servers=test2'
```

Doublequote

Formats single-value and multi-value variables into a comma-separated string, escapes " in each value by \", and quotes each value with ".

```
servers = ['test1', 'test2']
String to interpolate: '${servers:doublequote}'
Interpolation result: '"test1","test2"'
```

Glob - Graphite

Formats variables with multiple values into a glob (for Graphite queries).

```
servers = ['test1', 'test2']
String to interpolate: '${servers:glob}'
Interpolation result: '{test1,test2}'
```

JSON

Formats variables with multiple values as a comma-separated string.

```
servers = ['test1', 'test2']
String to interpolate: '${servers:json}'
Interpolation result: '["test1", "test2"]'
```

Lucene - OpenSearch

Formats variables with multiple values in Lucene format for OpenSearch.

```
servers = ['test1', 'test2']
String to interpolate: '${servers:lucene}'
Interpolation result: '("test1" OR "test2")'
```

Percentencode

Formats single-value and multi-value variables for use in URL parameters.

```
servers = ['foo()bar BAZ', 'test2']
String to interpolate: '${servers:percentencode}'
Interpolation result: 'foo%28%29bar%20BAZ%2Ctest2'
```

Pipe

Formats variables with multiple values into a pipe-separated string.

```
servers = ['test1.', 'test2']
String to interpolate: '${servers:pipe}'
Interpolation result: 'test1.|test2'
```

Raw

Turns off data source–specific formatting, such as single quotes in an SQL query.

```
servers = ['test1.', 'test2']
String to interpolate: '${var_name:raw}'
Interpolation result: '{test.1,test2}'
```

Regex

Formats variables with multiple values into a regex string.

```
servers = ['test1.', 'test2']
String to interpolate: '${servers:regex}'
Interpolation result: '(test1\.|test2)'
```

Singlequote

Formats single- and multi-value variables into a comma-separated string, escapes ' in each value by \' and quotes each value with '.

```
servers = ['test1', 'test2']
String to interpolate: '${servers:singlequote}'
Interpolation result: "'test1','test2'"
```

Sqlstring

Formats single- and multi-value variables into a comma-separated string, escapes ' in each value by '' and quotes each value with '.

```
servers = ["test'1", "test2"]
String to interpolate: '${servers:sqlstring}'
Interpolation result: "'test''1','test2'"
```

Text

Formats single-value and multi-value variables into their text representation. For a single variable, it will just return the text representation. For multi-value variables, it will return the text representation combined with +.

```
servers = ["test1", "test2"]
String to interpolate: '${servers:text}'
Interpolation result: "test1 + test2"
```

Formatting multi-value variables

Interpolating a variable with multiple values selected is tricky as it is not straight forward how to format the multiple values into a string that is valid in the given context where the variable

is used. Grafana tries to solve this by enabling each data source plugin to inform the templating interpolation engine what format to use for multiple values.

> **Note**
>
> The **Custom all value** option on the variable must be blank for Grafana to format all values into a single string. If leave it blank, then the Grafana concatenates (adds together) all the values in the query. Something like `value1,value2,value3`. If a custom `all` value is used, then instead the value will be something like `*` or `all`.

Multi-value variables with a Graphite data source

Graphite uses glob expressions. A variable with multiple values would, in this case, be interpolated as `{host1,host2,host3}` if the current variable value was *host1*, *host2*, and *host3*.

Multi-value variables with a Prometheus or InfluxDB data source

InfluxDB and Prometheus use regex expressions, so the same variable would be interpolated as `(host1|host2|host3)`. Every value would also be regex escaped. If not, a value with a regex control character would break the regex expression.

Multi-value variables with an Elastic data source

Amazon OpenSearch uses Lucene query syntax, so the same variable would be formatted as `("host1" OR "host2" OR "host3")`. In this case, every value must be escaped so that the value contains only Lucene control words and quotation marks.

Troubleshooting formatting

Automatic escaping and formatting can cause problems. It can be tricky to grasp the logic behind a problem, especially for InfluxDB and Prometheus, where the use of regex syntax requires that the variable is used in regex operator context.

If you do not want Grafana to do this automatic regex escaping and formatting, you must do one of the following:

- Turn off the **Multi-value Include All option** options.
- Use the [raw variable format]({{< relref "advanced-variable-format-options.md#raw" >}}).

Filtering variables with regex

Using the Regex Query option, you can filter the list of options returned by the variable query or modify the options returned.

This section shows how to use regex to filter and modify values in the variable dropdown list.

Using the Regex Query option, you filter the list of options returned by the Variable query or modify the options returned. For more information, see Regular expressions.

Examples of filtering on the following list of options:

```
backend_01
backend_02
backend_03
backend_04
```

Filtering so that only the options that end with `01` or `02` are returned

Regex:

```
/.*[01|02]/
```

Result:

```
backend_01
backend_02
```

Filtering and modifying the options using a regex capture group to return part of the text

Regex:

```
/.*(01|02)/
```

Result:

```
01
02
```

Filtering and modifying - Prometheus Example

List of options:

```
up{instance="demo.robustperception.io:9090",job="prometheus"} 1 1521630638000
up{instance="demo.robustperception.io:9093",job="alertmanager"} 1 1521630638000
up{instance="demo.robustperception.io:9100",job="node"} 1 1521630638000
```

Regex:

```
/.*instance="([^"]*).*/
```

Result:

```
demo.robustperception.io:9090
demo.robustperception.io:9093
demo.robustperception.io:9100
```

Filtering and modifying using named text and value capture groups

Using named capture groups, you can capture separate "text" and "value" parts from the options returned by the variable query. The variable dropdown list can contain a friendly name for each value that can be selected.

For example, when querying the node_hwmon_chip_names Prometheus metric, the chip_name is friendlier than the chip value. Start with the following variable query result.

```
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_0",chip_name="enp216s0f0np0"} 1
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_1",chip_name="enp216s0f0np1"} 1
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_2",chip_name="enp216s0f0np2"} 1
node_hwmon_chip_names{chip="0000:d7:00_0_0000:d8:00_3",chip_name="enp216s0f0np3"} 1
```

Pass it through the following Regex.

```
/chip_name="(?<text>[^"]+)|chip="(?<value>[^"]+)/g
```

The following dropdown list is produced.

```
Display Name          Value
------------          -------------------------
enp216s0f0np0         0000:d7:00_0_0000:d8:00_0
enp216s0f0np1         0000:d7:00_0_0000:d8:00_1
enp216s0f0np2         0000:d7:00_0_0000:d8:00_2
enp216s0f0np3         0000:d7:00_0_0000:d8:00_3
```

Note: Only `text` and `value` capture group names are supported.

Repeating panels or rows

You can create dynamic dashboards using *template variables*. All variables in your queries expand to the current value of the variable before the query is sent to the database. With variables, you can reuse a single dashboard for all your services.

Template variables can be very useful for dynamically changing your queries across a whole dashboard. If you want Grafana to dynamically create new panels or rows based on the values that you have selected, you can use the *Repeat* feature.

Repeating panels

If you have a variable with `Multi-value` or `Include all value` options turned on, you can choose one panel and have Grafana repeat that panel for every selected value. You can find the *Repeat* feature under the *General tab* in panel edit mode.

The `direction` controls how the panels are arranged.

If you choose `horizontal`, the panels are arranged side-by-side. Grafana automatically adjusts the width of each repeated panel so that the whole row is filled. Currently, you cannot mix other panels on a row with a repeated panel.

Set `Max per row` to tell Grafana how many panels per row you want at most. It defaults to *4*.

If you choose `vertical`, the panels are arranged from top to bottom in a column. The width of the repeated panels is the same as of the first panel (the original template) that is being repeated.

Make changes only to the first panel (the original template). For the changes take effect on all panels, you need to start a dynamic dashboard re-build. You can do this by either changing the variable value (that is, the basis for the repeat) or reloading the dashboard.

> **Note**
>
> Repeating panels require variables to have one or more items selected. You cannot repeat a panel zero times to hide it.

Repeating rows

As seen above with the panels you can also repeat rows if you have variables set with `Multi-value` or `Include all value` selection option.

To turn on this feature, you must first add a new *Row* by using the *Add Panel* menu. Then pause on the row title and choose the cog button to access the `Row Options` configuration panel. You can then select the variable you want to repeat the row for.

A best practice is to use a variable in the row title as well.

Grafana alerting

> This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Grafana alerting provides you with robust and actionable alerts that help you learn about problems in the systems moments after they occur, minimizing disruption to your services.

Amazon Managed Grafana includes access to an updated alerting system, *Grafana alerting*, that centralizes alerting information in a single, searchable view. It includes the following features:

- Create and manage Grafana alerts in a centralized view.

- Create and manage Cortex and Loki managed alerts through a single interface. For more information, see Manage your alert rules.
- View alerting information from Prometheus, Amazon Managed Service for Prometheus, and other Alertmanager compatible data sources.
- Create multiple alert instances from a single alert rule. For more information, see Single and multidimensional rules.
- Manage your alerting resources with terraform or provisioning APIs. For more information, see Provisioning Grafana Alerting resources.

For existing Amazon Managed Grafana workspace, the default is the Classic dashboard alerts. To migrate to Grafana alerting, you must migrate to Grafana alerting.

To learn more about Grafana alerting, see What's new in Grafana alerting.

Grafana alerting has four key components:

- Alerting rule - Evaluation criteria that determines whether an alert is initiated. It consists of one or more queries and expressions, a condition, the frequency of evaluation, and optionally, the duration over which the condition is met.
- Contact point - Channel for sending notifications when the conditions of an alerting rule are met.
- Notification policy - Set of matching and grouping criteria used to determine the frequency of notifications.
- Silences - Date and matching criteria used to silence notifications.

When Grafana alerting is enabled, you can:

- Create Grafana managed alerting rules
- Create Cortex or Loki managed alerting rules
- View existing alerting rules and manage their current state
- View the state and health of alerting rules
- Add or edit an alert contact point
- Add or edit notification policies
- Add or edit silences

Limitations

- The Grafana alerting system can retrieve rules from all available Amazon Managed Service for Prometheus, Prometheus, Loki, and Alertmanager data sources. It might not be able to fetch rules from other supported data sources.
- Alert rules defined in Grafana, rather than in Prometheus, send multiple notifications to your contact point. If you are using native Grafana alerts, we recommend that you stay on classic dashboard alerting and not enable the new Grafana alerting feature. If you would like to view Alerts defined in your Prometheus data source, then we recommend you enable Grafana Alerting, which sends only a single notification for alerts created in Prometheus Alertmanager.

> **Note**
>
> This limitation is no longer a limitation in Amazon Managed Grafana workspaces that support Grafana v10.4 and later.

Topics

- What's new in Grafana alerting
- Migrating classic dashboard alerts to Grafana alerting
- Alerting fundamentals
- Create and manage Grafana alerting rules
- Alert groups
- Silencing alert notifications for Prometheus data sources
- Working with contact points
- Using messaging templates
- Working with notification policies

What's new in Grafana alerting

> This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.

For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Grafana alerting has several enhancements over classic dashboard alerting.

Create multidimensional alerting

You can now create a single alerting rule that gives you system-wide visibility, generating multiple alert instances from a single alert rule. For example, you can create a rule to monitor the disk usage of multiple mount points on a single host. The evaluation engine returns multiple time series from a single query, with each time series identified by its label set.

Note

Each alert instance counts toward the alert quota. Multidimensional rules that create more instances than can be accommodated within the alert quota are not evaluated and return a quota error. For more information, see Quota reached errors.

Create alerts outside of dashboards

Unlike classic dashboard alerts, with Grafana alerting you can create queries and expressions that combine data from multiple sources in unique ways. You can still link dashboards and panels to alerting rules using their ID and quickly troubleshoot the system under observation.

Since unified alerts are no longer directly tied to panel queries, they do not include images or query values in the notification email. You can use customized notification templates to view query values.

Create Loki and Cortex alerting rules

In Grafana alerting, you can manage Loki and Cortex alerting rules using the same UI and API as your Grafana managed alerts.

View and search alerts from Amazon Managed Service for Prometheus and other Prometheus compatible data sources

Alerts for Amazon Managed Service for Prometheus and Prometheus compatible data sources are now listed in the Alerting interface. You can search for labels across multiple data sources to quickly find relevant alerts.

Special alerts for alert state NoData and Error

Grafana alerting generates special alerts that have the following labels, when evaluation of an alerting rule produces state `NoData` or `Error`:

- `alertname` with value `DatasourceNoData` or `DatasourceError` depending on the state.
- `rulename` with the name of the alert rule the special alert belongs to.
- `datasource_uid` has the UID of the data source that caused the state.
- All labels and annotations of the original rule.

You can handle these alerts the same as regular alerts, for example, by adding a silence, or routing to a contact point.

> **Note**
>
> If the rule uses multiple data sources and one or more returns no data, the special alert is created for each data source that caused the alert state.

Migrating classic dashboard alerts to Grafana alerting

> This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Existing workspaces, or workspaces that choose not to use Grafana alerting, use the classic dashboard alerting. To migrate to the new Grafana alerting, you must opt in to the feature.

You can configure your Amazon Managed Grafana instance to use Grafana alerting using the AWS Management Console, the AWS CLI, or the Amazon Managed Grafana API. For details about how to configure Amazon Managed Grafana, including turning Grafana alerting on or off, see Configure a Amazon Managed Grafana workspace.

Note

When using Grafana alerting, alert rules defined in Grafana, rather than in Prometheus, send multiple notifications to your contact point. If you are using native Grafana alerts, we recommend that you stay on classic dashboard alerting and not enable the new Grafana alerting feature. If you would like to view Alerts defined in your Prometheus data source, then we recommend you enable Grafana Alerting, which sends only a single notification for alerts created in Prometheus Alertmanager.
This limitation is removed in Amazon Managed Grafana workspaces that support Grafana v10.4 and later.

Migrating to Grafana alerting system

When Grafana alerting is turned on, existing classic dashboard alerts migrate in a format compatible with the Grafana alerting. In the Alerting page of your Grafana instance, you can view the migrated alerts alongside new alerts. With Grafana alerting, your Grafana-managed alert rules send multiple notifications rather than a single alert when they are matched.

Read and write access to classic dashboard alerts and Grafana alerts are governed by the permissions of the folders storing them. During migration, classic dashboard alert permissions are matched to the new rules permissions as follows:

- If the original alert's dashboard has permissions, migration creates a folder named with this format `Migrated {"dashboardUid": "UID", "panelId": 1, "alertId": 1}` to match permissions of the original dashboard (including the inherited permissions from the folder).
- If there are no dashboard permissions and the dashboard is under a folder, then the rule is linked to this folder and inherits its permissions.
- If there are no dashboard permissions and the dashboard is under the General folder, then the rule is linked to the General Alerting folder, and the rule inherits the default permissions.

Note

Since there is no `Keep Last State` option for `NoData` in Grafana alerting, this option becomes `NoData` during the classic rules migration. Option `Keep Last State` for `Error` handling is migrated to a new option `Error`. To match the behavior of the `Keep Last State`, in both cases, during the migration Amazon Managed Grafana automatically creates a silence for each alert rule with a duration of one year.

Notification channels are migrated to an Alertmanager configuration with the appropriate routes and receivers. Default notification channels are added as contact points to the default route. Notification channels not associated with any Dashboard alert go to the `autogen-unlinked-channel-recv` route.

Limitations

- Grafana alerting system can retrieve rules from all available Prometheus, Loki, and Alertmanager data sources. It might not be able to fetch alerting rules from other supported data sources.
- Migrating back and forth between Grafana alerts and the classic dashboard alerting can result in data loss for features supported in one system, but not the other.

Note

If you migrate back to the classic dashboard alerting, you lose all changes made to alerting configuration made while you had Grafana alerting enabled, including any new alert rules that were created.

Alerting fundamentals

This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.

For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

This section provides information about the fundamental concepts of Grafana alerting.

Alerting concepts

The following table describes the key concepts in Grafana alerting.

Key concept or feature	Definition
Data sources for Alerting	Select data sources from which you want to query and visualize metrics, logs and traces.
Scheduler	Evaluates your alert rules; the component that periodically runs queries against data sources. It is only applicable to Grafana-managed rules.
Alertmanager	Manages the routing and grouping of alert instances.
Alert rule	A set of evaluation criteria for when an alert rule should fire. An alert rule consists of one or more queries and expressions, a condition, the frequency of evaluation, and the duration over which the condition is met. An alert rule can produce multiple alert instances.
Alert instance	An alert instance is an instance of an alert rule. A single-dimensional alert rule has one alert instance. A multidimensional alert rule has one or more alert instances. A single alert rule that matches to multiple results, such as CPU against 10 VMs, is counted as multiple (in this case 10) alert instances. This number can vary over time. For example, an alert rule that monitors CPU usage for all VMs in a system has more alert instances as VMs are added. For

Key concept or feature	Definition
	more information about alert-instance quotas, see Quota reached errors.
Alert group	The Alertmanager groups alert instances by default using the labels for the root notificat ion policy. This controls de-duplication and groups of alert instances which are sent to contact points.
Contact point	Define how your contacts are notified when an alert rule fires.
Message templating	Create reusable custom templates and use them in contact points.
Notification policy	Set of rules for where, when, and how the alerts are grouped and routed to contact points.
Labels and label matchers	Labels uniquely identify alert rules. They link alert rules to notification policies and silences, determining which policy should handle them and which alert rules should be silenced.
Silences	Stop notifications from one or more alert instances. The difference between a silence and a mute timing is that a silence lasts for a specified window of time where a mute timing happens on a recurring schedule. Uses label matchers to silence alert instances.
Mute timings	Specify a time interval when you don't want new notifications to be generated or sent. You can freeze alert notifications for recurring periods of time, such as during a maintenance period. Must be linked to an existing notificat ion policy.

Alert data sources

Grafana managed alerts query the following backend data sources that have alerting enabled.

- Data sources built-in, or developed and maintained by Grafana: `Alertmanager`, `Graphite`, `Prometheus` (including Amazon Managed Service for Prometheus), `Loki`, `InfluxDB`, `Amazon OpenSearch Service`, `Google Cloud Monitoring`, `Amazon CloudWatch`, `Azure Monitor`, `MySQL`, `PostgreSQL`, `MSSQL`, `OpenTSDB`, `Oracle`, and `Azure Monitor`.

Alerting on numeric data

Numeric data that is not in a time series format can be directly alerted on, or passed into Server Side Expressions. This allows for more processing and resulting efficiency within the data source, and it can also simplify alert rules. When alerting on numeric data instead of time series data, there is no need to reduce each labeled time series into a single number. Instead labeled numbers are returned to Grafana instead.

Tabular data

This feature is supported with backend data sources that query tabular data, including SQL data sources, such as MySQL, Postgres, MSSQL, and Oracle.

A query with Grafana managed alerts or Server Side Expressions is considered numeric with these data sources:

- If the `Format AS` option is set to `Table` in the data source query.
- If the table response returned to Grafana from the query includes only one numeric (for example, int, double, or float) column, and optionally additional string columns.

If there are string columns, then those columns become labels. The name of column becomes the label name, and the value for each row becomes the value of the corresponding label. If multiple rows are returned, then each row should be uniquely identified by their labels.

Example

If you have a MySQL table called Diskspace, as the following.

Time	Host	Disk	PercentFree
2021-June-7	web1	/etc	3
2021-June-7	web2	/var	4
2021-June-7	web3	/var	8
…	…	…	…

You can query the data filtering on time, but without returning the time series to Grafana. For example, an alert that would be initiate per Host, Disk when there is less than 5% free space could look like the following.

```
SELECT Host, Disk, CASE WHEN PercentFree < 5.0 THEN PercentFree ELSE 0 END FROM (
  SELECT
      Host,
      Disk,
      Avg(PercentFree)
  FROM DiskSpace
  Group By
    Host,
    Disk
  Where __timeFilter(Time)
```

This query returns the following table response to Grafana.

Host	Disk	PercentFree
web1	/etc	3
web2	/var	4
web3	/var	0

When this query is used as the **condition** in an alert rule, then the cases where the value is non-zero alert. As a result, three alert instances are produced, as the following table.

Labels	Status
{Host=web1,disk=/etc}	Alerting
{Host=web2,disk=/var}	Alerting
{Host=web3,disk=/var}	Normal

Alertmanager

Grafana includes built-in support for Prometheus Alertmanager. The Alertmanager helps both group and manage alert rules, adding a layer of orchestration on top of the alerting engines. By default, notifications for Grafana managed alerts are handled by the embedded Alertmanager that is part of core Grafana. You can configure the Alertmanager's contact points, notification policies, and templates from the Grafana alerting UI by selecting the Grafana option from the Alertmanager dropdown.

Grafana alerting has support for external Alertmanager configuration (for more information on Alertmanager as an external datasource, see Connect to an Alertmanager data source). When you add an external Alertmanager, the Alertmanager dropdown shows a list of available external Alertmanager data sources. Select a data source to create and manage alerting for standalone Cortex or Loki data sources.

State and health of alerting rules

The state and health of alerting rules help you understand several key status indicators about your alerts. There are three key components: alert state, alerting rule state, and alerting rule health. Although related, each component conveys slightly different information.

Alerting rule state

- **Normal** – None of the time series returned by the evaluation engine is in a `Pending` or `Firing` state.
- **Pending** – At least one of the time series returned by the evaluation engine is `Pending`.
- **Firing** – At least one of the time series returned by the evaluation engine is `Firing`.

Alert state

- **Normal** – Condition for the alerting rule is **false** for every time series returned by the evaluation engine.
- **Alerting** – Condition of the alerting rule is **true** for at least one time series returned by the evaluation engine. The duration for which the condition must be true before an alert is initiated, if set, is met or has exceeded.
- **Pending** – Condition of the alerting rule is **true** for at least one time series returned by the evaluation engine. The duration for which the condition must be true before an alert is initiated, if set, has not been met.
- **NoData** – The alerting rule has not returned a time series, all values for the time series are null, or all values for the time series are zero.
- **Error** – Error when attempting to evaluate an alerting rule.

Alerting rule health

- **Ok** – No error when evaluating an alerting rule.
- **Error** – Error when evaluating an alerting rule.
- **NoData** – The absence of data in at least one time series returned during a rule evaluation.

Create and manage Grafana alerting rules

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

An alerting rule is a set of evaluation criteria that determines whether an alert is initiated. The rule consists of one or more queries and expressions, a condition, the frequency of evaluation, and optionally the duration over which the condition is met.

While queries and expressions select the dataset to evaluate, a condition sets the threshold that an alert must meet, or exceed to create an alert. An interval specifies how frequently an alerting rule

is evaluated. Duration, when configured, indicates how long a condition must be met. The rules can also define alerting behavior in the absence of data.

The following sections describe creating and managing different kinds of Grafana alert rules.

Topics

- Creating Cortex or Loki managed alert rules
- Creating Cortex or Loki managed recording rules
- Creating Grafana managed alert rules
- Annotations and labels for alerting rules
- Managing alerting rules
- Cortex or Loki rule groups and namespaces

Creating Cortex or Loki managed alert rules

Using Grafana, you can create alerting rules for an external Cortex or Loki instance.

> **Note**
>
> Cortex is the time series database used by Amazon Managed Service for Prometheus and Prometheus data sources.

Prerequisites

- Verify that you have write permissions to the Prometheus data source. Otherwise, you are not able to create or update Cortex managed alerting rules.
- For Cortex and Loki data sources, enable the ruler API by configuring their respective services.
  - **Loki** – The `local` rule storage type, default for the Loki data source, supports only viewing of rules. To edit rules, configure one of the other storage types.
  - **Cortex** – Use the legacy `/api/prom` prefix, not `/prometheus`. The Prometheus data source supports both Cortex and Prometheus, and Grafana expects that both the Query API and Ruler API are under the same URL. You cannot provide a separate URL for the Ruler API.

> **Note**
>
> If you do not want to manage alerting rules for a particular Loki or Prometheus data source, go to its settings and clear the **Manage alerts via Alerting UI** checkbox.

To add a Cortex or Loki managed alerting rule

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page listing existing alerts.
2. Choose **New alert rule**.
3. In **Step 1**, add the rule name, type and storage location, as follows:
   - In **Rule name**, add a descriptive name. This name is displayed in the alert rules list. It is also the `alertname` label for every alert instance that is created from this rule.
   - From the **Rule type** dropdown, select **Cortex/Loki managed alert**.
   - From the **Select data source** dropdown, select a Prometheus, or Loki data source.
   - From the **Namespace** dropdown, select an existing rule namespace. Otherwise, choose **Add new** and enter a name to create one. Namespaces can contain one or more rule groups and only have an organizational purpose. For more information, see Cortex or Loki rule groups and namespaces.
   - From the **Group** dropdown, select an existing group within the selected namespace. Otherwise, choose **Add new** and enter a name to create one. Newly created rules are appended to the end of the group. Rules within a group run sequentially at a regular interval, with the same evaluation time.
4. In **Step 2**, add the query to evaluate.

   The value can be a PromQL or LogQL expression. The rule initiates an alert if the evaluation result has at least one series with a value that is greater than 0. An alert is created for each series.
5. In **Step 3**, add conditions.

   In the For text box of the condition, specify the duration for which the condition must be true before the alert is initiated. If you specify `5m`, the conditions must be true for five minutes before the alert is initiated.

> **Note**
>
> After a condition is met, the alert goes into `Pending` state. If the condition remains active for the duration specified, the alert transitions to the `Firing` state. If it is no longer met, it reverts to the `Normal` state.

6. In **Step 4**, add additional metadata associated with the rule.
   - Add a description and summary to customize alert messages. Use the guidelines in Annotations and labels for alerting rules.
   - Add Runbook URL, panel, dashboard, and alert IDs.
   - Add custom labels.
7. Choose **Preview alerts** to evaluate the rule and see what alerts it would produce. It displays a list of alerts with state and value of each one.
8. Choose **Save** to save the rule or **Save and exit** to save the rule and go back to the **Alerting** page.

Creating Cortex or Loki managed recording rules

You can create and manage recording rules for an external Cortex or Loki instance. Recording rules calculate frequently needed expressions or computationally expensive expressions in advance and save the result as a new set of time series. Querying this new time series is faster, especially for dashboards since they query the same expression every time the dashboards refresh.

Prerequisites

For Cortex and Loki data sources, enable the ruler API by configuring their respective services.

- **Loki** – The `local` rule storage type, default for the Loki data source, supports only viewing of rules. To edit rules, configure one of the other storage types.
- **Cortex** – When configuring a Grafana Prometheus data source to point to Cortex, use the legacy `/api/prom` prefix, not `/prometheus`. The Prometheus data source supports both Cortex and Prometheus, and Grafana expects that both the Query API and Ruler API are under the same URL. You cannot provide a separate URL for the Ruler API.

> **Note**
>
> If you do not want to manage alerting rules for a particular Loki or Prometheus data source, go to its settings and clear the **Manage alerts via Alerting UI** check box.

To add a Cortex or Loki managed recording rule

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page listing existing alerts.
2. Choose **New alert rule**.
3. In **Step 1**, add the rule name, type and storage location, as follows.
   - In **Rule name**, add a descriptive name. This name is displayed in the alert rules list. It is also the `alertname` label for every alert instance that is created from this rule.
   - From the **Rule type** dropdown, select **Cortex/Loki managed alert**.
   - From the **Select data source** dropdown, select a Prometheus, or Loki data source.
   - From the **Namespace** dropdown, select an existing rule namespace. Otherwise, choose **Add new** and enter a name to create one. Namespaces can contain one or more rule groups and only have an organizational purpose. For more information, see Cortex or Loki rule groups and namespaces.
   - From the **Group** dropdown, select an existing group within the selected namespace. Otherwise, choose **Add new** and enter a name to create one. Newly created rules are appended to the end of the group. Rules within a group run sequentially at a regular interval, with the same evaluation time.
4. In **Step 2**, add the query to evaluate.

   The value can be a PromQL or LogQL expression. The rule initiates an alert if the evaluation result has at least one series with a value that is greater than 0. An alert is created for each series.
5. In **Step 3**, add additional metadata associated with the rule.
   - Add a description and summary to customize alert messages. Use the guidelines in Annotations and labels for alerting rules.
   - Add Runbook URL, panel, dashboard, and alert IDs.
   - Add custom labels.

6. Choose **Save** to save the rule or **Save and exit** to save the rule and go back to the **Alerting** page.

Creating Grafana managed alert rules

Grafana allows you to create alerting rules that query one or more data sources, reduce or transform the results and compare them to each other or to fix thresholds. When these are processed, Grafana sends notifications to the contact point.

> **Note**
>
> Creating Grafana managed alert rules while using Grafana alerting causes multiple notifications to be sent when the rule is matched. Some contact point providers might have configurable options to de-duplicate the notifications.

To add a Grafana managed rule

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page listing existing alerts.
2. Choose **New alert rule**.
3. In **Step 1**, add the rule name, type and storage location, as follows:
   - In **Rule name**, add a descriptive name. This name is displayed in the alert rules list. It is also the `alertname` label for every alert instance that is created from this rule.
   - From the **Rule type** dropdown, select **Grafana managed alert**.
   - From the **Folder** dropdown, select the folder where you want to store the rule. If you do not select a folder, the rule is stored in the `General` folder. To create a folder, select the dropdown and enter a new folder name.
4. In **Step 2**, add the queries and expressions to evaluate.
   - Keep the default name or hover over and choose the edit icon to change the name.
   - For queries, select a data source from the dropdown.
   - Add one or more queries or expressions (for details on expressions, see Expressions in the *Grafana documentation*).

- For each expression, select either **Classic condition** to create a single alert rule, or choose from **Math**, **Reduce**, **Resample** options to generate separate alerts for each series. For details on these options, see Single and multidimensional rules.
- Choose **Run queries** to verify that the query is successful.

5. In **Step 3**, add conditions.

- From the **Condition** dropdown, select the query or expression to initiate the alert rule.
- For **Evaluate every**, specify the frequency of evaluation. Must be a multiple of 10 seconds. For example, `1m`, `30s`.
- For **Evaluate for**, specify the duration for which the condition must be true before an alert is initiated.

> **Note**
>
> After a condition is breached, the alert goes into `Pending` state. If the condition remains breached for the duration specified, the alert transitions to the `Firing` state. If it is no longer met, it reverts to the `Normal` state.

- In **Configure no data and error handling**, configure alerting behavior in the absence of data. use the guidelines in Handling no data or error cases.
- Choose **Preview alerts** to check the result of running the query at this moment. Preview excludes no data and error handling conditions.

6. In **Step 4**, add additional metadata associated with the rule.

- Add a description and summary to customize alert messages. Use the guidelines in Annotations and labels for alerting rules.
- Add Runbook URL, panel, dashboard, and alert IDs.
- Add custom labels.

7. Choose **Save** to save the rule or **Save and exit** to save the rule and go back to the **Alerting** page.

Single and multidimensional rules

For Grafana managed alert rules, you can create a rule with a classic condition or you can create a multidimensional rule.

Single dimensional rule (classic condition)

Use a classic condition expression to create a rule that initiates a single alert when its condition is met. For a query that returns multiple series, Grafana does not track the alert state of each series. As a result, Grafana sends only a single alert even when alert conditions are met for multiple series.

For more information about how to format expressions, see Expressions in the *Grafana documentation*.

Multidimensional rule

To generate a separate alert instance for each series returned in the query, create a multidimensional rule.

> **Note**
>
> Each alert instance generated by a multi-dimensional rule counts toward your total quota of alerts. Rules are not evaluated when you reach your quota of alerts. For more information about quotas for multi-dimensional rules, see Quota reached errors.

To create multiple instances from a single rule, use `Math`, `Reduce`, or `Resample` expressions to create a multidimensional rule. For example, you can:

- Add a `Reduce` expression for each query to aggregate values in the selected time range into a single value. (Not needed for rules using numeric data).
- Add a `Math` expression with the condition for the rule. This is not needed in case a query or a reduce expression already returns 0 if rule should not initiate an alert, or a positive number if it should initiate an alert.

  Some examples:

  - `$B > 70` if it should initiate an alert in case value of B query/expression is more than 70.
  - `$B < $C * 100` in case it should initiate an alert if value of B is less than value of C multiplied by 100. If queries being compared have multiple series in their results, series from different queries are matched if they have the same labels, or one is a subset of the other.

> **Note**
>
> Grafana does not support alert queries with template variables. More information is available at the community page Template variables are not supported in alert queries while setting up Alert.

Performance considerations for multidimensional rules

Each alert instance counts toward the alert quota. Multidimensional rules that create more instances than can be accommodated within the alert quota are not evaluated and return a quota error. For more information, see Quota reached errors.

Multidimensional alerts can have a high impact on the performance of your Grafana workspace, as well as on the performance of your data sources as Grafana queries them to evaluate your alert rules. The following considerations can be helpful as you are trying to optimize the performance of your monitoring system.

- **Frequency of rule evaluation** – The **Evaluate Every** property of an alert rule controls the frequency of rule evaluation. We recommend using the lowest acceptable evaluation frequency.
- **Result set cardinality** – The number of alert instances you create with a rule affects its performance. Suppose you are monitoring API response errors for every API path, on every VM in your fleet. This set has a cardinality of the number of paths multiplied by the number of VMs. You can reduce the cardinality of the result set, for example, by monitoring total errors per VM instead of per path per VM.
- **Complexity of the query** – Queries that data sources can process and respond to quickly consume fewer resources. Although this consideration is less important than the other considerations listed above, if you have reduced those as much as possible, looking at individual query performance could make a difference. You should also be aware of the performance impact that evaluating these rules has on your data sources. Alerting queries are often the vast majority of queries handled by monitoring databases, so the same load factors that affect the Grafana instance affect them as well.

Quota reached errors

There is a quota for the number of alert instances you can have within a single workspace. When you reach that number, you can no longer create new alert rules in that workspace. With multidimensional alerts, the number of alert instances can vary over time.

The following are important to remember when working with alert instances.

- If you create only single-dimensional rules, each rule is a single alert instance. You can create the same number of rules in a single workspace as your alert-instance quota, and no more.
- Multidimensional rules create multiple alert instances, however, the number is not known until they are evaluated. For example, if you create an alert rule that tracks the CPU usage of your Amazon EC2 instances, there might be 50 EC2 instances when you create it (and therefore 50 alert instances), but if you add 10 more EC2 instances a week later, the next evaluation has 60 alert instances.

  The number of alert instances is evaluated when you create a multidimensional alert, and you can't create one that immediately puts you over your alert instance quota. Because the number of alert instances can change, your quota is checked each time that your rules are evaluated.
- At rule evaluation time, if a rule causes you to go beyond your quota for alert instances, that rule is not evaluated until an update is made to the alert rule that brings the total count of alert instances below the service quota. When this happens, you receive an alert notification letting you know that your quota has been reached (the notification uses the notification policy for the rule being evaluated). The notification includes an `Error` annotation with the value `QuotaReachedError`.
- A rule that causes a `QuotaReachedError` stops being evaluated. Evaluation is only resumed when an update is made and the evaluation after the update does not itself cause a `QuotaReachedError`. A rule that is not being evaluated shows the **Quota reached** error in the Grafana console.
- You can lower the number of alert instances by removing alert rules, or by editing multidimensional alerts to have fewer alert instances (for example, by having one alert on errors per VM, rather than one alert on error per API in a VM).
- To resume evaluations, update the alert and save it. You can update it to lower the number of alert instances, or if you have made other changes to lower the number of alert instances, you can save it with no changes. If it can be resumed, it is. If it causes another `QuotaReachedError`, you are not able to save it.

- When an alert is saved and resumes evaluation without going over the alerts quota, the **Quota reached** error can continue to show in the Grafana console for some time (up to its evaluation interval), however, the alert rule evaluation does start and alerts are sent if the rule threshold is met.
- For details on the alerts quota, as well as other quotas, see Amazon Managed Grafana service quotas.

Handling no data or error cases

Choose options for how to handle alerting behavior in the absence of data or when there are errors.

The options for handling no data are listed in the following table.

No Data option	Behavior
No Data	Create an alert `DatasourceNoData` with the name and UID of the alert rule, and UID of the data source that returned no data as labels.
Alerting	Set alert rule state to `Alerting`.
OK	Set alert rule state to `Normal`.

The options for handling error cases are listed in the following table.

Error or timeout option	Behavior
Alerting	Set alert rule state to `Alerting`
OK	Set alert rule state to `Normal`
Error	Create an alert `DatasourceError` with the name and UID of the alert rule, and UID of the data source that returned no data as labels.

Annotations and labels for alerting rules

Annotations and labels are key-value pairs associated with alerts originating from the alerting rule, datasource response, and as a result of alerting rule evaluation. They can be used in alert notifications directly or in templates and template functions to create notification contact dynamically.

Annotations

Annotations are key-value pairs that provide additional information about an alert. You can use the following annotations: `description`, `summary`, `runbook_url`, `alertId`, `dashboardUid`, and `panelId`. These are displayed in rule and alert details in the UI and can be used in contact point message templates.

Labels

Labels are key-value pairs that contain information about an alert. The label set for an alert is generated and added to throughout the alerting evaluation and notification process. They are used in the following ways.

- The complete set of labels for an alert uniquely identifies that alert within Grafana Alerts.
- The Alertmanager uses labels to match alerts for silences and alert groups in notification policies.
- The alerting UI displays labels for every alert instance generated by the evaluation of that rule.
- Contact points can access labels to dynamically generate notifications that contain information specific to the alert that is resulting in a notification.
- Labels can be added to an alerting rule. These manually configured labels are able to use template functions and reference other labels. Labels added to an alerting rule here take precedence in the event of a collision between labels.

The following template variables are available when expanding annotations and labels.

Name	Description
`$labels`	The labels from the query or condition. For example, `{{ $labels.instance }}` and `{{ $labels.job }}` . This is unavailable when the rule uses a classic condition.

Name	Description
`$values`	The values of all reduce and math expressions that were evaluated for this alert rule. For example, `{{ $values.A }}`, `{{ $values.A.Labels }}` and `{{ $values.A.Value }}` where `A` is the `refID` of the expression. This is unavailable when the rule uses a classic condition
`$value`	The value string of the alert instance. For example, `[ var='A' labels={instance=foo} value=10 ]` .

Managing alerting rules

The **Alerting** page lists alerting rules. By default, rules are grouped by types of data sources. The **Grafana** section lists rules managed by Grafana, and the **Cortex/Loki** section lists rules for Prometheus compatible data sources. You can view alerting rules for Prometheus compatible data sources but you cannot edit them.

View alerting rules

Using Grafana alerts, you can view all of your alerts in one page.

To view alerting details

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page. By default, rules are displayed in groups by data source type. You can also view by the current state of each alert (these are described in more detail in the following text).
2. In **View as**, you can toggle between the group and state views by choosing the option you prefer.
3. Choose the arrow next to a row to view more details for that row. The details for a rule include the rule labels, annotations, data sources, and queries, as well as a list of alert instances resulting from the rule.

Group view

Group view shows Grafana alert rules grouped by folder and Loki or Prometheus alert rules grouped by `namespace + group`. This is the default rule list view, intended for managing rules. You can expand each group to view a list of rules in this group. Expand a rule further to view its details. You can also expand action buttons and alerts resulting from the rule to view their details.

State view

State view shows alert rules grouped by state. Use this view to get an overview of which rules are in what state. Each rule can be expanded to view its details. Action buttons and any alerts generated by this rule, and each alert can be further expanded to view its details.

Filter alerting rules

You can filter the alerting rules that appear on the **Alerting** page in several ways.

- You can filter to display the rules that query a specific data source by choosing **Select data sources**, then selecting a data source to filter to.
- You can filter by labels by choosing search criteria in **Search by label**. Some sample criteria include `environment=production`, `region=~US|EU`, `severity!=warning`.
- You can filter to display the rules in a specific state by choosing **Filter alerts by state**, and then selecting the state you want to view.

Edit or delete alerting rules

Grafana managed alerting rules can only be edited or deleted by users with Edit permissions for the folder storing the rules. Alerting rules for an external Cortex or Loki instance can be edited or deleted by users with Editor or Admin roles.

To edit or delete a rule

1. Expand a rule until you can see the rule controls for **View**, **Edit**, and **Delete**.
2. Choose **Edit** to open the create rule page. Make updates in the same way that you create a rule. For details, see the instructions in Creating Grafana managed alert rules or Creating Cortex or Loki managed alert rules.
3. Optionally, choose **Delete** to delete a rule.

Cortex or Loki rule groups and namespaces

You can organize your rules. Rules are created within rule groups, and rule groups are organized into namespaces. The rules within a rule group are run sequentially at a regular interval. The default interval is one minute. You can rename Cortex or Loki namespaces and rule groups, and edit rule group evaluation intervals.

To edit a rule group or namespace

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page.
2. Navigate to a rule within the rule group or namespace you want to edit.
3. Choose the **Edit** (pen) icon.
4. Make changes to the rule group or namespace.

   > **Note**
   >
   > For namespaces, you can only edit the name. For rule groups, you change the name, or the evaluation interval for rules in the group. For example, you can choose `1m` to have the rules be evaluated once per minute, or `30s` to evaluate once every 30 seconds.

5. Choose **Save changes**.

Alert groups

> This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Alert groups show grouped alerts from an Alertmanager instance. By default, the alerts are grouped by the label keys for the root policy in Working with notification policies. Grouping common alerts into a single alert group prevents duplicate alerts from being initiated.

To view alert groupings

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon, then the **Alert grouping** item to open the page listing existing groups.
2. From the **Alertmanager** drop-down, select an external Alertmanager as your data source. By default, the `Grafana` Alertmanager is selected.
3. From the **custom group by** drop-down, select a combination of labels to view a grouping other than the default. You can use this view to debug or verify your grouping of notification policies.

Alerts without labels specified in the grouping of the root policy or the custom grouping, are added to a group with a header of `No grouping`.

Filter alerts

You can use the following filters to view alerts that match specific criteria:

- **Search by label** – In **Search**, enter an existing label to view alerts matching the label. For example, `environment=production`, `region=~US|EU`, `severity!=warning`.
- **Filter alerts by state** – In **States**, select from `Active`, `Suppressed`, or `Unprocessed` states to view alerts in that state.

Silencing alert notifications for Prometheus data sources

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

For external Alertmanager data sources (including Amazon Managed Service for Prometheus), you can suppress alert notifications with a *silence*. A silence only stops notifications from being created: Silences do not prevent alert rules from being evaluated, and they do not stop alerting instances

from being shown in the user interface. When you silence an alert, you specify a window of time for it to be suppressed.

You can configure silences for an external Alertmanager data source.

> **Note**
>
> To suppress alert notifications at regular time intervals (for example, during regular maintenance periods), use Mute timings rather than silences.

To add a silence

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page.
2. Choose **Silences** to open a page listing existing contact points.
3. Choose the external Alertmanager from the **Alertmanager** dropdown.
4. Select **New Silence**.
5. Select the start and end date in **Silence start and end** to indicate when the silence should go into effect and when it should end.

   As an alternative to setting an end time, in **Duration**, specify how long the silence is enforced. This automatically updates the end time in the **Silence start and end** field.
6. In the **Name** and **Value** fields, enter one or more *Matching Labels*. Matchers determine which rules the silence applies to. Label matching is discussed in more detail following this procedure.
7. Optionally, add a **Comment**, or modify the **Creator** to set the owner of the silence.

Label matching for alert suppression

When you create a silence, you create a set of *matching labels* as part of the silence. This is a set of rules about labels that must match for the alert to be suppressed. The matching labels consist of three parts:

- **Label** – The name of the label to match. It must exactly match the label name of the alert.
- **Operator** – The operator used to compare the label value with the matching label value. The available operators are:

- = Select labels whose value exactly matches the provided string.
- != Select labels whose value does not match the provided string.
- =~ Select labels whose value match the regex interpreted value of the provided string (the provided string is interpreted as a regular expression).
- != Select labels that do not match the provided regular expression.

- **Value** – The value to match the label value to. It can match as a string or as a regular expression, depending on the operator chosen.

A silence ends at the indicated end date, but you can manually end the suppression at any time.

To end a silence manually

1. In the **Alerting** page, choose **Silences** to view the list of existing silences.
2. Select the silence that you want to end, and choose **Unsilence**. This ends the alert suppression.

> **Note**
>
> Unsilencing ends the alert suppression, as if the end time was set for the current time. Silences that have ended (automatically or manually) are retained and listed for five days. You cannot remove a silence from the list manually.

Creating a link to the silence creation form

You can create a URL to the silence creation form with details already filled in. Operators can use this to suppress an alarm quickly during an operational event.

When creating a link to a silence form, use a `matchers` query parameter to specify the matching labels, and a `comment` query parameter to specify a comment. The `matchers` parameter requires one or more values in the form `[label][operator][value]`, separated by commas.

Example URL

To link to a silence form, with matching labels `severity=critical` and `cluster!~europe-.*`, with a comment that says `Silencing critical EU alerts`, use a URL like the following. Replace *`mygrafana`* with the hostname of your Grafana instance.

```
https://mygrafana/alerting/silence/new?matchers=severity%3Dcritical%2Ccluster!~europe-
*&comment=Silence%20critical%20EU%20alert
```

To link to a new silence page for an external Alertmanager, add an `alertmanager` query parameter with the Alertmanage data source name, such as `alertmanager=myAlertmanagerdatasource`.

Working with contact points

> ⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Use contact points to define how your contacts are notified when an alert is initiated. A contact point can have one or more contact point types, for example, Amazon Simple Notification Service or Slack. When an alert is initiated, a notification is sent to all contact point types listed for a contact point. Optionally, use Using messaging templates to customize the notification messages for the contact point types.

> **Note**
> You can create and edit contact points for Grafana managed alerts. Contact points for Alertmanager alerts are read-only.

Working with contact points

The following procedures detail how to add, edit, test, and delete contact points.

To add a contact point

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page.

2. Choose **Contact points**, then **New contact point**.
3. From the **Alertmanager** dropdown, select an Alertmanager. The Grafana Alertmanager is selected by default.
4. Enter a **Name** for the contact point.
5. From **Contact point type**, choose a type, and the mandatory fields based on that type. For example, if you choose Slack, enter the Slack channels and users who should be contacted.
6. If available for the contact point you selected, optionally choose the **Optional settings** to specify additional settings.
7. Under **Notification settings**, optionally select **Disable resolved message** if you do not want to be notified when an alert resolves.
8. If your contact point needs more contact points types, you can chooose **New contact point type** and repeat the steps for each contact point type needed.
9. Choose **Save contact point** to save your changes.

To edit a contact point

1. Choose **Contact points** to see a list of existing contact points.
2. Select the contact point to edit, then choose the **Edit** icon (pen).
3. Make any necessary changes, and then choose **Save contact point** to save your changes.

After your contact point is created, you can send a test notification to verify that it is configured properly.

To send a test notification

1. Choose **Contact points** to open the list of existing contact points.
2. Select the contact point to test, then choose the **Edit** icon (pen).
3. Select the **Test** icon (paper airplane).
4. Choose whether to send a predefined test notification or choose **Custom** to add your own custom annotations and labels in the test notification.
5. Choose **Send test notification** to test the alert with the given contact points.

You can delete contact points that are not in use by a notification policy.

To delete a contact point

1. Choose **Contact points** to open the list of existing contact points.
2. Select the contact point to delete, then choose the **Delete** icon (trash can).
3. In the confirmation dialog box, choose **Yes, delete**.

> **Note**
>
> If the contact point is in use by a notification policy, you must delete the notification policy or edit it to use a different contact point before deleting the contact point.

List of supported notifiers

Name	Type
Amazon SNS	`sns`
OpsGenie	`opsgenie`
Pager Duty	`pagerduty`
Slack	`slack`
VictorOps	`victorops`

Using messaging templates

> This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
> For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
> For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
> For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Notifications sent via Working with contact points are built using *messaging templates*. Grafana's default templates are based on the Go templating system where some fields are evaluated as text, while others are evaluated as HTML (which can affect escaping).

Since most of the contact point fields can be templated, you can create reusable custom templates and use them in multiple contact points. The Template data topic lists variables that are available for templating.

Using templates

Templates are used to create a message. For example, with a Slack alert message, you can set the title and body in the contact point. The following example shows how to use default templates to create a title that contains a count of alerts firing and resolved, and a body that lists the alerts and their statuses.

- **Title**:

```
{{ len .Alerts.Firing }} firing, {{ len .Alerts.Resolved }} resolved
```

- **Text Body**:

```
{{ range .Alerts }}{{ .Status }}: {{ .Labels.alertname }}
{{end }}
```

You can create your own custom templates, as in the following example.

- **Title**:

```
{{ template "slack.default.title" .}}
```

- **Text Body**:

```
{{ template "mymessage" .}}
```

The following is a sample template.

```
{{ define "myalert" }}
  [{{.Status}}] {{ .Labels.alertname }}
```

```
  Labels:
  {{ range .Labels.SortedPairs }}
    {{ .Name }}: {{ .Value }}
  {{ end }}

  {{ if gt (len .Annotations) 0 }}
  Annotations:
  {{ range .Annotations.SortedPairs }}
    {{ .Name }}: {{ .Value }}
  {{ end }}
  {{ end }}

  {{ if gt (len .SilenceURL ) 0 }}
    Silence alert: {{ .SilenceURL }}
  {{ end }}
  {{ if gt (len .DashboardURL ) 0 }}
    Go to dashboard: {{ .DashboardURL }}
  {{ end }}
{{ end }}
```

The following procedures show how to create, edit, and delete custom message templates.

To create a message template

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page.
2. Choose **Contact points**.
3. From the **Alertmanager** dropdown, select the Alertmanager instance you want to create a message template for. The default is the Grafana Alertmanager.
4. Choose **Add template**.
5. Add a descriptive **Name**.
6. Add the **Content** for the template, for example:

   ```
   {{ define "mymessage" }}
     {{ range .Alerts }}
       [{{ .Status }}] {{ range .Labels }} {{ .Name }}={{.Value }}{{end}}
     {{ end }}
   {{ end }}
   ```

The `define` tag in the Content section assigns the template name. This tag is optional, and when omitted, the template name is derived from the **Name** field. When both are specified, it is a best practice to keep them the same.

7. Choose **Save template**.

> **Note**
>
> HTML in alerting message templates is rendered as text, with control characters escaped. Rendering of HTML in the resulting notification is not supported by Grafana.

To edit a message template

1. In the **Alerting** page, choose **Contact points** to open the list of contact points.
2. In the **Template table**, find the template you want to edit, then choose the **Edit** icon (pen).
3. Make your changes, then choose **Save template**.

To delete a message template

1. In the **Alerting** page, choose **Contact points** to open the list of contact points.
2. In the **Template table**, find the template you want to remove, then choose the **Delete** icon (trash can).
3. Choose **Yes, delete** to delete the template.

Nested templates

You can embed templates within other templates.

For example, you can define a template fragment using the `define` keyword:

```
{{ define "mytemplate" }}
  {{ len .Alerts.Firing }} firing. {{ len .Alerts.Resolved }} resolved.
{{ end }}
```

You can then embed custom templates within this fragment using the `template` keyword. For example:

```
Alert summary:
{{ template "mytemplate" . }}
```

You can use the following built-in template options to embed custom templates.

Name	Notes
`default.title`	Displays high-level status information.
`default.message`	Provides a formatted summary of firing and resolved alerts.

Custom template examples

Here are examples of how to use custom templates.

Template to render a single alert:

```
{{ define "myalert" }}
  [{{.Status}}] {{ .Labels.alertname }}

  Labels:
  {{ range .Labels.SortedPairs }}
    {{ .Name }}: {{ .Value }}
  {{ end }}

  {{ if gt (len .Annotations) 0 }}
  Annotations:
  {{ range .Annotations.SortedPairs }}
    {{ .Name }}: {{ .Value }}
  {{ end }}
  {{ end }}

  {{ if gt (len .SilenceURL ) 0 }}
    Silence alert: {{ .SilenceURL }}
  {{ end }}
  {{ if gt (len .DashboardURL ) 0 }}
    Go to dashboard: {{ .DashboardURL }}
  {{ end }}
{{ end }}
```

Template to render entire notification message:

```
{{ define "mymessage" }}
  {{ if gt (len .Alerts.Firing) 0 }}
    {{ len .Alerts.Firing }} firing:
    {{ range .Alerts.Firing }} {{ template "myalert" .}} {{ end }}
  {{ end }}
  {{ if gt (len .Alerts.Resolved) 0 }}
    {{ len .Alerts.Resolved }} resolved:
    {{ range .Alerts.Resolved }} {{ template "myalert" .}} {{ end }}
  {{ end }}
{{ end }}
```

Template data

The following data is passed to message templates.

Name	Type	Notes
`Receiver`	string	Name of the contact point that the notification is being sent to.
`Status`	string	firing if at least one alert is firing, otherwise resolved.
`Alerts`	Alert	List of alert objects that are included in this notification (see below).
`GroupLabels`	KeyValue	Labels these alerts were grouped by.
`CommonLabels`	KeyValue	Labels common to all the alerts included in this notification.
`CommonAnnotations`	KeyValue	Annotations common to all the alerts included in this notification.

Name	Type	Notes
`ExternalURL`	string	Back link to the Grafana that sent the notification. If using external Alertmanager, back link to this Alertmanager.

The `Alerts` type exposes two functions for filtering the alerts returned.

- `Alerts.Firing` – Returns a list of firing alerts.
- `Alerts.Resolved` – Returns a list of resolved alerts.

Alert (type)

The alert type contains the following data.

Name	Type	Notes
Status	string	`firing` or `resolved`.
Labels	KeyValue	A set of labels attached to the alert.
Annotations	KeyValue	A set of annotations attached to the alert.
StartsAt	time.Time	Time the alert started firing.
EndsAt	time.Time	Only set if the end time of an alert is known. Otherwise set to a configurable timeout period from the time since the last alert was received.
GeneratorURL	string	A back link to Grafana or external Alertmanager.

Name	Type	Notes
SilenceURL	string	Link to grafana silence for with labels for this alert pre-filled. Only for Grafana managed alerts.
DashboardURL	string	Link to grafana dashboard, if alert rule belongs to one. Only for Grafana managed alerts.
PanelURL	string	Link to grafana dashboard panel, if alert rule belongs to one. Only for Grafana managed alerts.
Fingerprint	string	Fingerprint that can be used to identify the alert.
ValueString	string	A string that contains the labels and value of each reduced expression in the alert.

KeyValue type

The `KeyValue` type is a set of key/value string pairs that represent labels and annotations.

In addition to direct access of the data stored as a `KeyValue`, there are also methods for sorting, removing and transforming the data.

Name	Arguments	Returns	Notes
SortedPairs		Sorted list of key and value string pairs	

Name	Arguments	Returns	Notes
Remove	[]string	KeyValue	Returns a copy of the Key/Value map without the given keys.
Names		[]string	List of label names
Values		[]string	List of label values

Template functions

Using template functions you can process labels and annotations to generate dynamic notifications. The following functions are available.

Name	Argument type	Return type	Description
`humanize`	number or string	string	Converts a number to a more readable format, using metric prefixes.
`humanize1024`	number or string	string	Like humanize, but uses 1024 as the base rather than 1000.
`humanizeDuration`	number or string	string	Converts a duration in seconds to a more readable format.
`humanizePercentage`	number or string	string	Converts a ratio value to a fraction of 100.
`humanizeTimestamp`	number or string	string	Converts a Unix timestamp in seconds

Name	Argument type	Return type	Description
			to a more readable format.
`title`	string	string	strings.Title, capitalizes first character of each word.
`toUpper`	string	string	strings.ToUpper, converts all characters to upper case.
`toLower`	string	string	strings.ToLower, converts all characters to lower case.
`match`	pattern, text	boolean	regexp.MatchString Tests for an unanchored regexp match.
`reReplaceAll`	pattern, replacement, text	string	Regexp.ReplaceAllString Regexp substitution, unanchored.
`graphLink`	string - JSON Object with `expr` and `datasource` fields	string	Returns the path to graphical view in Explore for the given expression and data source.
`tableLink`	string - JSON Object with `expr` and `datasource` fields	string	Returns the path to tabular view in Explore for the given expression and data source.

Name	Argument type	Return type	Description
`args`	[]interface{}	map[string]interface{}	Converts a list of objects to a map with keys, for example, arg0, arg1. Use this function to pass multiple arguments to templates.
`externalURL`	nothing	string	Returns a string representing the external URL.
`pathPrefix`	nothing	string	Returns the path of the external URL.

The following table shows examples of using each function.

Function	TemplateString	Input	Expected
humanize	{ humanize $value }	1234567.0	1.235M
humanize1024	{ humanize1024 $value }	1048576.0	1Mi
humanizeDuration	{ humanizeDuration $value }	899.99	14m 59s
humanizePercentage	{ humanizePercentage $value }	0.1234567	12.35%
humanizeTimestamp	{ humanizeTimestamp $value }	1435065584.128	2015-06-23 13:19:44.128 +0000 UTC
title	{ $value | title }	aa bB CC	Aa Bb Cc

Function	TemplateString	Input	Expected
toUpper	{ $value | toUpper }	aa bB CC	AA BB CC
toLower	{ $value | toLower }	aa bB CC	aa bb cc
match	{ match "a+" $labels.instance }	aa	true
reReplaceAll	{{ reReplaceAll "localhost:(.*)" "my.domain:$1" $labels.instance }}	localhost:3000	my.domain:3000
graphLink	{{ graphLink "{\"expr\": \"up\", \"datasource\": \"gdev-prometheus\"}" }}		/explore?left=["now-1h","now","gdev-prometheus",{"datasource":"gdev-prometheus","expr":"up","instant":false,"range":true}]
tableLink	{{ tableLink "{\"expr\":\"up\", \"datasource\":\"gdev-prometheus\"}" }}		/explore?left=["now-1h","now","gdev-prometheus",{"datasource":"gdev-prometheus","expr":"up","instant":true,"range":false}]
args	{{define "x"}}{{.arg0}} {{.arg1}}{{end}}{{ template "x" (args 1 "2")}}		1 2
externalURL	{ externalURL }		http://localhost/path/prefix
pathPrefix	{ pathPrefix }		/path/prefix

Working with notification policies

⚠ This documentation topic is designed for Grafana workspaces that support **Grafana version 8.x**.
For Grafana workspaces that support Grafana version 12.x, see Working in Grafana version 12.
For Grafana workspaces that support Grafana version 10.x, see Working in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Working in Grafana version 9.

Notification policies determine how alerts are routed to contact points. Policies have a tree structure, where each policy can have one or more child policies. Each policy, except for the root policy, can also match specific alert labels. Each alert is evaluated by the root policy and then by each child policy. If you enable the `Continue matching subsequent sibling nodes` option for a specific policy, then evaluation continues even after one or more matches. A parent policy's configuration settings and contact point information govern the behavior of an alert that does not match any of the child policies. A root policy governs any alert that does not match a specific policy.

Note

You can create and edit notification policies for Grafana managed alerts. Notification policies for Alertmanager alerts are read-only.

Grouping notifications

Grouping categorizes alert notifications of similar nature into a single funnel. This allows you to control alert notifications during larger outages when many parts of a system fail at once causing a high number of alerts to initiate simultaneously.

Grouping example

Suppose you have 100 services connected to a database in different environments. These services are differentiated by the label `env=environmentname`. An alert rule is in place to monitor whether your services can reach the database. The alert rule creates alerts named `alertname=DatabaseUnreachable`.

If a network partition occurs, where half of your services can no longer reach the database, 50 different alerts are initiated. For this situation, you want to receive a single-page notification (as opposed to 50) with a list of the environments that are affected.

You can configure grouping to be `group_by: [alertname]` (not using the `env` label, which is different for each service). With this configuration in place, Grafana sends a single compact notification that has all the affected environments for this alert rule.

Special Groups

Grafana has two special groups. The default group, `group_by: null` groups *all* alerts together into a single group. You can also use a special label named `...` to group alerts by all labels, effectively disabling grouping, and sending each alert into its own group.

Working with notifications

The following procedures show you how to create and manage notification policies.

To edit the root notification policy

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page.
2. Choose **Notification policies**.
3. From the **Alertmanager** dropdown, select the Alertmanager you want to edit.
4. In the **Root policy** section, choose the **Edit** icon (pen).
5. In **Default contact point**, update the contact point where notifications should be sent for rules when alert rules do not match any specific policy.
6. In **Group by**, choose the labels (or special groups) to group alerts by.
7. In **Timing options**, select from the following options.
   - **Group wait** – Time to wait to buffer alerts of the same group before sending an initial notification. The default is 30 seconds.
   - **Group interval** – Minimum time interval between two notifications for a group. The default is 5 minutes.
   - **Repeat interval** – Minimum time interval before resending a notification if no new alerts were added to the group. The default is 4 hours.
8. Choose **Save** to save your changes.

To add a new, top-level specific policy

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page.
2. Choose **Notification policies**.
3. From the **Alertmanager** dropdown, select the Alertmanager you want to edit.
4. In the **Specific routing** section, choose **New specific policy**.
5. In the **Matching labels** section, add one or more matching alert labels. More information about label matching is later in this topic.
6. In **Contact point**, add the contact point to send notifications to if the alert matches this specific policy. Nested policies override this contact point.
7. Optionally select **Override grouping** to specify a grouping different from the root policy.
8. Optionally select **Override general timings** to override the timing options in the group notification policy.
9. Choose **Save policy** to save your changes.

To add a nested policy

1. Expand the specific policy you want to create a nested policy under.
2. Choose **Add nested policy**, then add the details (as when adding a top-level specific policy).
3. Choose **Save policy** to save your changes.

To edit a specific policy

1. From the **Alerting** page, choose **Notification policies** to open the page that listing existing policies.
2. Select the policy that you want to edit, then choose the **Edit** icon (pen).
3. Make any changes (as when adding a top-level specific policy).
4. Choose **Save policy**.

How label matching works

A policy matches an alert if the alert's labels match all the *Matching Labels* specified on the policy.

- **Label** – The name of the label to match. It must exactly match the label name of the alert.

- **Operator** – The operator used to compare the label value with the matching label value. The available operators are:
  - = Select labels whose value exactly matches the provided string.
  - != Select labels whose value does not match the provided string.
  - =~ Select labels whose value match the regex interpreted value of the provided string (the provided string is interpreted as a regular expression.
  - != Select labels that do not match the provided regular expression.
- **Value** – The value to match the label value to. It can match as a string or as a regular expression, depending on the operator chosen.

Mute timings

A mute timing is a recurring interval of time when no new notifications for a policy are generated or sent. Use them to prevent alerts from firing a specific and reoccurring period, for example, a regular maintenance period.

Similar to silences, mute timings do not prevent alert rules from being evaluated, nor do they stop alert instances from being shown in the user interface. They only prevent notifications from being created.

You can configure Grafana managed mute timings as well as mute timings for an external Alertmanager data source.

Mute timings compared to silences

The following table highlights the differences between mute timings and silences.

Mute timing	Silence
Uses time interval definitions that can reoccur	Has a fixed start and end time
Is created and then added to notification policies	Uses labels to match against an alert to determine whether to silence or not

To create a mute timing

1. From your Grafana console, in the Grafana menu, choose the **Alerting** (bell) icon to open the **Alerting** page.

2. Choose **Notification policies**.
3. From the **Alertmanager** dropdown, select the Alertmanager you want to edit.
4. In the **Mute timings** section, choose the **Add mute timing** button.
5. Choose the time interval for which you want the mute timting to apply.
6. Choose **Submit** to create the mute timing.

To add a mute timing to a notification policy

1. Select the notification policy you would like to add the mute timing to, and choose the **Edit** button.
2. From the **Mute timings** dropdown, select the mute timings you would like to add to the policy.

   Choose the **Save policy** button.

Time intervals

A time interval is a definition for a range of time. If an alert is initiated during this interval it is suppressed. Ranges are supported using `:` (for example, `monday:thursday`). A mute timing can contain multiple time intervals. A time interval consists of multiple fields (details in the following list), all of which must match in order to suppress the alerts. For example, if you specify days of the week `monday:friday` and time range from 8:00-9:00, then alerts are suppressed from 8-9, Monday through Friday, but not, for example, 8-9 on Saturday.

- **Time range** – The time of day to suppress notifications. Consists of two sub-fields, **Start time** and **End time**. An example time is `14:30`. Time is in 24 hour notation, in UTC.
- **Days of the week** – The days of the week. Can be a single day, such as `monday`, a range, such as `monday:friday`, or a comma-separate list of days, such as `monday, tuesday, wednesday`.
- **Months** – The months to select. You can specify months with numeric designations, or with the full month name, for example `1` or `january` both specify January. You can specify a single month, a range of months, or a comma-separated list of months.
- **Days of the month** – The dates within a month. Values can range from `1-31`. Negative values specify days of the month in reverse order, so `-1` represents the last day of the month. Days of the month can be specified as a single day, a range of days, or a comma-separate list of days.

Change your preferences

You can perform several tasks on the **Preferences** tab. You can edit your profile, change your Amazon Managed Grafana preferences, and view information about your profile and Amazon Managed Grafana usage.

> **Note**
>
> To make changes to the configuration of your workspace, see Configure a Amazon Managed Grafana workspace.

Edit your Amazon Managed Grafana profile

Your profile includes your name, user name, and email address.

To edit your profile

1. Pause on your user icon in the lower left corner of the screen, and then choose **Preferences**.
2. In the **Edit Profile** section, you can edit any of the following:
   - **Name** – Edit this field to change the display name associated with your profile.
   - **Email** – Edit this field to change the email address associated with your profile.
   - **Username** – Edit this field to change your user name.
3. Choose **Save**.

Edit your preferences

Your preferences include whether uses the dark or light theme, your home dashboard, and your timezone.

> **Note**
>
> Settings on your personal instance override settings made by your administrator at the instance or team level.

To change your preferences

1. Pause on your user icon in the lower left corner of the screen, and then choose **Preferences**.
2. In the Preferences section, you can edit any of the following:

   - **UI Theme** – To set a theme, choose **Dark** or **Light**. **Default** is either the dark theme or the theme selected by your Grafana administrator.
   - **Home Dashboard**
   - **Timezone** – Choose to select an option in the **Timezone** list. **Default** is either the browser local timezone or the timezone selected by your Grafana administrator. For more information, see Time range controls..
3. Choose **Save**.

View your Amazon Managed Grafana sessions

Amazon Managed Grafana logs your sessions in each Grafana workspace. If you suspect someone has misused your Amazon Managed Grafana credentials you can review this section.

To view your sessions information

1. Pause on your user icon in the lower left corner of the screen, and then choose **Preferences.**
2. Scroll down to the **Sessions** section. Grafana displays the following:

   - **Last seen** – How long ago you logged on.
   - **Logged on** – The date you logged on to the current Grafana instance.
   - **IP address** – The IP address that you logged on from.
   - **Browser & OS** – The web browser and operating system used to log on to Grafana.
   - If you are a Grafana admin for the instance, you can revoke a session by choosing the red signout icon in the session row.

Gather information for support

Support bundles provide a simple way to collect information about your Grafana workspace through the user interface. When you encounter problems with your Grafana workspace, you can send product support a support bundle that contains information about your workspace, including:

- Grafana version
- Installed plugins
- Grafana configuration
- Database information and migrations

> **Note**
>
> Support bundles are only available in workspaces compatible with Grafana version 10 or newer.

Support bundle components

A support bundle can include any of the following components:

- **Usage statistics** – Usage statistics for the Grafana workspace.
- **User information** – A list of users of the Grafana workspace.
- **Database and migration information** – Database information, and migration log.
- **Plugin information** – Information about installed plugins in the workspace.
- **Basic information** – Basic information about the Grafana workspace, including version, and memory usage.
- **Settings** – The settings of the Grafana workspace.
- **SAML** – Healthcheck connection and metadata for SAML (only displayed if SAML is enabled).
- **LDAP** – Healthcheck connection and metadata for LDAP (only displayed if LDAP is enabled).
- **OAuth2** – Healthcheck connection and metadata for each OAuth2 provider (only displayed if OAuth provider is enabled).

Creating a support bundle

Use the following procedure to create a support bundle.

> **Note**
>
> This procedure requires `admin` permissions in the workspace.

To create a support bundle

1. Sign into your Grafana workspace.
2. Select the Help icon
3. From the help menu, choose **Support Bundles**.
4. Select **New Support Bundle**.
5. Choose the components that you want to include in the support bundle.
6. Select **Create**.
7. After the support bundle is ready, select **Download**.

   Grafana downloads the support bundle to an archive (`tar.gz`) file.

You can open the file to view the contents of the support bundle. You can directly send the file to support, if needed. If the bundle contains private information, and you must send it via a channel that is not private, you may consider encrypting it. You can use a tool like age to encrypt the file before sending.

Classic dashboard alerts

⚠ This documentation topic discusses legacy alerting in Grafana. Legacy alerting is removed in Amazon Managed Grafana version 12. You must migrate to Grafana alerting before upgrading to v12. For more information, see one of the following topics.
For Grafana workspaces that support Grafana version 10.x, see Alerts in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Alerts in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Grafana alerting.

Note

This section describes the classic dashboard alerts system in Grafana. To learn about migrating to, and using, the new Grafana alerting, which is designed to view Prometheus Alertmanager alerts along with Grafana alerts, see Alerts in Grafana version 10.

Classic dashboard alerts consist of two parts:

- Alert rules – When the alert is triggered. Alert rules are defined by one or more conditions that are regularly evaluated by Grafana.
- Notification channel – How the alert is delivered. When the conditions of an alert rule are met, the Grafana notifies the channels configured for that alert.

Currently, only the graph panel visualization supports alerts.

Alert configuration

You can configure alerts in your Amazon Managed Grafana workspace.

- Add or edit an alert notification channel. For more information, see Notifications.
- Create an alert rule. For more information, see Creating alerts.
- View existing alert rules and their current state. For more information, see Viewing existing alert rules.
- Test alert rules and troubleshoot. For more information, see Troubleshooting alerts.

Clustering

Currently, alerting supports a limited form of high availability. Alert notifications are deduplicated when you run multiple workspaces. This means that all alerts are run on every server, but no duplicate alert notifications are sent due to the deduping logic.

Notifications

You can create alert rules with detailed messages including information such as how you might solve the issue, a link to a runbook, and so on.

The actual notifications are configured and shared between multiple alerts.

Alert execution

Alert rules are evaluated in Amazon Managed Grafana in a scheduler and query execution engine.

Alert notifications

⚠ This documentation topic discusses legacy alerting in Grafana. Legacy alerting is removed in Amazon Managed Grafana version 12. You must migrate to Grafana alerting before upgrading to v12. For more information, see one of the following topics.
For Grafana workspaces that support Grafana version 10.x, see Alerts in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Alerts in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Grafana alerting.

When an alert changes state, it sends out notifications. Each alert rule can have multiple notifications. To add a notification to an alert rule, you first must add and configure a `notification` channel.

This is done from the Notification channels page.

Adding a notification channel

1. In the side bar, pause on the **Alerting** (bell) icon, and then choose **Notification channels**.
2. Choose **Add channel**.
3. Fill out the fields or select options described in the following sections.

New notification channel fields

Default (send on all alerts)

- **Name** – Enter a name for this channel. It will be displayed when users add notifications to alert rules.
- **Type** – Select the channel type. For more information, see List of supported notifiers.
- **Default (send on all alerts)** – When selected, this option sends a notification on this channel for all alert rules.
- **Disable Resolve Message** – When selected, this option disables the resolve message [OK] that is sent when the alerting state returns to false.
- **Send reminders** – When this option is selected, additional notifications (reminders) will be sent for alerts. You can specify how often reminders should be sent by using the number of seconds (s), minutes (m), or hours (h); for example, `30s`, `3m`, `5m` or `1h`.

Important

Alert reminders are sent after rules are evaluated. Therefore, a reminder can't be sent more frequently than a configured alert rule evaluation interval.

The following examples show how often and when reminders are sent for a triggered alert.

Alert rule evaluation interval	Send reminders every	Reminder sent every (after last alert notification)
`30s`	`15s`	~30 seconds
`1m`	`5m`	~5 minutes
`5m`	`15m`	~15 minutes
`6m`	`20m`	~24 minutes
`1h`	`15m`	~1 hour
`1h`	`2h`	~2 hours

List of supported notifiers

Name	Type	Supports images	Supports alert rule tags
Amazon Simple Notification Service	sns	No	Yes
OpsGenie	`opsgenie`	No	Yes
PagerDuty	`pagerduty`	No	Yes
Slack	`slack`	No	No
VictorOps	`victorops`	No	No

Amazon Simple Notification Service

If you have enabled service-managed permissions and included Amazon SNS as a notification channel for your workspace, you only need to provide the SNS Topic ARN when you create your notification channel. In the **Name** field, provide the name of the SNS topic that you have created. If you created the workspace using service-managed permissions, the SNS topic name must be prefixed with `grafana` for notifications to successfully publish to the topic. If you selected customer-managed permissions when you created the workspace, the SNS Topic name does not need to be prefixed with `grafana`.

In the **Topic** field, copy and paste the ARN of the SNS topic. In the **Message body format**, you can choose either the JSON or the text option.

In the **Optional AWS SNS Settings** field, check the checkbox **Include all tags in the message** to see all the Grafana tags in the message body.

If you use customer-managed permissions for the workplace, the IAM role that you supply should include SNS Publish permissions for your SNS Topic.

Slack

To set up Slack, you must configure an incoming Slack webhook URL. For more information, see Sending messages using Incoming Webhooks.

For more information about setting up a Slack bot integration, see Follow Slack's guide to set up a bot integration. Use the token provided, which starts with "xoxb".

Setting	Description
Url	Slack incoming webhook URL, or eventually the chat.postMessage Slack API endpoint.
Username	Set the user name for the bot's message.
Recipient	Use this to override the Slack recipient. You must provide either a channel Slack ID, a user Slack ID, a user name reference (@<user>, all lowercase, no whitespace), or a channel reference (#<channel>, all lowercase, no white

Setting	Description
	space). If you use the `chat.postMessage` Slack API endpoint, this is required.
Icon emoji	Provide an emoji to use as the icon for the bot's message. For example, :smile:
Icon URL	Provide a URL to an image to use as the icon for the bot's message.
Mention Users	Optionally mention one or more users in the Slack notification sent by Grafana. To see users, comma-separated, via their corresponding Slack IDs, choose the overflow button on each user's Slack profile.
Mention Groups	Optionally mention one or more groups in the Slack notification sent by Grafana. You can see groups, comma-separated, via their correspon ding Slack IDs (which you can get from each group's Slack profile URL).
Mention Channel	Optionally mention either all channel members or only active ones.
Token	If provided, Amazon Managed Grafana will upload the generated image via the Slack file.upload API operation, not the external image destination. If you use the `chat.postMessage` Slack API endpoint, this is required.

If you are using the token for a slack bot, you have to invite the bot to the channel that you want to send notifications. Then add the channel to the recipient field.

PagerDuty

To set up PagerDuty, provide an integration key.

Setting	Description
Integration Key	Integration key for PagerDuty.
Severity	Level for dynamic notifications; default is `critical` (1) .
Auto resolve incidents	Resolve incidents in PagerDuty after the alert goes back to ok.
Message in details	Removes the Alert message from the PD summary field and puts it into custom details instead (2).

Note

The tags `Severity`, `Class`, `Group`, `dedup_key`, and `Component` have special meaning in the PagerDuty Common Event Format – PD-CEF. If an alert panel defines these tag keys, they are transposed to the root of the event sent to PagerDuty. This means they will be available within the PagerDuty UI and Filtering tools. A Severity tag set on an alert overrides the global Severity set on the notification channel if it's a valid level.

Note

Using Message In Details will change the structure of the `custom_details` field in the PagerDuty Event. This might break custom event rules in your PagerDuty rules if you rely on the fields in `payload.custom_details`. Move any existing rules that use `custom_details.myMetric` to `custom_details.queries.myMetric`.

Note

Using `dedup_key` tag will override the Grafana generated `dedup_key` with a custom key.

Configuring the link back to Grafana from alert notifications

All alert notifications contain a link back to the triggered alert in the Grafana workspace.

Creating alerts

⚠ This documentation topic discusses legacy alerting in Grafana. Legacy alerting is removed in Amazon Managed Grafana version 12. You must migrate to Grafana alerting before upgrading to v12. For more information, see one of the following topics.
For Grafana workspaces that support Grafana version 10.x, see Alerts in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Alerts in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Grafana alerting.

When you use Amazon Managed Grafana alerting, you can attach rules to your dashboard panels. When you save the dashboard, Amazon Managed Grafana extracts the alert rules into a separate alert rule storage and schedules them for evaluation.

On the **Alert** tab of the graph panel, you can configure how often the alert rule should be evaluated and the conditions that must be met for the alert to change state and initiate its notifications.

Currently, only the graph panel supports alert rules.

Adding or editing an alert rule

1. Navigate to the panel where add or edit an alert rule, choose the title, and then choose **Edit**.
2. On the **Alert** tab, choose **Create Alert**. If an alert already exists for this panel, you can edit the fields on the **Alert** tab.
3. Fill out the fields. For more information, see Alert rule fields.
4. When you have finished writing your rule, choose **Save** in the upper right corner to save the alert rule and the dashboard.
5. (Optional but recommended) To make sure that the rule returns the results you expect, choose **Test rule**.

Deleting an alert rule

To delete an alert, scroll to the bottom of the alert, and then choose **Delete**.

Alert rule fields

This section describes the fields that you fill out to create an alert.

Rule

- **Name** – Enter a descriptive name. The name will be displayed in the **Alert Rules** list.
- **Evaluate every** – Specify how often the scheduler should evaluate the alert rule. This is referred to as the *evaluation interval*.
- **For** – Specify how long the query must violate the configured thresholds before the alert notification triggers.

> **Warning**
>
> Do not use `For` with the `If no data or all values are null` setting set to `No Data`. The triggering of `No Data` will trigger instantly and not take `For` into consideration. This can also result in an OK notification not being sent if alert transitions from `No Data -> Pending -> OK`.

If an alert rule has a configured `For` and the query violates the configured threshold, it will first go from `OK` to `Pending`. Going from `OK` to `Pending`, Amazon Managed Grafana does not send any notifications. When the alert rule has been firing for more than the `For` duration, it will change to `Alerting` and send alert notifications.

Typically, we recommend using this setting because it's often worse to get false positive than to wait a few minutes before the alert notification initiates. Looking at the `Alert list` or `Alert list panels`, you will be able to see alerts that are in the pending state.

Conditions

Currently, the only existing condition type is a `Query` condition that allows you to specify a query letter, a time range, and an aggregation function.

Query condition example

```
avg() OF query(A, 15m, now) IS BELOW 14
```

- `avg()` Controls how the values for **each** series should be reduced to a value that can be compared against the threshold. Choose the function to change it to another aggregation function.
- `query(A, 15m, now)` The letter defines what query to run from the **Metrics** tab. The second two parameters define the time range: `15m, now` means 15 minutes ago to now. You can also use `10m, now-2m` to define a time range that will be 10 minutes ago to 2 minutes ago. This is useful if you want to ignore the last 2 minutes of data.
- `IS BELOW 14` Defines the type of threshold and the threshold value. You can choose `IS BELOW` to change the type of threshold.

The query used in an alert rule cannot contain any template variables. Currently, we support only AND and OR operators between conditions, and they are run serially. For example, we have three conditions in the following order: *condition:A(evaluates to: TRUE) OR condition:B(evaluates to: FALSE) AND condition:C(evaluates to: TRUE)* so the result will be calculated as ((TRUE OR FALSE) AND TRUE) = TRUE.

Multiple series

If a query returns multiple series, the aggregation function and threshold check will be evaluated for each series. Currently, Amazon Managed Grafana does not track the alert rule state **per series**. The implications of this are detailed in the following scenario.

- An alert condition with query that returns two series: **server1** and **server2**.
- The **server1** series causes the alert rule to fire and switch to state `Alerting`.
- Notifications are sent out with message: *load peaking (server1)*
- In a subsequent evaluation of the same alert rule, the **server2** series also causes the alert rule to fire.
- No new notifications are sent because the alert rule is already in state `Alerting`.

As you can see from the previous scenario, if the rule already is in state `Alerting`, Grafana doesn't send out notifications when other series cause the alert to fire.

Note

You can configure reminders to be sent for triggered alerts. This will send additional notifications when an alert continues to fire. If other series (such as server2 in the previous example) also cause the alert rule to fire, they are included in the reminder notification. Depending on which notification channel you're using, you might be able to take advantage of this feature for identifying new or existing series that are causing alerts to fire.

No data and error handling

The following table contains conditions for controlling how the rule evaluation engine handles queries that return no data or only null values.

No Data Option	Description
No Data	Set alert rule state to `NoData`.
Alerting	Set alert rule state to `Alerting`.
Keep Last State	Keep the current alert rule state, whatever it is.
Ok	Supported, but usually not useful.

Execution errors or timeouts

The following options tell Amazon Managed Grafana how to handle execution or timeout errors.

Error or timeout option	Description
Alerting	Set alert rule state to `Alerting`.
Keep Last State	Keep the current alert rule state, whatever it is.

If you have an unreliable time series store from which queries sometimes time out or fail randomly, you can set this option to `Keep Last State` to basically ignore them.

Notifications

On **Alert** tab, you can also specify alert rule notifications and a detailed message about the alert rule. The message can contain anything: information about how you might solve the issue, link to runbook, and so on.

The actual notifications are configured and shared between multiple alerts. For information on how to configure and set up notifications, see Alert notifications.

- **Send to** – Select an alert notification channel if you have one set up.
- **Message** – Enter a text message to be sent on the notification channel. Some alert notifiers support transforming the text to HTML or other rich formats.
- **Tags** – Specify a list of tags (key-value) to be included in the notification. It is supported by only some notifiers.

Alert state history and annotations

Alert state changes are recorded in the internal annotation table in the Amazon Managed Grafana database. The state changes are visualized as annotations in the graph panel of the alert rulel. You can also go into the `State history` submenu on the **Alert** tab to view and clear state history.

Pausing an alert rule

⚠ This documentation topic discusses legacy alerting in Grafana. Legacy alerting is removed in Amazon Managed Grafana version 12. You must migrate to Grafana alerting before upgrading to v12. For more information, see one of the following topics.
For Grafana workspaces that support Grafana version 10.x, see Alerts in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Alerts in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Grafana alerting.

Pausing the evaluation of an alert rule can sometimes be useful. For example, during a maintenance window, pausing alert rules can avoid initiating a flood of alerts.

1. In the Grafana side bar, pause on the **Alerting** (bell) icon and then choose **Alert Rules**. All configured alert rules are listed, along with their current state.
2. Find your alert in the list, and choose the **Pause** icon on the right. The **Pause** icon turns into a **Play** icon.
3. Choose the **Play** icon to resume evaluation of your alert.

Viewing existing alert rules

⚠ This documentation topic discusses legacy alerting in Grafana. Legacy alerting is removed in Amazon Managed Grafana version 12. You must migrate to Grafana alerting before upgrading to v12. For more information, see one of the following topics.
For Grafana workspaces that support Grafana version 10.x, see Alerts in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Alerts in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Grafana alerting.

Amazon Managed Grafana stores individual alert rules in the panels where they are defined, but you can also view a list of all existing alert rules and their current state.

In the Grafana side bar, pause on the **Alerting** (bell) icon, and then choose **Alert Rules**. All configured alert rules are listed, along with their current state.

While viewing alerts, you can do the following:

- **Filter alerts by name** – Type an alert name in the **Search alerts** field.
- **Filter alerts by state** – In **States**, select which alert states you want to see. All others will be hidden.
- **Pause or resume an alert** – choose the **Pause** or **Play** icon next to the alert to pause or resume evaluation.
- **Access alert rule settings** – Choose the alert name or the **Edit alert rule** (gear) icon. Amazon Managed Grafana opens the **Alert** tab of the panel where the alert rule is defined. This is helpful when an alert is firing, but you don't know which panel it is defined in.

Notification templating

⚠ This documentation topic discusses legacy alerting in Grafana. Legacy alerting is removed in Amazon Managed Grafana version 12. You must migrate to Grafana alerting before upgrading to v12. For more information, see one of the following topics.
For Grafana workspaces that support Grafana version 10.x, see Alerts in Grafana version 10.
For Grafana workspaces that support Grafana version 9.x, see Alerts in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Grafana alerting.

The alert notification template feature allows you to take the label value from an alert query and inject that into alert notifications.

Labels that exist from the evaluation of the alert query can be used in the alert rule name and in the alert notification message fields. The alert label data is injected into the notification fields when the alert is in the alerting state. When there are multiple unique values for the same label, the values are comma-separated.

To add alert label data into your alert notification

1. Navigate to the panel that you want to add or edit an alert rule for.
2. Choose the panel title and choose **Edit**.
3. On the **Alert** tab, choose **Create Alert**. If an alert already exists for this panel, you can edit it directly.
4. Refer to the alert query labels in the alert rule name or the alert notification message field by using the `${Label}` syntax. For more information about alert query labels, see Message templating in the Grafana documentation.
5. Choose **Save** in the upper right corner.

Troubleshooting alerts

⚠ This documentation topic discusses legacy alerting in Grafana. Legacy alerting is removed in Amazon Managed Grafana version 12. You must migrate to Grafana alerting before upgrading to v12. For more information, see one of the following topics.
For Grafana workspaces that support Grafana version 10.x, see Alerts in Grafana version 10.

For Grafana workspaces that support Grafana version 9.x, see Alerts in Grafana version 9.
For Grafana workspaces that support Grafana version 8.x, see Grafana alerting.

If alerts are not behaving as you expect, the following steps can help you troubleshoot and figure out what is going wrong.

The first level of troubleshooting that you can do is choose **Test Rule**. You can expand the result to the point where you can see the raw data that was returned from your query.

Using Grafana HTTP APIs

You can use Grafana HTTP APIs with Amazon Managed Grafana workspaces. The following sections describe how to use the APIs, and then list which Grafana APIs are supported.

> **Note**
>
> Amazon Managed Grafana also provides AWS APIs for creating and managing workspaces. For more information about the AWS APIs, see the Amazon Managed Grafana API Reference.

Topics

- Grafana version 12 API reference
- Grafana version 10 API reference

Grafana version 12 API reference

You can use Grafana HTTP APIs with Amazon Managed Grafana workspaces. The following sections describe how to use the APIs, and then list which Grafana APIs are supported.

> **Note**
>
> Amazon Managed Grafana also provides AWS APIs for creating and managing workspaces. For more information about the AWS APIs, see the Amazon Managed Grafana API Reference.

Topics

- Authenticate with service accounts
- New API structure
- Annotations API
- Alerting Notification Channels API
- Dashboard API

- Dashboard Permissions API
- Dashboard Versions API
- Data Source API
- Data Source Permissions API
- External Group Synchronization API
- Folder API
- Folder/Dashboard Search API
- Folder Permissions API
- Organization API
- Playlist API
- Plugin API
- Preferences API
- Snapshot API
- Short URL API
- Team API
- User API

Authenticate with service accounts

Using service accounts

To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. Service accounts are used to run automated workloads in Grafana, such as dashboard provisioning, configuration, or report generation. Create service accounts and tokens to authenticate applications, such as Terraform, with the Grafana console or the Amazon Managed Grafana API.

A common use case for creating a service account is to perform operations on automated or triggered tasks. You can use service accounts to:

- Define alerts in your system to be used in Grafana
- Interact with Grafana without signing in as a user

> **Note**
>
> Each service account is considered a user for billing purposes.

Service account tokens

A service account token is a generated string that acts as a key to authenticate with Grafana's HTTP API.

When you create a service account, you can associate one or more access tokens with it. You can use service account tokens to access Grafana HTTP APIs programmatically.

You can create multiple tokens for the same service account. You might want to do this if:

- multiple applications use the same permissions, but you would like to audit or manage their actions separately.
- you need to rotate or replace a compromised token.

Service account access tokens inherit permissions from the service account.

Amazon Managed Grafana has a quota on the number of service account tokens you can have at one time. This includes active and expired tokens. You must delete tokens to remove them from your quota.

Service account benefits

Service accounts provide the following benefits:

- Service accounts resemble Grafana users and can be enabled/disabled, granted specific permissions, and remain active until they are deleted or disabled.
- Service accounts can be associated with multiple tokens.
- Service account tokens are not associated with a specific user, which means that applications can be authenticated even if a Grafana user is deleted.
- You can grant permissions to service accounts in the same way that you grant permissions to users.

  For more information about permissions, see Using permissions.

Creating a service account

> **Note**
>
> The user who creates a service account is also able to read, update and delete the service account that they created, as well as permissions associated with that service account.

Prerequisite

Ensure you have permission to create and edit service accounts. By default, the organization administrator role is required to create and edit service accounts. For more information about permissions, see Using permissions.

To create a service account

1. Sign in to your Amazon Managed Grafana workspace and select **Administration** from the left-side menu.
2. Select **Service accounts**.
3. Select **Add service account**.
4. Enter a **Display name**.
5. The display name must be unique as it determines the ID associated with the service account.
   - We recommend that you use a consistent naming convention when you name service accounts. A consistent naming convention can help you scale and maintain service accounts in the future.
   - You can change the display name at any time.
6. Choose **Create**.

> **Note**
>
> You can also create service accounts using the Amazon Managed Grafana AWS APIs. Use the CreateWorkspaceServiceAccount to create a service account programmatically.

Adding a token to a service account

A service account token is a generated random string that acts as an alternative to a password when authenticating with Grafana's HTTP API.

Prerequisite

Ensure you have permission to create and edit service accounts. By default, the organization administrator role is required to create and edit service accounts. For more information about permissions, see Using permissions.

To add a token to a service account

1. Sign in to your Grafana workspace and choose **Administration** in the left-side menu.
2. Expand the **Users and Access** menu.
3. Choose **Service accounts**.
4. Select the service account to which you want to add a token.
5. Choose **Add service account token**.
6. Enter a name for the token.
7. Select **Set expiration date** and enter an expiration date for the token.
   - The expiration date specifies how long you want the key to be valid.
   - You can set an expiration date up to 30 days in the future.
   - If you are unsure of an expiration date, we recommend that you set the token to expire after a short time, such as a few hours or less. This limits the risk associated with a token that is valid for a long time.
8. Choose **Generate token**.

> **Note**
>
> You can also create service account tokens using the Amazon Managed Grafana AWS APIs. Use the CreateWorkspaceServiceAccountToken to create a service account token programmatically.

Delete a service token

When you are done with a service token, you must delete it to remove it from your workspace. Expired, but not yet deleted, tokens count toward your quota of service account tokens.

Prerequisite

Ensure you have permission to create and edit service accounts. By default, the organization administrator role is required to create and edit service accounts. For more information about permissions, see Using permissions.

To remove a token to a service account

1. Sign in to your Grafana workspace and choose **Administration** in the left-side menu.
2. Expand the **Users and Access** menu.
3. Choose **Service accounts**.
4. Select the service account from which you want to delete a token.
5. In the list of tokens, select the red icon with an **x** next to the service account token you wish to delete.
6. Select **Delete**.

Your token is deleted.

> **Note**
>
> You can also delete service account tokens using the Amazon Managed Grafana AWS APIs. Use the DeleteWorkspaceServiceAccountToken to delete a service account token programmatically.

Assign roles to a service account

You can assign roles to a Grafana service account to control access for the associated service account tokens. You can assign roles to a service account using the Grafana UI or via the API.

Prerequisite

Ensure you have permission to create and edit service accounts. By default, the organization administrator role is required to create and edit service accounts. For more information about permissions, see Using permissions.

To assign a role to a service account

1. Sign in to Grafana and choose **Administration** in the left-side menu.
2. Choose **Service accounts**.
3. Select the service account to which you want to assign a role. As an alternative, find the service account in the list view.
4. Assign a role using the role picker to update.

New API structure

Starting in Grafana version 12, Grafana HTTP APIs are transitioning to a new standardized API structure with consistent versioning. The new APIs follow a Kubernetes-style path format:

```
/apis/<group>/<version>/namespaces/default/<resource>[/<name>]
```

Where:

- **group** – Organizes related functionality into logical collections. For example, `playlist.grafana.app` for playlist operations.
- **version** – Uses semantic versioning with stability levels: `v1alpha1` (experimental), `v1beta1` (pre-release), and `v1` (GA, stable with backward compatibility guarantees).
- **resource** – The resource type, such as `playlists` or `dashboards`.
- **name** – The unique identifier for a specific resource instance. Used for Get, Update, and Delete operations.

API responses follow a standardized format with `kind`, `apiVersion`, `metadata`, and `spec` fields.

For more information about the new API structure, see New API Structure in the *Grafana Labs documentation*.

Annotations API

Use the Annotations API to create, update, delete, and work with annotations in the Amazon Managed Grafana workspace.

Annotations are saved in the workspace's Grafana database (sqlite, mysql or postgres). Annotations can be global annotations that can be shown on any dashboard by configuring an annotation data source. Annotations are filtered by tags. Or they can be tied to a panel on a dashboard, and displayed only on that panel.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Find annotations

```
GET /api/annotations?from=1506676478816&to=1507281278816&tags=tag1&tags=tag2&limit=100
```

Example request

```
GET /api/annotations?from=1506676478816&to=1507281278816&tags=tag1&tags=tag2&limit=100
 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Query parameters:

- **from**— (Optional) Epoch datetime in milliseconds.
- **to**— (Optional) Epoch datetime in milliseconds.
- **limit**— (Optional) Maximum number of results returned. The default is 100.
- **alertid**— (Optional) FInd annotations for the specified alert.
- **dashboardId**— (Optional) Find annotations that are scoped to the specified dashboard.
- **panelId**— (Optional) Find annotations that are scoped to the specified panel.

- **userId**— (Optional) Find annotations created by the specified user.
- **type**— (Optional) Specify to return alerts or user-created annotations. Value values are `alert` and `annotation`.
- **tags**— (Optional) Use this to filter global annotations. Global annotations are annotations from an annotation data source that are not connected specifically to a dashboard or panel. To do an "AND" filtering with multiple tags, specify the tags parameter multiple times. For example, `tags=tag1&tags=tag2`. These are Grafana tags, not AWS tags.

Example response

```
HTTP/1.1 200
Content-Type: application/json
[
    {
        "id": 1124,
        "alertId": 0,
        "dashboardId": 468,
        "panelId": 2,
        "userId": 1,
        "userName": "",
        "newState": "",
        "prevState": "",
        "time": 1507266395000,
        "timeEnd": 1507266395000,
        "text": "test",
        "metric": "",
        "tags": [
            "tag1",
            "tag2"
        ],
        "data": {}
    },
    {
        "id": 1123,
        "alertId": 0,
        "dashboardId": 468,
        "panelId": 2,
        "userId": 1,
        "userName": "",
        "newState": "",
        "prevState": "",
```

```
        "time": 1507265111000,
        "text": "test",
        "metric": "",
        "tags": [
            "tag1",
            "tag2"
        ],
        "data": {}
    }
]
```

Create annotation

```
POST /api/annotations
```

Creates an annotation in the workspace's Grafana database. The `dashboardId` and `panelId` fields are optional. If they are not specified, a global annotation is created and can be queried in any dashboard that adds the Grafana annotations data source. When creating a region annotation, be sure to include the `timeEnd` property.

The format for `time` and `timeEnd` should be epoch numbers in millisecond resolution.

Example request

```
POST /api/annotations HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "dashboardId":468,
  "panelId":1,
  "time":1507037197339,
  "timeEnd":1507180805056,
  "tags":["tag1","tag2"],
  "text":"Annotation Description"
}
```

Example response

```
HTTP/1.1 200
```

```
Content-Type: application/json

{
    "message":"Annotation added",
    "id": 1,
}
```

Create annotation in graphite format

```
POST /api/annotations/graphite
```

Creates an annotation by using a Graphite-compatible event format. The `when` and `data` fields are optional. If `when` is not specified, the current time is used as annotation's timestamp. The `tags` field can also be in prior to Graphite 0.10.0 format (string with multiple tags being separated by a space).

Example request

```
POST /api/annotations/graphite HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "what": "Event - deploy",
  "tags": ["deploy", "production"],
  "when": 1467844481,
  "data": "deploy of master branch happened at Wed Jul 6 22:34:41 UTC 2016"
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
    "message":"Graphite annotation added",
    "id": 1
}
```

Update annotation

```
PUT /api/annotations/:id
```

pdates all properties of an annotation that matches the specified id. To only update certain properties, use the Patch Annotation operation.

Example request

```
PUT /api/annotations/1141 HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
Content-Type: application/json

{
  "time":1507037197339,
  "timeEnd":1507180805056,
  "text":"Annotation Description",
  "tags":["tag3","tag4","tag5"]
}
```

Example response:

```
HTTP/1.1 200
Content-Type: application/json

{
    "message":"Annotation updated"
}
```

Patch annotation

```
PATCH /api/annotations/:id
```

Updates one or more properties of an annotation that matches the specified id. This operation currently supports updating the `text`, `tags`, `time`, and `timeEnd` properties.

Example request:

```
PATCH /api/annotations/1145 HTTP/1.1
```

```
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
Content-Type: application/json

{
   "text":"New Annotation Description",
   "tags":["tag6","tag7","tag8"]
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
    "message":"Annotation patched"
}
```

Delete annotation by Id

```
DELETE /api/annotations/:id
```

Deletes the annotation that matches the specified Id.

Example request

```
DELETE /api/annotations/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
    "message":"Annotation deleted"
}
```

Alerting Notification Channels API

Use the Alerting Notification Channels API to create, update, delete, and retrieve notification channels.

The identifier (id) of a notification channel is an auto-incrementing numeric value and is only unique per workspace. The unique identifier (uid) of a notification channel can be used to uniquely identify a folder between multiple workspaces. It's automatically generated if you don't provide one when you create a notification channel. The uid allows having consistent URLs for accessing notification channels and when synchronizing notification channels between multiple Amazon Managed Grafana workspaces.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get all notification channels

Returns all notification channels that the authenticated user has permission to view.

```
GET /api/alert-notifications
```

Example request

```
GET /api/alert-notifications HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 1,
    "uid": "sns-uid",
```

```
    "name": "test",
    "type": "sns",
    "isDefault": false,
    "sendReminder": false,
    "disableResolveMessage": false,
    "frequency": "",
    "created": "2023-09-08T19:57:56Z",
    "updated": "2023-09-08T19:57:56Z",
    "settings": {
      "authProvider": "default",
      "autoResolve": true,
      "httpMethod": "POST",
      "messageFormat": "json",
      "severity": "critical",
      "topic": "<SNS-TOPIC-ARN>",
      "uploadImage": false
    },
    "secureFields": {}
  }
]
```

Get all notification channels (lookup)

Returns all notification channels, but with less detailed information. Accessible by any authenticated user and is mainly used to provide alert notification channels in the Grafana workspace console UI when configuring alert rules.

```
GET /api/alert-notifications/lookup
```

Example request

```
GET /api/alert-notifications/lookup HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
```

```
[
  {
    "id": 1,
    "uid": "sns-uid",
    "name": "test",
    "type": "sns",
    "isDefault": false
  },
  {
    "id": 2,
    "uid": "slack-uid",
    "name": "Slack",
    "type": "slack",
    "isDefault": false
  }
]
```

Get all notification channels by UID

```
GET /api/alert-notifications/uid/:uid
```

Example request

```
GET /api/alert-notifications/uid/sns-uid HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "sns-uid",
  "name": "test",
  "type": "sns",
  "isDefault": false,
  "sendReminder": false,
  "disableResolveMessage": false,
  "frequency": "",
```

```
  "created": "2023-09-08T19:57:56Z",
  "updated": "2023-09-08T19:57:56Z",
  "settings": {
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
    "severity": "critical",
    "topic": "<SNS-TOPIC-ARN>",
    "uploadImage": false
  },
  "secureFields": {}
}
```

Get all notification channels by Id

```
GET /api/alert-notifications/:id
```

Example request

```
GET /api/alert-notifications/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "sns-uid",
  "name": "test",
  "type": "sns",
  "isDefault": false,
  "sendReminder": false,
  "disableResolveMessage": false,
  "frequency": "",
  "created": "2023-09-08T19:57:56Z",
  "updated": "2023-09-08T19:57:56Z",
  "settings": {
```

```
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
    "severity": "critical",
    "topic": "<SNS-TOPIC-ARN>",
    "uploadImage": false
  },
  "secureFields": {}
}
```

Create notification channel

To see what notification channels are supported by Amazon Managed Grafana, see the list of supported notifiers in Working with contact points.

In Grafana version 12, use the Alerting Provisioning API to create contact points. The legacy `/api/alert-notifications` endpoint is no longer available.

Example request

```
POST /api/v1/provisioning/contact-points HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890

{
  "uid": "new-sns-uid",
  "name": "sns alert notification",
  "type": "sns",
  "isDefault": false,
  "sendReminder": false,
  "settings": {
    "topic_arn": "<SNS-TOPIC-ARN>",
    "subject": "<SUBJECT>"
  }
}
```

Update notification channel by UID

```
PUT /api/alert-notifications/uid/:uid
```

Example request

```
PUT /api/alert-notifications/uid/sns-uid HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890

{
  "uid": "sns-uid", // optional
  "name": "sns alert notification",  //Required
  "type":  "sns", //Required
  "isDefault": false,
  "sendReminder": true,
  "frequency": "15m",
  "settings": {
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
    "severity": "critical",
    "topic": "<SNS-TOPIC-ARN>",
    "uploadImage": false
  }
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "sns-uid",
  "name": "sns alert notification",
  "type": "sns",
  "isDefault": false,
  "sendReminder": true,
  "frequency": "15m",
  "settings": {
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
```

```
    "severity": "critical",
    "topic": "<SNS-TOPIC-ARN>",
    "uploadImage": false
  },
  "created": "2017-01-01 12:34",
  "updated": "2017-01-01 12:34"
}
```

Update notification channel by Id

```
PUT /api/alert-notifications/:id
```

Example request

```
PUT /api/alert-notifications/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890

{
  "id": 1,
  "uid": "sns-uid", // optional
  "name": "sns alert notification",  //Required
  "type":  "sns", //Required
  "isDefault": false,
  "sendReminder": true,
  "frequency": "15m",
  "settings": {
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
    "severity": "critical",
    "topic": "<SNS-TOPIC-ARN>",
    "uploadImage": false
  }
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
```

```
{
  "id": 1,
  "uid": "sns-uid",
  "name": "sns alert notification",
  "type": "sns",
  "isDefault": false,
  "sendReminder": true,
  "frequency": "15m",
  "settings": {
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
    "severity": "critical",
    "topic": "<SNS-TOPIC-ARN>",
    "uploadImage": false
  },
  "created": "2017-01-01 12:34",
  "updated": "2017-01-01 12:34"
}
```

Delete notification channel by UID

```
DELETE /api/alert-notifications/uid/:uid
```

Example request

```
DELETE /api/alert-notifications/uid/sns-uid HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "message": "Notification deleted"
}
```

Delete notification channel by Id

```
DELETE /api/alert-notifications/:id
```

Example request

```
DELETE /api/alert-notifications/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "message": "Notification deleted"
}
```

Test notification channel

Sends a test notification message for the given notification channel type and settings.

```
POST /api/alert-notifications/test
```

Example request

```
POST /api/alert-notifications/test HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890

{
  "type":  "sns",
  "settings": {
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
    "severity": "critical",
```

```
    "topic": "<SNS-TOPIC-ARN>",
    "uploadImage": false
  }
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "message": "Test notification sent"
}
```

Dashboard API

Use the Dashboard API to create, update, delete, and work with dashboards in the Amazon Managed Grafana workspace.

The identifier (id) of a dashboard is an auto-incrementing numeric value and is only unique per workspace. The unique identifier (uid) of a dashboard can be used to uniquely identify a dashboard between multiple Amazon Managed Grafana workspaces. It's automatically generated if you don't provide one when you create a dashboard. The uid allows having consistent URLs for accessing dashboards and when synchronizing dashboards between multiple workspaces. The use of the uid means that changing the title of a dashboard does not break any bookmarked links to that dashboard.

The uid can have a maximum length of 40 characters.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Create/Update dashboard

```
POST /api/dashboards/db
```

Creates a new dashboard or updates an existing dashboard.

Example request to create a new dashboard

```
POST /api/dashboards/db HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "dashboard": {
    "id": null,
    "uid": null,
    "title": "Production Overview",
    "tags": [ "templated" ],
    "timezone": "browser",
    "schemaVersion": 16,
    "version": 0,
    "refresh": "25s"
  },
  "folderId": 0,
  "folderUid": "l3KqBxCMz",
  "message": "Made changes to xyz",
  "overwrite": false
}
```

JSON body schema:

- **dashboard**— The complete dashboard model. Use null to create a new dashboard.
- **dashboard.id**— Use null to create a new dashboard.
- **dashboard.uid**— Optional unique identifer when you use this to create a new dashboard. If null, a new uid is generated.
- **folderid**— The id of the folder to save the dashboard in.
- **folderUid**— The Uid of the folder to save the dashboard in. Overrides the value of `folderid`
- **overwrite**— Specify `true` to overwrite an existing dashboard with a newer version, same dashboard title in folder or same dashboard uid.
- **message**— Set a commit message for the version history.
- **refresh**— Set the dashboard refresh interval. If this is lower than the minimum refresh interval, it is ignored and the minimum refresh interval is used.

To add or update an alert rule for a dashboard panel, declare a `dashboard.panels.alert` block.

Example request to update a dashboard alert rule

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 78

{
 "dashboard":  {
        "id": 104,
        "panels": [
            {
                "alert": {
                    "alertRuleTags": {},
                    "conditions": [
                        {
                            "evaluator": {
                                "params": [
                                    25
                                ],
                                "type": "gt"
                            },
                            "operator": {
                                "type": "and"
                            },
                            "query": {
                                "params": [
                                    "A",
                                    "5m",
                                    "now"
                                ]
                            },
                            "reducer": {
                                "params": [],
                                "type": "avg"
                            },
                            "type": "query"
                        }
                    ],
                    "executionErrorState": "alerting",
                    "for": "5m",
                    "frequency": "1m",
                    "handler": 1,
                    "name": "Panel Title alert",
                    "noDataState": "no_data",
```

```
            "notifications": []
        },
        "aliasColors": {},
        "bars": false,
        "dashLength": 10,
        "dashes": false,
        "datasource": null,
        "fieldConfig": {
            "defaults": {
                "custom": {}
            },
            "overrides": []
        },
        "fill": 1,
        "fillGradient": 0,
        "gridPos": {
            "h": 9,
            "w": 12,
            "x": 0,
            "y": 0
        },
        "hiddenSeries": false,
        "id": 2,
        "legend": {
            "avg": false,
            "current": false,
            "max": false,
            "min": false,
            "show": true,
            "total": false,
            "values": false
        },
        "lines": true,
        "linewidth": 1,
        "nullPointMode": "null",
        "options": {
            "dataLinks": []
        },
        "percentage": false,
        "pointradius": 2,
        "points": false,
        "renderer": "flot",
        "seriesOverrides": [],
        "spaceLength": 10,
```

```
                "stack": false,
                "steppedLine": false,
                "targets": [
                    {
                        "refId": "A",
                        "scenarioId": "random_walk"
                    }
                ],
                "thresholds": [
                    {
                        "colorMode": "critical",
                        "fill": true,
                        "line": true,
                        "op": "gt",
                        "value": 50
                    }
                ],
                "timeFrom": null,
                "timeRegions": [],
                "timeShift": null,
                "title": "Panel Title",
                "tooltip": {
                    "shared": true,
                    "sort": 0,
                    "value_type": "individual"
                },
                "type": "graph",
                "xaxis": {
                    "buckets": null,
                    "mode": "time",
                    "name": null,
                    "show": true,
                    "values": []
                },
                "yaxes": [
                    {
                        "format": "short",
                        "label": null,
                        "logBase": 1,
                        "max": null,
                        "min": null,
                        "show": true
                    },
                    {
```

```
                        "format": "short",
                        "label": null,
                        "logBase": 1,
                        "max": null,
                        "min": null,
                        "show": true
                    }
                ],
                "yaxis": {
                    "align": false,
                    "alignLevel": null
                }
            }
        ],
        "title": "Update alert rule via API",
        "uid": "dHEquNzGz",
        "version": 1
    }
}
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 78

{
  "id":      1,
  "uid":     "cIBgcSjkk",
  "url":     "/d/cIBgcSjkk/production-overview",
  "status":  "success",
  "version": 1,
  "slug":    "production-overview" //deprecated in Grafana v5.0
}
```

Status Codes:

- **200**— Created
- **400**— Error such as invalid JSON, invalid or missing fields
- **401**— Unauthorized
- **403**— Access denied

- **412**— Precondition failed

The **412** status code is used to explain why the dashboard can't be created.

- The dashboard has been changed by someone else `status=version-mismatch`
- A dashboard with the same name in the folder already exists `status=name-exists`
- A dashboard with the same uid already exists `status=name-exists`
- The dashboard belongs to plugin `plugin title` `status=plugin-dashboard`

The response body has the following properties. If another dashboard has the same title, the `status` value is `name-exists`.

```
HTTP/1.1 412 Precondition Failed
Content-Type: application/json; charset=UTF-8
Content-Length: 97

{
  "message": "The dashboard has been changed by someone else",
  "status": "version-mismatch"
}
```

Get dashboard by uid

```
GET /api/dashboards/uid/:uid
```

Returns the dashboard matching the uid. The metadata returned might contain information about the UID of the folder that contains the dashboard.

Example request

```
GET /api/dashboards/uid/cIBgcSjkk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
```

```
Content-Type: application/json

{
  "dashboard": {
    "id": 1,
    "uid": "cIBgcSjkk",
    "title": "Production Overview",
    "tags": [ "templated" ],
    "timezone": "browser",
    "schemaVersion": 16,
    "version": 0
  },
  "meta": {
    "isStarred": false,
    "url": "/d/cIBgcSjkk/production-overview",
    "folderId": 2,
    "folderUid": "l3KqBxCMz",
    "slug": "production-overview" //deprecated in Grafana v5.0
  }
}
```

Status Codes:

- **200**— Found
- **401**— Unauthorized
- **403**— Access denied
- **404**— Not found

Delete dashboard by uid

```
DELETE /api/dashboards/uid/:uid
```

Deletes the dashboard matching the uid.

Example request

```
DELETE /api/dashboards/uid/cIBgcSjkk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "title": "Production Overview",
  "message": "Dashboard Production Overview deleted",
  "id": 2
}
```

Status Codes:

- **200**— Deleted
- **401**— Unauthorized
- **403**— Access denied
- **404**— Not found

Gets the home dashboard

```
GET /api/dashboards/home
```

Returns the home dashboard.

Example request

```
GET /api/dashboards/home HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "dashboard": {
    "editable":false,
```

```
    "hideControls":true,
    "nav":[
      {
        "enable":false,
        "type":"timepicker"
      }
    ],
    "style":"dark",
    "tags":[],
    "templating":{
      "list":[
      ]
    },
    "time":{
    },
    "timezone":"browser",
    "title":"Home",
    "version":5
  },
  "meta": {
    "isHome":true,
    "canSave":false,
    "canEdit":false,
    "canStar":false,
    "url":"",
    "expires":"0001-01-01T00:00:00Z",
    "created":"0001-01-01T00:00:00Z"
  }
}
```

Get dashboard tags

```
GET /api/dashboards/tags
```

Returns all tags of dashboards.

Example request

```
GET /api/dashboards/tags HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "term":"tag1",
    "count":1
  },
  {
    "term":"tag2",
    "count":4
  }
]
```

Dashboard Permissions API

Use the Dashboard Permissions API to update or retrieve the permissions for a dashboard.

Permissions with `dashboardId=-1` are the default permissions for users with the Viewer and Editor roles. Permissions can be set for a user, a team or a role (Viewer or Editor). Permissions cannot be set for Admins - they always have access to everything.

The permission levels for the `permission` field are as follows:

- 1 = View
- 2 = Edit
- 4 = Admin

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get permissions for a dashboard

```
GET /api/dashboards/id/:dashboardId/permissions
```

Gets all existing permissions for the dashboard with the given `dashboardId`.

Example request

```
GET /api/dashboards/id/1/permissions HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 551

[
  {
    "id": 1,
    "dashboardId": -1,
    "created": "2017-06-20T02:00:00+02:00",
    "updated": "2017-06-20T02:00:00+02:00",
    "userId": 0,
    "userLogin": "",
    "userEmail": "",
    "teamId": 0,
    "team": "",
    "role": "Viewer",
    "permission": 1,
    "permissionName": "View",
    "uid": "",
    "title": "",
    "slug": "",
    "isFolder": false,
    "url": ""
  },
  {
    "id": 2,
    "dashboardId": -1,
    "created": "2017-06-20T02:00:00+02:00",
    "updated": "2017-06-20T02:00:00+02:00",
    "userId": 0,
    "userLogin": "",
    "userEmail": "",
```

```
    "teamId": 0,
    "team": "",
    "role": "Editor",
    "permission": 2,
    "permissionName": "Edit",
    "uid": "",
    "title": "",
    "slug": "",
    "isFolder": false,
    "url": ""
  }
]
```

Status Codes:

- **200**— OK
- **401**— Unauthorized
- **403**— Access denied
- **404**— Dashboard not found

Update permissions for a dashboard

```
POST /api/dashboards/id/:dashboardId/permissions
```

Updates the permissions for a dashboard. This operation removes existing permissions if they are not included in the request.

Example request

```
POST /api/dashboards/id/1/permissions
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "items": [
    {
      "role": "Viewer",
      "permission": 1
    },
```

```
    {
      "role": "Editor",
      "permission": 2
    },
    {
      "teamId": 1,
      "permission": 1
    },
    {
      "userId": 11,
      "permission": 4
    }
  ]
}
```

JSON body schema:

- **items**— The permission items to add or update. Existing items that are omitted from the list are removed.

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Dashboard permissions updated"}
```

Status Codes:

- **200**— Created
- **401**— Unauthorized
- **403**— Access denied
- **404**— Dashboard not found

Dashboard Versions API

Use the Dashboard Versions API to retrieve dashboard versions and restore a dashboard to a specified version.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get all dashboard versions

```
GET /api/dashboards/uid/:uid/versions
```

Gets all existing dashboard versions for the dashboard with the given `uid`.

Query parameters:

- **limit**— Maximum number of results to return.
- **start**— Version to start from when returning queries.

Example request

```
GET /api/dashboards/id/1/versions?limit=2?start=0 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 428

[
  {
    "id": 2,
    "dashboardId": 1,
    "parentVersion": 1,
    "restoredFrom": 0,
    "version": 2,
    "created": "2017-06-08T17:24:33-04:00",
    "createdBy": "admin",
    "message": "Updated panel title"
```

```
  },
  {
    "id": 1,
    "dashboardId": 1,
    "parentVersion": 0,
    "restoredFrom": 0,
    "version": 1,
    "created": "2017-06-08T17:23:33-04:00",
    "createdBy": "admin",
    "message": "Initial save"
  }
]
```

Status Codes:

- **200**— OK
- **400**— Errors
- **401**— Unauthorized
- **404**— Dashboard version not found

Get dashboard version

```
GET /api/dashboards/uid/:uid/versions/:id
```

Get the dashboard version with the given id, for the dashboard with the given `dashboardId`.

Example request

```
GET /api/dashboards/id/1/versions/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 1300

{
```

```
  "id": 1,
  "dashboardId": 1,
  "parentVersion": 0,
  "restoredFrom": 0,
  "version": 1,
  "created": "2017-04-26T17:18:38-04:00",
  "message": "Initial save",
  "data": {
    "annotations": {
      "list": [

      ]
    },
    "editable": true,
    "gnetId": null,
    "graphTooltip": 0,
    "hideControls": false,
    "id": 1,
    "links": [

    ],
    "rows": [
      {
        "collapse": false,
        "height": "250px",
        "panels": [

        ],
        "repeat": null,
        "repeatIteration": null,
        "repeatRowId": null,
        "showTitle": false,
        "title": "Dashboard Row",
        "titleSize": "h6"
      }
    ],
    "schemaVersion": 14,
    "style": "dark",
    "tags": [

    ],
    "templating": {
      "list": [
```

```
      ]
    },
    "time": {
      "from": "now-6h",
      "to": "now"
    },
    "timepicker": {
      "refresh_intervals": [
        "5s",
        "10s",
        "30s",
        "1m",
        "5m",
        "15m",
        "30m",
        "1h",
        "2h",
        "1d"
      ],
      "time_options": [
        "5m",
        "15m",
        "1h",
        "6h",
        "12h",
        "24h",
        "2d",
        "7d",
        "30d"
      ]
    },
    "timezone": "browser",
    "title": "test",
    "version": 1
  },
  "createdBy": "admin"
}
```


Status Codes:

- **200**— OK
- **401**— Unauthorized
- **404**— Dashboard version not found

Restore dashboard

```
POST /api/dashboards/uid/:uid/restore
```

Restores a dashboard to the dashboard version that you specify.

Example request

```
POST /api/dashboards/id/1/restore
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "version": 1
}
```

JSON body schema:

- **version**— The dashboard version to restore to.

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 67

{
  "slug": "my-dashboard",
  "status": "success",
  "version": 3
}
```

JSON response body schema:

- **slug**— The URL-friendly slug of the dashboard's title.
- **status**— Whether the restore was successful or not.
- **version**— The new dashboard version following the restore.

Status Codes:

- **200**— Created
- **401**— Unauthorized
- **404**— Dashboard or dashboard version not found
- **500**— Internal server error (indicates issue retrieving dashboard tags from database)

Example error response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json; charset=UTF-8
Content-Length: 46

{
  "message": "Dashboard version not found"
}
```

JSON response body schema:

- **message**— A message explaining the reason for the failure.

Compare dashboard versions

```
POST /api/dashboards/calculate-diff
```

Compares two dashboard versions by calculating the JSON diff of them.

Example request

```
POST /api/dashboards/calculate-diff HTTP/1.1
Accept: text/html
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "base": {
    "dashboardId": 1,
    "version": 1
  },
  "new": {
```

```
    "dashboardId": 1,
    "version": 2
  },
  "diffType": "json"
}
```

JSON body schema:

- **base**— An object representing the base dashboard version.
- **new**— An object representing the new dashboard version.
- **difftype**— The type of diff to return. Valid values are `json` and `basic`.

Example response (JSON diff)

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8

<p id="v12-l1" class="diff-line diff-json-same">
  <!-- Diff omitted -->
</p>
```

The response is a textual representation of the diff, with the dashboard values being in JSON, similar to the diffs seen on sites like GitHub or GitLab.

Status Codes:

- **200**— OK
- **200**— Bad request, invalid JSON sent
- **401**— Unauthorized
- **404**— Not found

Example response (Basic diff)

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8

<div class="diff-group">
  <!-- Diff omitted -->
```

```
</div>
```

The response is a summary of the changes, derived from the diff between the two JSON objects.

Status Codes:

- **200**— OK
- **200**— Bad request, invalid JSON sent
- **401**— Unauthorized
- **404**— Not found

Data Source API

Use the Data Source API to create, update, delete, and work with data sources in the Amazon Managed Grafana workspace.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get all data sources

```
GET /api/datasources
```

Example request

```
GET /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
```

```
[
   {
     "id": 1,
     "orgId": 1,
     "uid": "H8joYFVGz"
     "name": "datasource_elastic",
     "type": "elasticsearch",
     "typeLogoUrl": "public/app/plugins/datasource/elasticsearch/img/
elasticsearch.svg",
     "access": "proxy",
     "url": "http://mydatasource.com",
     "password": "",
     "user": "",
     "database": "grafana-dash",
     "basicAuth": false,
     "isDefault": false,
     "jsonData": {
         "esVersion": 5,
         "logLevelField": "",
         "logMessageField": "",
         "maxConcurrentShardRequests": 256,
         "timeField": "@timestamp"
     },
     "readOnly": false
   }
]
```

Get a single data source by UID

```
GET /api/datasources/uid/:uid
```

Example request

```
GET /api/datasources/uid/kLtEtcRGk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
```

```
Content-Type: application/json

{
  "id": 1,
  "uid": "kLtEtcRGk",
  "orgId": 1,
  "name": "test_datasource",
  "type": "graphite",
  "typeLogoUrl": "",
  "access": "proxy",
  "url": "http://mydatasource.com",
  "password": "",
  "user": "",
  "database": "",
  "basicAuth": false,
  "basicAuthUser": "",
  "basicAuthPassword": "",
  "withCredentials": false,
  "isDefault": false,
  "jsonData": {
    "graphiteType": "default",
    "graphiteVersion": "1.1"
  },
  "secureJsonFields": {},
  "version": 1,
  "readOnly": false
}
```

Get a single data source by name

```
GET /api/datasources/name/:name
```

Example request

```
GET /api/datasources/name/test_datasource HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
```

```
Content-Type: application/json

{
  "id": 1,
  "uid": "kLtEtcRGk",
  "orgId": 1,
  "name": "test_datasource",
  "type": "graphite",
  "typeLogoUrl": "",
  "access": "proxy",
  "url": "http://mydatasource.com",
  "password": "",
  "user": "",
  "database": "",
  "basicAuth": false,
  "basicAuthUser": "",
  "basicAuthPassword": "",
  "withCredentials": false,
  "isDefault": false,
  "jsonData": {
    "graphiteType": "default",
    "graphiteVersion": "1.1"
  },
  "secureJsonFields": {},
  "version": 1,
  "readOnly": false
}
```

Get data source Id by name

```
GET /api/datasources/id/:name
```

Example request

```
GET /api/datasources/id/test_datasource HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
```

```
Content-Type: application/json

{
  "id":1
}
```

Create a data source

```
POST /api/datasources
```

Example Graphite request

```
POST /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name":"test_datasource",
  "type":"graphite",
  "url":"http://mydatasource.com",
  "access":"proxy",
  "basicAuth":false
}
```

Example Graphite response

```
HTTP/1.1 200
Content-Type: application/json

{
  "datasource": {
    "id": 1,
    "orgId": 1,
    "name": "test_datasource",
    "type": "graphite",
    "typeLogoUrl": "",
    "access": "proxy",
    "url": "http://mydatasource.com",
    "password": "",
    "user": "",
    "database": "",
```

```
    "basicAuth": false,
    "basicAuthUser": "",
    "basicAuthPassword": "",
    "withCredentials": false,
    "isDefault": false,
    "jsonData": {},
    "secureJsonFields": {},
    "version": 1,
    "readOnly": false
  },
  "id": 1,
  "message": "Datasource added",
  "name": "test_datasource"
}
```

Note

When you define `password` and `basicAuthPassword` within `secureJsonData`, Amazon Managed Grafana encryptes them securely as an encrypted blob in the database. The response then lists the encrypte fields in `secureJsonFields`.

Example Graphite request with basic auth enabled

```
POST /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "test_datasource",
  "type": "graphite",
  "url": "http://mydatasource.com",
  "access": "proxy",
  "basicAuth": true,
  "basicAuthUser": "basicuser",
  "secureJsonData": {
    "basicAuthPassword": "basicpassword"
  }
}
```

Example response with basic auth enabled

```
HTTP/1.1 200
Content-Type: application/json

{
  "datasource": {
    "id": 1,
    "orgId": 1,
    "name": "test_datasource",
    "type": "graphite",
    "typeLogoUrl": "",
    "access": "proxy",
    "url": "http://mydatasource.com",
    "password": "",
    "user": "",
    "database": "",
    "basicAuth": true,
    "basicAuthUser": "basicuser",
    "basicAuthPassword": "",
    "withCredentials": false,
    "isDefault": false,
    "jsonData": {},
    "secureJsonFields": {
      "basicAuthPassword": true
    },
    "version": 1,
    "readOnly": false
  },
  "id": 102,
  "message": "Datasource added",
  "name": "test_datasource"
}
```

Example CloudWatch request

```
POST /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "test_datasource",
  "type": "cloudwatch",
  "url": "http://monitoring.us-west-1.amazonaws.com",
```

```
  "access": "proxy",
  "jsonData": {
    "authType": "keys",
    "defaultRegion": "us-west-1"
  },
  "secureJsonData": {
    "accessKey": "Ol4pIDpeKSA6XikgOl4p",
    "secretKey": "dGVzdCBrZXkgYmxlYXNlIGRvbid0IHN0ZWFs"
  }
}
```

Update a data source by UID

```
PUT /api/datasources/uid/:uid
```

Updates the data source matching the given UID.

Example request

```
PUT /api/datasources/uid/kLtEtcRGk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "test_datasource",
  "type": "graphite",
  "url": "http://mydatasource.com",
  "access": "proxy",
  "basicAuth": false,
  "isDefault": false
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "datasource": {
    "id": 1,
```

```
    "uid": "kLtEtcRGk",
    "orgId": 1,
    "name": "test_datasource",
    "type": "graphite",
    "access": "proxy",
    "url": "http://mydatasource.com",
    "basicAuth": false,
    "isDefault": false
  },
  "name": "test_datasource"
}
```

Delete data source by UID

```
DELETE /api/datasources/uid/:uid
```

Example request

```
DELETE /api/datasources/uid/kLtEtcRGk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Data source deleted"}
```

Delete data source by name

```
DELETE /api/datasources/name/:datasourceName
```

Example request

```
DELETE /api/datasources/name/test_datasource HTTP/1.1
Accept: application/json
Content-Type: application/json
```

```
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "message":"Data source deleted",
  "id": 1
}
```

Data source proxy calls

```
GET /api/datasources/proxy/:datasourceId/*
```

Proxies all calls to the actual data source.

Query data sources

```
POST /api/ds/query
```

Queries a data source having a backend implementation. Most built-in data sources have backend implementation.

Example request

```
POST /api/ds/query HTTP/1.1
Accept: application/json
Content-Type: application/json

{
   "queries":[
      {
         "refId":"A",
         "scenarioId":"csv_metric_values",
         "datasource":{
            "uid":"PD8C576611E62080A"
         },
         "format": "table",
         "maxDataPoints":1848,
```

```
         "intervalMs":200,
         "stringInput":"1,20,90,30,5,0"
      }
   ],
   "from":"now-5m",
   "to":"now"
}
```

JSON body schema:

- **from/to**— Specifies the time range for the queries. The time can be either epoch timestamps in milliseconds or relative using Grafana time units. For example, `now-5m`.
- **queries**— Specifies one or more queries. Must contain at least 1.
- **queries.datasource.uid**— Specifies the UID of data source to be queried. Each query in the request must have a unique `datasource`.
- **queries.refId**— Specifies an identifier of the query. Defaults to “A”.
- **queries.format**— Specifies the format the data should be returned in. Valid options are `time_series` or `table` depending on the data source.
- **queries.maxDataPoints**— Species the maximum amount of data points that a dashboard panel can render. Defaults to 100.
- **queries.intervalMs**— Specifies the time series time interval in milliseconds. Defaults to 1000.

In addition, specific properties of each data source should be added in a request (for example **queries.stringInput** as shown in the request above). To better understand how to form a query for a certain data source, use the Developer Tools in your browser of choice and inspect the HTTP requests being made to `/api/ds/query`.

Example test data source time series query response

```
HTTP/1.1 200
Content-Type: application/json

{
  "results": {
    "A": {
      "frames": [
        {
          "schema": {
```

```
            "refId": "A",
            "fields": [
              {
                "name": "time",
                "type": "time",
                "typeInfo": {
                  "frame": "time.Time"
                }
              },
              {
                "name": "A-series",
                "type": "number",
                "typeInfo": {
                  "frame": "int64",
                  "nullable": true
                }
              }
            ]
          },
          "data": {
            "values": [
              [1644488152084, 1644488212084, 1644488272084, 1644488332084,
 1644488392084, 1644488452084],
              [1, 20, 90, 30, 5, 0]
            ]
          }
        }
      ]
    }
  }
}
```

Update an existing data source (deprecated)

> **Important**
>
> This endpoint is deprecated. Use Update a data source by UID instead.

```
PUT /api/datasources/:datasourceId
```

Example request

```
PUT /api/datasources/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "id":1,
  "orgId":1,
  "name":"test_datasource",
  "type":"graphite",
  "access":"proxy",
  "url":"http://mydatasource.com",
  "password":"",
  "user":"",
  "database":"",
  "basicAuth":true,
  "basicAuthUser":"basicuser",
  "secureJsonData": {
    "basicAuthPassword": "basicpassword"
  },
  "isDefault":false,
  "jsonData":null
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "datasource": {
    "id": 1,
    "orgId": 1,
    "name": "test_datasource",
    "type": "graphite",
    "typeLogoUrl": "",
    "access": "proxy",
    "url": "http://mydatasource.com",
    "password": "",
    "user": "",
    "database": "",
    "basicAuth": true,
    "basicAuthUser": "basicuser",
```

```
    "basicAuthPassword": "",
    "withCredentials": false,
    "isDefault": false,
    "jsonData": {},
    "secureJsonFields": {
      "basicAuthPassword": true
    },
    "version": 1,
    "readOnly": false
  },
  "id": 102,
  "message": "Datasource updated",
  "name": "test_datasource"
}
```

Note

We recommend that you define `password` and `basicAuthPassword` within `secureJsonData` so that they are stored securely as an encrypted blob in the database. The response then lists the encrypte fields in `secureJsonFields`.

Get a single data source by Id (deprecated)

Important

This endpoint is deprecated. Use Get a single data source by UID instead.

```
GET /api/datasources/:datasourceId
```

Example request

```
GET /api/datasources/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "kLtEtcRGk",
  "orgId": 1,
  "name": "test_datasource",
  "type": "graphite",
  "typeLogoUrl": "",
  "access": "proxy",
  "url": "http://mydatasource.com",
  "password": "",
  "user": "",
  "database": "",
  "basicAuth": false,
  "basicAuthUser": "",
  "basicAuthPassword": "",
  "withCredentials": false,
  "isDefault": false,
  "jsonData": {
    "graphiteType": "default",
    "graphiteVersion": "1.1"
  },
  "secureJsonFields": {},
  "version": 1,
  "readOnly": false
}
```

Delete data source by Id (deprecated)

Important

This endpoint is deprecated. Use Delete data source by UID instead.

```
DELETE /api/datasources/:datasourceId
```

Example request

```
DELETE /api/datasources/1 HTTP/1.1
```

```
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Data source deleted"}
```

Data Source Permissions API

Use the Data Source Permissions API to enable, disable, list, add, and remove permissions for data sources.

You can set permissions for a user or a team. Permissions can't be set for Admins, because they always have access to everything.

The permission levels for the permission field are as follows:

- 1 = Query

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Enable permissions for a data source

```
POST /api/datasources/:id/enable-permissions
```

Enables permissions for the data source with the given id. No one except Org Admins are able to query the data source until permissions have been added to permit certain users or teams to query the data source.

Example request

```
POST /api/datasources/1/enable-permissions
```

```
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Datasource permissions enabled"}
```

Status Codes:

- **200**— Created
- **400**— Permissions can't be enabled, see the response body for details.
- **401**— Unauthorized
- **403**— Access denied
- **404**— Data source not found

Disable permissions for a data source

```
POST /api/datasources/:id/disable-permissions
```

Disables permissions for the data source with the given id. All existing permissions are removed and anyone is able to query the data source.

Example request

```
POST /api/datasources/1/disable-permissions
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{}
```

Example response

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Datasource permissions disabled"}
```

Status Codes:

- **200**— Ok
- **400**— Permissions can't be disabled, see the response body for details.
- **401**— Unauthorized
- **403**— Access denied
- **404**— Data source not found

Get permissions for a data source

```
GET /api/datasources/:id/permissions
```

Gets all existing permissions for the data source with the given `id`.

Example request

```
GET /api/datasources/1/permissions HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 551

{
  "datasourceId": 1,
  "enabled": true,
  "permissions":
  [
    {
      "id": 1,
```

```
      "datasourceId": 1,
      "userId": 1,
      "userLogin": "user",
      "userEmail": "user@test.com",
      "userAvatarUrl": "/avatar/46d229b033af06a191ff2267bca9ae",
      "permission": 1,
      "permissionName": "Query",
      "created": "2017-06-20T02:00:00+02:00",
      "updated": "2017-06-20T02:00:00+02:00",
    },
    {
      "id": 2,
      "datasourceId": 1,
      "teamId": 1,
      "team": "A Team",
      "teamAvatarUrl": "/avatar/46d229b033af06a191ff2267bca9ae",
      "permission": 1,
      "permissionName": "Query",
      "created": "2017-06-20T02:00:00+02:00",
      "updated": "2017-06-20T02:00:00+02:00",
    }
  ]
}
```

Status Codes:

- **200**— Ok
- **401**— Unauthorized
- **403**— Access denied
- **404**— Data source not found

Add permission for a data source

```
POST /api/datasources/:id/permissions
```

Adds a user permission for the data source with the given `id`.

Example request to add user permission

```
POST /api/datasources/1/permissions
Accept: application/json
```

```
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "userId": 1,
  "permission": 1
}
```

Example response for adding a user permission

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Datasource permission added"}
```

Example request to add team permission

```
POST /api/datasources/1/permissions
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "teamId": 1,
  "permission": 1
}
```

Example response for adding a team permission

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Datasource permission added"}
```

Status Codes:

- **200**— Ok
- **400**— Permission can't be added, see response body for details.
- **401**— Unauthorized

- **403**— Access denied
- **404**— Data source not found

Remove permission for a data source

```
DELETE /api/datasources/:id/permissions/:permissionId
```

Removes the permission with the given permissionId for the data source with the given `id`.

Example request

```
DELETE /api/datasources/1/permissions/2
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Datasource permission removed"}
```

Status Codes:

- **200**— Ok
- **401**— Unauthorized
- **403**— Access denied
- **404**— Data source not found or permission not found

External Group Synchronization API

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get external groups

```
GET /api/teams/:teamId/groups
```

Example request

```
GET /api/teams/1/groups HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk]
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "orgId": 1,
    "teamId": 1,
    "groupId": "cn=editors,ou=groups,dc=grafana,dc=org"
  }
]
```

Status Codes:

- **200**— Ok
- **401**— Unauthorized
- **403**— Access denied

Add external group

```
POST /api/teams/:teamId/groups
```

Example request

```
POST /api/teams/1/members HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk]
```

```
{
  "groupId": "cn=editors,ou=groups,dc=grafana,dc=org"
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Group added to Team"}
```

Status Codes:

- **200**— Ok
- **400**— Group is already added to this team
- **401**— Unauthorized
- **403**— Permission denied
- **404**— Team not found

Remove external group

```
DELETE /api/teams/:teamId/groups/:groupId
```

Example request

```
DELETE /api/teams/1/groups/cn=editors,ou=groups,dc=grafana,dc=org HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk]
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Team Group removed"}
```

Status Codes:

- **200**— Ok
- **401**— Unauthorized
- **403**— Permission denied
- **404**— Team not found or group not found

Folder API

Use the Folder API to work with folders in the Amazon Managed Grafana workspace.

The identifier (id) of a folder is an auto-incrementing numeric value and is only unique per workspace. The unique identifier (uid) of a folder can be used to uniquely identify a folder between multiple workspaces. It's automatically generated if you don't provide one when you create a folder. The uid allows having consistent URLs for accessing folder and when synchronizing folder between multiple Amazon Managed Grafana workspaces. The use of the uid means that changing the title of a folder does not break any bookmarked links to that folder.

The uid can have a maximum length of 40 characters.

Folders can't be nested.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

The **General** folder, with an `id` of 0, is not part of the Folder API. You can't use the Folder API to retrieve information about the general folder.

Create folder

```
POST /api/folders
```

Creates a new folder.

Example request

```
POST /api/folders HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "uid": "nErXDvCkzz",
  "title": "Department ABC"
}
```

JSON body schema:

- **uid**— Optional unique identifer. If null, a new uid is generated.
- **title**— The title for the folder.

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
  "uid": "nErXDvCkzz",
  "title": "Department ABC",
  "url": "/dashboards/f/nErXDvCkzz/department-abc",
  "hasAcl": false,
  "canSave": true,
  "canEdit": true,
  "canAdmin": true,
  "createdBy": "admin",
  "created": "2018-01-31T17:43:12+01:00",
  "updatedBy": "admin",
  "updated": "2018-01-31T17:43:12+01:00",
  "version": 1
}
```

Status Codes:

- **200**— Created
- **400**— Error such as invalid JSON, invalid or missing fields
- **401**— Unauthorized
- **403**— Access denied

Update folder

```
PUT /api/folders/:uid
```

Updates the existing folder that matches the uid.

Example request

```
PUT /api/folders/nErXDvCkzz HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "title":"Department DEF",
  "version": 1
}
```

JSON body schema:

- **uid**— Changes the unique identifer, if provided.
- **title**— The title of the folder.
- **version**— Provide the current version to be able to overwrite the folder. Not needed if `overwrite=true`.
- **overwrite**— Set to `true` to overwrite the existing folder with a newer version.

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
  "uid": "nErXDvCkzz",
  "title": "Department DEF",
  "url": "/dashboards/f/nErXDvCkzz/department-def",
  "hasAcl": false,
  "canSave": true,
  "canEdit": true,
```

```
  "canAdmin": true,
  "createdBy": "admin",
  "created": "2018-01-31T17:43:12+01:00",
  "updatedBy": "admin",
  "updated": "2018-01-31T17:43:12+01:00",
  "version": 1
}
```

Status Codes:

- **200**— Created
- **400**— Error such as invalid JSON, invalid or missing fields
- **401**— Unauthorized
- **403**— Access denied
- **404**— Folder not found
- **412**— Precondition failed

The **412** status code is used to explain why the folder can't be updated.

- The folder has been changed by someone else `status=version-mismatch`

The response body has the following properties:

```
HTTP/1.1 412 Precondition Failed
Content-Type: application/json; charset=UTF-8
Content-Length: 97

{
  "message": "The folder has been changed by someone else",
  "status": "version-mismatch"
}
```

Get all folders

```
GET /api/folders
```

Returns all folders that you have permission to view. You can control the maximum number of folders returned by using the `limit` query parameter. The default is 1000.

Example request

```
GET /api/folders?limit=10 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id":1,
    "uid": "nErXDvCkzz",
    "title": "Department ABC"
  },
  {
    "id":2,
    "uid": "k3S1cklGk",
    "title": "Department RND"
  }
]
```

Get folder by uid

```
GET /api/folders/:uid
```

Returns all folders that matches the given uid.

Example request

```
GET /api/folders/nErXDvCkzzh HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
```

```
Content-Type: application/json

{
  "id":1,
  "uid": "nErXDvCkzz",
  "title": "Department ABC",
  "url": "/dashboards/f/nErXDvCkzz/department-abc",
  "hasAcl": false,
  "canSave": true,
  "canEdit": true,
  "canAdmin": true,
  "createdBy": "admin",
  "created": "2018-01-31T17:43:12+01:00",
  "updatedBy": "admin",
  "updated": "2018-01-31T17:43:12+01:00",
  "version": 1
}
```

Status Codes:

- **200**— Found
- **401**— Unauthorized
- **403**— Access denied
- **404**— Not found

Get folder by id

```
GET /api/folders/id/:id
```

Returns the folder that matches the given id.

Example request

```
GET /api/folders/id/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
  "uid": "nErXDvCkzz",
  "title": "Department ABC",
  "url": "/dashboards/f/nErXDvCkzz/department-abc",
  "hasAcl": false,
  "canSave": true,
  "canEdit": true,
  "canAdmin": true,
  "createdBy": "admin",
  "created": "2018-01-31T17:43:12+01:00",
  "updatedBy": "admin",
  "updated": "2018-01-31T17:43:12+01:00",
  "version": 1
}
```

Status Codes:

- **200**— Found
- **401**— Unauthorized
- **403**— Access denied
- **404**— Not found

Delete folder by uid

```
DELETE /api/folders/:uid
```

Deletes the folder matching the uid, and also deletes all dashboards stored in the folder. This oepration can't be reverted.

Example request

```
DELETE /api/folders/nErXDvCkzz HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "message":"Folder deleted",
  "id": 2
}
```

Status Codes:

- **200**— Deleted
- **401**— Unauthorized
- **403**— Access denied
- **404**— Not found

Folder/Dashboard Search API

Use the FolderDashboard-Search API to search folders and dashboards in an Amazon Managed Grafana workspace.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Search folders and dashboards

```
GET /api/search/
```

Query parameters:

- **query**— Search query
- **tag**— List of tags to search for. These are Grafana tags, not AWS tags.
- **type**— The type to search for, either `dash-folder` or `dash-db`.
- **dashboardIds**— List of dashboard Id's to search for.

- **folderIds**— List of dashboard Id's to search for in dashboards.
- **starred**— Flag to specify that only starred dashboards are to be returned.
- **limit**— Limit the number of returned results (maximum is 5000).
- **page**— Use this parameter to access hits beyond limit. Numbering starts at 1. The `limit` parameter acts as page size.

Example request for retrieving folders and dashboards of the general folder

```
GET /api/search?folderIds=0&query=&starred=false HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response for retrieving folders and dashboards of the general folder

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 163,
    "uid": "000000163",
    "title": "Folder",
    "url": "/dashboards/f/000000163/folder",
    "type": "dash-folder",
    "tags": [],
    "isStarred": false,
    "uri":"db/folder" // deprecated in Grafana v5.0
  },
  {
    "id":1,
    "uid": "cIBgcSjkk",
    "title":"Production Overview",
    "url": "/d/cIBgcSjkk/production-overview",
    "type":"dash-db",
    "tags":[prod],
    "isStarred":true,
    "uri":"db/production-overview" // deprecated in Grafana v5.0
  }
]
```

Example request for searching for starred dashboards

```
GET /api/search?query=Production%20Overview&starred=true&tag=prod HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response for searching for starred dashboards

```
HTTP/1.1 200
Content-Type: application/json

[HTTP/1.1 200
Content-Type: application/json

[
  {
    "id":1,
    "uid": "cIBgcSjkk",
    "title":"Production Overview",
    "url": "/d/cIBgcSjkk/production-overview",
    "type":"dash-db",
    "tags":[prod],
    "isStarred":true,
    "folderId": 2,
    "folderUid": "000000163",
    "folderTitle": "Folder",
    "folderUrl": "/dashboards/f/000000163/folder",
    "uri":"db/production-overview" // deprecated in Grafana v5.0
  }
]
```

Folder Permissions API

Use the Folder API to update or retrieve the permissions for a folder.

Permissions with `folderId=-1` are the default permissions for users with the Viewer and Editor roles. Permissions can be set for a user, a team or a role (Viewer or Editor). Permissions cannot be set for Admins - they always have access to everything.

The permission levels for the `permission` field are as follows:

- 1 = View
- 2 = Edit
- 4 = Admin

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get permissions for a folder

```
GET /api/folders/:uid/permissions
```

Gets all existing permissions for the folder with the given `uid`.

Example request

```
GET /api/folders/nErXDvCkzz/permissions HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 551

[
  {
    "id": 1,
    "folderId": -1,
    "created": "2017-06-20T02:00:00+02:00",
    "updated": "2017-06-20T02:00:00+02:00",
    "userId": 0,
    "userLogin": "",
    "userEmail": "",
    "teamId": 0,
```

```
    "team": "",
    "role": "Viewer",
    "permission": 1,
    "permissionName": "View",
    "uid": "nErXDvCkzz",
    "title": "",
    "slug": "",
    "isFolder": false,
    "url": ""
  },
  {
    "id": 2,
    "dashboardId": -1,
    "created": "2017-06-20T02:00:00+02:00",
    "updated": "2017-06-20T02:00:00+02:00",
    "userId": 0,
    "userLogin": "",
    "userEmail": "",
    "teamId": 0,
    "team": "",
    "role": "Editor",
    "permission": 2,
    "permissionName": "Edit",
    "uid": "",
    "title": "",
    "slug": "",
    "isFolder": false,
    "url": ""
  }
]
```


Status Codes:

- **200**— OK
- **401**— Unauthorized
- **403**— Access denied
- **404**— Folder not found

Update permissions for a folder

```
POST /api/folders/:uid/permissions
```

Updates the permissions for a folder. This operation removes existing permissions if they are not included in the request.

Example request

```
POST /api/folders/nErXDvCkzz/permissions
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
{
  "items": [
    {
      "role": "Viewer",
      "permission": 1
    },
    {
      "role": "Editor",
      "permission": 2
    },
    {
      "teamId": 1,
      "permission": 1
    },
    {
      "userId": 11,
      "permission": 4
    }
  ]
}
```

JSON body schema:

- **items**— The permission items to add or update. Existing items that are omitted from the list are removed.

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Folder permissions updated","id":1,"title":"Department ABC"}
```

Status Codes:

- **200**— Created
- **401**— Unauthorized
- **403**— Access denied
- **404**— Dashboard not found

Organization API

Use the Organization API to work with organizations in an Amazon Managed Grafana workspace.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get current organization

```
GET /api/org/
```

Example request

```
GET /api/org/ HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
  "name":"Main Org."
}
```

Get all users within the current organization

```
GET /api/org/users
```

Required permissions: the `org.users:read` action with the scope `users:*`

Example request

```
GET /api/org/users HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "orgId": 1,
    "userId": 1,
    "email": "admin@localhost",
    "avatarUrl": "/avatar/46d229b033af06a191ff2267bca9ae",
    "login": "admin",
    "role": "Admin",
    "lastSeenAt": "2019-08-09T11:02:49+02:00",
    "lastSeenAtAge": "< 1m"
  }
]
```

Get all users within the current organization (lookup)

```
GET /api/org/users/lookup
```

Returns all users within the current organization, but with less detailed information. Accessible to users with org admin role, admin in any folder or admin of any team. Used mostly by the Grafana UI to provide a list of users when adding team members and when e diting folder/dashboard permissions.

Example request

```
GET /api/org/users/lookup HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "userId": 1,
    "login": "admin",
    "avatarUrl": "/avatar/46d229b033af06a191ff2267bca9ae"
  }
]
```

Updates the given user

```
PATCH /api/org/users/:userId
```

Required permissions: the `org.users.role:update` action with the scope `users:*`

Example request

```
PATCH /api/org/users/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "role": "Viewer",
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Organization user updated"}
```

Deletes user in current organization

```
DELETE /api/org/users/:userId
```

Required permissions: the `org.users:remove` action with the scope `users:*`

Example request

```
DELETE /api/org/users/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"User removed from organization"}
```

Update the current organization

```
PUT /api/org
```

Example request

```
PUT /api/org HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name":"Main Org."
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
```

```
{"message":"Organization updated"}
```

Add user to the current organization

```
POST /api/org/users
```

Required permissions: the `org.users:add` action with the scope `users:*`

Example request

```
POST /api/org/users HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "role": "Admin",
  "loginOrEmail": "admin"
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"User added to organization","userId":1}
```

Playlist API

Use the Playlist API to work with playlists in the Amazon Managed Grafana workspace.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

List playlists

```
GET /apis/playlist.grafana.app/v1/namespaces/default/playlists
```

Lists all playlists in the workspace.

Example request

```
GET /apis/playlist.grafana.app/v1/namespaces/default/playlists HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "kind": "PlaylistList",
  "apiVersion": "playlist.grafana.app/v1",
  "metadata": {},
  "items": [
    {
      "kind": "Playlist",
      "apiVersion": "playlist.grafana.app/v1",
      "metadata": {
        "name": "my-playlist-uid",
        "namespace": "default"
      },
      "spec": {
        "title": "My Playlist",
        "interval": "5m",
        "items": [
          {"type": "dashboard_by_uid", "value": "dashboard-uid-1"}
        ]
      }
    }
  ]
}
```

Get a playlist

```
GET /apis/playlist.grafana.app/v1/namespaces/default/playlists/:name
```

Retrieves a specific playlist by name (UID).

Example request

```
GET /apis/playlist.grafana.app/v1/namespaces/default/playlists/my-playlist-uid HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "kind": "Playlist",
  "apiVersion": "playlist.grafana.app/v1",
  "metadata": {
    "name": "my-playlist-uid",
    "namespace": "default"
  },
  "spec": {
    "title": "My Playlist",
    "interval": "5m",
    "items": [
      {"type": "dashboard_by_uid", "value": "dashboard-uid-1"}
    ]
  }
}
```

Create a playlist

```
POST /apis/playlist.grafana.app/v1/namespaces/default/playlists
```

Creates a new playlist.

Example request

```
POST /apis/playlist.grafana.app/v1/namespaces/default/playlists HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
```

```
  "metadata": {"name": "my-new-playlist-uid"},
  "spec": {
    "title": "My New Playlist",
    "interval": "5m",
    "items": [
      {"type": "dashboard_by_uid", "value": "dashboard-uid-1"}
    ]
  }
}
```

Update a playlist

```
PUT /apis/playlist.grafana.app/v1/namespaces/default/playlists/:name
```

Updates an existing playlist. The entire playlist spec must be provided.

Example request

```
PUT /apis/playlist.grafana.app/v1/namespaces/default/playlists/my-playlist-uid HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "metadata": {
    "name": "my-playlist-uid",
    "resourceVersion": "1234567890"
  },
  "spec": {
    "title": "Updated Playlist",
    "interval": "10m",
    "items": [
      {"type": "dashboard_by_uid", "value": "dashboard-uid-1"},
      {"type": "dashboard_by_tag", "value": "monitoring"}
    ]
  }
}
```

Delete a playlist

```
DELETE /apis/playlist.grafana.app/v1/namespaces/default/playlists/:name
```

Deletes a playlist by name (UID).

Example request

```
DELETE /apis/playlist.grafana.app/v1/namespaces/default/playlists/my-playlist-uid
 HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "kind": "Status",
  "apiVersion": "v1",
  "status": "Success",
  "code": 200
}
```

Search playlist (deprecated)

> **Important**
>
> This endpoint is deprecated. Use List playlists instead.

```
GET /api/playlists
```

Returns all playlists for the current Amazon Managed Grafana workspace, using pagination.

Example request

```
GET /api/playlists HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Querystring parameters:

- **query**— Limit the responses to playlists that have a name like this value.
- **limit**— Limit the response to X number of playlists.

Example response

```
HTTP/1.1 200
Content-Type: application/json
[
  {
    "id": 1,
    "name": "my playlist",
    "interval": "5m"
  }
]
```

Get one playlist (deprecated)

> **Important**
>
> This endpoint is deprecated. Use Get a playlist instead.

```
GET /api/playlists/:id
```

Example request

```
GET /api/playlists/1 HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
{
  "id" : 1,
  "name": "my playlist",
  "interval": "5m",
  "orgId": "my org",
```

```
  "items": [
    {
      "id": 1,
      "playlistId": 1,
      "type": "dashboard_by_id",
      "value": "3",
      "order": 1,
      "title":"my third dashboard"
    },
    {
      "id": 2,
      "playlistId": 1,
      "type": "dashboard_by_tag",
      "value": "myTag",
      "order": 2,
      "title":"my other dashboard"
    }
  ]
}
```

Get playlist items (deprecated)

> **Important**
>
> This endpoint is deprecated. Use Get a playlist instead. Playlist items are included in the playlist response.

```
GET /api/playlists/:id/items
```

Example request

```
GET /api/playlists/1/items HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
[
```

```
  {
    "id": 1,
    "playlistId": 1,
    "type": "dashboard_by_id",
    "value": "3",
    "order": 1,
    "title":"my third dashboard"
  },
  {
    "id": 2,
    "playlistId": 1,
    "type": "dashboard_by_tag",
    "value": "myTag",
    "order": 2,
    "title":"my other dashboard"
  }
]
```

Get playlist dashboards (deprecated)

> **Important**
>
> This endpoint is deprecated. Use Get a playlist instead.

```
GET /api/playlists/:id/dashboards
```

Example request

```
GET /api/playlists/1/dashboards HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
[
  {
    "id": 3,
    "title": "my third dashboard",
```

```
    "order": 1,
  },
  {
    "id": 5,
    "title":"my other dashboard"
    "order": 2,

  }
]
```

Create a playlist (deprecated)

> **Important**
>
> This endpoint is deprecated. Use Create a playlist instead.

```
POST /api/playlists/
```

Example request

```
PUT /api/playlists/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
  {
    "name": "my playlist",
    "interval": "5m",
    "items": [
      {
        "type": "dashboard_by_id",
        "value": "3",
        "order": 1,
        "title":"my third dashboard"
      },
      {
        "type": "dashboard_by_tag",
        "value": "myTag",
        "order": 2,
        "title":"my other dashboard"
      }
```

```
    ]
  }
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
  {
    "id": 1,
    "name": "my playlist",
    "interval": "5m"
  }
```

Update a playlist (deprecated)

> **Important**
>
> This endpoint is deprecated. Use Update a playlist instead.

```
PUT /api/playlists/:id
```

Example request

```
PUT /api/playlists/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
  {
    "name": "my playlist",
    "interval": "5m",
    "items": [
      {
        "playlistId": 1,
        "type": "dashboard_by_id",
        "value": "3",
        "order": 1,
        "title":"my third dashboard"
      },
      {
        "playlistId": 1,
```

```
        "type": "dashboard_by_tag",
        "value": "myTag",
        "order": 2,
        "title":"my other dashboard"
      }
    ]
  }
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
{
  "id" : 1,
  "name": "my playlist",
  "interval": "5m",
  "orgId": "my org",
  "items": [
    {
      "id": 1,
      "playlistId": 1,
      "type": "dashboard_by_id",
      "value": "3",
      "order": 1,
      "title":"my third dashboard"
    },
    {
      "id": 2,
      "playlistId": 1,
      "type": "dashboard_by_tag",
      "value": "myTag",
      "order": 2,
      "title":"my other dashboard"
    }
  ]
}
```

Delete a playlist (deprecated)

> **Important**
>
> This endpoint is deprecated. Use Delete a playlist instead.

```
DELETE /api/playlists/:id
```

Example request

```
DELETE /api/playlists/1 HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
{}
```

Plugin API

Use the Plugin API to manage plugins in the Amazon Managed Grafana workspace. To make changes to plugins with this API, the workspace must have plugin management enabled for your workspace. The user defined by the service account token must also be an admin for the Amazon Managed Grafana workspace.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Install plugin

```
POST /api/plugins/:id/install
```

Example request

```
POST /api/plugins/grafana-athena-datasource/install HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

```
{
  "version": "2.12.0" # optional, uses the latest compatible version if not provided
}
```

Example response

```
HTTP/1.1 200
```

Uninstall plugin

```
POST /api/plugins/:id/uninstall
```

Example request

```
POST /api/plugins/grafana-athena-datasource/uninstall HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "version": "2.12.0" # optional, uninstalls whatever is installed if not provided
}
```

Example response

```
HTTP/1.1 200
```

Get all plugins

```
GET /api/gnet/plugins
```

Example request

```
GET /api/gnet/plugins HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
{
  "items": [
    {
      "status": "active",
      "id": 74,
      "typeId": 1,
      "typeName": "Application",
      "typeCode": "app",
      "slug": "alexanderzobnin-zabbix-app",
      "name": "Zabbix",
      "description": "Zabbix plugin for Grafana",
      "version": "4.4.3",
      "versionStatus": "active",
      "versionSignatureType": "grafana",
      "versionSignedByOrg": "grafana",
      "versionSignedByOrgName": "Grafana Labs",
      "userId": 0,
      "orgId": 13056,
      "orgName": "Alexander Zobnin",
      "orgSlug": "alexanderzobnin",
      "orgUrl": "https://github.com/alexanderzobnin",
      "url": "https://github.com/grafana/grafana-zabbix/",
      "createdAt": "2016-04-06T20:23:41.000Z",
      "updatedAt": "2023-10-10T12:53:51.000Z",
      "downloads": 90788771,
      "verified": false,
      "featured": 180,
      "internal": false,
      "downloadSlug": "alexanderzobnin-zabbix-app",
      "popularity": 0.2485,
      "signatureType": "grafana",
      "packages": {
        "linux-amd64": {
          "md5": "baa06e8f26731f99748c58522cd4ffb6",
          "sha256": "a4a108f2e04a2114810c7b60419b4b04bf80d3377e2394b0586e2dc96b5a929c",
          "packageName": "linux-amd64",
          "downloadUrl": "/api/plugins/alexanderzobnin-zabbix-app/versions/4.4.3/
download?os=linux&arch=amd64"
        },
        <... further packages>
```

```
      },
      "links": [
        {
          "rel": "self",
          "href": "/plugins/alexanderzobnin-zabbix-app"
        },
        <... further links>
      ],
      "angularDetected": false
    },
    <... further plugins>
  ],
  "orderBy": "weight",
  "direction": "asc",
  "links": [
    {
      "rel": "self",
      "href": "/plugins"
    }
  ]
}
```

Get plugin

```
GET /api/gnet/plugins/:id
```

Example request

```
GET /api/gnet/plugins/grafana-athena-datasource HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
{
  "status": "active",
  "id": 764,
  "typeId": 2,
  "typeName": "Data Source",
```

```
  "typeCode": "datasource",
  "slug": "grafana-athena-datasource",
  "name": "Amazon Athena",
  "description": "Use Amazon Athena with Grafana",
  "version": "2.13.0",
  "versionStatus": "active",
  "versionSignatureType": "grafana",
  "versionSignedByOrg": "grafana",
  "versionSignedByOrgName": "Grafana Labs",
  "userId": 0,
  "orgId": 5000,
  "orgName": "Grafana Labs",
  "orgSlug": "grafana",
  "orgUrl": "https://grafana.org",
  "url": "https://github.com/grafana/athena-datasource/",
  "createdAt": "2021-11-24T08:55:41.000Z",
  "updatedAt": "2023-10-31T17:20:32.000Z",
  "json": {
    "$schema": "https://raw.githubusercontent.com/grafana/grafana/master/docs/sources/
developers/plugins/plugin.schema.json",
    "alerting": true,
    "annotations": true,
    "backend": true,
    "dependencies": {
      "grafanaDependency": ">=8.0.0",
      "plugins": []
    },
    "executable": "gpx_athena",
    "id": "grafana-athena-datasource",
    "includes": [
      {
        "name": "Cost Usage Report Monitoring",
        "path": "dashboards/cur-monitoring.json",
        "type": "dashboard"
      },
      {
        "name": "Amazon VPC Flow Logs",
        "path": "dashboards/vpc-flow-logs.json",
        "type": "dashboard"
      }
    ],
    "info": {
      "author": {
        "name": "Grafana Labs",
```

```
        "url": "https://grafana.com"
      },
      "build": {
        "time": 1698764559022,
        "repo": "https://github.com/grafana/athena-datasource",
        "branch": "main",
        "hash": "25cc131300f1ed22593bc3ba08b2bef7d23fbcd01",
        "build": 1462
      },
      "description": "Use Amazon Athena with Grafana",
      "keywords": [
        "datasource",
        "athena"
      ],
      "links": [
        {
          "name": "Website",
          "url": "https://github.com/grafana/athena-datasource"
        },
        {
          "name": "License",
          "url": "https://github.com/grafana/athena-datasource/blob/master/LICENSE"
        }
      ],
      "logos": {
        "large": "img/logo.svg",
        "small": "img/logo.svg"
      },
      "screenshots": [],
      "updated": "2023-10-31",
      "version": "2.13.0"
    },
    "metrics": true,
    "name": "Amazon Athena",
    "type": "datasource"
  },
  "readme": "<... full HTML readme>",
  "statusContext": "",
  "downloads": 2505825,
  "verified": false,
  "featured": 0,
  "internal": false,
  "downloadSlug": "grafana-athena-datasource",
  "popularity": 0.0594,
```

```
  "signatureType": "grafana",
  "grafanaDependency": ">=8.0.0",
  "packages": {
    "linux-amd64": {
      "md5": "7efef359bf917b4ca6b149de42a1282d",
      "sha256": "cd2fc5737c321dc3d8bbe2852c801c01adb64eacc9f60420bd21dc18bee43531",
      "packageName": "linux-amd64",
      "downloadUrl": "/api/plugins/grafana-athena-datasource/versions/2.13.0/download?
os=linux&arch=amd64"
    },
    <... other packages>
  },
  "links": [
    {
      "rel": "self",
      "href": "/plugins/grafana-athena-datasource"
    },
    <... other links>
  ],
  "angularDetected": false
}
```

Get plugin versions

```
POST /api/gnet/plugins/:id/versions
```

Example request

```
GET /api/gnet/plugins/grafana-athena-datasource/versions HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
{
  "items": [
    {
      "id": 5306,
      "pluginId": 764,
```

```
      "pluginSlug": "grafana-athena-datasource",
      "version": "2.13.0",
      "url": "https://github.com/grafana/athena-datasource/",
      "commit": "",
      "description": "Use Amazon Athena with Grafana",
      "createdAt": "2023-10-31T17:20:31.000Z",
      "updatedAt": null,
      "downloads": 33790,
      "verified": false,
      "status": "active",
      "statusContext": "",
      "downloadSlug": "grafana-athena-datasource",
      "packages": {},
      "links": [
        {
          "rel": "self",
          "href": "/plugins/grafana-athena-datasource/versions/2.13.0"
        },
        {
          "rel": "images",
          "href": "/plugins/grafana-athena-datasource/versions/2.13.0/images"
        },
        {
          "rel": "thumbnails",
          "href": "/plugins/grafana-athena-datasource/versions/2.13.0/thumbnails"
        },
        {
          "rel": "plugin",
          "href": "/plugins/grafana-athena-datasource"
        },
        {
          "rel": "download",
          "href": "/plugins/grafana-athena-datasource/versions/2.13.0/download"
        }
      ],
      "grafanaDependency": ">=8.0.0",
      "angularDetected": false
    },
    {
      "id": 5244,
      "pluginId": 764,
      "pluginSlug": "grafana-athena-datasource",
      "version": "2.12.0",
      "url": "https://github.com/grafana/athena-datasource/",
```

```
      "commit": "",
      "description": "Use Amazon Athena with Grafana",
      "createdAt": "2023-10-17T12:42:13.000Z",
      "updatedAt": null,
      "downloads": 60742,
      "verified": false,
      "status": "active",
      "statusContext": "",
      "downloadSlug": "grafana-athena-datasource",
      "packages": {},
      "links": [
        {
          "rel": "self",
          "href": "/plugins/grafana-athena-datasource/versions/2.12.0"
        },
        {
          "rel": "images",
          "href": "/plugins/grafana-athena-datasource/versions/2.12.0/images"
        },
        {
          "rel": "thumbnails",
          "href": "/plugins/grafana-athena-datasource/versions/2.12.0/thumbnails"
        },
        {
          "rel": "plugin",
          "href": "/plugins/grafana-athena-datasource"
        },
        {
          "rel": "download",
          "href": "/plugins/grafana-athena-datasource/versions/2.12.0/download"
        }
      ],
      "grafanaDependency": ">=8.0.0",
      "angularDetected": false
    },
    <... other versions>
  ]
}
```


Preferences API

Use the Preferences API to work with user preferences in the Amazon Managed Grafana workspace.

Keys:

- **theme**— Valid values are `light`, `dark`, or an empty string to use the default theme.
- **homeDashboardId**— The numerical `:id` of a favorited dashboard. The default is 0.
- **timezone**— Valid values are `utc`, `browser`, or an empty string to use the default.

Omitting a key causes the current value to be replaced with the system default value.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get current user preferences

```
GET /api/user/preferences
```

Example request

```
GET /api/user/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"theme":"","homeDashboardId":0,"timezone":""}
```

Update current user preferences

```
PUT /api/user/preferences
```

Example request

```
PUT /api/user/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "theme": "",
  "homeDashboardId":0,
  "timezone":"utc"
}
```

Example response

```
HTTP/1.1 200
Content-Type: text/plain; charset=utf-8

{"message":"Preferences updated"}
```

Get current org preferences

```
GET /api/org/preferences
```

Example request

```
GET /api/org/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"theme":"","homeDashboardId":0,"timezone":""}
```

Update current org preferences

```
PUT /api/org/preferences
```

Example request

```
PUT /api/org/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "theme": "",
  "homeDashboardId":0,
  "timezone":"utc"
}
```

Example response

```
HTTP/1.1 200
Content-Type: text/plain; charset=utf-8

{"message":"Preferences updated"}
```

Snapshot API

Use the Snapshot API to work with snapshots in an Amazon Managed Grafana workspace.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Create new shapshot

```
POST /api/snapshots
```

Example request

```
POST /api/snapshots HTTP/1.1
    Accept: application/json
    Content-Type: application/json
    Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

```
{
  "dashboard": {
    "editable":false,
    "hideControls":true,
    "nav":[
    {
      "enable":false,
    "type":"timepicker"
    }
    ],
    "rows": [
      {

      }
    ],
    "style":"dark",
    "tags":[],
    "templating":{
      "list":[
      ]
    },
    "time":{
    },
    "timezone":"browser",
    "title":"Home",
    "version":5
    },
  "expires": 3600
}
```

JSON body schema:

- **dashboard**— (Required) The complete dashboard model.
- **name**— (Optional) A name for the snapshot.
- **expires**— (Optional) When the snapshot should expire, in seconds. The default is to never expire.
- **external**— (Optional) Save the snapshot on an external server rather than locally. Default is false.
- **key**— (Required if `external` is `true`) Define a unique key.
- **deletekey**— (Required if `external` is `true`) A unique key to be used to delete the snapshot. It is different than `key` so that only the creator can delete the snapshot.

> **Note**
>
> When creating a snapshot using the API, you have to provide the full dashboard payload including the snapshot data. This endpoint is designed for the Grafana UI.

Example response

```
HTTP/1.1 200
    Content-Type: application/json
    {
      "deleteKey":"XXXXXXX",
      "deleteUrl":"myurl/api/snapshots-delete/XXXXXXX",
      "key":"YYYYYYY",
      "url":"myurl/dashboard/snapshot/YYYYYYY",
      "id": 1,
    }
```

Keys:

- **deleteKey**— A key generated to be used to delete the snapshot.
- **key**— A key generated to share the dashboard.

Get list of snapshots

```
GET /api/dashboard/snapshots
```

Query parameters:

- **query**— Search query
- **limit**— Limit the number of returned results

Example request

```
GET /api/dashboard/snapshots HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id":8,
    "name":"Home",
    "key":"YYYYYYY",
    "orgId":1,
    "userId":1,
    "external":false,
    "externalUrl":"",
    "expires":"2200-13-32T25:23:23+02:00",
    "created":"2200-13-32T28:24:23+02:00",
    "updated":"2200-13-32T28:24:23+02:00"
  }
]
```

Get snapshot by key

```
GET /api/snapshots/:key
```

Example request

```
GET /api/snapshots/YYYYYYY HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "meta":{
    "isSnapshot":true,
    "type":"snapshot",
    "canSave":false,
    "canEdit":false,
    "canStar":false,
    "slug":"",
```

```
    "expires":"2200-13-32T25:23:23+02:00",
    "created":"2200-13-32T28:24:23+02:00"
  },
  "dashboard": {
    "editable":false,
    "hideControls":true,
    "nav": [
      {
        "enable":false,
        "type":"timepicker"
      }
    ],
    "rows": [
      {

      }
    ],
    "style":"dark",
    "tags":[],
    "templating":{
      "list":[
      ]
    },
    "time":{
    },
    "timezone":"browser",
    "title":"Home",
    "version":5
  }
}
```

Delete snapshot by key

```
DELETE /api/snapshots/:key
```

Example request

```
DELETE /api/snapshots/YYYYYYY HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Snapshot deleted. It might take an hour before it's cleared from any CDN
 caches.", "id": 1}
```

Delete snapshot by deleteKey

This API call can be used without authentication by using the secret delete key for the snapshot.

```
GET /api/snapshots-delete/:deleteKey
```

Example request

```
GET /api/snapshots-delete/XXXXXXX HTTP/1.1
Accept: application/json
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Snapshot deleted. It might take an hour before it's cleared from any CDN
 caches.", "id": 1}
```

Short URL API

Use the Short URL API to create shortened URLs. A short URL represents a longer URL containing complex query parameters in a smaller and simpler format.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Create short URL

```
POST /api/short-urls
```

Creates a short URL.

Example request

```
POST /api/short-urls HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "path": "d/TxKARsmGz/new-dashboard?orgId=1&from=1599389322894&to=1599410922894"
}
```

JSON body schema:

- **path** – The path to shorten, relative to the Grafana root URL.

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "uid": "AT76wBvGk",
  "url": "/goto/AT76wBvGk?orgId=1"
}
```

Status codes:

- **200** – Created
- **400** – Errors (invalid JSON, missing or invalid fields)

Get short URL

```
GET /api/short-urls/:uid
```

Retrieves a short URL by its UID.

Example request

```
GET /api/short-urls/AT76wBvGk HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "uid": "AT76wBvGk",
  "path": "d/TxKARsmGz/new-dashboard?orgId=1&from=1599389322894&to=1599410922894"
}
```

Team API

Use the Team API to work with teams in an Amazon Managed Grafana workspace. All actions in this API require that you have the Admin role.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Team search with pagination

```
GET /api/teams/search?perpage=50&page=1&query=myteam
```

or

```
GET /api/teams/search?name=myteam
```

Example request

```
GET /api/teams/search?perpage=10&page=1&query=myteam HTTP/1.1
Accept: application/json
Content-Type: application/json
```

```
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Using the query parameter

The default value for the `perpage` parameter is 1000 and for the `page` parameter is 1.

The `totalCount` field in the response can be used for pagination of the teams list. For example, if `totalCount` is 100 teams and the`perpage` parameter is set to 10, then there are 10 pages of teams.

The `query` parameter is optional and returns results where the query value is contained in the `name` field. Query values with spaces need to be URL-encoded. For example, `query=my%20team`.

Using the name parameter

The `name` parameter returns a single team if the parameter matches the `name` field.

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "totalCount": 1,
  "teams": [
    {
      "id": 1,
      "orgId": 1,
      "name": "MyTestTeam",
      "email": "",
      "avatarUrl": "\/avatar\/3f49c15916554246daa714b9bd0ee39",
      "memberCount": 1
    }
  ],
  "page": 1,
  "perPage": 1000
```

Status Codes:

- **200**— Created
- **401**— Unauthorized

- **403**— Permission denied
- **404**— Team not found (if searching by name)

Get team by Id

```
GET /api/teams/:id
```

Example request

```
GET /api/teams/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HHTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "orgId": 1,
  "name": "MyTestTeam",
  "email": "",
  "created": "2017-12-15T10:40:45+01:00",
  "updated": "2017-12-15T10:40:45+01:00"
}
```

Add a team

The `name` of the team must be unique. The `name` field is required and the `email` and `orgId` fields are optional.

```
POST /api/teams
```

Example request

```
POST /api/teams HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "MyTestTeam",
  "email": "email@test.com",
  "orgId": 2
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Team created","teamId":2}
```

Status Codes:

- **200**— Created
- **401**— Unauthorized
- **403**— Permission denied
- **409**— Team name already exists

Update team

```
PUT /api/teams/:id
```

Only the `name` and `email` fields can be updated.

Example request

```
PUT /api/teams/2 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "MyTestTeam",
  "email": "email@test.com"
```

```
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Team updated"}
```

Status Codes:

- **200**— Created
- **401**— Unauthorized
- **403**— Permission denied
- **404**— Team not found
- **409**— Team name already exists

Delete team by Id

```
DELETE /api/teams/:id
```

Example request

```
DELETE /api/teams/2 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Team deleted"}
```

Status Codes:

- **200**— Created

- **401**— Unauthorized
- **403**— Permission denied
- **404**— Team not found

Get team members

```
GET /api/teams/:teamId/members
```

Example request

```
GET /api/teams/1/members HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "orgId": 1,
    "teamId": 1,
    "userId": 3,
    "email": "user1@email.com",
    "login": "user1",
    "avatarUrl": "\/avatar\/1b3c32f6386b0185c40d359cdc733a7"
  },
  {
    "orgId": 1,
    "teamId": 1,
    "userId": 2,
    "email": "user2@email.com",
    "login": "user2",
    "avatarUrl": "\/avatar\/cad3c68da76e45d10269e8ef02f8e7"
  }
]
```

Status Codes:

- **200**— Created
- **401**— Unauthorized
- **403**— Permission denied

Add team member

```
POST /api/teams/:teamId/members
```

Example request

```
POST /api/teams/1/members HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "userId": 2
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Member added to Team"}
```

Status Codes:

- **200**— Created
- **400**— User is already in team
- **401**— Unauthorized
- **403**— Permission denied
- **404**— Team not found

Remove member from team

```
DELETE /api/teams/:teamId/members/:userId
```

Example request

```
DELETE /api/teams/2/members/3 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Team Member removed"}
```

Status Codes:

- **200**— Created
- **401**— Unauthorized
- **403**— Permission denied
- **404**— Team not found/team member not found

Get team preferences

```
GET /api/teams/:teamId/preferences
```

Example request

```
GET /api/teams/2/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
```

```
{
  "theme": "",
  "homeDashboardId": 0,
  "timezone": ""
}
```

Update team preferences

```
PUT /api/teams/:teamId/preferences
```

Example request

```
PUT /api/teams/2/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "theme": "dark",
  "homeDashboardId": 39,
  "timezone": "utc"
}
```

JSON body schema:

- **theme**— Specify either `light`, `dark`, or an empty string to use the default theme.
- **homeDashboardId**— The numerical `:id` of a dashboard. The default is 0.
- **timezone**— Specify either `utc`, `browser`, or an empty string to use the default.

Omitting a parameter causes the current value to be replaced with the system default value.

Example response

```
HTTP/1.1 200
Content-Type: text/plain; charset=utf-8

{
  "message":"Preferences updated"
}
```

User API

Use the User API to work with users in an Amazon Managed Grafana workspace.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get teams that the user is a member of

```
GET /api/user/teams
```

Example request

```
GET /api/user/teams HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 1,
    "orgId": 1,
    "name": "MyTestTeam",
    "email": "",
    "avatarUrl": "\/avatar\/3f49c15916554246daa714b9bd0ee3",
    "memberCount": 1
  }
]
```

Get list of snapshots

Stars the given Dashboard for the actual user.

```
POST /api/user/stars/dashboard/:dashboardId
```

Example request

```
POST /api/user/stars/dashboard/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Dashboard starred!"}
```

Unstar a dashboard

Deletes the starring of the given Dashboard for the actual user.

```
DELETE /api/user/stars/dashboard/:dashboardId
```

Example request

```
DELETE /api/user/stars/dashboard/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Dashboard unstarred"}
```

Get auth tokens of the actual user

```
GET /api/user/auth-tokens
```

Example request

```
GET /api/user/auth-tokens HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 361,
    "isActive": true,
    "clientIp": "127.0.0.1",
    "browser": "Chrome",
    "browserVersion": "72.0",
    "os": "Linux",
    "osVersion": "",
    "device": "Other",
    "createdAt": "2019-03-05T21:22:54+01:00",
    "seenAt": "2019-03-06T19:41:06+01:00"
  },
  {
    "id": 364,
    "isActive": false,
    "clientIp": "127.0.0.1",
    "browser": "Mobile Safari",
    "browserVersion": "11.0",
    "os": "iOS",
    "osVersion": "11.0",
    "device": "iPhone",
    "createdAt": "2019-03-06T19:41:19+01:00",
    "seenAt": "2019-03-06T19:41:21+01:00"
  }
]
```

Revoke an auth token of the actual user

```
POST /api/user/revoke-auth-token
```

Revokes the given auth token (device) for the actual user. User of issued auth token (device) are no longer logged in and are required to authenticate again at their next activity.

Example request

```
POST /api/user/revoke-auth-token HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "authTokenId": 364
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "message": "User auth token revoked"
}
```

Grafana version 10 API reference

You can use Grafana HTTP APIs with Amazon Managed Grafana workspaces. The following sections describe how to use the APIs, and then list which Grafana APIs are supported.

> **Note**
>
> Amazon Managed Grafana also provides AWS APIs for creating and managing workspaces. For more information about the AWS APIs, see the Amazon Managed Grafana API Reference.

Topics

- Authenticate with tokens
- Alerting API
- Alerting Notification Channels API

- Annotations API
- Authentication API
- Dashboard API
- Dashboard Permissions API
- Dashboard Versions API
- Data Source API
- Data Source Permissions API
- External Group Synchronization API
- Folder API
- Folder/Dashboard Search API
- Folder Permissions API
- Organization API
- Playlist API
- Plugin API
- Preferences API
- Snapshot API
- Team API
- User API

Authenticate with tokens

To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid Grafana *token*. The token provides authentication and authorization for caller of the API. There are two ways to create tokens.

- **Service account** – A service account is used to run automated workloads in Grafana, such as provisioning, configuration, or report generation. You can create service account tokens for your service account. Service accounts are available in Grafana workspaces compatible with version 9.x, and are planned to replace API keys as the primary way to authenticate applications that interact with Grafana APIs.
- **API Key** – An API key (also called an API token) is a randomly generated string that external systems can use to interact with Grafana HTTP APIs. API keys are available in versions 8, 9, and

10 of Grafana workspaces, however, GrafanaLabs has announced that they are deprecating them in a future release.

Topics

- Use service accounts to authenticate with the Grafana HTTP APIs
- Use API keys to authenticate with Grafana HTTP APIs

Use service accounts to authenticate with the Grafana HTTP APIs

You can use a service account to run automated workloads in Grafana, such as dashboard provisioning, configuration, or report generation. Create service accounts and tokens to authenticate applications, such as Terraform, with the Grafana console or the Amazon Managed Grafana API.

> **Note**
>
> Service accounts are available in Grafana 9.x and newer, and have replaced API keys as the primary way to authenticate applications that interact with Grafana. API keys are deprecated and have been removed in Amazon Managed Grafana version 12.

A common use case for creating a service account is to perform operations on automated or triggered tasks. You can use service accounts to:

- Define alerts in your system to be used in Grafana
- Interact with Grafana without signing in as a user

> **Note**
>
> Each service account is considered a user for billing purposes.

Service account tokens

A service account token is a generated string that acts as a key to authenticate with Grafana's HTTP API.

When you create a service account, you can associate one or more access tokens with it. You can use service access tokens the same way as API Keys, for example to access Grafana HTTP APIs programmatically.

You can create multiple tokens for the same service account. You might want to do this if:

- multiple applications use the same permissions, but you would like to audit or manage their actions separately.
- you need to rotate or replace a compromised token.

Service account access tokens inherit permissions from the service account.

Amazon Managed Grafana has a quota on the number of service account tokens you can have at one time. This includes active and expired tokens. You must delete tokens to remove them from your quota.

Service account benefits

The added benefits of service accounts to API keys include:

- Service accounts resemble Grafana users and can be enabled/disabled, granted specific permissions, and remain active until they are deleted or disabled. API keys are only valid until their expiry date.
- Service accounts can be associated with multiple tokens.
- Unlike API keys, service account tokens are not associated with a specific user, which means that applications can be authenticated even if a Grafana user is deleted.
- You can grant permissions to service accounts in the same way that you grant permissions to users.

  For more information about permissions, see Using permissions.

Creating a service account

> **Note**
>
> The user who creates a service account is also able to read, update and delete the service account that they created, as well as permissions associated with that service account.

Prerequisite

Ensure you have permission to create and edit service accounts. By default, the organization administrator role is required to create and edit service accounts. For more information about permissions, see Using permissions.

To create a service account

1. Sign in to your Amazon Managed Grafana workspace and select **Administration** from the left-side menu.
2. Select **Service accounts**.
3. Select **Add service account**.
4. Enter a **Display name**.
5. The display name must be unique as it determines the ID associated with the service account.
   - We recommend that you use a consistent naming convention when you name service accounts. A consistent naming convention can help you scale and maintain service accounts in the future.
   - You can change the display name at any time.
6. Choose **Create**.

> **Note**
>
> You can also create service accounts using the Amazon Managed Grafana AWS APIs. Use the CreateWorkspaceServiceAccount to create a service account programmatically.

Adding a token to a service account

A service account token is a generated random string that acts as an alternative to a password when authenticating with Grafana's HTTP API.

Prerequisite

Ensure you have permission to create and edit service accounts. By default, the organization administrator role is required to create and edit service accounts. For more information about permissions, see Using permissions.

To add a token to a service account

1. Sign in to your Grafana workspace and choose **Administration** in the left-side menu.
2. Expand the **Users and Access** menu.
3. Choose **Service accounts**.
4. Select the service account to which you want to add a token.
5. Choose **Add service account token**.
6. Enter a name for the token.
7. Select **Set expiration date** and enter an expiration date for the token.
   - The expiration date specifies how long you want the key to be valid.
   - You can set an expiration date up to 30 days in the future.
   - If you are unsure of an expiration date, we recommend that you set the token to expire after a short time, such as a few hours or less. This limits the risk associated with a token that is valid for a long time.
8. Choose **Generate token**.

> **Note**
>
> You can also create service account tokens using the Amazon Managed Grafana AWS APIs. Use the CreateWorkspaceServiceAccountToken to create a service account token programmatically.

Delete a service token

When you are done with a service token, you must delete it to remove it from your workspace. Expired, but not yet deleted, tokens count toward your quota of service account tokens.

Prerequisite

Ensure you have permission to create and edit service accounts. By default, the organization administrator role is required to create and edit service accounts. For more information about permissions, see Using permissions.

To remove a token to a service account

1. Sign in to your Grafana workspace and choose **Administration** in the left-side menu.
2. Expand the **Users and Access** menu.
3. Choose **Service accounts**.
4. Select the service account from which you want to delete a token.
5. In the list of tokens, select the red icon with an **x** next to the service account token you wish to delete.
6. Select **Delete**.

Your token is deleted.

> **Note**
>
> You can also delete service account tokens using the Amazon Managed Grafana AWS APIs. Use the DeleteWorkspaceServiceAccountToken to delete a service account token programmatically.

Assign roles to a service account

You can assign roles to a Grafana service account to control access for the associated service account tokens. You can assign roles to a service account using the Grafana UI or via the API.

Prerequisite

Ensure you have permission to create and edit service accounts. By default, the organization administrator role is required to create and edit service accounts. For more information about permissions, see Using permissions.

To assign a role to a service account

1. Sign in to Grafana and choose **Administration** in the left-side menu.
2. Choose **Service accounts**.
3. Select the service account to which you want to assign a role. As an alternative, find the service account in the list view.

4. Assign a role using the role picker to update.

Use API keys to authenticate with Grafana HTTP APIs

One way to access Grafana APIs is to use an *API key*, which is also called an *API token*. To create an API key, use one of the following procedures. An API key is valid for a limited time that you specify when you create it, up to 30 days.

Topics

- Creating a Grafana API key to use with Grafana APIs in the workspace (Console)
- Creating an Amazon Managed Grafana workspace API key using AWS CLI

> **Important**
>
> API keys are deprecated and have been removed in Amazon Managed Grafana version 12. Use service accounts instead. For more information, see Use service accounts to authenticate with the Grafana HTTP APIs.

When you create an API key, you specify a *role* for the key. The role determines the level of administrative power that users of the key have.

The following tables show the permissions granted to the Admin, Editor, and Viewer roles. The first table shows general organizational permissions. In this table, **Full** means the ability to view, edit, add permissions, and delete permissions. The **Explore** column shows whether the role can use the *Explore* view. The **Other** permissions column shows whether the role has permissions for managing users, teams, plug-ins, and organizational settings.

Role	Dashboards	Playlists	Folders	Explore	Data sources	Other permissions
Viewer	View	View	No	No	No	No
Editor	Full	Full	Full	Yes	No	No
Admin	Full	Full	Full	Yes	Full	Full

The following table shows the additional dashboard- and folder-level permissions that you can set. These are different than the Admin, Editor, and Viewer roles.

Role	Dashboards	Folders	Change permissions
View	View	View	No
Edit	Create, edit	View	No
Admin	Create, edit, delete	Create, edit, delete	Yes

> **Note**
>
> A more scoped permission with a lower permission level does not have effect if a more general rule with more permission exists. For example, if you give a user the organizational **Editor** role and then assign that user only the **View** permissions for a dashboard, the more restrictive **View** permission has no effect because the user has full **Edit** access because of their **Editor** role.

Creating a Grafana API key to use with Grafana APIs in the workspace (Console)

> **Note**
>
> In Amazon Managed Grafana workspaces compatible with Grafana version 10 and above, the ability to create API keys in the workspace was removed. If your workspace is a Grafana version 10 workspace, you can only create API keys through the AWS CLI or API.
> API keys are deprecated and have been removed in Amazon Managed Grafana version 12. Use service accounts instead. For more information, see Use service accounts to authenticate with the Grafana HTTP APIs.

To create a Grafana API key to use with Grafana APIs in the workspace console

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the upper left corner of the page, choose the menu icon and then choose **All workspaces**.
3. Choose the name of the Amazon Managed Grafana workspace.

4. In the workspace details page, choose the URL displayed under **Grafana workspace URL**.
5. In the Grafana console side menu, pause on the **Configuration** (gear) icon, then choose **API Keys**.
6. Choose **New API Key**.
7. Enter a unique name for the key.
8. For **Role**, select the access level that the key is to be granted. Select **Admin** to allow a user with this key to use APIs at the broadest, most powerful administrative level. Select **Editor** or **Viewer** to limit the key's users to those levels of power. For more information, see the previous tables.
9. For **Time to live**, specify how long you want the key to be valid. The maximum is 30 days (one month). You enter a number and a letter. The valid letters are **s** for seconds, **m** for minutes, **h** for hours, **d** for days, **w** for weeks, and **M** for month. For example, **12h** is 12 hours and **1M** is 1 month (30 days).

   We strongly recommend that you set the key's time to live for a shorter time, such as a few hours or less. This creates much less risk than having API keys that are valid for a long time.
10. Choose **Add**.
11. (Optional) You can automate creating API keys with the Create API Key API using Terraform. For more information on automating API key creation using Terraform, see Creating Grafana API Key using Terraform.

Creating an Amazon Managed Grafana workspace API key using AWS CLI

To create an Amazon Managed Grafana workspace API key using AWS CLI

In the following example, replace the *key_name*, *key_role*, *seconds_to_live* and *workspace_id* with your own information. To find out about the format of the key-name, key-role and seconds-to-live, see https://docs.aws.amazon.com/grafana/latest/APIReference/API_CreateWorkspaceApiKey.html in the API guide.

```
aws grafana create-workspace-api-key --key-name "key_name" --key-role "key_role" --
seconds-to-live seconds_to_live --workspace-id "workspace_id"
```

The following is a sample CLI response:

create-workspace-api-key output example

You can find the *workspace_id* of your workspace by running the following command:

```
aws grafana list-workspaces
```

Alerting API

> **Note**
>
> This section only applies to classic alerting. For more information, see Grafana alerting.

Use the Preferences API to get information about classic dashboard alerts and their states. However, you can't use this API to modify the alert. To create new alerts or modify them you need to update the dashboard JSON that contains the alerts.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get alerts

```
GET /api/alerts
```

Example request

```
GET /api/alerts HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Querystring parameters:

These parameters are used as querystring parameters. For example: `/api/alerts?dashboardId=1`

- **dashboardId**— Limit the responses to alerts in the specified dashboards value. You can specify multiple dashboards. For example, `dashboardId=23&dashboardId=35`

- **panelId**— Limit the response to alert for a specified panel on a dashboard.
- **query**— Limit the response to alerts having a name like this value.
- **state**— Return the alerts that have one ore more of the following alert states: `ALL`, `alerting`, ok, `no_data`, `paused`, or `pending`. To specify multiple states, use the following format: `?state=paused&state=alerting`
- **limit**— Limit the response to X number of alerts.
- **folderId**— Limit the response to alerts of dashboards in the specified folders. You can specify multiple folders. For example, `folderId=23&folderId=35`
- **dashboardQuery**— Limit the responses to alerts having a dashboard name like this value.
- **dashboardTag**— Limit the response alerts of dashboards with specified tags. To do "AND" filtering with multiple tags, specify the tags parameter multiple times. For example, `dashboardTag=tag1&dashboardTag=tag2`. Note that these are Grafana tags, not AWS tags.

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 1,
    "dashboardId": 1,
    "dashboardUId": "ABcdEFghij"
    "dashboardSlug": "sensors",
    "panelId": 1,
    "name": "fire place sensor",
    "state": "alerting",
    "newStateDate": "2018-05-14T05:55:20+02:00",
    "evalDate": "0001-01-01T00:00:00Z",
    "evalData": null,
    "executionError": "",
    "url": "http://grafana.com/dashboard/db/sensors"
  }
]
```

Get alert by Id

```
GET /api/alerts/:id
```

Example request

```
GET /api/alerts/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "dashboardId": 1,
  "dashboardUId": "ABcdEFghij"
  "dashboardSlug": "sensors",
  "panelId": 1,
  "name": "fire place sensor",
  "state": "alerting",
  "message": "Someone is trying to break in through the fire place",
  "newStateDate": "2018-05-14T05:55:20+02:00",
  "evalDate": "0001-01-01T00:00:00Z",
  "evalData": "evalMatches": [
    {
      "metric": "movement",
      "tags": {
        "name": "fireplace_chimney"
      },
      "value": 98.765
    }
  ],
  "executionError": "",
  "url": "http://grafana.com/dashboard/db/sensors"
}
```

Important

`evalMatches` data is cached in the database when and only when the state of the alert changes. If data from one server triggers the alert first and, before that server is seen

leaving the alerting state, a second server also enters a state that would trigger the alert, the second server is not visible in the `evalMatches` data.

Pause alert by Id

```
POST /api/alerts/:id/pause
```

Example request

```
POST /api/alerts/1/pause HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "paused": true
}
```

The `:id` query parameter is the Id of the alert to be paused or unpaused. `paused` can be `true` to pause an alert or `false` to unpause the alert.

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "alertId": 1,
  "state":   "Paused",
  "message": "alert paused"
}
```

Alerting Notification Channels API

Use the Alerting Notification Channels API to create, update, delete, and retrieve notification channels.

The identifier (id) of a notification channel is an auto-incrementing numeric value and is only unique per workspace. The unique identifier (uid) of a notification channel can be used to uniquely

identify a folder between multiple workspaces. It's automatically generated if you don't provide one when you create a notification channel. The uid allows having consistent URLs for accessing notification channels and when synchronizing notification channels between multiple Amazon Managed Grafana workspaces.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get all notification channels

Returns all notification channels that the authenticated user has permission to view.

```
GET /api/alert-notifications
```

Example request

```
GET /api/alert-notifications HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 1,
    "uid": "sns-uid",
    "name": "test",
    "type": "sns",
    "isDefault": false,
    "sendReminder": false,
    "disableResolveMessage": false,
    "frequency": "",
    "created": "2023-09-08T19:57:56Z",
```

```
    "updated": "2023-09-08T19:57:56Z",
    "settings": {
      "authProvider": "default",
      "autoResolve": true,
      "httpMethod": "POST",
      "messageFormat": "json",
      "severity": "critical",
      "topic": "<SNS-TOPIC-ARN>",
      "uploadImage": false
    },
    "secureFields": {}
  }
]
```

Get all notification channels (lookup)

Returns all notification channels, but with less detailed information. Accessible by any authenticated user and is mainly used to provide alert notification channels in the Grafana workspace console UI when configuring alert rules.

```
GET /api/alert-notifications/lookup
```

Example request

```
GET /api/alert-notifications/lookup HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 1,
    "uid": "sns-uid",
    "name": "test",
    "type": "sns",
    "isDefault": false
```

```
  },
  {
    "id": 2,
    "uid": "slack-uid",
    "name": "Slack",
    "type": "slack",
    "isDefault": false
  }
]
```

Get all notification channels by UID

```
GET /api/alert-notifications/uid/:uid
```

Example request

```
GET /api/alert-notifications/uid/sns-uid HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "sns-uid",
  "name": "test",
  "type": "sns",
  "isDefault": false,
  "sendReminder": false,
  "disableResolveMessage": false,
  "frequency": "",
  "created": "2023-09-08T19:57:56Z",
  "updated": "2023-09-08T19:57:56Z",
  "settings": {
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
```

```
    "severity": "critical",
    "topic": "<SNS-TOPIC-ARN>",
    "uploadImage": false
  },
  "secureFields": {}
}
```

Get all notification channels by Id

```
GET /api/alert-notifications/:id
```

Example request

```
GET /api/alert-notifications/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "sns-uid",
  "name": "test",
  "type": "sns",
  "isDefault": false,
  "sendReminder": false,
  "disableResolveMessage": false,
  "frequency": "",
  "created": "2023-09-08T19:57:56Z",
  "updated": "2023-09-08T19:57:56Z",
  "settings": {
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
    "severity": "critical",
    "topic": "<SNS-TOPIC-ARN>",
    "uploadImage": false
```

```
  },
  "secureFields": {}
}
```

Create notification channel

To see what notification channels are supported by Amazon Managed Grafana, see the list of supported notifiers in Working with contact points.

```
POST /api/alert-notifications
```

Example request

```
POST /api/alert-notifications HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890

{
  "uid": "new-sns-uid", // optional
  "name": "sns alert notification",  //Required
  "type":  "sns", //Required
  "isDefault": false,
  "sendReminder": false,
  "settings": {
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
    "severity": "critical",
    "topic": "<SNS-TOPIC-ARN>",
    "uploadImage": false
  }
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
```

```
  "uid": "new-sns-uid",
  "name": "sns alert notification",
  "type": "sns",
  "isDefault": false,
  "sendReminder": false,
  "settings": {
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
    "severity": "critical",
    "topic": "<SNS-TOPIC-ARN>",
    "uploadImage": false
  },
  "created": "2018-04-23T14:44:09+02:00",
  "updated": "2018-08-20T15:47:49+02:00"
}
```

Update notification channel by UID

```
PUT /api/alert-notifications/uid/:uid
```

Example request

```
PUT /api/alert-notifications/uid/sns-uid HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890

{
  "uid": "sns-uid", // optional
  "name": "sns alert notification",  //Required
  "type":  "sns", //Required
  "isDefault": false,
  "sendReminder": true,
  "frequency": "15m",
  "settings": {
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
    "severity": "critical",
    "topic": "<SNS-TOPIC-ARN>",
```

```
    "uploadImage": false
  }
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "sns-uid",
  "name": "sns alert notification",
  "type": "sns",
  "isDefault": false,
  "sendReminder": true,
  "frequency": "15m",
  "settings": {
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
    "severity": "critical",
    "topic": "<SNS-TOPIC-ARN>",
    "uploadImage": false
  },
  "created": "2017-01-01 12:34",
  "updated": "2017-01-01 12:34"
}
```

Update notification channel by Id

```
PUT /api/alert-notifications/:id
```

Example request

```
PUT /api/alert-notifications/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890

{
```

```
  "id": 1,
  "uid": "sns-uid", // optional
  "name": "sns alert notification",  //Required
  "type":  "sns", //Required
  "isDefault": false,
  "sendReminder": true,
  "frequency": "15m",
  "settings": {
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
    "severity": "critical",
    "topic": "<SNS-TOPIC-ARN>",
    "uploadImage": false
  }
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "sns-uid",
  "name": "sns alert notification",
  "type": "sns",
  "isDefault": false,
  "sendReminder": true,
  "frequency": "15m",
  "settings": {
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
    "severity": "critical",
    "topic": "<SNS-TOPIC-ARN>",
    "uploadImage": false
  },
  "created": "2017-01-01 12:34",
  "updated": "2017-01-01 12:34"
}
```

Delete notification channel by UID

```
DELETE /api/alert-notifications/uid/:uid
```

Example request

```
DELETE /api/alert-notifications/uid/sns-uid HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "message": "Notification deleted"
}
```

Delete notification channel by Id

```
DELETE /api/alert-notifications/:id
```

Example request

```
DELETE /api/alert-notifications/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "message": "Notification deleted"
}
```

Test notification channel

Sends a test notification message for the given notification channel type and settings.

```
POST /api/alert-notifications/test
```

Example request

```
POST /api/alert-notifications/test HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer 1234abcd567exampleToken890

{
  "type":  "sns",
  "settings": {
    "authProvider": "default",
    "autoResolve": true,
    "httpMethod": "POST",
    "messageFormat": "json",
    "severity": "critical",
    "topic": "<SNS-TOPIC-ARN>",
    "uploadImage": false
  }
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "message": "Test notification sent"
}
```

Annotations API

Use the Annotations API to create, update, delete, and work with annotations in the Amazon Managed Grafana workspace.

Annotations are saved in the workspace's Grafana database (sqlite, mysql or postgres). Annotations can be global annotations that can be shown on any dashboard by configuring an annotation

data source. Annotations are filtered by tags. Or they can be tied to a panel on a dashboard, and displayed only on that panel.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Find annotations

```
GET /api/annotations?from=1506676478816&to=1507281278816&tags=tag1&tags=tag2&limit=100
```

Example request

```
GET /api/annotations?from=1506676478816&to=1507281278816&tags=tag1&tags=tag2&limit=100
 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Query parameters:

- **from**— (Optional) Epoch datetime in milliseconds.
- **to**— (Optional) Epoch datetime in milliseconds.
- **limit**— (Optional) Maximum number of results returned. The default is 100.
- **alertid**— (Optional) FInd annotations for the specified alert.
- **dashboardId**— (Optional) Find annotations that are scoped to the specified dashboard.
- **panelId**— (Optional) Find annotations that are scoped to the specified panel.
- **userId**— (Optional) Find annotations created by the specified user.
- **type**— (Optional) Specify to return alerts or user-created annotations. Value values are `alert` and `annotation`.
- **tags**— (Optional) Use this to filter global annotations. Global annotations are annotations from an annotation data source that are not connected specifically to a dashboard or panel. To do an “AND” filtering with multiple tags, specify the tags parameter multiple times. For example, `tags=tag1&tags=tag2`. These are Grafana tags, not AWS tags.

Example response

```
HTTP/1.1 200
Content-Type: application/json
[
    {
        "id": 1124,
        "alertId": 0,
        "dashboardId": 468,
        "panelId": 2,
        "userId": 1,
        "userName": "",
        "newState": "",
        "prevState": "",
        "time": 1507266395000,
        "timeEnd": 1507266395000,
        "text": "test",
        "metric": "",
        "tags": [
            "tag1",
            "tag2"
        ],
        "data": {}
    },
    {
        "id": 1123,
        "alertId": 0,
        "dashboardId": 468,
        "panelId": 2,
        "userId": 1,
        "userName": "",
        "newState": "",
        "prevState": "",
        "time": 1507265111000,
        "text": "test",
        "metric": "",
        "tags": [
            "tag1",
            "tag2"
        ],
        "data": {}
    }
]
```

Create annotation

```
POST /api/annotations
```

Creates an annotation in the workspace's Grafana database. The `dashboardId` and `panelId` fields are optional. If they are not specified, a global annotation is created and can be queried in any dashboard that adds the Grafana annotations data source. When creating a region annotation, be sure to include the `timeEnd` property.

The format for `time` and `timeEnd` should be epoch numbers in millisecond resolution.

Example request

```
POST /api/annotations HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "dashboardId":468,
  "panelId":1,
  "time":1507037197339,
  "timeEnd":1507180805056,
  "tags":["tag1","tag2"],
  "text":"Annotation Description"
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
    "message":"Annotation added",
    "id": 1,
}
```

Create annotation in graphite format

```
POST /api/annotations/graphite
```

Creates an annotation by using a Graphite-compatible event format. The `when` and `data` fields are optional. If `when` is not specified, the current time is used as annotation's timestamp. The `tags` field can also be in prior to Graphite 0.10.0 format (string with multiple tags being separated by a space).

Example request

```
POST /api/annotations/graphite HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "what": "Event - deploy",
  "tags": ["deploy", "production"],
  "when": 1467844481,
  "data": "deploy of master branch happened at Wed Jul 6 22:34:41 UTC 2016"
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
    "message":"Graphite annotation added",
    "id": 1
}
```

Update annotation

```
PUT /api/annotations/:id
```

pdates all properties of an annotation that matches the specified id. To only update certain properties, use the Patch Annotation operation.

Example request

```
PUT /api/annotations/1141 HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

```
Content-Type: application/json

{
  "time":1507037197339,
  "timeEnd":1507180805056,
  "text":"Annotation Description",
  "tags":["tag3","tag4","tag5"]
}
```

Example response:

```
HTTP/1.1 200
Content-Type: application/json

{
    "message":"Annotation updated"
}
```

Patch annotation

```
PATCH /api/annotations/:id
```

Updates one or more properties of an annotation that matches the specified id. This operation currently supports updating the `text`, `tags`, `time`, and `timeEnd` properties.

Example request:

```
PATCH /api/annotations/1145 HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
Content-Type: application/json

{
   "text":"New Annotation Description",
   "tags":["tag6","tag7","tag8"]
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
```

```
{
    "message":"Annotation patched"
}
```

Delete annotation by Id

```
DELETE /api/annotations/:id
```

Deletes the annotation that matches the specified Id.

Example request

```
DELETE /api/annotations/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
    "message":"Annotation deleted"
}
```

Authentication API

Use the Authentication API to work with authentication keys in an Amazon Managed Grafana workspace.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get API keys

```
GET /api/auth/keys
```

Example request

```
GET /api/auth/keys HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Query parameter:

- **includeExpired**— (Optional) Boolean parameter that specifies whether to include expired keys in the returned results. The default is `false`.

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {"id": 3,"name": "API","role": "Admin"},
  {"id": 1,"name": "TestAdmin","role": "Admin","expiration":
 "2019-06-26T10:52:03+03:00"}
]
```

Create API key

```
POST /api/auth/keys
```

Example request

```
POST /api/auth/keys HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "mykey",
  "role": "Admin",
  "secondsToLive": 86400
}
```

JSON body schema:

- **name**— The name for the key.
- **role**— Sets the access level (Grafana role) for the key. Valid values are `Admin`, `Editor`, or `Viewer`.
- **secondsToLive**— Sets the amount of time before the key expires. It must be 2592000 (30 days) or less.

Example response

```
{"name":"mykey","key":"eyJrIjoiWHZiSWd3NzdCYUZnNUtibE9obUpESmE3bzJYNDRIc0UiLCJuIjoibXlrZXkiLCJp
```

Error statuses:

- **400**— `secondsToLive` is greater than 2592000
- **500**— The key couldn't be stored in the database.

Delete API key

```
DELETE /api/auth/keys/:id
```

Example request

```
DELETE /api/auth/keys/3 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"API key deleted"}
```

Dashboard API

Use the Dashboard API to create, update, delete, and work with dashboards in the Amazon Managed Grafana workspace.

The identifier (id) of a dashboard is an auto-incrementing numeric value and is only unique per workspace. The unique identifier (uid) of a dashboard can be used to uniquely identify a dashboard between multiple Amazon Managed Grafana workspaces. It's automatically generated if you don't provide one when you create a dashboard. The uid allows having consistent URLs for accessing dashboards and when synchronizing dashboards between multiple workspaces. The use of the uid means that changing the title of a dashboard does not break any bookmarked links to that dashboard.

The uid can have a maximum length of 40 characters.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Create/Update dashboard

```
POST /api/dashboards/db
```

Creates a new dashboard or updates an existing dashboard.

Example request to create a new dashboard

```
POST /api/dashboards/db HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "dashboard": {
    "id": null,
    "uid": null,
    "title": "Production Overview",
    "tags": [ "templated" ],
    "timezone": "browser",
    "schemaVersion": 16,
    "version": 0,
    "refresh": "25s"
  },
  "folderId": 0,
```

```
  "folderUid": "l3KqBxCMz",
  "message": "Made changes to xyz",
  "overwrite": false
}
```

JSON body schema:

- **dashboard**— The complete dashboard model. Use null to create a new dashboard.
- **dashboard.id**— Use null to create a new dashboard.
- **dashboard.uid**— Optional unique identifer when you use this to create a new dashboard. If null, a new uid is generated.
- **folderid**— The id of the folder to save the dashboard in.
- **folderUid**— The Uid of the folder to save the dashboard in. Overrides the value of `folderid`
- **overwrite**— Specify `true` to overwrite an existing dashboard with a newer version, same dashboard title in folder or same dashboard uid.
- **message**— Set a commit message for the version history.
- **refresh**— Set the dashboard refresh interval. If this is lower than the minimum refresh interval, it is ignored and the minimum refresh interval is used.

To add or update an alert rule for a dashboard panel, declare a `dashboard.panels.alert` block.

Example request to update a dashboard alert rule

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 78

{
 "dashboard":  {
        "id": 104,
        "panels": [
            {
                "alert": {
                    "alertRuleTags": {},
                    "conditions": [
                        {
                            "evaluator": {
                                "params": [
                                    25
```

```
                        ],
                        "type": "gt"
                    },
                    "operator": {
                        "type": "and"
                    },
                    "query": {
                        "params": [
                            "A",
                            "5m",
                            "now"
                        ]
                    },
                    "reducer": {
                        "params": [],
                        "type": "avg"
                    },
                    "type": "query"
                }
            ],
            "executionErrorState": "alerting",
            "for": "5m",
            "frequency": "1m",
            "handler": 1,
            "name": "Panel Title alert",
            "noDataState": "no_data",
            "notifications": []
        },
        "aliasColors": {},
        "bars": false,
        "dashLength": 10,
        "dashes": false,
        "datasource": null,
        "fieldConfig": {
            "defaults": {
                "custom": {}
            },
            "overrides": []
        },
        "fill": 1,
        "fillGradient": 0,
        "gridPos": {
            "h": 9,
            "w": 12,
```

```
                    "x": 0,
                    "y": 0
                },
                "hiddenSeries": false,
                "id": 2,
                "legend": {
                    "avg": false,
                    "current": false,
                    "max": false,
                    "min": false,
                    "show": true,
                    "total": false,
                    "values": false
                },
                "lines": true,
                "linewidth": 1,
                "nullPointMode": "null",
                "options": {
                    "dataLinks": []
                },
                "percentage": false,
                "pointradius": 2,
                "points": false,
                "renderer": "flot",
                "seriesOverrides": [],
                "spaceLength": 10,
                "stack": false,
                "steppedLine": false,
                "targets": [
                    {
                        "refId": "A",
                        "scenarioId": "random_walk"
                    }
                ],
                "thresholds": [
                    {
                        "colorMode": "critical",
                        "fill": true,
                        "line": true,
                        "op": "gt",
                        "value": 50
                    }
                ],
                "timeFrom": null,
```

```
                "timeRegions": [],
                "timeShift": null,
                "title": "Panel Title",
                "tooltip": {
                    "shared": true,
                    "sort": 0,
                    "value_type": "individual"
                },
                "type": "graph",
                "xaxis": {
                    "buckets": null,
                    "mode": "time",
                    "name": null,
                    "show": true,
                    "values": []
                },
                "yaxes": [
                    {
                        "format": "short",
                        "label": null,
                        "logBase": 1,
                        "max": null,
                        "min": null,
                        "show": true
                    },
                    {
                        "format": "short",
                        "label": null,
                        "logBase": 1,
                        "max": null,
                        "min": null,
                        "show": true
                    }
                ],
                "yaxis": {
                    "align": false,
                    "alignLevel": null
                }
            }
        ],
        "title": "Update alert rule via API",
        "uid": "dHEquNzGz",
        "version": 1
    }
```

```
}
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 78

{
  "id":      1,
  "uid":     "cIBgcSjkk",
  "url":     "/d/cIBgcSjkk/production-overview",
  "status":  "success",
  "version": 1,
  "slug":    "production-overview" //deprecated in Grafana v5.0
}
```

Status Codes:

- **200**— Created
- **400**— Error such as invalid JSON, invalid or missing fields
- **401**— Unauthorized
- **403**— Access denied
- **412**— Precondition failed

The **412** status code is used to explain why the dashboard can't be created.

- The dashboard has been changed by someone else `status=version-mismatch`
- A dashboard with the same name in the folder already exists `status=name-exists`
- A dashboard with the same uid already exists `status=name-exists`
- The dashboard belongs to plugin `plugin title` `status=plugin-dashboard`

The response body has the following properties. If another dashboard has the same title, the `status` value is `name-exists`.

```
HTTP/1.1 412 Precondition Failed
Content-Type: application/json; charset=UTF-8
Content-Length: 97
```

```
{
  "message": "The dashboard has been changed by someone else",
  "status": "version-mismatch"
}
```

Get dashboard by uid

```
GET /api/dashboards/uid/:uid
```

Returns the dashboard matching the uid. The metadata returned might contain information about the UID of the folder that contains the dashboard.

Example request

```
GET /api/dashboards/uid/cIBgcSjkk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "dashboard": {
    "id": 1,
    "uid": "cIBgcSjkk",
    "title": "Production Overview",
    "tags": [ "templated" ],
    "timezone": "browser",
    "schemaVersion": 16,
    "version": 0
  },
  "meta": {
    "isStarred": false,
    "url": "/d/cIBgcSjkk/production-overview",
    "folderId": 2,
    "folderUid": "l3KqBxCMz",
    "slug": "production-overview" //deprecated in Grafana v5.0
  }
```

```
}
```

Status Codes:

- **200**— Found
- **401**— Unauthorized
- **403**— Access denied
- **404**— Not found

Delete dashboard by uid

```
DELETE /api/dashboards/uid/:uid
```

Deletes the dashboard matching the uid.

Example request

```
DELETE /api/dashboards/uid/cIBgcSjkk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "title": "Production Overview",
  "message": "Dashboard Production Overview deleted",
  "id": 2
}
```

Status Codes:

- **200**— Deleted
- **401**— Unauthorized
- **403**— Access denied
- **404**— Not found

Gets the home dashboard

```
GET /api/dashboards/home
```

Returns the home dashboard.

Example request

```
GET /api/dashboards/home HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "dashboard": {
    "editable":false,
    "hideControls":true,
    "nav":[
      {
        "enable":false,
        "type":"timepicker"
      }
    ],
    "style":"dark",
    "tags":[],
    "templating":{
      "list":[
      ]
    },
    "time":{
    },
    "timezone":"browser",
    "title":"Home",
    "version":5
  },
  "meta": {
    "isHome":true,
```

```
    "canSave":false,
    "canEdit":false,
    "canStar":false,
    "url":"",
    "expires":"0001-01-01T00:00:00Z",
    "created":"0001-01-01T00:00:00Z"
  }
}
```

Get dashboard tags

```
GET /api/dashboards/tags
```

Returns all tags of dashboards.

Example request

```
GET /api/dashboards/tags HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "term":"tag1",
    "count":1
  },
  {
    "term":"tag2",
    "count":4
  }
]
```

Dashboard Permissions API

Use the Dashboard Permissions API to update or retrieve the permissions for a dashboard.

Permissions with `dashboardId=-1` are the default permissions for users with the Viewer and Editor roles. Permissions can be set for a user, a team or a role (Viewer or Editor). Permissions cannot be set for Admins - they always have access to everything.

The permission levels for the `permission` field are as follows:

- 1 = View
- 2 = Edit
- 4 = Admin

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get permissions for a dashboard

```
GET /api/dashboards/id/:dashboardId/permissions
```

Gets all existing permissions for the dashboard with the given `dashboardId`.

Example request

```
GET /api/dashboards/id/1/permissions HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 551

[
  {
    "id": 1,
```

```
    "dashboardId": -1,
    "created": "2017-06-20T02:00:00+02:00",
    "updated": "2017-06-20T02:00:00+02:00",
    "userId": 0,
    "userLogin": "",
    "userEmail": "",
    "teamId": 0,
    "team": "",
    "role": "Viewer",
    "permission": 1,
    "permissionName": "View",
    "uid": "",
    "title": "",
    "slug": "",
    "isFolder": false,
    "url": ""
  },
  {
    "id": 2,
    "dashboardId": -1,
    "created": "2017-06-20T02:00:00+02:00",
    "updated": "2017-06-20T02:00:00+02:00",
    "userId": 0,
    "userLogin": "",
    "userEmail": "",
    "teamId": 0,
    "team": "",
    "role": "Editor",
    "permission": 2,
    "permissionName": "Edit",
    "uid": "",
    "title": "",
    "slug": "",
    "isFolder": false,
    "url": ""
  }
]
```

Status Codes:

- **200**— OK
- **401**— Unauthorized
- **403**— Access denied

- **404**— Dashboard not found

Update permissions for a dashboard

```
POST /api/dashboards/id/:dashboardId/permissions
```

Updates the permissions for a dashboard. This operation removes existing permissions if they are not included in the request.

Example request

```
POST /api/dashboards/id/1/permissions
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "items": [
    {
      "role": "Viewer",
      "permission": 1
    },
    {
      "role": "Editor",
      "permission": 2
    },
    {
      "teamId": 1,
      "permission": 1
    },
    {
      "userId": 11,
      "permission": 4
    }
  ]
}
```

JSON body schema:

- **items**— The permission items to add or update. Existing items that are omitted from the list are removed.

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Dashboard permissions updated"}
```

Status Codes:

- **200**— Created
- **401**— Unauthorized
- **403**— Access denied
- **404**— Dashboard not found

Dashboard Versions API

Use the Dashboard Versions API to retrieve dashboard versions and restore a dashboard to a specified version.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get all dashboard versions

```
GET /api/dashboards/id/:dashboardId/versions
```

Gets all existing dashboard versions for the dashboard with the given `dashboardId`.

Query parameters:

- **limit**— Maximum number of results to return.
- **start**— Version to start from when returning queries.

Example request

```
GET /api/dashboards/id/1/versions?limit=2?start=0 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 428

[
  {
    "id": 2,
    "dashboardId": 1,
    "parentVersion": 1,
    "restoredFrom": 0,
    "version": 2,
    "created": "2017-06-08T17:24:33-04:00",
    "createdBy": "admin",
    "message": "Updated panel title"
  },
  {
    "id": 1,
    "dashboardId": 1,
    "parentVersion": 0,
    "restoredFrom": 0,
    "version": 1,
    "created": "2017-06-08T17:23:33-04:00",
    "createdBy": "admin",
    "message": "Initial save"
  }
]
```

Status Codes:

- **200**— OK
- **400**— Errors
- **401**— Unauthorized
- **404**— Dashboard version not found

Get dashboard version

```
GET /api/dashboards/id/:dashboardId/versions/:id
```

Get the dashboard version with the given id, for the dashboard with the given `dashboardId`.

Example request

```
GET /api/dashboards/id/1/versions/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 1300

{
  "id": 1,
  "dashboardId": 1,
  "parentVersion": 0,
  "restoredFrom": 0,
  "version": 1,
  "created": "2017-04-26T17:18:38-04:00",
  "message": "Initial save",
  "data": {
    "annotations": {
      "list": [

      ]
    },
    "editable": true,
    "gnetId": null,
    "graphTooltip": 0,
    "hideControls": false,
    "id": 1,
    "links": [

    ],
    "rows": [
      {
```

```
        "collapse": false,
        "height": "250px",
        "panels": [

        ],
        "repeat": null,
        "repeatIteration": null,
        "repeatRowId": null,
        "showTitle": false,
        "title": "Dashboard Row",
        "titleSize": "h6"
      }
    ],
    "schemaVersion": 14,
    "style": "dark",
    "tags": [

    ],
    "templating": {
      "list": [

      ]
    },
    "time": {
      "from": "now-6h",
      "to": "now"
    },
    "timepicker": {
      "refresh_intervals": [
        "5s",
        "10s",
        "30s",
        "1m",
        "5m",
        "15m",
        "30m",
        "1h",
        "2h",
        "1d"
      ],
      "time_options": [
        "5m",
        "15m",
        "1h",
```

```
        "6h",
        "12h",
        "24h",
        "2d",
        "7d",
        "30d"
      ]
    },
    "timezone": "browser",
    "title": "test",
    "version": 1
  },
  "createdBy": "admin"
}
```

Status Codes:

- **200**— OK
- **401**— Unauthorized
- **404**— Dashboard version not found

Restore dashboard

```
POST /api/dashboards/id/:dashboardId/restore
```

Restores a dashboard to the dashboard version that you specify.

Example request

```
POST /api/dashboards/id/1/restore
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "version": 1
}
```

JSON body schema:

- **version**— The dashboard version to restore to.

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 67

{
  "slug": "my-dashboard",
  "status": "success",
  "version": 3
}
```

JSON response body schema:

- **slug**— The URL-friendly slug of the dashboard's title.
- **status**— Whether the restore was successful or not.
- **version**— The new dashboard version following the restore.

Status Codes:

- **200**— Created
- **401**— Unauthorized
- **404**— Dashboard or dashboard version not found
- **500**— Internal server error (indicates issue retrieving dashboard tags from database)

Example error response:

```
HTTP/1.1 404 Not Found
Content-Type: application/json; charset=UTF-8
Content-Length: 46

{
  "message": "Dashboard version not found"
}
```

JSON response body schema:

- **message**— A message explaining the reason for the failure.

Compare dashboard versions

```
POST /api/dashboards/calculate-diff
```

Compares two dashboard versions by calculating the JSON diff of them.

Example request

```
POST /api/dashboards/calculate-diff HTTP/1.1
Accept: text/html
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "base": {
    "dashboardId": 1,
    "version": 1
  },
  "new": {
    "dashboardId": 1,
    "version": 2
  },
  "diffType": "json"
}
```

JSON body schema:

- **base**— An object representing the base dashboard version.
- **new**— An object representing the new dashboard version.
- **difftype**— The type of diff to return. Valid values are `json` and `basic`.

Example response (JSON diff)

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8

<p id="v10-l1" class="diff-line diff-json-same">
  <!-- Diff omitted -->
</p>
```

The response is a textual representation of the diff, with the dashboard values being in JSON, similar to the diffs seen on sites like GitHub or GitLab.

Status Codes:

- **200**— OK
- **200**— Bad request, invalid JSON sent
- **401**— Unauthorized
- **404**— Not found

Example response (Basic diff)

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8

<div class="diff-group">
  <!-- Diff omitted -->
</div>
```

The response is a summary of the changes, derived from the diff between the two JSON objects.

Status Codes:

- **200**— OK
- **200**— Bad request, invalid JSON sent
- **401**— Unauthorized
- **404**— Not found

Data Source API

Use the Data Source API to create, update, delete, and work with data sources in the Amazon Managed Grafana workspace.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get all data sources

```
GET /api/datasources
```

Example request

```
GET /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
   {
     "id": 1,
     "orgId": 1,
     "uid": "H8joYFVGz"
     "name": "datasource_elastic",
     "type": "elasticsearch",
     "typeLogoUrl": "public/app/plugins/datasource/elasticsearch/img/
elasticsearch.svg",
     "access": "proxy",
     "url": "http://mydatasource.com",
     "password": "",
     "user": "",
     "database": "grafana-dash",
     "basicAuth": false,
     "isDefault": false,
     "jsonData": {
         "esVersion": 5,
         "logLevelField": "",
         "logMessageField": "",
         "maxConcurrentShardRequests": 256,
         "timeField": "@timestamp"
     },
     "readOnly": false
   }
]
```

Get a single data source by Id

```
GET /api/datasources/:datasourceId
```

Example request

```
GET /api/datasources/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "kLtEtcRGk",
  "orgId": 1,
  "name": "test_datasource",
  "type": "graphite",
  "typeLogoUrl": "",
  "access": "proxy",
  "url": "http://mydatasource.com",
  "password": "",
  "user": "",
  "database": "",
  "basicAuth": false,
  "basicAuthUser": "",
  "basicAuthPassword": "",
  "withCredentials": false,
  "isDefault": false,
  "jsonData": {
    "graphiteType": "default",
    "graphiteVersion": "1.1"
  },
  "secureJsonFields": {},
  "version": 1,
  "readOnly": false
}
```

Get a single data source by UID

```
GET /api/datasources/uid/:uid
```

Example request

```
GET /api/datasources/uid/kLtEtcRGk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "kLtEtcRGk",
  "orgId": 1,
  "name": "test_datasource",
  "type": "graphite",
  "typeLogoUrl": "",
  "access": "proxy",
  "url": "http://mydatasource.com",
  "password": "",
  "user": "",
  "database": "",
  "basicAuth": false,
  "basicAuthUser": "",
  "basicAuthPassword": "",
  "withCredentials": false,
  "isDefault": false,
  "jsonData": {
    "graphiteType": "default",
    "graphiteVersion": "1.1"
  },
  "secureJsonFields": {},
  "version": 1,
  "readOnly": false
}
```

Get a single data source by name

```
GET /api/datasources/name/:name
```

Example request

```
GET /api/datasources/name/test_datasource HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "kLtEtcRGk",
  "orgId": 1,
  "name": "test_datasource",
  "type": "graphite",
  "typeLogoUrl": "",
  "access": "proxy",
  "url": "http://mydatasource.com",
  "password": "",
  "user": "",
  "database": "",
  "basicAuth": false,
  "basicAuthUser": "",
  "basicAuthPassword": "",
  "withCredentials": false,
  "isDefault": false,
  "jsonData": {
    "graphiteType": "default",
    "graphiteVersion": "1.1"
  },
  "secureJsonFields": {},
  "version": 1,
  "readOnly": false
}
```

Get data source Id by name

```
GET /api/datasources/id/:name
```

Example request

```
GET /api/datasources/id/test_datasource HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1
}
```

Create a data source

```
POST /api/datasources
```

Example Graphite request

```
POST /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name":"test_datasource",
  "type":"graphite",
  "url":"http://mydatasource.com",
  "access":"proxy",
  "basicAuth":false
}
```

Example Graphite response

```
HTTP/1.1 200
Content-Type: application/json

{
  "datasource": {
    "id": 1,
    "orgId": 1,
    "name": "test_datasource",
    "type": "graphite",
    "typeLogoUrl": "",
    "access": "proxy",
    "url": "http://mydatasource.com",
    "password": "",
    "user": "",
    "database": "",
    "basicAuth": false,
    "basicAuthUser": "",
    "basicAuthPassword": "",
    "withCredentials": false,
    "isDefault": false,
    "jsonData": {},
    "secureJsonFields": {},
    "version": 1,
    "readOnly": false
  },
  "id": 1,
  "message": "Datasource added",
  "name": "test_datasource"
}
```

Note

When you define `password` and `basicAuthPassword` within `secureJsonData`, Amazon Managed Grafana encryptes them securely as an encrypted blob in the database. The response then lists the encrypte fields in `secureJsonFields`.

Example Graphite request with basic auth enabled

```
POST /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
```

```
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "test_datasource",
  "type": "graphite",
  "url": "http://mydatasource.com",
  "access": "proxy",
  "basicAuth": true,
  "basicAuthUser": "basicuser",
  "secureJsonData": {
    "basicAuthPassword": "basicpassword"
  }
}
```

Example response with basic auth enabled

```
HTTP/1.1 200
Content-Type: application/json

{
  "datasource": {
    "id": 1,
    "orgId": 1,
    "name": "test_datasource",
    "type": "graphite",
    "typeLogoUrl": "",
    "access": "proxy",
    "url": "http://mydatasource.com",
    "password": "",
    "user": "",
    "database": "",
    "basicAuth": true,
    "basicAuthUser": "basicuser",
    "basicAuthPassword": "",
    "withCredentials": false,
    "isDefault": false,
    "jsonData": {},
    "secureJsonFields": {
      "basicAuthPassword": true
    },
    "version": 1,
    "readOnly": false
  },
```

```
  "id": 102,
  "message": "Datasource added",
  "name": "test_datasource"
}
```

Example CloudWatch request

```
POST /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "test_datasource",
  "type": "cloudwatch",
  "url": "http://monitoring.us-west-1.amazonaws.com",
  "access": "proxy",
  "jsonData": {
    "authType": "keys",
    "defaultRegion": "us-west-1"
  },
  "secureJsonData": {
    "accessKey": "Ol4pIDpeKSA6XikgOl4p",
    "secretKey": "dGVzdCBrZXkgYmxlYXNlIGRvbid0IHN0ZWFs"
  }
}
```

Update an existing data source

```
PUT /api/datasources/:datasourceId
```

Example request

```
PUT /api/datasources/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "id":1,
  "orgId":1,
  "name":"test_datasource",
```

```
  "type":"graphite",
  "access":"proxy",
  "url":"http://mydatasource.com",
  "password":"",
  "user":"",
  "database":"",
  "basicAuth":true,
  "basicAuthUser":"basicuser",
  "secureJsonData": {
    "basicAuthPassword": "basicpassword"
  },
  "isDefault":false,
  "jsonData":null
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "datasource": {
    "id": 1,
    "orgId": 1,
    "name": "test_datasource",
    "type": "graphite",
    "typeLogoUrl": "",
    "access": "proxy",
    "url": "http://mydatasource.com",
    "password": "",
    "user": "",
    "database": "",
    "basicAuth": true,
    "basicAuthUser": "basicuser",
    "basicAuthPassword": "",
    "withCredentials": false,
    "isDefault": false,
    "jsonData": {},
    "secureJsonFields": {
      "basicAuthPassword": true
    },
    "version": 1,
    "readOnly": false
```

```
  },
  "id": 102,
  "message": "Datasource updated",
  "name": "test_datasource"
}
```

> **Note**
>
> We recommend that you define `password` and `basicAuthPassword` within `secureJsonData` so that they are stored securely as an encrypted blob in the database. The response then lists the encrypte fields in `secureJsonFields`.

Delete data source by Id

```
DELETE /api/datasources/:datasourceId
```

Example request

```
DELETE /api/datasources/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Data source deleted"}
```

Delete data source by UID

```
DELETE /api/datasources/uid/:uid
```

Example request

```
DELETE /api/datasources/uid/kLtEtcRGk HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Data source deleted"}
```

Delete data source by name

```
DELETE /api/datasources/name/:datasourceName
```

Example request

```
DELETE /api/datasources/name/test_datasource HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "message":"Data source deleted",
  "id": 1
}
```

Data source proxy calls

```
GET /api/datasources/proxy/:datasourceId/*
```

Proxies all calls to the actual data source.

Query data sources

```
POST /api/ds/query
```

Queries a data source having a backend implementation. Most built-in data sources have backend implementation.

Example request

```
POST /api/ds/query HTTP/1.1
Accept: application/json
Content-Type: application/json

{
   "queries":[
      {
         "refId":"A",
         "scenarioId":"csv_metric_values",
         "datasource":{
            "uid":"PD8C576611E62080A"
         },
         "format": "table",
         "maxDataPoints":1848,
         "intervalMs":200,
         "stringInput":"1,20,90,30,5,0"
      }
   ],
   "from":"now-5m",
   "to":"now"
}
```

JSON body schema:

- **from/to**— Specifies the time range for the queries. The time can be either epoch timestamps in milliseconds or relative using Grafana time units. For example, `now-5m`.
- **queries**— Specifies one or more queries. Must contain at least 1.
- **queries.datasource.uid**— Specifies the UID of data source to be queried. Each query in the request must have a unique `datasource`.
- **queries.refId**— Specifies an identifier of the query. Defaults to “A”.
- **queries.format**— Specifies the format the data should be returned in. Valid options are `time_series` or `table` depending on the data source.
- **queries.maxDataPoints**— Species the maximum amount of data points that a dashboard panel can render. Defaults to 100.
- **queries.intervalMs**— Specifies the time series time interval in milliseconds. Defaults to 1000.

In addition, specific properties of each data source should be added in a request (for example **queries.stringInput** as shown in the request above). To better understand how to form a query for a certain data source, use the Developer Tools in your browser of choice and inspect the HTTP requests being made to `/api/ds/query`.

Example test data source time series query response

```
HTTP/1.1 200
Content-Type: application/json

{
  "results": {
    "A": {
      "frames": [
        {
          "schema": {
            "refId": "A",
            "fields": [
              {
                "name": "time",
                "type": "time",
                "typeInfo": {
                  "frame": "time.Time"
                }
              },
              {
                "name": "A-series",
                "type": "number",
                "typeInfo": {
                  "frame": "int64",
                  "nullable": true
                }
              }
            ]
          },
          "data": {
            "values": [
              [1644488152084, 1644488212084, 1644488272084, 1644488332084,
 1644488392084, 1644488452084],
              [1, 20, 90, 30, 5, 0]
            ]
          }
        }
```

```
      ]
    }
  }
}
```

Query data source by Id

```
POST /api/tsdb/query
```

> **Important**
>
> Starting Version 9, `/api/tsdb/query` is not supported. Use Query data sources.

Queries a data source having backend implementation. Most built-in data sources have backend implementation.

Example request

```
POST /api/tsdb/query HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "from": "1420066800000",
  "to": "1575845999999",
  "queries": [
    {
      "refId": "A",
      "intervalMs": 86400000,
      "maxDataPoints": 1092,
      "datasourceId": 86,
      "rawSql": "SELECT 1 as valueOne, 2 as valueTwo",
      "format": "table"
    }
  ]
}
```

> **Note**
>
> The `from`, `to`, and `queries` properties are required.

JSON body schema:

- **from/to**— Must be either absolute in epoch timestamps in milliseconds, or relative using Grafana time units. For example, `now-1h`.
- **queries.refId**— (Optional) Specifies an identifier for the query. The default is `A`.
- **queries.datasourceId**— Specifies the data source to be queried. Each query in the request must have a unique `datasourceId`.
- **queries.maxDataPoints**— (Optional) Specifies the maximum amount of data points that a dashboard panel can render. The default is 100.
- **queries.intervalIMs**— (Optional) Specifies the time interval in milliseconds of time series. The default is 1000

Example request for the MySQL data source:

```
POST /api/tsdb/query HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "from": "1420066800000",
  "to": "1575845999999",
  "queries": [
    {
      "refId": "A",
      "intervalMs": 86400000,
      "maxDataPoints": 1092,
      "datasourceId": 86,
      "rawSql": "SELECT\n  time,\n  sum(opened) AS \"Opened\",\n  sum(closed) AS
 \"Closed\"\nFROM\n  issues_activity\nWHERE\n  $__unixEpochFilter(time) AND\n  period
 = 'm' AND\n  repo IN('grafana/grafana') AND\n  opened_by IN('Contributor','Grafana
 Labs')\nGROUP BY 1\nORDER BY 1\n",
      "format": "time_series"
    }
  ]
```

```
}
```

Example response for the MySQL data source request:

```
HTTP/1.1 200
Content-Type: application/json

{
  "results": {
    "A": {
      "refId": "A",
      "meta": {
        "rowCount": 0,
        "sql": "SELECT\n  time,\n  sum(opened) AS \"Opened\",\n  sum(closed) AS
 \"Closed\"\nFROM\n  issues_activity\nWHERE\n  time <= 1420066800 AND time >=
 1575845999 AND\n  period = 'm' AND\n  repo IN('grafana/grafana') AND\n  opened_by
 IN('Contributor','Grafana Labs')\nGROUP BY 1\nORDER BY 1\n"
      },
      "series": [
        {
          "name": "Opened",
          "points": [
            [
              109,
              1420070400000
            ],
            [
              122,
              1422748800000
            ]
          ]
        },
        {
          "name": "Closed",
          "points": [
            [
              89,
              1420070400000
            ]
          ]
        }
      ]
    }
```

```
  }
}
```

Data Source Permissions API

Use the Data Source Permissions API to enable, disable, list, add, and remove permissions for data sources.

You can set permissions for a user or a team. Permissions can't be set for Admins, because they always have access to everything.

The permission levels for the permission field are as follows:

- 1 = Query

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Enable permissions for a data source

```
POST /api/datasources/:id/enable-permissions
```

Enables permissions for the data source with the given id. No one except Org Admins are able to query the data source until permissions have been added to permit certain users or teams to query the data source.

Example request

```
POST /api/datasources/1/enable-permissions
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Datasource permissions enabled"}
```

Status Codes:

- **200**— Created
- **400**— Permissions can't be enabled, see the response body for details.
- **401**— Unauthorized
- **403**— Access denied
- **404**— Data source not found

Disable permissions for a data source

```
POST /api/datasources/:id/disable-permissions
```

Disables permissions for the data source with the given id. All existing permissions are removed and anyone is able to query the data source.

Example request

```
POST /api/datasources/1/disable-permissions
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{}
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Datasource permissions disabled"}
```

Status Codes:

- **200**— Ok
- **400**— Permissions can't be disabled, see the response body for details.
- **401**— Unauthorized
- **403**— Access denied
- **404**— Data source not found

Get permissions for a data source

```
GET /api/datasources/:id/permissions
```

Gets all existing permissions for the data source with the given `id`.

Example request

```
GET /api/datasources/1/permissions HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 551

{
  "datasourceId": 1,
  "enabled": true,
  "permissions":
  [
    {
      "id": 1,
      "datasourceId": 1,
      "userId": 1,
      "userLogin": "user",
      "userEmail": "user@test.com",
      "userAvatarUrl": "/avatar/46d229b033af06a191ff2267bca9ae",
      "permission": 1,
      "permissionName": "Query",
      "created": "2017-06-20T02:00:00+02:00",
```

```
      "updated": "2017-06-20T02:00:00+02:00",
    },
    {
      "id": 2,
      "datasourceId": 1,
      "teamId": 1,
      "team": "A Team",
      "teamAvatarUrl": "/avatar/46d229b033af06a191ff2267bca9ae",
      "permission": 1,
      "permissionName": "Query",
      "created": "2017-06-20T02:00:00+02:00",
      "updated": "2017-06-20T02:00:00+02:00",
    }
  ]
}
```

Status Codes:

- **200**— Ok
- **401**— Unauthorized
- **403**— Access denied
- **404**— Data source not found

Add permission for a data source

```
POST /api/datasources/:id/permissions
```

Adds a user permission for the data source with the given `id`.

Example request to add user permission

```
POST /api/datasources/1/permissions
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "userId": 1,
  "permission": 1
}
```

Example response for adding a user permission

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Datasource permission added"}
```

Example request to add team permission

```
POST /api/datasources/1/permissions
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "teamId": 1,
  "permission": 1
}
```

Example response for adding a team permission

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Datasource permission added"}
```

Status Codes:

- **200**— Ok
- **400**— Permission can't be added, see response body for details.
- **401**— Unauthorized
- **403**— Access denied
- **404**— Data source not found

Remove permission for a data source

```
DELETE /api/datasources/:id/permissions/:permissionId
```

Removes the permission with the given permissionId for the data source with the given `id`.

Example request

```
DELETE /api/datasources/1/permissions/2
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Datasource permission removed"}
```

Status Codes:

- **200**— Ok
- **401**— Unauthorized
- **403**— Access denied
- **404**— Data source not found or permission not found

External Group Synchronization API

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get external groups

```
GET /api/teams/:teamId/groups
```

Example request

```
GET /api/teams/1/groups HTTP/1.1
```

```
Accept: application/json
Content-Type: application/json
Authorization Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk]
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "orgId": 1,
    "teamId": 1,
    "groupId": "cn=editors,ou=groups,dc=grafana,dc=org"
  }
]
```

Status Codes:

- **200**— Ok
- **401**— Unauthorized
- **403**— Access denied

Add external group

```
POST /api/teams/:teamId/groups
```

Example request

```
POST /api/teams/1/members HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk]

{
  "groupId": "cn=editors,ou=groups,dc=grafana,dc=org"
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Group added to Team"}
```

Status Codes:

- **200**— Ok
- **400**— Group is already added to this team
- **401**— Unauthorized
- **403**— Permission denied
- **404**— Team not found

Remove external group

```
DELETE /api/teams/:teamId/groups/:groupId
```

Example request

```
DELETE /api/teams/1/groups/cn=editors,ou=groups,dc=grafana,dc=org HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk]
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Team Group removed"}
```

Status Codes:

- **200**— Ok
- **401**— Unauthorized
- **403**— Permission denied
- **404**— Team not found or group not found

Folder API

Use the Folder API to work with folders in the Amazon Managed Grafana workspace.

The identifier (id) of a folder is an auto-incrementing numeric value and is only unique per workspace. The unique identifier (uid) of a folder can be used to uniquely identify a folder between multiple workspaces. It's automatically generated if you don't provide one when you create a folder. The uid allows having consistent URLs for accessing folder and when synchronizing folder between multiple Amazon Managed Grafana workspaces. The use of the uid means that changing the title of a folder does not break any bookmarked links to that folder.

The uid can have a maximum length of 40 characters.

Folders can't be nested.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

The **General** folder, with an `id` of 0, is not part of the Folder API. You can't use the Folder API to retrieve information about the general folder.

Create folder

```
POST /api/folders
```

Creates a new folder.

Example request

```
POST /api/folders HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "uid": "nErXDvCkzz",
  "title": "Department ABC"
```

```
}
```

JSON body schema:

- **uid**— Optional unique identifer. If null, a new uid is generated.
- **title**— The title for the folder.

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
  "uid": "nErXDvCkzz",
  "title": "Department ABC",
  "url": "/dashboards/f/nErXDvCkzz/department-abc",
  "hasAcl": false,
  "canSave": true,
  "canEdit": true,
  "canAdmin": true,
  "createdBy": "admin",
  "created": "2018-01-31T17:43:12+01:00",
  "updatedBy": "admin",
  "updated": "2018-01-31T17:43:12+01:00",
  "version": 1
}
```

Status Codes:

- **200**— Created
- **400**— Error such as invalid JSON, invalid or missing fields
- **401**— Unauthorized
- **403**— Access denied

Update folder

```
PUT /api/folders/:uid
```

Updates the existing folder that matches the uid.

Example request

```
PUT /api/folders/nErXDvCkzz HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "title":"Department DEF",
  "version": 1
}
```

JSON body schema:

- **uid**— Changes the unique identifer, if provided.
- **title**— The title of the folder.
- **version**— Provide the current version to be able to overwrite the folder. Not needed if `overwrite=true`.
- **overwrite**— Set to `true` to overwrite the existing folder with a newer version.

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
  "uid": "nErXDvCkzz",
  "title": "Department DEF",
  "url": "/dashboards/f/nErXDvCkzz/department-def",
  "hasAcl": false,
  "canSave": true,
  "canEdit": true,
  "canAdmin": true,
  "createdBy": "admin",
  "created": "2018-01-31T17:43:12+01:00",
  "updatedBy": "admin",
  "updated": "2018-01-31T17:43:12+01:00",
  "version": 1
```

```
}
```

Status Codes:

- **200**— Created
- **400**— Error such as invalid JSON, invalid or missing fields
- **401**— Unauthorized
- **403**— Access denied
- **404**— Folder not found
- **412**— Precondition failed

The **412** status code is used to explain why the folder can't be updated.

- The folder has been changed by someone else `status=version-mismatch`

The response body has the following properties:

```
HTTP/1.1 412 Precondition Failed
Content-Type: application/json; charset=UTF-8
Content-Length: 97

{
  "message": "The folder has been changed by someone else",
  "status": "version-mismatch"
}
```

Get all folders

```
GET /api/folders
```

Returns all folders that you have permission to view. You can control the maximum number of folders returned by using the `limit` query parameter. The default is 1000.

Example request

```
GET /api/folders?limit=10 HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id":1,
    "uid": "nErXDvCkzz",
    "title": "Department ABC"
  },
  {
    "id":2,
    "uid": "k3S1cklGk",
    "title": "Department RND"
  }
]
```

Get folder by uid

```
GET /api/folders/:uid
```

Returns all folders that matches the given uid.

Example request

```
GET /api/folders/nErXDvCkzzh HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
```

```
  "uid": "nErXDvCkzz",
  "title": "Department ABC",
  "url": "/dashboards/f/nErXDvCkzz/department-abc",
  "hasAcl": false,
  "canSave": true,
  "canEdit": true,
  "canAdmin": true,
  "createdBy": "admin",
  "created": "2018-01-31T17:43:12+01:00",
  "updatedBy": "admin",
  "updated": "2018-01-31T17:43:12+01:00",
  "version": 1
}
```

Status Codes:

- **200**— Found
- **401**— Unauthorized
- **403**— Access denied
- **404**— Not found

Get folder by id

```
GET /api/folders/id/:id
```

Returns the folder that matches the given id.

Example request

```
GET /api/folders/id/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
```

```
  "id":1,
  "uid": "nErXDvCkzz",
  "title": "Department ABC",
  "url": "/dashboards/f/nErXDvCkzz/department-abc",
  "hasAcl": false,
  "canSave": true,
  "canEdit": true,
  "canAdmin": true,
  "createdBy": "admin",
  "created": "2018-01-31T17:43:12+01:00",
  "updatedBy": "admin",
  "updated": "2018-01-31T17:43:12+01:00",
  "version": 1
}
```

Status Codes:

- **200**— Found
- **401**— Unauthorized
- **403**— Access denied
- **404**— Not found

Delete folder by uid

```
DELETE /api/folders/:uid
```

Deletes the folder matching the uid, and also deletes all dashboards stored in the folder. This oepration can't be reverted.

Example request

```
DELETE /api/folders/nErXDvCkzz HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
```

```
Content-Type: application/json

{
  "message":"Folder deleted",
  "id": 2
}
```

Status Codes:

- **200**— Deleted
- **401**— Unauthorized
- **403**— Access denied
- **404**— Not found

Folder/Dashboard Search API

Use the FolderDashboard-Search API to search folders and dashboards in an Amazon Managed Grafana workspace.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Search folders and dashboards

```
GET /api/search/
```

Query parameters:

- **query**— Search query
- **tag**— List of tags to search for. These are Grafana tags, not AWS tags.
- **type**— The type to search for, either `dash-folder` or `dash-db`.
- **dashboardIds**— List of dashboard Id's to search for.
- **folderIds**— List of dashboard Id's to search for in dashboards.

- **starred**— Flag to specify that only starred dashboards are to be returned.
- **limit**— Limit the number of returned results (maximum is 5000).
- **page**— Use this parameter to access hits beyond limit. Numbering starts at 1. The `limit` parameter acts as page size.

Example request for retrieving folders and dashboards of the general folder

```
GET /api/search?folderIds=0&query=&starred=false HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response for retrieving folders and dashboards of the general folder

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 163,
    "uid": "000000163",
    "title": "Folder",
    "url": "/dashboards/f/000000163/folder",
    "type": "dash-folder",
    "tags": [],
    "isStarred": false,
    "uri":"db/folder" // deprecated in Grafana v5.0
  },
  {
    "id":1,
    "uid": "cIBgcSjkk",
    "title":"Production Overview",
    "url": "/d/cIBgcSjkk/production-overview",
    "type":"dash-db",
    "tags":[prod],
    "isStarred":true,
    "uri":"db/production-overview" // deprecated in Grafana v5.0
  }
]
```

Example request for searching for starred dashboards

```
GET /api/search?query=Production%20Overview&starred=true&tag=prod HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response for searching for starred dashboards

```
HTTP/1.1 200
Content-Type: application/json

[HTTP/1.1 200
Content-Type: application/json

[
  {
    "id":1,
    "uid": "cIBgcSjkk",
    "title":"Production Overview",
    "url": "/d/cIBgcSjkk/production-overview",
    "type":"dash-db",
    "tags":[prod],
    "isStarred":true,
    "folderId": 2,
    "folderUid": "000000163",
    "folderTitle": "Folder",
    "folderUrl": "/dashboards/f/000000163/folder",
    "uri":"db/production-overview" // deprecated in Grafana v5.0
  }
]
```

Folder Permissions API

Use the Folder API to update or retrieve the permissions for a folder.

Permissions with `folderId=-1` are the default permissions for users with the Viewer and Editor roles. Permissions can be set for a user, a team or a role (Viewer or Editor). Permissions cannot be set for Admins - they always have access to everything.

The permission levels for the `permission` field are as follows:

- 1 = View

- 2 = Edit
- 4 = Admin

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get permissions for a folder

```
GET /api/folders/:uid/permissions
```

Gets all existing permissions for the folder with the given `uid`.

Example request

```
GET /api/folders/nErXDvCkzz/permissions HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 551

[
  {
    "id": 1,
    "folderId": -1,
    "created": "2017-06-20T02:00:00+02:00",
    "updated": "2017-06-20T02:00:00+02:00",
    "userId": 0,
    "userLogin": "",
    "userEmail": "",
    "teamId": 0,
    "team": "",
```

```
    "role": "Viewer",
    "permission": 1,
    "permissionName": "View",
    "uid": "nErXDvCkzz",
    "title": "",
    "slug": "",
    "isFolder": false,
    "url": ""
  },
  {
    "id": 2,
    "dashboardId": -1,
    "created": "2017-06-20T02:00:00+02:00",
    "updated": "2017-06-20T02:00:00+02:00",
    "userId": 0,
    "userLogin": "",
    "userEmail": "",
    "teamId": 0,
    "team": "",
    "role": "Editor",
    "permission": 2,
    "permissionName": "Edit",
    "uid": "",
    "title": "",
    "slug": "",
    "isFolder": false,
    "url": ""
  }
]
```

Status Codes:

- **200**— OK
- **401**— Unauthorized
- **403**— Access denied
- **404**— Folder not found

Update permissions for a folder

```
POST /api/folders/:uid/permissions
```

Updates the permissions for a folder. This operation removes existing permissions if they are not included in the request.

Example request

```
POST /api/folders/nErXDvCkzz/permissions
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
{
  "items": [
    {
      "role": "Viewer",
      "permission": 1
    },
    {
      "role": "Editor",
      "permission": 2
    },
    {
      "teamId": 1,
      "permission": 1
    },
    {
      "userId": 11,
      "permission": 4
    }
  ]
}
```

JSON body schema:

- **items**— The permission items to add or update. Existing items that are omitted from the list are removed.

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 35

{"message":"Folder permissions updated","id":1,"title":"Department ABC"}
```

Status Codes:

- **200**— Created
- **401**— Unauthorized
- **403**— Access denied
- **404**— Dashboard not found

Organization API

Use the Organization API to work with organizations in an Amazon Managed Grafana workspace.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get current organization

```
GET /api/org/
```

Example request

```
GET /api/org/ HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "id":1,
  "name":"Main Org."
}
```

Get all users within the current organization

```
GET /api/org/users
```

Required permissions: the `org.users:read` action with the scope `users:*`

Example request

```
GET /api/org/users HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "orgId": 1,
    "userId": 1,
    "email": "admin@localhost",
    "avatarUrl": "/avatar/46d229b033af06a191ff2267bca9ae",
    "login": "admin",
    "role": "Admin",
    "lastSeenAt": "2019-08-09T11:02:49+02:00",
    "lastSeenAtAge": "< 1m"
  }
]
```

Get all users within the current organization (lookup)

```
GET /api/org/users/lookup
```

Returns all users within the current organization, but with less detailed information. Accessible to users with org admin role, admin in any folder or admin of any team. Used mostly by the Grafana UI to provide a list of users when adding team members and when e diting folder/dashboard permissions.

Example request

```
GET /api/org/users/lookup HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "userId": 1,
    "login": "admin",
    "avatarUrl": "/avatar/46d229b033af06a191ff2267bca9ae"
  }
]
```

Updates the given user

```
PATCH /api/org/users/:userId
```

Required permissions: the `org.users.role:update` action with the scope `users:*`

Example request

```
PATCH /api/org/users/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "role": "Viewer",
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Organization user updated"}
```

Deletes user in current organization

```
DELETE /api/org/users/:userId
```

Required permissions: the `org.users:remove` action with the scope `users:*`

Example request

```
DELETE /api/org/users/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"User removed from organization"}
```

Update the current organization

```
PUT /api/org
```

Example request

```
PUT /api/org HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name":"Main Org."
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
```

```
{"message":"Organization updated"}
```

Add user to the current organization

```
POST /api/org/users
```

Required permissions: the `org.users:add` action with the scope `users:*`

Example request

```
POST /api/org/users HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "role": "Admin",
  "loginOrEmail": "admin"
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"User added to organization","userId":1}
```

Playlist API

Use the Playlist API to work with playlists in the Amazon Managed Grafana workspace.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Search playlist

```
GET /api/playlists
```

Returns all playlists for the current Amazon Managed Grafana workspace, using pagination.

Example request

```
GET /api/playlists HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Querystring parameters:

- **query**— Limit the responses to playlists that have a name like this value.
- **limit**— Limit the response to X number of playlists.

Example response

```
HTTP/1.1 200
Content-Type: application/json
[
  {
    "id": 1,
    "name": "my playlist",
    "interval": "5m"
  }
]
```

Get one playlist

```
GET /api/playlists/:id
```

Example request

```
GET /api/playlists/1 HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
```

```
Content-Type: application/json
{
  "id" : 1,
  "name": "my playlist",
  "interval": "5m",
  "orgId": "my org",
  "items": [
    {
      "id": 1,
      "playlistId": 1,
      "type": "dashboard_by_id",
      "value": "3",
      "order": 1,
      "title":"my third dashboard"
    },
    {
      "id": 2,
      "playlistId": 1,
      "type": "dashboard_by_tag",
      "value": "myTag",
      "order": 2,
      "title":"my other dashboard"
    }
  ]
}
```

Get playlist items

```
GET /api/playlists/:id/items
```

Example request

```
GET /api/playlists/1/items HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
[
  {
```

```
    "id": 1,
    "playlistId": 1,
    "type": "dashboard_by_id",
    "value": "3",
    "order": 1,
    "title":"my third dashboard"
  },
  {
    "id": 2,
    "playlistId": 1,
    "type": "dashboard_by_tag",
    "value": "myTag",
    "order": 2,
    "title":"my other dashboard"
  }
]
```

Get playlist dashboards

```
GET /api/playlists/:id/dashboards
```

Example request

```
GET /api/playlists/1/dashboards HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
[
  {
    "id": 3,
    "title": "my third dashboard",
    "order": 1,
  },
  {
    "id": 5,
    "title":"my other dashboard"
    "order": 2,
```

```
  }
]
```

Create a playlist

```
POST /api/playlists/
```

Example request

```
PUT /api/playlists/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
  {
    "name": "my playlist",
    "interval": "5m",
    "items": [
      {
        "type": "dashboard_by_id",
        "value": "3",
        "order": 1,
        "title":"my third dashboard"
      },
      {
        "type": "dashboard_by_tag",
        "value": "myTag",
        "order": 2,
        "title":"my other dashboard"
      }
    ]
  }
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
  {
    "id": 1,
    "name": "my playlist",
    "interval": "5m"
  }
```

Update a playlist

```
PUT /api/playlists/:id
```

Example request

```
PUT /api/playlists/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
  {
    "name": "my playlist",
    "interval": "5m",
    "items": [
      {
        "playlistId": 1,
        "type": "dashboard_by_id",
        "value": "3",
        "order": 1,
        "title":"my third dashboard"
      },
      {
        "playlistId": 1,
        "type": "dashboard_by_tag",
        "value": "myTag",
        "order": 2,
        "title":"my other dashboard"
      }
    ]
  }
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
{
  "id" : 1,
  "name": "my playlist",
  "interval": "5m",
  "orgId": "my org",
  "items": [
    {
```

```
      "id": 1,
      "playlistId": 1,
      "type": "dashboard_by_id",
      "value": "3",
      "order": 1,
      "title":"my third dashboard"
    },
    {
      "id": 2,
      "playlistId": 1,
      "type": "dashboard_by_tag",
      "value": "myTag",
      "order": 2,
      "title":"my other dashboard"
    }
  ]
}
```

Delete a playlist

```
DELETE /api/playlists/:id
```

Example request

```
DELETE /api/playlists/1 HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
{}
```

Plugin API

Use the Plugin API to manage plugins in the Amazon Managed Grafana workspace. To make changes to plugins with this API, the workspace must have plugin management enabled for your workspace. The user defined by the Grafana API key must also be an admin for the Amazon Managed Grafana workspace.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Install plugin

```
POST /api/plugins/:id/install
```

Example request

```
POST /api/plugins/grafana-athena-datasource/install HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "version": "2.12.0" # optional, uses the latest compatible version if not provided
}
```

Example response

```
HTTP/1.1 200
```

Uninstall plugin

```
POST /api/plugins/:id/uninstall
```

Example request

```
POST /api/plugins/grafana-athena-datasource/uninstall HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "version": "2.12.0" # optional, uninstalls whatever is installed if not provided
}
```

Example response

```
HTTP/1.1 200
```

Get all plugins

```
GET /api/gnet/plugins
```

Example request

```
GET /api/gnet/plugins HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
{
  "items": [
    {
      "status": "active",
      "id": 74,
      "typeId": 1,
      "typeName": "Application",
      "typeCode": "app",
      "slug": "alexanderzobnin-zabbix-app",
      "name": "Zabbix",
      "description": "Zabbix plugin for Grafana",
      "version": "4.4.3",
      "versionStatus": "active",
      "versionSignatureType": "grafana",
      "versionSignedByOrg": "grafana",
      "versionSignedByOrgName": "Grafana Labs",
      "userId": 0,
      "orgId": 13056,
      "orgName": "Alexander Zobnin",
      "orgSlug": "alexanderzobnin",
      "orgUrl": "https://github.com/alexanderzobnin",
      "url": "https://github.com/grafana/grafana-zabbix/",
```

```
      "createdAt": "2016-04-06T20:23:41.000Z",
      "updatedAt": "2023-10-10T12:53:51.000Z",
      "downloads": 90788771,
      "verified": false,
      "featured": 180,
      "internal": false,
      "downloadSlug": "alexanderzobnin-zabbix-app",
      "popularity": 0.2485,
      "signatureType": "grafana",
      "packages": {
        "linux-amd64": {
          "md5": "baa06e8f26731f99748c58522cd4ffb6",
          "sha256": "a4a108f2e04a2114810c7b60419b4b04bf80d3377e2394b0586e2dc96b5a929c",
          "packageName": "linux-amd64",
          "downloadUrl": "/api/plugins/alexanderzobnin-zabbix-app/versions/4.4.3/
download?os=linux&arch=amd64"
        },
        <... further packages>
      },
      "links": [
        {
          "rel": "self",
          "href": "/plugins/alexanderzobnin-zabbix-app"
        },
        <... further links>
      ],
      "angularDetected": false
    },
    <... further plugins>
  ],
  "orderBy": "weight",
  "direction": "asc",
  "links": [
    {
      "rel": "self",
      "href": "/plugins"
    }
  ]
}
```

Get plugin

```
GET /api/gnet/plugins/:id
```

Example request

```
GET /api/gnet/plugins/grafana-athena-datasource HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
{
  "status": "active",
  "id": 764,
  "typeId": 2,
  "typeName": "Data Source",
  "typeCode": "datasource",
  "slug": "grafana-athena-datasource",
  "name": "Amazon Athena",
  "description": "Use Amazon Athena with Grafana",
  "version": "2.13.0",
  "versionStatus": "active",
  "versionSignatureType": "grafana",
  "versionSignedByOrg": "grafana",
  "versionSignedByOrgName": "Grafana Labs",
  "userId": 0,
  "orgId": 5000,
  "orgName": "Grafana Labs",
  "orgSlug": "grafana",
  "orgUrl": "https://grafana.org",
  "url": "https://github.com/grafana/athena-datasource/",
  "createdAt": "2021-11-24T08:55:41.000Z",
  "updatedAt": "2023-10-31T17:20:32.000Z",
  "json": {
    "$schema": "https://raw.githubusercontent.com/grafana/grafana/master/docs/sources/
developers/plugins/plugin.schema.json",
    "alerting": true,
    "annotations": true,
    "backend": true,
    "dependencies": {
      "grafanaDependency": ">=8.0.0",
      "plugins": []
    },
```

```
    "executable": "gpx_athena",
    "id": "grafana-athena-datasource",
    "includes": [
      {
        "name": "Cost Usage Report Monitoring",
        "path": "dashboards/cur-monitoring.json",
        "type": "dashboard"
      },
      {
        "name": "Amazon VPC Flow Logs",
        "path": "dashboards/vpc-flow-logs.json",
        "type": "dashboard"
      }
    ],
    "info": {
      "author": {
        "name": "Grafana Labs",
        "url": "https://grafana.com"
      },
      "build": {
        "time": 1698764559022,
        "repo": "https://github.com/grafana/athena-datasource",
        "branch": "main",
        "hash": "25cc131300f1ed22593bc3ba08b2bef7d23fbcd01",
        "build": 1462
      },
      "description": "Use Amazon Athena with Grafana",
      "keywords": [
        "datasource",
        "athena"
      ],
      "links": [
        {
          "name": "Website",
          "url": "https://github.com/grafana/athena-datasource"
        },
        {
          "name": "License",
          "url": "https://github.com/grafana/athena-datasource/blob/master/LICENSE"
        }
      ],
      "logos": {
        "large": "img/logo.svg",
        "small": "img/logo.svg"
```

```
      },
      "screenshots": [],
      "updated": "2023-10-31",
      "version": "2.13.0"
    },
    "metrics": true,
    "name": "Amazon Athena",
    "type": "datasource"
  },
  "readme": "<... full HTML readme>",
  "statusContext": "",
  "downloads": 2505825,
  "verified": false,
  "featured": 0,
  "internal": false,
  "downloadSlug": "grafana-athena-datasource",
  "popularity": 0.0594,
  "signatureType": "grafana",
  "grafanaDependency": ">=8.0.0",
  "packages": {
    "linux-amd64": {
      "md5": "7efef359bf917b4ca6b149de42a1282d",
      "sha256": "cd2fc5737c321dc3d8bbe2852c801c01adb64eacc9f60420bd21dc18bee43531",
      "packageName": "linux-amd64",
      "downloadUrl": "/api/plugins/grafana-athena-datasource/versions/2.13.0/download?
os=linux&arch=amd64"
    },
    <... other packages>
  },
  "links": [
    {
      "rel": "self",
      "href": "/plugins/grafana-athena-datasource"
    },
    <... other links>
  ],
  "angularDetected": false
}
```


Get plugin versions

```
POST /api/gnet/plugins/:id/versions
```

Example request

```
GET /api/gnet/plugins/grafana-athena-datasource/versions HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
{
  "items": [
    {
      "id": 5306,
      "pluginId": 764,
      "pluginSlug": "grafana-athena-datasource",
      "version": "2.13.0",
      "url": "https://github.com/grafana/athena-datasource/",
      "commit": "",
      "description": "Use Amazon Athena with Grafana",
      "createdAt": "2023-10-31T17:20:31.000Z",
      "updatedAt": null,
      "downloads": 33790,
      "verified": false,
      "status": "active",
      "statusContext": "",
      "downloadSlug": "grafana-athena-datasource",
      "packages": {},
      "links": [
        {
          "rel": "self",
          "href": "/plugins/grafana-athena-datasource/versions/2.13.0"
        },
        {
          "rel": "images",
          "href": "/plugins/grafana-athena-datasource/versions/2.13.0/images"
        },
        {
          "rel": "thumbnails",
          "href": "/plugins/grafana-athena-datasource/versions/2.13.0/thumbnails"
        },
        {
```

```
          "rel": "plugin",
          "href": "/plugins/grafana-athena-datasource"
        },
        {
          "rel": "download",
          "href": "/plugins/grafana-athena-datasource/versions/2.13.0/download"
        }
      ],
      "grafanaDependency": ">=8.0.0",
      "angularDetected": false
    },
    {
      "id": 5244,
      "pluginId": 764,
      "pluginSlug": "grafana-athena-datasource",
      "version": "2.12.0",
      "url": "https://github.com/grafana/athena-datasource/",
      "commit": "",
      "description": "Use Amazon Athena with Grafana",
      "createdAt": "2023-10-17T12:42:13.000Z",
      "updatedAt": null,
      "downloads": 60742,
      "verified": false,
      "status": "active",
      "statusContext": "",
      "downloadSlug": "grafana-athena-datasource",
      "packages": {},
      "links": [
        {
          "rel": "self",
          "href": "/plugins/grafana-athena-datasource/versions/2.12.0"
        },
        {
          "rel": "images",
          "href": "/plugins/grafana-athena-datasource/versions/2.12.0/images"
        },
        {
          "rel": "thumbnails",
          "href": "/plugins/grafana-athena-datasource/versions/2.12.0/thumbnails"
        },
        {
          "rel": "plugin",
          "href": "/plugins/grafana-athena-datasource"
        },
```

```
        {
          "rel": "download",
          "href": "/plugins/grafana-athena-datasource/versions/2.12.0/download"
        }
      ],
      "grafanaDependency": ">=8.0.0",
      "angularDetected": false
    },
    <... other versions>
  ]
}
```

Preferences API

Use the Preferences API to work with user preferences in the Amazon Managed Grafana workspace.

Keys:

- **theme**— Valid values are `light`, `dark`, or an empty string to use the default theme.
- **homeDashboardId**— The numerical `:id` of a favorited dashboard. The default is 0.
- **timezone**— Valid values are `utc`, `browser`, or an empty string to use the default.

Omitting a key causes the current value to be replaced with the system default value.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get current user preferences

```
GET /api/user/preferences
```

Example request

```
GET /api/user/preferences HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"theme":"","homeDashboardId":0,"timezone":""}
```

Update current user preferences

```
PUT /api/user/preferences
```

Example request

```
PUT /api/user/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "theme": "",
  "homeDashboardId":0,
  "timezone":"utc"
}
```

Example response

```
HTTP/1.1 200
Content-Type: text/plain; charset=utf-8

{"message":"Preferences updated"}
```

Get current org preferences

```
GET /api/org/preferences
```

Example request

```
GET /api/org/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"theme":"","homeDashboardId":0,"timezone":""}
```

Update current org preferences

```
PUT /api/org/preferences
```

Example request

```
PUT /api/org/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "theme": "",
  "homeDashboardId":0,
  "timezone":"utc"
}
```

Example response

```
HTTP/1.1 200
Content-Type: text/plain; charset=utf-8

{"message":"Preferences updated"}
```

Snapshot API

Use the Snapshot API to work with snapshots in an Amazon Managed Grafana workspace.

Note

To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Create new shapshot

```
POST /api/snapshots
```

Example request

```
POST /api/snapshots HTTP/1.1
    Accept: application/json
    Content-Type: application/json
    Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

    {
      "dashboard": {
        "editable":false,
        "hideControls":true,
        "nav":[
        {
          "enable":false,
        "type":"timepicker"
        }
        ],
        "rows": [
          {

          }
        ],
        "style":"dark",
        "tags":[],
        "templating":{
          "list":[
          ]
        },
        "time":{
        },
        "timezone":"browser",
        "title":"Home",
```

```
        "version":5
        },
      "expires": 3600
    }
```

JSON body schema:

- **dashboard**— (Required) The complete dashboard model.
- **name**— (Optional) A name for the snapshot.
- **expires**— (Optional) When the snapshot should expire, in seconds. The default is to never expire.
- **external**— (Optional) Save the snapshot on an external server rather than locally. Default is false.
- **key**— (Required if `external` is `true`) Define a unique key.
- **deletekey**— (Required if `external` is `true`) A unique key to be used to delete the snapshot. It is different than `key` so that only the creator can delete the snapshot.

> **Note**
>
> When creating a snapshot using the API, you have to provide the full dashboard payload including the snapshot data. This endpoint is designed for the Grafana UI.

Example response

```
HTTP/1.1 200
    Content-Type: application/json
    {
      "deleteKey":"XXXXXXX",
      "deleteUrl":"myurl/api/snapshots-delete/XXXXXXX",
      "key":"YYYYYYY",
      "url":"myurl/dashboard/snapshot/YYYYYYY",
      "id": 1,
    }
```

Keys:

- **deleteKey**— A key generated to be used to delete the snapshot.
- **key**— A key generated to share the dashboard.

Get list of snapshots

```
GET /api/dashboard/snapshots
```

Query parameters:

- **query**— Search query
- **limit**— Limit the number of returned results

Example request

```
GET /api/dashboard/snapshots HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id":8,
    "name":"Home",
    "key":"YYYYYYY",
    "orgId":1,
    "userId":1,
    "external":false,
    "externalUrl":"",
    "expires":"2200-13-32T25:23:23+02:00",
    "created":"2200-13-32T28:24:23+02:00",
    "updated":"2200-13-32T28:24:23+02:00"
  }
]
```

Get snapshot by key

```
GET /api/snapshots/:key
```

Example request

```
GET /api/snapshots/YYYYYYY HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "meta":{
    "isSnapshot":true,
    "type":"snapshot",
    "canSave":false,
    "canEdit":false,
    "canStar":false,
    "slug":"",
    "expires":"2200-13-32T25:23:23+02:00",
    "created":"2200-13-32T28:24:23+02:00"
  },
  "dashboard": {
    "editable":false,
    "hideControls":true,
    "nav": [
      {
        "enable":false,
        "type":"timepicker"
      }
    ],
    "rows": [
      {

      }
    ],
    "style":"dark",
    "tags":[],
    "templating":{
      "list":[
      ]
    },
    "time":{
    },
    "timezone":"browser",
```

```
    "title":"Home",
    "version":5
  }
}
```

Delete snapshot by key

```
DELETE /api/snapshots/:key
```

Example request

```
DELETE /api/snapshots/YYYYYYY HTTP/1.1
Accept: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Snapshot deleted. It might take an hour before it's cleared from any CDN
 caches.", "id": 1}
```

Delete snapshot by deleteKey

This API call can be used without authentication by using the secret delete key for the snapshot.

```
GET /api/snapshots-delete/:deleteKey
```

Example request

```
GET /api/snapshots-delete/XXXXXXX HTTP/1.1
Accept: application/json
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
```

```
{"message":"Snapshot deleted. It might take an hour before it's cleared from any CDN
 caches.", "id": 1}
```

Team API

Use the Team API to work with teams in an Amazon Managed Grafana workspace. All actions in this API require that you have the Admin role.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Team search with pagination

```
GET /api/teams/search?perpage=50&page=1&query=myteam
```

or

```
GET /api/teams/search?name=myteam
```

Example request

```
GET /api/teams/search?perpage=10&page=1&query=myteam HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Using the query parameter

The default value for the `perpage` parameter is 1000 and for the `page` parameter is 1.

The `totalCount` field in the response can be used for pagination of the teams list. For example, if `totalCount` is 100 teams and the`perpage` parameter is set to 10, then there are 10 pages of teams.

The `query` parameter is optional and returns results where the query value is contained in the `name` field. Query values with spaces need to be URL-encoded. For example, `query=my%20team`.

Using the name parameter

The name parameter returns a single team if the parameter matches the name field.

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "totalCount": 1,
  "teams": [
    {
      "id": 1,
      "orgId": 1,
      "name": "MyTestTeam",
      "email": "",
      "avatarUrl": "\/avatar\/3f49c15916554246daa714b9bd0ee39",
      "memberCount": 1
    }
  ],
  "page": 1,
  "perPage": 1000
```

Status Codes:

- **200**— Created
- **401**— Unauthorized
- **403**— Permission denied
- **404**— Team not found (if searching by name)

Get team by Id

```
GET /api/teams/:id
```

Example request

```
GET /api/teams/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
```

```
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HHTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "orgId": 1,
  "name": "MyTestTeam",
  "email": "",
  "created": "2017-12-15T10:40:45+01:00",
  "updated": "2017-12-15T10:40:45+01:00"
}
```

Add a team

The `name` of the team must be unique. The `name` field is required and the `email` and `orgId` fields are optional.

```
POST /api/teams
```

Example request

```
POST /api/teams HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "MyTestTeam",
  "email": "email@test.com",
  "orgId": 2
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
```

```
{"message":"Team created","teamId":2}
```

Status Codes:

- **200**— Created
- **401**— Unauthorized
- **403**— Permission denied
- **409**— Team name already exists

Update team

```
PUT /api/teams/:id
```

Only the `name` and `email` fields can be updated.

Example request

```
PUT /api/teams/2 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "MyTestTeam",
  "email": "email@test.com"
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Team updated"}
```

Status Codes:

- **200**— Created
- **401**— Unauthorized

- **403**— Permission denied
- **404**— Team not found
- **409**— Team name already exists

Delete team by Id

```
DELETE /api/teams/:id
```

Example request

```
DELETE /api/teams/2 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Team deleted"}
```

Status Codes:

- **200**— Created
- **401**— Unauthorized
- **403**— Permission denied
- **404**— Team not found

Get team members

```
GET /api/teams/:teamId/members
```

Example request

```
GET /api/teams/1/members HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "orgId": 1,
    "teamId": 1,
    "userId": 3,
    "email": "user1@email.com",
    "login": "user1",
    "avatarUrl": "\/avatar\/1b3c32f6386b0185c40d359cdc733a7"
  },
  {
    "orgId": 1,
    "teamId": 1,
    "userId": 2,
    "email": "user2@email.com",
    "login": "user2",
    "avatarUrl": "\/avatar\/cad3c68da76e45d10269e8ef02f8e7"
  }
]
```

Status Codes:

- **200**— Created
- **401**— Unauthorized
- **403**— Permission denied

Add team member

```
POST /api/teams/:teamId/members
```

Example request

```
POST /api/teams/1/members HTTP/1.1
```

```
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "userId": 2
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Member added to Team"}
```

Status Codes:

- **200**— Created
- **400**— User is already in team
- **401**— Unauthorized
- **403**— Permission denied
- **404**— Team not found

Remove member from team

```
DELETE /api/teams/:teamId/members/:userId
```

Example request

```
DELETE /api/teams/2/members/3 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
```

```
{"message":"Team Member removed"}
```

Status Codes:

- **200**— Created
- **401**— Unauthorized
- **403**— Permission denied
- **404**— Team not found/team member not found

Get team preferences

```
GET /api/teams/:teamId/preferences
```

Example request

```
GET /api/teams/2/preferences HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "theme": "",
  "homeDashboardId": 0,
  "timezone": ""
}
```

Update team preferences

```
PUT /api/teams/:teamId/preferences
```

Example request

```
PUT /api/teams/2/preferences HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "theme": "dark",
  "homeDashboardId": 39,
  "timezone": "utc"
}
```

JSON body schema:

- **theme**— Specify either `light`, `dark`, or an empty string to use the default theme.
- **homeDashboardId**— The numerical `:id` of a dashboard. The default is 0.
- **timezone**— Specify either `utc`, `browser`, or an empty string to use the default.

Omitting a parameter causes the current value to be replaced with the system default value.

Example response

```
HTTP/1.1 200
Content-Type: text/plain; charset=utf-8

{
  "message":"Preferences updated"
}
```

User API

Use the User API to work with users in an Amazon Managed Grafana workspace.

> **Note**
>
> To use a Grafana API with your Amazon Managed Grafana workspace, you must have a valid service account token. You include this in the `Authorization` field in the API request.

Get teams that the user is a member of

```
GET /api/user/teams
```

Example request

```
GET /api/user/teams HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
    "id": 1,
    "orgId": 1,
    "name": "MyTestTeam",
    "email": "",
    "avatarUrl": "\/avatar\/3f49c15916554246daa714b9bd0ee3",
    "memberCount": 1
  }
]
```

Get list of snapshots

Stars the given Dashboard for the actual user.

```
POST /api/user/stars/dashboard/:dashboardId
```

Example request

```
POST /api/user/stars/dashboard/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json
```

```
{"message":"Dashboard starred!"}
```

Unstar a dashboard

Deletes the starring of the given Dashboard for the actual user.

```
DELETE /api/user/stars/dashboard/:dashboardId
```

Example request

```
DELETE /api/user/stars/dashboard/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Dashboard unstarred"}
```

Get auth tokens of the actual user

```
GET /api/user/auth-tokens
```

Example request

```
GET /api/user/auth-tokens HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

[
  {
```

```
    "id": 361,
    "isActive": true,
    "clientIp": "127.0.0.1",
    "browser": "Chrome",
    "browserVersion": "72.0",
    "os": "Linux",
    "osVersion": "",
    "device": "Other",
    "createdAt": "2019-03-05T21:22:54+01:00",
    "seenAt": "2019-03-06T19:41:06+01:00"
  },
  {
    "id": 364,
    "isActive": false,
    "clientIp": "127.0.0.1",
    "browser": "Mobile Safari",
    "browserVersion": "11.0",
    "os": "iOS",
    "osVersion": "11.0",
    "device": "iPhone",
    "createdAt": "2019-03-06T19:41:19+01:00",
    "seenAt": "2019-03-06T19:41:21+01:00"
  }
]
```

Revoke an auth token of the actual user

```
POST /api/user/revoke-auth-token
```

Revokes the given auth token (device) for the actual user. User of issued auth token (device) are no longer logged in and are required to authenticate again at their next activity.

Example request

```
POST /api/user/revoke-auth-token HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "authTokenId": 364
}
```

Example response

```
HTTP/1.1 200
Content-Type: application/json

{
  "message": "User auth token revoked"
}
```

Observability solutions

You can use Amazon Managed Grafana to monitor systems or applications. AWS provides solutions to help you create a base monitoring setup for different kinds of computing systems. These solutions use Amazon Managed Service for Prometheus an Amazon Managed Grafana to monitor Amazon EKS solutions that give you insights into your application or service.

There are solutions for monitoring the **Amazon EKS infrastructure**, a **Java Virtual Machine (JVM) application** running in Amazon EKS, and an **Apache Kafka application** running on JVM in Amazon EKS.

Observability solutions

- Solution for Monitoring Amazon EKS infrastructure with Amazon Managed Grafana
- Solution for monitoring JVM applications with Amazon Managed Grafana
- Solution for monitoring Kafka applications with Amazon Managed Grafana

Solution for Monitoring Amazon EKS infrastructure with Amazon Managed Grafana

Monitoring Amazon Elastic Kubernetes Service infrastructure is one of the most common scenarios for which Amazon Managed Grafana are used. This page describes a template that provides you with a solution for this scenario. The solution can be installed using AWS Cloud Development Kit (AWS CDK) or with Terraform.

This solution configures:

- Your Amazon Managed Service for Prometheus workspace to store metrics from your Amazon EKS cluster, and creates a managed collector to scrape the metrics and push them to that workspace. For more information, see Ingest metrics with AWS managed collectors.
- Gathering logs from your Amazon EKS cluster using a CloudWatch agent. The logs are stored in CloudWatch, and queried by Amazon Managed Grafana. For more information, see Logging for Amazon EKS
- Your Amazon Managed Grafana workspace to pull those logs and metrics, and create dashboards and alerts to help you monitor your cluster.

Applying this solution will create dashboards and alerts that:

- Assess the overall Amazon EKS cluster health.
- Show the health and performance of the Amazon EKS control plane.
- Show the health and performance of the Amazon EKS data plane.
- Display insights on Amazon EKS workloads across Kubernetes namespaces.
- Display resource usage across namespaces, including CPU, memory, disk, and network usage.

About this solution

This solution configures an Amazon Managed Grafana workspace to provide metrics for your Amazon EKS cluster. The metrics are used to generate dashboards and alerts.

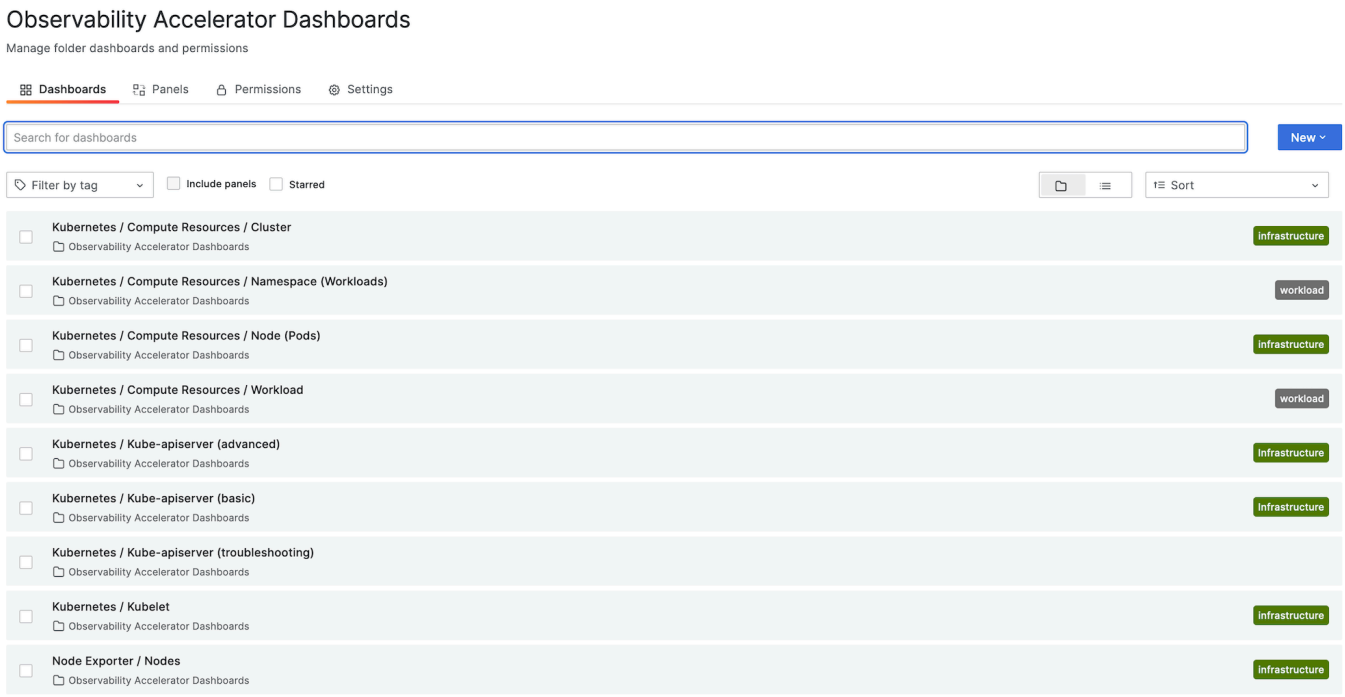
The metrics help you to operate Amazon EKS clusters more effectively by providing insights into the health and performance of the Kubernetes control and data plane. You can understand your Amazon EKS cluster from the node level, to pods, down to the Kubernetes level, including detailed monitoring of resource usage.

The solution provides both anticipatory and corrective capabilities:

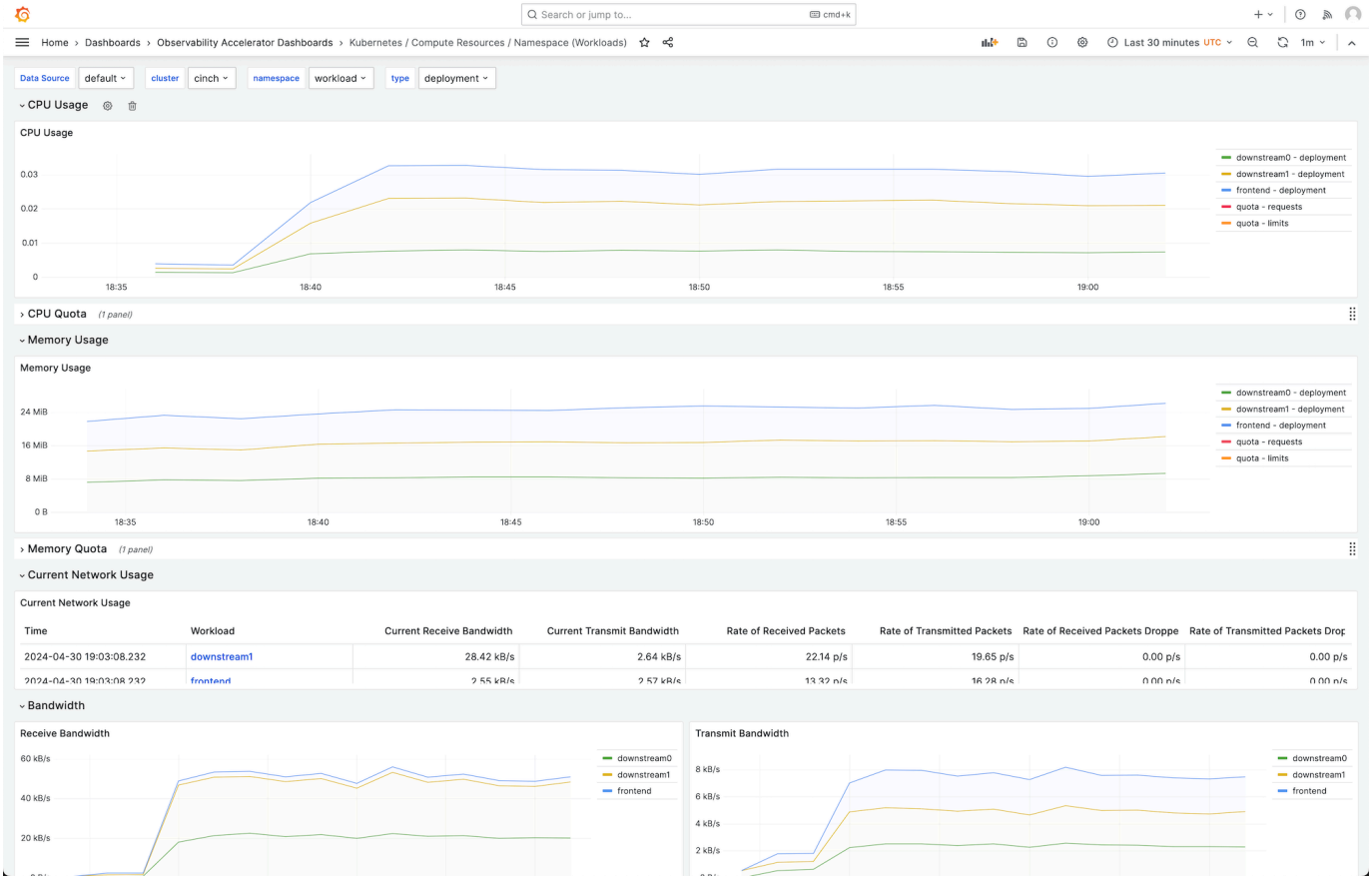
- **Anticipatory** capabilities include:
  - Manage resource efficiency by driving scheduling decisions. For example, to provide performance and reliability SLAs to your internal users of the Amazon EKS cluster you can allocate enough CPU and memory resources to their workloads based on tracking historical usage.
  - Usage forecasts: Based on the current utilization of your Amazon EKS cluster resources such as nodes, Persistent Volumes backed by Amazon EBS, or Application Load Balancers you can plan ahead, for example, for a new product or project with similar demands.
  - Detect potential issues early: For example, by analyzing resource consumption trends on a Kubernetes namespace level, you can understand the seasonality of the workload's usage.
- **Corrective** capabilities include:
  - Decrease the mean time to detection (MTTD) of issues on the infrastructure and the Kubernetes workload level. For example, by looking at the troubleshooting dashboard, you can quickly test hypotheses about what went wrong and eliminate them.

- Determine where in the stack a problem is happening. For example, the Amazon EKS control plane is fully managed by AWS and certain operations such as updating a Kubernetes deployment may fail if the API server is overloaded or connectivity is impacted.

The following image shows a sample of the dashboard folder for the solution.

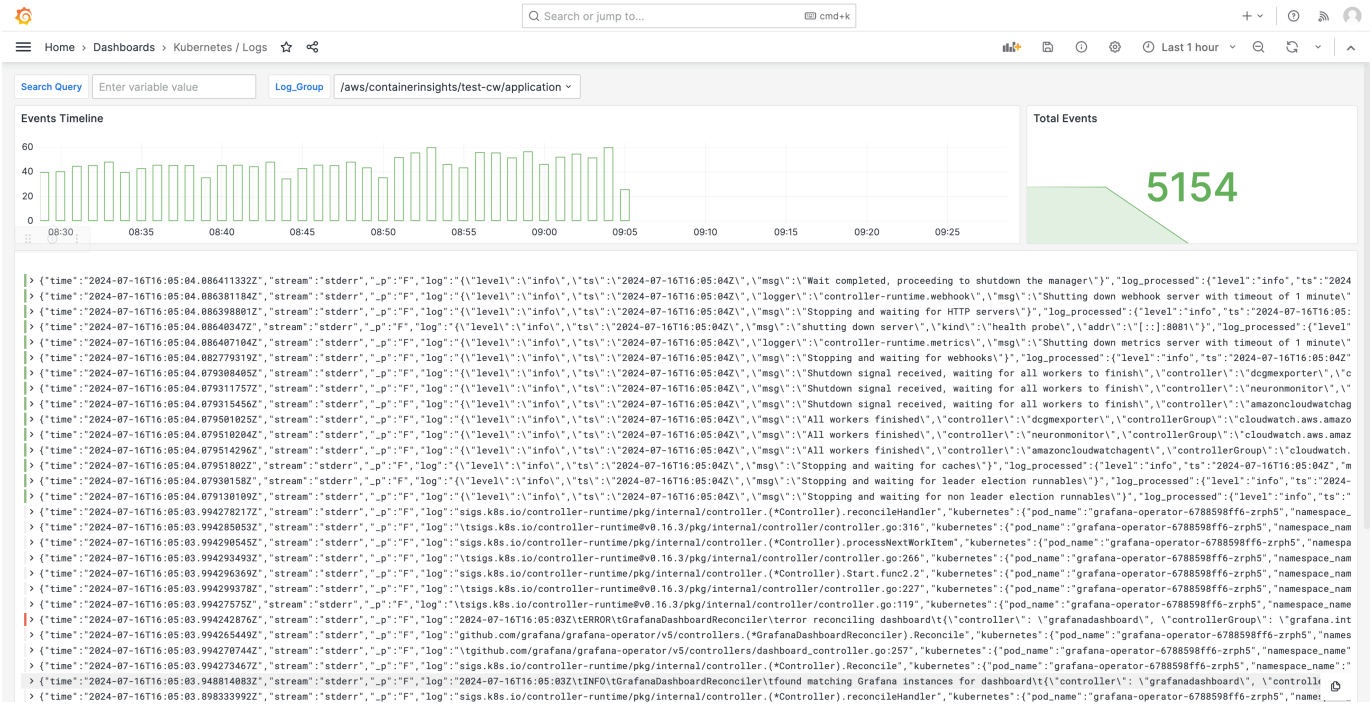


You can choose a dashboard to see more details, for example, choosing to view the Compute Resources for workloads will show a dashboard, such as that shown in the following image.



The metrics are scraped with a 1 minute scrape interval. The dashboards show metrics aggregated to 1 minute, 5 minutes, or more, based on the specific metric.

Logs are shown in dashboards, as well, so that you can query and analyze logs to find root causes of issues. The following image shows a log dashboard.



For a list of metrics tracked by this solution, see List of metrics tracked.

For a list of alerts created by the solution, see List of alerts created.

Costs

This solution creates and uses resources in your workspace. You will be charged for standard usage of the resources created, including:

- Amazon Managed Grafana workspace access by users. For more information about pricing, see Amazon Managed Grafana pricing.
- Amazon Managed Service for Prometheus metric ingestion and storage, including use of the Amazon Managed Service for Prometheus agentless collector, and metric analysis (query sample processing). The number of metrics used by this solution depends on the Amazon EKS cluster configuration and usage.

  You can view the ingestion and storage metrics in Amazon Managed Service for Prometheus using CloudWatch For more information, see CloudWatch metrics in the *Amazon Managed Service for Prometheus User Guide*.

  You can estimate the cost using the pricing calculator on the Amazon Managed Service for Prometheus pricing page. The number of metrics will depend on the number of nodes in your cluster, and the metrics your applications produce.

- CloudWatch Logs ingestion, storage, and analysis. By default, the log retention is set to never expire. You can adjust this in CloudWatch. For more information on pricing, see Amazon CloudWatch Pricing.
- Networking costs. You may incur standard AWS network charges for cross availability zone, Region, or other traffic.

The pricing calculators, available from the pricing page for each product, can help you understand potential costs for your solution. The following information can help get a base cost, for the solution running in the same availability zone as the Amazon EKS cluster.

Product	Calculator metric	Value
Amazon Managed Service for Prometheus	Active series	8000 (base) 15,000 (per node)
	Avg Collection Interval	60 (seconds)
Amazon Managed Service for Prometheus (managed collector)	Number of collectors	1
	Number of samples	15 (base) 150 (per node)
	Number of rules	161
	Average rules extraction interval	60 (seconds)
Amazon Managed Grafana	Number of active editors/administrators	1 (or more, based on your users)
CloudWatch (Logs)	Standard Logs: Data ingested	24.5 GB (base) 0.5 GB (per node)

Product	Calculator metric	Value
	Log Storage/Archival (Standard and Vended Logs)	Yes to store logs: Assuming 1 month retention
	Expected Logs Data Scanned	Each log insights query from Grafana will scan all log contents from the group over the specified time period.

These numbers are the base numbers for a solution running EKS with no additional software. This will give you an estimate of the base costs. It also leaves out network usage costs, which will vary based on whether the Amazon Managed Grafana workspace, Amazon Managed Service for Prometheus workspace, and Amazon EKS cluster are in the same availability zone, AWS Region, and VPN.

> **Note**
>
> When an item in this table includes a `(base)` value and a value per resource (for example, `(per node)`), you should add the base value to the per resource value times the number you have of that resource. For example, for **Average active time series**, enter a number that is `8000 + the number of nodes in your cluster * 15,000`. If you have 2 nodes, you would enter `38,000`, which is `8000 + ( 2 * 15,000 )`.

Prerequisites

This solution requires that you have done the following before using the solution.

1. You must have or **create an Amazon Elastic Kubernetes Service cluster** that you wish to monitor, and the cluster must have at least one node. The cluster must have API server endpoint access set to include private access (it can also allow public access).

   The authentication mode must include API access (it can be set to either `API` or `API_AND_CONFIG_MAP`). This allows the solution deployment to use access entries.

The following should be installed in the cluster (true by default when creating the cluster via the console, but must be added if you create the cluster using the AWS API or AWS CLI): AWS CNI, CoreDNS and Kube-proxy AddOns.

Save the Cluster name to specify later. This can be found in the cluster details in the Amazon EKS console.

> **Note**
>
> For details about how to create an Amazon EKS cluster, see Getting started with Amazon EKS.

2. You must **create an Amazon Managed Service for Prometheus workspace** in the same AWS account as your Amazon EKS cluster. For details, see Create a workspace in the *Amazon Managed Service for Prometheus User Guide*.

   Save the Amazon Managed Service for Prometheus workspace ARN to specify later.

3. You must **create an Amazon Managed Grafana workspace** with Grafana version 9 or newer, in the same AWS Region as your Amazon EKS cluster. For details about creating a new workspace, see Create an Amazon Managed Grafana workspace.

   The workspace role must have permissions to access Amazon Managed Service for Prometheus and Amazon CloudWatch APIs. The easiest way to do this is to use Service-managed permissions and select Amazon Managed Service for Prometheus and CloudWatch. You can also manually add the AmazonPrometheusQueryAccess and AmazonGrafanaCloudWatchAccess policies to your workspace IAM role.

   Save the Amazon Managed Grafana workspace ID and endpoint to specify later. The ID is in the form `g-123example`. The ID and the endpoint can be found in the Amazon Managed Grafana console. The endpoint is the URL for the workspace, and includes the ID. For example, `https://g-123example.grafana-workspace.<region>.amazonaws.com/`.

4. If you are deploying the solution with Terraform, you must create an **Amazon S3 bucket** that is accessible from your account. This will be used to store Terraform state files for the deployment.

   Save the Amazon S3 bucket ID to specify later.

5. In order to view the Amazon Managed Service for Prometheus alert rules, you must enable Grafana alerting for the Amazon Managed Grafana workspace.

Additionally, Amazon Managed Grafana must have the following permissions for your Prometheus resources. You must add them to either the service-managed or customer-managed policies described in Amazon Managed Grafana permissions and policies for AWS data sources.

- `aps:ListRules`
- `aps:ListAlertManagerSilences`
- `aps:ListAlertManagerAlerts`
- `aps:GetAlertManagerStatus`
- `aps:ListAlertManagerAlertGroups`
- `aps:PutAlertManagerSilences`
- `aps:DeleteAlertManagerSilence`

Note

While not strictly required to set up the solution, you must set up user authentication in your Amazon Managed Grafana workspace before users can access the dashboards created. For more information, see Authenticate users in Amazon Managed Grafana workspaces.

Using this solution

This solution configures AWS infrastructure to support reporting and monitoring metrics from an Amazon EKS cluster. You can install it using either AWS Cloud Development Kit (AWS CDK) or with Terraform.

Using AWS CDK

One way this solution is provided to you is as an AWS CDK application. You will provide information about the resources you want to use, and the solution will create the scraper, logs, and dashboards for you.

Note

The steps here assume that you have an environment with the AWS CLI, and AWS CDK, and both Node.js and NPM installed. You will use `make` and `brew` to simplify build and other common actions.

To use this solution to monitor an Amazon EKS cluster with AWS CDK

1. Make sure that you have completed all of the prerequisites steps.
2. Download all files for the solution from Amazon S3. The files are located at `s3://aws-observability-solutions/EKS/OSS/CDK/v3.0.0/iac`, and you can download them with the following Amazon S3 command. Run this command from a folder in your command line environment.

   ```
   aws s3 sync s3://aws-observability-solutions/EKS/OSS/CDK/v3.0.0/iac/ .
   ```

   You do not need to modify these files.
3. In your command line environment (from the folder where you downloaded the solution files), run the following commands.

   Set up the needed environment variables. Replace *REGION*, *AMG_ENDPOINT*, *EKS_CLUSTER*, and *AMP_ARN* with your AWS Region, Amazon Managed Grafana workspace endpoint (n the form `http://g-123example.grafana-workspace.us-east-1.amazonaws.com`), Amazon EKS cluster name, and Amazon Managed Service for Prometheus workspace ARN.

   ```
   export AWS_REGION=REGION
   export AMG_ENDPOINT=AMG_ENDPOINT
   export EKS_CLUSTER_NAME=EKS_CLUSTER
   export AMP_WS_ARN=AMP_ARN
   ```
4. You must create a service account token with ADMIN access for calling Grafana HTTP APIs. For details, see Using service accounts. You can use the AWS CLI with the following commands to create the token. You will need to replace the *GRAFANA_ID* with the ID of your Grafana workspace (it will be in the form `g-123example`). This key will expire after 7,200 seconds, or 2 hours. You can change the time (`seconds-to-live`), if you need to. The deployment takes under one hour.

   ```
   GRAFANA_SA_ID=$(aws grafana create-workspace-service-account \
     --workspace-id GRAFANA_ID \
     --grafana-role ADMIN \
     --name grafana-operator-key \
     --query 'id' \
     --output text)
   ```

```
# creates a new token for calling APIs
export AMG_API_KEY=$(aws grafana create-workspace-service-account-token \
  --workspace-id $managed_grafana_workspace_id \
  --name "grafana-operator-key-$(date +%s)" \
  --seconds-to-live 7200 \
  --service-account-id $GRAFANA_SA_ID \
  --query 'serviceAccountToken.key' \
  --output text)
```

Make the API Key available to the AWS CDK by adding it to AWS Systems Manager with the following command. Replace *AWS_REGION* with the Region that your solution will run in (in the form `us-east-1`).

```
aws ssm put-parameter --name "/observability-aws-solution-eks-infra/grafana-api-
key" \
    --type "SecureString" \
    --value $AMG_API_KEY \
    --region AWS_REGION \
    --overwrite
```

5. Run the following `make` command, which will install any other dependencies for the project.

```
make deps
```

6. Finally, run the AWS CDK project:

```
make build && make pattern aws-observability-solution-eks-infra-
$EKS_CLUSTER_NAME deploy
```

7. [Optional] After the stack creation is complete, you may use the same environment to create more instances of the stack for other Amazon EKS clusters in the same region, as long as you complete the other prerequisites for each (including separate Amazon Managed Grafana and Amazon Managed Service for Prometheus workspaces). You will need to redefine the `export` commands with the new parameters.

When the stack creation is completed, your Amazon Managed Grafana workspace will be populated with a dashboard showing metrics for your Amazon EKS cluster. It will take a few minutes for metrics to be shown, as the scraper begins to collect metrics.

Using Terraform

One way this solution is provided to you is as a Terraform solution. You will provide information about the resources you want to use, and the solution will create the scraper, logs, and dashboards for you.

To use this solution to monitor an Amazon EKS cluster with Terraform

1. Make sure that you have completed all of the prerequisites steps.
2. Download all files for the solution from Amazon S3. The files are located at `s3://aws-observability-solutions/EKS/OSS/Terraform/v3.0.0/`, and you can download them with the following Amazon S3 command. Run this command from a folder in your command line environment, then change directory to the folder from which you will deploy.

   ```
   aws s3 sync s3://aws-observability-solutions/EKS/OSS/Terraform/v3.0.0/ .
   cd eks-monitoring
   ```

   You do not need to modify these files.
3. In your command line environment (from the folder where you downloaded the solution files), run the following commands.

   Set up the needed environment variables. Replace *REGION*, *AMG_ENDPOINT*, *EKS_CLUSTER*, *AMP_ARN*, and *S3_ID*, with the AWS Region where you want new resources deployed (for example, `us-east-1`), Amazon Managed Grafana workspace endpoint (in the form `http://g-123example.grafana-workspace.us-east-1.amazonaws.com`), Amazon EKS cluster name, Amazon Managed Service for Prometheus workspace ARN, and Amazon S3 bucket ID.

   ```
   export TF_VAR_aws_region=REGION
   export TF_VAR_amg_endpoint=AMG_ENDPOINT
   export TF_VAR_eks_cluster_name=EKS_CLUSTER
   export TF_VAR_amp_ws_arn=AMP_ARN
   export TF_VAR_s3_bucket_id=S3_ID
   ```
4. You must create a service account token with ADMIN access for calling Grafana HTTP APIs. For details, see Using service accounts. You can use the AWS CLI with the following commands to create the token. You will need to replace the *GRAFANA_ID* with the ID of your Grafana workspace (it will be in the form `g-123example`). This key will expire after

7,200 seconds, or 2 hours. You can change the time (`seconds-to-live`), if you need to. The deployment takes under one hour.

```
GRAFANA_SA_ID=$(aws grafana create-workspace-service-account \
  --workspace-id GRAFANA_ID \
  --grafana-role ADMIN \
  --name grafana-operator-key \
  --query 'id' \
  --output text)

# creates a new token for running Terraform
export TF_VAR_grafana_api_key=$(aws grafana create-workspace-service-account-
token \
  --workspace-id $managed_grafana_workspace_id \
  --name "grafana-operator-key-$(date +%s)" \
  --seconds-to-live 7200 \
  --service-account-id $GRAFANA_SA_ID \
  --query 'serviceAccountToken.key' \
  --output text)
```

> **Note**
>
> The first step above, creating a service account for the workspace is not required if you already have a service account. In this case, replace the *$GRAFANA_SA_ID* with the ID of your service account.

5. Run the following `terraform` command to initialize Terraform with the solution.

```
terraform init -reconfigure \
-backend-config="bucket=${TF_VAR_s3_bucket_id}" \
-backend-config="region=${TF_VAR_aws_region}" \
-backend-config="key=state/${TF_VAR_eks_cluster_name}/terraform.tfstate"
```

6. Finally, deploy the Terraform project:

```
terraform apply
```

When the solution creation is completed, your Amazon Managed Grafana workspace will be populated with a dashboard showing metrics for your Amazon EKS cluster. It will take a few minutes for metrics to be shown, as the scraper begins to collect metrics.

List of metrics tracked

This solution creates a scraper that collects metrics from your Amazon EKS cluster. Those metrics are stored in Amazon Managed Service for Prometheus, and then displayed in Amazon Managed Grafana dashboards. By default, the scraper collects all Prometheus-compatible metrics that are exposed by the cluster. Installing software in your cluster that produces more metrics will increase the metrics collected. If you want, you can reduce the number of metrics by updating the scraper with a configuration that filters the metrics.

The following metrics are tracked with this solution, in a base Amazon EKS cluster configuration with no additional software installed.

Metric	Description / Purpose
`aggregator_unavailable_apiservice`	Gauge of APIServices which are marked as unavailable broken down by APIService name.
`apiserver_admission_webhook_admission_duration_seconds_bucket`	Admission webhook latency histogram in seconds, identified by name and broken out for each operation and API resource and type (validate or admit).
`apiserver_current_inflight_requests`	Maximal number of currently used inflight request limit of this apiserver per request kind in last second.
`apiserver_envelope_encryption_dek_cache_fill_percent`	Percent of the cache slots currently occupied by cached DEKs.
`apiserver_flowcontrol_current_executing_requests`	Number of requests in initial (for a WATCH) or any (for a non-WATCH) execution stage in the API Priority and Fairness subsystem.

Metric	Description / Purpose
apiserver_flowcontrol_rejected_requests_total	Number of requests in initial (for a WATCH) or any (for a non-WATCH) execution stage in the API Priority and Fairness subsystem that were rejected.
apiserver_flowcontrol_request_concurrency_limit	Nominal number of execution seats configured for each priority level.
apiserver_flowcontrol_request_execution_seconds_bucket	The bucketed histogram of duration of initial stage (for a WATCH) or any (for a non-WATCH) stage of request execution in the API Priority and Fairness subsystem.
apiserver_flowcontrol_request_queue_length_after_enqueue_count	The count of initial stage (for a WATCH) or any (for a non-WATCH) stage of request execution in the API Priority and Fairness subsystem.
apiserver_request	Indicates an API server request.
apiserver_requested_deprecated_apis	Gauge of deprecated APIs that have been requested, broken out by API group, version, resource, subresource, and removed_release.
apiserver_request_duration_seconds	Response latency distribution in seconds for each verb, dry run value, group, version, resource, subresource, scope and component.
apiserver_request_duration_seconds_bucket	The bucketed histogram of response latency distribution in seconds for each verb, dry run value, group, version, resource, subresource, scope and component.
apiserver_request_slo_duration_seconds	The Service Level Objective (SLO) response latency distribution in seconds for each verb, dry run value, group, version, resource, subresource, scope and component.

Metric	Description / Purpose
apiserver_request_terminations_total	Number of requests which apiserver terminated in self-defense.
apiserver_request_total	Counter of apiserver requests broken out for each verb, dry run value, group, version, resource, scope, component, and HTTP response code.
container_cpu_usage_seconds_total	Cumulative cpu time consumed.
container_fs_reads_bytes_total	Cumulative count of bytes read.
container_fs_reads_total	Cumulative count of reads completed.
container_fs_writes_bytes_total	Cumulative count of bytes written.
container_fs_writes_total	Cumulative count of writes completed.
container_memory_cache	Total page cache memory.
container_memory_rss	Size of RSS.
container_memory_swap	Container swap usage.
container_memory_working_set_bytes	Current working set.
container_network_receive_bytes_total	Cumulative count of bytes received.
container_network_receive_packets_dropped_total	Cumulative count of packets dropped while receiving.
container_network_receive_packets_total	Cumulative count of packets received.

Metric	Description / Purpose
container_network_transmit_bytes_total	Cumulative count of bytes transmitted.
container_network_transmit_packets_dropped_total	Cumulative count of packets dropped while transmitting.
container_network_transmit_packets_total	Cumulative count of packets transmitted.
etcd_request_duration_seconds_bucket	The bucketed histogram of etcd request latency in seconds for each operation and object type.
go_goroutines	Number of goroutines that currently exist.
go_threads	Number of OS threads created.
kubelet_cgroup_manager_duration_seconds_bucket	The bucketed histogram of duration in seconds for cgroup manager operations. Broken down by method.
kubelet_cgroup_manager_duration_seconds_count	Duration in seconds for cgroup manager operations. Broken down by method.
kubelet_node_config_error	This metric is true (1) if the node is experienc ing a configuration-related error, false (0) otherwise.
kubelet_node_name	The node's name. The count is always 1.
kubelet_pleg_relist_duration_seconds_bucket	The bucketed histogram of duration in seconds for relisting pods in PLEG.
kubelet_pleg_relist_duration_seconds_count	The count of duration in seconds for relisting pods in PLEG.
kubelet_pleg_relist_interval_seconds_bucket	The bucketed histogram of interval in seconds between relisting in PLEG.

Metric	Description / Purpose
kubelet_pod_start_duration_seconds_count	The count of duration in seconds from kubelet seeing a pod for the first time to the pod starting to run.
kubelet_pod_worker_duration_seconds_bucket	The bucketed histogram of duration in seconds to sync a single pod. Broken down by operation type: create, update, or sync.
kubelet_pod_worker_duration_seconds_count	The count of duration in seconds to sync a single pod. Broken down by operation type: create, update, or sync.
kubelet_running_containers	Number of containers currently running.
kubelet_running_pods	Number of pods that have a running pod sandbox.
kubelet_runtime_operations_duration_seconds_bucket	The bucketed histogram of duration in seconds of runtime operations. Broken down by operation type.
kubelet_runtime_operations_errors_total	Cumulative number of runtime operation errors by operation type.
kubelet_runtime_operations_total	Cumulative number of runtime operations by operation type.
kube_node_status_allocatable	The amount of resources allocatable for pods (after reserving some for system daemons).
kube_node_status_capacity	The total amount of resources available for a node.
kube_pod_container_resource_limits (CPU)	The number of requested limit resource by a container.

Metric	Description / Purpose
kube_pod_container_resource_limits (Memory)	The number of requested limit resource by a container.
kube_pod_container_resource_requests (CPU)	The number of requested request resource by a container.
kube_pod_container_resource_requests (Memory)	The number of requested request resource by a container.
kube_pod_owner	Information about the Pod's owner.
kube_resourcequota	Resource quotas in Kubernetes enforce usage limits on resources such as CPU, memory, and storage within namespaces.
node_cpu	The CPU usage metrics for a node, including usage per core and total usage.
node_cpu_seconds_total	Seconds the CPUs spent in each mode.
node_disk_io_time_seconds	The cumulative amount of time spent performing I/O operations on disk by a node.
node_disk_io_time_seconds_total	The total amount of time spent performing I/O operations on disk by the node.
node_disk_read_bytes_total	The total number of bytes read from disk by the node.
node_disk_written_bytes_total	The total number of bytes written to disk by the node.
node_filesystem_avail_bytes	The amount of available space in bytes on the filesystem of a node in a Kubernetes cluster.
node_filesystem_size_bytes	The total size of the filesystem on the node.

Metric	Description / Purpose
node_load1	The 1-minute load average of a node's CPU usage.
node_load15	The 15-minute load average of a node's CPU usage.
node_load5	The 5-minute load average of a node's CPU usage.
node_memory_Buffers_bytes	The amount of memory used for buffer caching by the node's operating system.
node_memory_Cached_bytes,	The amount of memory used for disk caching by the node's operating system.
node_memory_MemAvailable_bytes	The amount of memory available for use by applications and caches.
node_memory_MemFree_bytes	The amount of free memory available on the node.
node_memory_MemTotal_bytes	The total amount of physical memory available on the node.
node_network_receive_bytes_total	The total number of bytes received over the network by the node.
node_network_transmit_bytes_total	The total number of bytes transmitted over the network by the node.
process_cpu_seconds_total	Total user and system CPU time spent in seconds.
process_resident_memory_bytes	Resident memory size in bytes.
rest_client_requests_total	Number of HTTP requests, partitioned by status code, method, and host.

Metric	Description / Purpose
rest_client_request_duration_seconds_bucket	The bucketed histogram of request latency in seconds. Broken down by verb, and host.
storage_operation_duration_seconds_bucket	The bucketed histogram of duration of storage operations.
storage_operation_duration_seconds_count	The count of duration of storage operations.
storage_operation_errors_total	Cumulative number of errors during storage operations.
up	A metric indicating whether the monitored target (e.g., node) is up and running.
volume_manager_total_volumes	The total number of volumes managed by the volume manager.
workqueue_adds_total	Total number of adds handled by workqueue.
workqueue_depth	Current depth of workqueue.
workqueue_queue_duration_seconds_bucket	The bucketed histogram of how long in seconds an item stays in workqueue before being requested.
workqueue_work_duration_seconds_bucket	The bucketed histogram of how long in seconds processing an item from workqueue takes.

List of alerts created

The following tables list the alerts that are created by this solution. The alerts are created as rules in Amazon Managed Service for Prometheus, and are displayed in your Amazon Managed Grafana workspace.

You can modify the rules, including adding or deleting rules by editing the rules configuration file in your Amazon Managed Service for Prometheus workspace.

These two alerts are special alerts that are handled slightly differently than typical alerts. Instead of alerting you to an issue, they give you information that is used to monitor the system. The description includes details about how to use these alerts.

Alert	Description and usage
`Watchdog`	This is an alert meant to ensure that the entire alerting pipeline is functional. This alert is always firing, therefore it should always be firing in Alertmanager and always fire against a receiver. You can integrate this with your notification mechanism to send a notification when this alert is *not* firing. For example, you could use the **DeadMansSnitch** integration in PagerDuty.
`InfoInhibitor`	This is an alert that is used to inhibit info alerts. By themselves, info-level alerts can be very noisy, but they are relevant when combined with other alerts. This alert fires whenever there's a `severity=info` alert, and stops firing when another alert with a severity of `warning` or `critical` starts firing on the same namespace. This alert should be routed to a null receiver and configured to inhibit alerts with `severity=info` .

The following alerts give you information or warnings about your system.

Alert	Severity	Description
`NodeNetworkInterfaceFlapping`	`warning`	Network interface is often changing its status

Alert	Severity	Description
NodeFilesystemSpaceFillingUp	warning	File system is predicted to run out of space within the next 24 hours.
NodeFilesystemSpaceFillingUp	critical	File system is predicted to run out of space within the next 4 hours.
NodeFilesystemAlmostOutOfSpace	warning	File system has less than 5% space left.
NodeFilesystemAlmostOutOfSpace	critical	File system has less than 3% space left.
NodeFilesystemFilesFillingUp	warning	File system is predicted to run out of inodes within the next 24 hours.
NodeFilesystemFilesFillingUp	critical	File system is predicted to run out of inodes within the next 4 hours.
NodeFilesystemAlmostOutOfFiles	warning	File system has less than 5% inodes left.
NodeFilesystemAlmostOutOfFiles	critical	File system has less than 3% inodes left.
NodeNetworkReceiveErrs	warning	Network interface is reporting many receive errors.
NodeNetworkTransmitErrs	warning	Network interface is reporting many transmit errors.
NodeHighNumberConntrackEntriesUsed	warning	Number of conntrack entries are getting close to the limit.

Alert	Severity	Description
NodeTextFileCollectorScrapeError	warning	Node Exporter text file collector failed to scrape.
NodeClockSkewDetected	warning	Clock skew detected.
NodeClockNotSynchronizing	warning	Clock not synchronizing.
NodeRAIDDegraded	critical	RAID Array is degraded
NodeRAIDDiskFailure	warning	Failed device in RAID array
NodeFileDescriptorLimit	warning	Kernel is predicted to exhaust file descriptors limit soon.
NodeFileDescriptorLimit	critical	Kernel is predicted to exhaust file descriptors limit soon.
KubeNodeNotReady	warning	Node is not ready.
KubeNodeUnreachable	warning	Node is unreachable.
KubeletTooManyPods	info	Kubelet is running at capacity.
KubeNodeReadinessFlapping	warning	Node readiness status is flapping.
KubeletPlegDurationHigh	warning	Kubelet Pod Lifecycle Event Generator is taking too long to relist.
KubeletPodStartUpLatencyHigh	warning	Kubelet Pod startup latency is too high.
KubeletClientCertificateExpiration	warning	Kubelet client certificate is about to expire.

Alert	Severity	Description
KubeletClientCertificateExpiration	critical	Kubelet client certificate is about to expire.
KubeletServerCertificateExpiration	warning	Kubelet server certificate is about to expire.
KubeletServerCertificateExpiration	critical	Kubelet server certificate is about to expire.
KubeletClientCertificateRenewalErrors	warning	Kubelet has failed to renew its client certificate.
KubeletServerCertificateRenewalErrors	warning	Kubelet has failed to renew its server certificate.
KubeletDown	critical	Target disappeared from Prometheus target discovery.
KubeVersionMismatch	warning	Different semantic versions of Kubernetes components running.
KubeClientErrors	warning	Kubernetes API server client is experiencing errors.
KubeClientCertificateExpiration	warning	Client certificate is about to expire.
KubeClientCertificateExpiration	critical	Client certificate is about to expire.
KubeAggregatedAPIErrors	warning	Kubernetes aggregated API has reported errors.
KubeAggregatedAPIDown	warning	Kubernetes aggregated API is down.

Alert	Severity	Description
KubeAPIDown	critical	Target disappeared from Prometheus target discovery.
KubeAPITerminatedRequests	warning	The kubernetes apiserver has terminated {{ $value | humanizePercentage }} of its incoming requests.
KubePersistentVolumeFillingUp	critical	Persistent Volume is filling up.
KubePersistentVolumeFillingUp	warning	Persistent Volume is filling up.
KubePersistentVolumeInodesFillingUp	critical	Persistent Volume Inodes is filling up.
KubePersistentVolumeInodesFillingUp	warning	Persistent Volume Inodes are filling up.
KubePersistentVolumeErrors	critical	Persistent Volume is having issues with provisioning.
KubeCPUOvercommit	warning	Cluster has overcommitted CPU resource requests.
KubeMemoryOvercommit	warning	Cluster has overcommitted memory resource requests.
KubeCPUQuotaOvercommit	warning	Cluster has overcommitted CPU resource requests.
KubeMemoryQuotaOvercommit	warning	Cluster has overcommitted memory resource requests.
KubeQuotaAlmostFull	info	Namespace quota is going to be full.

Alert	Severity	Description
KubeQuotaFullyUsed	info	Namespace quota is fully used.
KubeQuotaExceeded	warning	Namespace quota has exceeded the limits.
CPUThrottlingHigh	info	Processes experience elevated CPU throttling.
KubePodCrashLooping	warning	Pod is crash looping.
KubePodNotReady	warning	Pod has been in a non-ready state for more than 15 minutes.
KubeDeploymentGenerationMismatch	warning	Deployment generation mismatch due to possible roll-back
KubeDeploymentReplicasMismatch	warning	Deployment has not matched the expected number of replicas.
KubeStatefulSetReplicasMismatch	warning	StatefulSet has not matched the expected number of replicas.
KubeStatefulSetGenerationMismatch	warning	StatefulSet generation mismatch due to possible roll-back
KubeStatefulSetUpdateNotRolledOut	warning	StatefulSet update has not been rolled out.
KubeDaemonSetRolloutStuck	warning	DaemonSet rollout is stuck.

Alert	Severity	Description
KubeContainerWaiting	warning	Pod container waiting longer than 1 hour
KubeDaemonSetNotScheduled	warning	DaemonSet pods are not scheduled.
KubeDaemonSetMisScheduled	warning	DaemonSet pods are misscheduled.
KubeJobNotCompleted	warning	Job did not complete in time
KubeJobFailed	warning	Job failed to complete.
KubeHpaReplicasMismatch	warning	HPA has not matched desired number of replicas.
KubeHpaMaxedOut	warning	HPA is running at max replicas
KubeStateMetricsListErrors	critical	kube-state-metrics is experiencing errors in list operations.
KubeStateMetricsWatchErrors	critical	kube-state-metrics is experiencing errors in watch operations.
KubeStateMetricsShardingMismatch	critical	kube-state-metrics sharding is misconfigured.
KubeStateMetricsShardsMissing	critical	kube-state-metrics shards are missing.
KubeAPIErrorBudgetBurn	critical	The API server is burning too much error budget.
KubeAPIErrorBudgetBurn	critical	The API server is burning too much error budget.

Alert	Severity	Description
KubeAPIErrorBudgetBurn	warning	The API server is burning too much error budget.
KubeAPIErrorBudgetBurn	warning	The API server is burning too much error budget.
TargetDown	warning	One or more targets are down.
etcdInsufficientMembers	critical	Etcd cluster insufficient members.
etcdHighNumberOfLeaderChanges	warning	Etcd cluster high number of leader changes.
etcdNoLeader	critical	Etcd cluster has no leader.
etcdHighNumberOfFailedGRPCRequests	warning	Etcd cluster high number of failed gRPC requests.
etcdGRPCRequestsSlow	critical	Etcd cluster gRPC requests are slow.
etcdMemberCommunicationSlow	warning	Etcd cluster member communication is slow.
etcdHighNumberOfFailedProposals	warning	Etcd cluster high number of failed proposals.
etcdHighFsyncDurations	warning	Etcd cluster high fsync durations.
etcdHighCommitDurations	warning	Etcd cluster has higher than expected commit durations.
etcdHighNumberOfFailedHTTPRequests	warning	Etcd cluster has failed HTTP requests.

Alert	Severity	Description
`etcdHighNumberOfFailedHTTPRequests`	`critical`	Etcd cluster has a high number of failed HTTP requests.
`etcdHTTPRequestsSlow`	`warning`	Etcd cluster HTTP requests are slow.
`HostClockNotSynchronizing`	`warning`	Host clock not synchronizing.
`HostOomKillDetected`	`warning`	Host OOM kill detected.

Troubleshooting

There are a few things that can cause the setup of the project to fail. Be sure to check the following.

- You must complete all Prerequisites before installing the solution.
- The cluster must have at least one node in it before attempting to create the solution or access the metrics.
- Your Amazon EKS cluster must have the `AWS CNI`, `CoreDNS` and `kube-proxy` add-ons installed. If they are not installed, the solution will not work correctly. They are installed by default, when creating the cluster through the console. You may need to install them if the cluster was created through an AWS SDK.
- Amazon EKS pods installation timed out. This can happen if there is not enough node capacity available. There are multiple causes of these issues, including:
  - The Amazon EKS cluster was initialized with Fargate instead of Amazon EC2. This project requires Amazon EC2.
  - The nodes are tainted and therefore unavailable.

    You can use `kubectl describe node` *`NODENAME`* `| grep Taints` to check the taints. Then `kubectl taint node` *`NODENAME TAINT_NAME`*`-` to remove the taints. Make sure to include the `-` after the taint name.

  - The nodes have reached the capacity limit. In this case you can create a new node or increase the capacity.
- You do not see any dashboards in Grafana: using the incorrect Grafana workspace ID.

  Run the following command to get information about Grafana:

  ```
  kubectl describe grafanas external-grafana -n grafana-operator
  ```

  You can check the results for the correct workspace URL. If it is not the one you are expecting, re-deploy with the correct workspace ID.

  ```
  Spec:
    External:
      API Key:
        Key:   GF_SECURITY_ADMIN_APIKEY
        Name:  grafana-admin-credentials
      URL:     https://g-123example.grafana-workspace.aws-region.amazonaws.com
  Status:
    Admin URL:  https://g-123example.grafana-workspace.aws-region.amazonaws.com
    Dashboards:
      ...
  ```

- You do not see any dashboards in Grafana: You are using an expired API key.

  To look for this case, you will need to get the grafana operator and check the logs for errors. Get the name of the Grafana operator with this command:

  ```
  kubectl get pods -n grafana-operator
  ```

  This will return the operator name, for example:

  ```
  NAME                                 READY   STATUS    RESTARTS   AGE
  grafana-operator-1234abcd5678ef90    1/1     Running   0          1h2m
  ```

  Use the operator name in the following command:

  ```
  kubectl logs grafana-operator-1234abcd5678ef90 -n grafana-operator
  ```

  Error messages such as the following indicate an expired API key:

```
ERROR   error reconciling datasource    {"controller": "grafanadatasource",
 "controllerGroup": "grafana.integreatly.org", "controllerKind": "GrafanaDatasource",
 "GrafanaDatasource": {"name":"grafanadatasource-sample-amp","namespace":"grafana-
operator"}, "namespace": "grafana-operator", "name": "grafanadatasource-sample-
amp", "reconcileID": "72cfd60c-a255-44a1-bfbd-88b0cbc4f90c", "datasource":
 "grafanadatasource-sample-amp", "grafana": "external-grafana", "error": "status:
 401, body: {\"message\":\"Expired API key\"}\n"}
github.com/grafana-operator/grafana-operator/controllers.
(*GrafanaDatasourceReconciler).Reconcile
```

In this case, create a new API key and deploy the solution again. If the problem persists, you can force synchronization by using the following command before redeploying:

```
kubectl delete externalsecret/external-secrets-sm -n grafana-operator
```

- *CDK installs* – Missing SSM parameter. If you see an error like the following, run `cdk bootstrap` and try again.

```
Deployment failed: Error: aws-observability-solution-eks-infra-$EKS_CLUSTER_NAME:
 SSM
parameter /cdk-bootstrap/xxxxxxx/version not found. Has the environment been
bootstrapped? Please run 'cdk bootstrap' (see https://docs.aws.amazon.com/cdk/latest/
guide/bootstrapping.html)
```

- Deployment can fail if the OIDC provider already exists. You will see an error like the following (in this case, for CDK installs):

```
| CREATE_FAILED | Custom::AWSCDKOpenIdConnectProvider | OIDCProvider/Resource/Default
Received response status [FAILED] from custom resource. Message returned:
EntityAlreadyExistsException: Provider with url https://
oidc.eks.REGION.amazonaws.com/id/PROVIDER ID already exists.
```

In this case, go to the IAM portal and delete the OIDC provider and try again.

- *Terraform installs* – You see an error message that includes `cluster-secretstore-sm failed to create kubernetes rest client for update of resource` and `failed to create kubernetes rest client for update of resource`.

This error typically indicates that the External Secrets Operator is not installed or enabled in your Kubernetes cluster. This is installed as part of the solution deployment, but sometimes is not ready when the solution needs it.

You can verify that it's installed with the following command:

```
kubectl get deployments -n external-secrets
```

If it's installed, it can take some time for the operator to be fully ready to be used. You can check the status of the needed Custom Resource Definitions (CRDs) by running the following command:

```
kubectl get crds|grep external-secrets
```

This command should list the CRDs related to the external secrets operator, including `clustersecretstores.external-secrets.io` and `externalsecrets.external-secrets.io`. If they are not listed, wait a few minutes and check again.

Once the CRDs are registered, you can run `terraform apply` again to deploy the solution.

Solution for monitoring JVM applications with Amazon Managed Grafana

Applications built with Java Virtual Machines (JVM) have specialized monitoring needs. This page describes a template that provides a solution for monitoring JVM-based applications running on your Amazon EKS cluster. The solution can be installed using AWS Cloud Development Kit (AWS CDK).

> **Note**
>
> This solution provides monitoring for a JVM application. If your JVM application is specifically an Apache Kafka application, you can instead choose to use the Kafka monitoring solution, which includes both JVM and Kafka monitoring.

This solution configures:

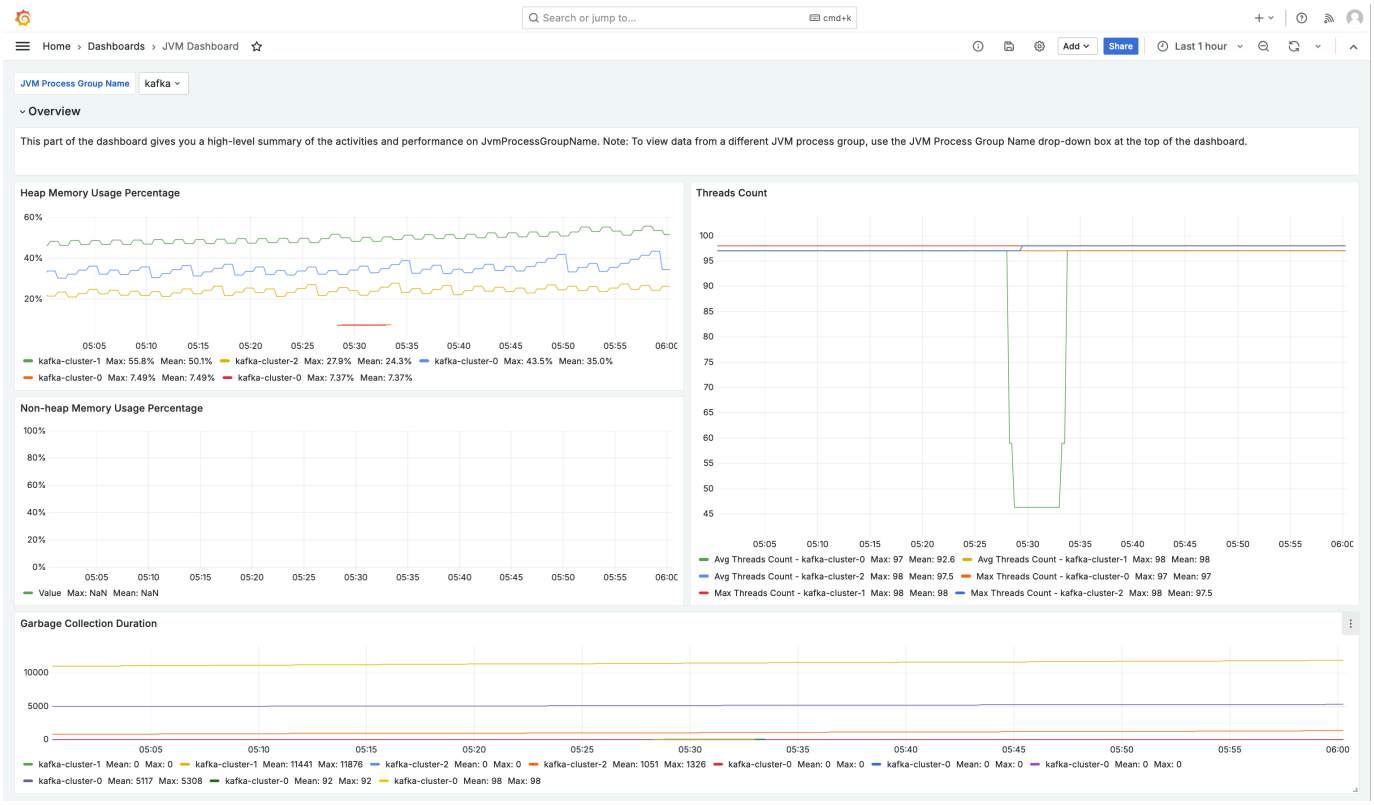
- Your Amazon Managed Service for Prometheus workspace to store Java Virtual Machine (JVM) metrics from your Amazon EKS cluster.
- Gathering specific JVM metrics using the CloudWatch agent, as well as a CloudWatch agent add-on. The metrics are configured to be sent to the Amazon Managed Service for Prometheus workspace.
- Your Amazon Managed Grafana workspace to pull those metrics, and create dashboards to help you monitor your cluster.

> **Note**
>
> This solution provides JVM metrics for your application running on Amazon EKS, but does not include Amazon EKS metrics. You can additionally use the Observability solution for monitoring Amazon EKS to see metrics and alerts for your Amazon EKS cluster.

About this solution

This solution configures an Amazon Managed Grafana workspace to provide metrics for your Java Virtual Machine (JVM) application. The metrics are used to generate dashboards that help you to operate your application more effectively by providing insights into the health and performance of the application.

The following image shows a sample of one of the dashboards created by this solution.



The metrics are scraped with a 1 minute scrape interval. The dashboards show metrics aggregated to 1 minute, 5 minutes, or more, based on the specific metric.

For a list of metrics tracked by this solution, see List of metrics tracked.

Costs

This solution creates and uses resources in your workspace. You will be charged for standard usage of the resources created, including:

- Amazon Managed Grafana workspace access by users. For more information about pricing, see Amazon Managed Grafana pricing.
- Amazon Managed Service for Prometheus metric ingestion and storage, and metric analysis (query sample processing). The number of metrics used by this solution depends on your application configuration and usage.

  You can view the ingestion and storage metrics in Amazon Managed Service for Prometheus using CloudWatch For more information, see CloudWatch metrics in the *Amazon Managed Service for Prometheus User Guide*.

You can estimate the cost using the pricing calculator on the Amazon Managed Service for Prometheus pricing page. The number of metrics will depend on the number of nodes in your cluster, and the metrics your applications produce.

- Networking costs. You may incur standard AWS network charges for cross availability zone, Region, or other traffic.

The pricing calculators, available from the pricing page for each product, can help you understand potential costs for your solution. The following information can help get a base cost, for the solution running in the same availability zone as the Amazon EKS cluster.

Product	Calculator metric	Value
Amazon Managed Service for Prometheus	Active series	50 (per application pod)
	Avg Collection Interval	60 (seconds)
Amazon Managed Grafana	Number of active editors/administrators	1 (or more, based on your users)

These numbers are the base numbers for a JVM application running on Amazon EKS. This will give you an estimate of the base costs. As you add pods to your application, the costs will grow, as shown. These costs leave out network usage costs, which will vary based on whether the Amazon Managed Grafana workspace, Amazon Managed Service for Prometheus workspace, and Amazon EKS cluster are in the same availability zone, AWS Region, and VPN.

Prerequisites

This solution requires that you have done the following before using the solution.

1. You must have or **create an Amazon Elastic Kubernetes Service cluster** that you wish to monitor, and the cluster must have at least one node. The cluster must have API server endpoint access set to include private access (it can also allow public access).

   The authentication mode must include API access (it can be set to either `API` or `API_AND_CONFIG_MAP`). This allows the solution deployment to use access entries.

The following should be installed in the cluster (true by default when creating the cluster via the console, but must be added if you create the cluster using the AWS API or AWS CLI): Amazon EKS Pod Identity Agent, AWS CNI, CoreDNS, Kube-proxy and Amazon EBS CSI Driver AddOns (the Amazon EBS CSI Driver AddOn is not technically required for the solution, but is required for some JVM applications).

Save the Cluster name to specify later. This can be found in the cluster details in the Amazon EKS console.

> **Note**
>
> For details about how to create an Amazon EKS cluster, see Getting started with Amazon EKS.

2. You must be running an application on Java Virtual Machines on your Amazon EKS cluster.
3. You must **create an Amazon Managed Service for Prometheus workspace** in the same AWS account as your Amazon EKS cluster. For details, see Create a workspace in the *Amazon Managed Service for Prometheus User Guide*.

   Save the Amazon Managed Service for Prometheus workspace ARN to specify later.
4. You must **create an Amazon Managed Grafana workspace** with Grafana version 9 or newer, in the same AWS Region as your Amazon EKS cluster. For details about creating a new workspace, see Create an Amazon Managed Grafana workspace.

   The workspace role must have permissions to access Amazon Managed Service for Prometheus and Amazon CloudWatch APIs. The easiest way to do this is to use Service-managed permissions and select Amazon Managed Service for Prometheus and CloudWatch. You can also manually add the AmazonPrometheusQueryAccess and AmazonGrafanaCloudWatchAccess policies to your workspace IAM role.

   Save the Amazon Managed Grafana workspace ID and endpoint to specify later. The ID is in the form `g-123example`. The ID and the endpoint can be found in the Amazon Managed Grafana console. The endpoint is the URL for the workspace, and includes the ID. For example, `https://g-123example.grafana-workspace.<region>.amazonaws.com/`.

> **Note**
>
> While not strictly required to set up the solution, you must set up user authentication in your Amazon Managed Grafana workspace before users can access the dashboards created. For more information, see Authenticate users in Amazon Managed Grafana workspaces.

Using this solution

This solution configures AWS infrastructure to support reporting and monitoring metrics from a Java Virtual Machine (JVM) application running in an Amazon EKS cluster. You can install it using AWS Cloud Development Kit (AWS CDK).

> **Note**
>
> The steps here assume that you have an environment with the AWS CLI, and AWS CDK, and both Node.js and NPM installed. You will use `make` and `brew` to simplify build and other common actions.

To use this solution to monitor an Amazon EKS cluster with AWS CDK

1. Make sure that you have completed all of the prerequisites steps.
2. Download all files for the solution from Amazon S3. The files are located at `s3://aws-observability-solutions/JVM_EKS/OSS/CDK/v1.0.0/iac`, and you can download them with the following Amazon S3 command. Run this command from a folder in your command line environment.

   ```
   aws s3 sync s3://aws-observability-solutions/JVM_EKS/OSS/CDK/v1.0.0/iac/ .
   ```

   You do not need to modify these files.
3. In your command line environment (from the folder where you downloaded the solution files), run the following commands.

   Set up the needed environment variables. Replace *`REGION`*, *`AMG_ENDPOINT`*, *`EKS_CLUSTER`*, and *`AMP_ARN`* with your AWS Region, Amazon Managed Grafana workspace endpoint (n the form `http://g-123example.grafana-workspace.us-east-1.amazonaws.com`), Amazon EKS cluster name, and Amazon Managed Service for Prometheus workspace ARN.

```
export AWS_REGION=REGION
export AMG_ENDPOINT=AMG_ENDPOINT
export EKS_CLUSTER_NAME=EKS_CLUSTER
export AMP_WS_ARN=AMP_ARN
```

4. Create annotations that can be used by the solution. You can choose to annotate a namespace, deployment, statefulset, daemonset, or your pods directly. The JSM solution requires two annotations. You will use `kubectl` to annotation your resources with the following commands:

```
kubectl annotate <resource-type> <resource-value> instrumentation.opentelemetry.io/
inject-java=true
kubectl annotate <resource-type> <resource-value> cloudwatch.aws.amazon.com/inject-
jmx-jvm=true
```

   Replace *<resource-type>* and *<resource-value>* with the correct values for your system. For example, to annotate your `foo` deployment, your first command would be:

```
kubectl annotate deployment foo instrumentation.opentelemetry.io/inject-java=true
```

5. Create a service account token with ADMIN access for calling Grafana HTTP APIs. For details, see Using service accounts. You can use the AWS CLI with the following commands to create the token. You will need to replace the *GRAFANA_ID* with the ID of your Grafana workspace (it will be in the form `g-123example`). This key will expire after 7,200 seconds, or 2 hours. You can change the time (`seconds-to-live`), if you need to. The deployment takes under one hour.

```
# creates a new service account (optional: you can use an existing account)
GRAFANA_SA_ID=$(aws grafana create-workspace-service-account \
  --workspace-id GRAFANA_ID \
  --grafana-role ADMIN \
  --name grafana-operator-key \
  --query 'id' \
  --output text)

# creates a new token for calling APIs
export AMG_API_KEY=$(aws grafana create-workspace-service-account-token \
  --workspace-id $managed_grafana_workspace_id \
  --name "grafana-operator-key-$(date +%s)" \
  --seconds-to-live 7200 \
```

```
  --service-account-id $GRAFANA_SA_ID \
  --query 'serviceAccountToken.key' \
  --output text)
```

Make the API Key available to the AWS CDK by adding it to AWS Systems Manager with the following command. Replace *AWS_REGION* with the Region that your solution will run in (in the form `us-east-1`).

```
aws ssm put-parameter --name "/observability-aws-solution-jvm-eks/grafana-api-key"
 \
  --type "SecureString" \
  --value $AMG_API_KEY \
  --region AWS_REGION \
  --overwrite
```

6. Run the following `make` command, which will install any other dependencies for the project.

```
make deps
```

7. Finally, run the AWS CDK project:

```
make build && make pattern aws-observability-solution-jvm-eks-$EKS_CLUSTER_NAME
 deploy
```

8. [Optional] After the stack creation is complete, you may use the same environment to create more instances of the stack for other JVM applications running on Amazon EKS clusters in the same region, as long as you complete the other prerequisites for each (including separate Amazon Managed Grafana and Amazon Managed Service for Prometheus workspaces). You will need to redefine the `export` commands with the new parameters.

When the stack creation is completed, your Amazon Managed Grafana workspace will be populated with a dashboard showing metrics for your application and Amazon EKS cluster. It will take a few minutes for metrics to be shown, as the metrics are collected.

List of metrics tracked

This solution collects metrics from your JVM-based application. Those metrics are stored in Amazon Managed Service for Prometheus, and then displayed in Amazon Managed Grafana dashboards.

The following metrics are tracked with this solution.

- jvm.classes.loaded
- jvm.gc.collections.count
- jvm.gc.collections.elapsed
- jvm.memory.heap.init
- jvm.memory.heap.max
- jvm.memory.heap.used
- jvm.memory.heap.committed
- jvm.memory.nonheap.init
- jvm.memory.nonheap.max
- jvm.memory.nonheap.used
- jvm.memory.nonheap.committed
- jvm.memory.pool.init
- jvm.memory.pool.max
- jvm.memory.pool.used
- jvm.memory.pool.committed
- jvm.threads.count

Troubleshooting

There are a few things that can cause the setup of the project to fail. Be sure to check the following.

- You must complete all Prerequisites before installing the solution.
- The cluster must have at least one node in it before attempting to create the solution or access the metrics.
- Your Amazon EKS cluster must have the `AWS CNI`, `CoreDNS` and `kube-proxy` add-ons installed. If they are not installed, the solution will not work correctly. They are installed by default, when creating the cluster through the console. You may need to install them if the cluster was created through an AWS SDK.
- Amazon EKS pods installation timed out. This can happen if there is not enough node capacity available. There are multiple causes of these issues, including:

- The Amazon EKS cluster was initialized with Fargate instead of Amazon EC2. This project requires Amazon EC2.
- The nodes are tainted and therefore unavailable.

  You can use `kubectl describe node NODENAME | grep Taints` to check the taints. Then `kubectl taint node NODENAME TAINT_NAME-` to remove the taints. Make sure to include the `-` after the taint name.
- The nodes have reached the capacity limit. In this case you can create a new node or increase the capacity.

- You do not see any dashboards in Grafana: using the incorrect Grafana workspace ID.

  Run the following command to get information about Grafana:

  ```
  kubectl describe grafanas external-grafana -n grafana-operator
  ```

  You can check the results for the correct workspace URL. If it is not the one you are expecting, re-deploy with the correct workspace ID.

  ```
  Spec:
    External:
      API Key:
        Key:   GF_SECURITY_ADMIN_APIKEY
        Name:  grafana-admin-credentials
      URL:     https://g-123example.grafana-workspace.aws-region.amazonaws.com
  Status:
    Admin URL:  https://g-123example.grafana-workspace.aws-region.amazonaws.com
    Dashboards:
      ...
  ```

- You do not see any dashboards in Grafana: You are using an expired API key.

  To look for this case, you will need to get the grafana operator and check the logs for errors. Get the name of the Grafana operator with this command:

  ```
  kubectl get pods -n grafana-operator
  ```

  This will return the operator name, for example:

  ```
  NAME                                READY   STATUS    RESTARTS   AGE
  ```

```
grafana-operator-1234abcd5678ef90   1/1     Running   0          1h2m
```

Use the operator name in the following command:

```
kubectl logs grafana-operator-1234abcd5678ef90 -n grafana-operator
```

Error messages such as the following indicate an expired API key:

```
ERROR   error reconciling datasource    {"controller": "grafanadatasource",
 "controllerGroup": "grafana.integreatly.org", "controllerKind": "GrafanaDatasource",
 "GrafanaDatasource": {"name":"grafanadatasource-sample-amp","namespace":"grafana-
operator"}, "namespace": "grafana-operator", "name": "grafanadatasource-sample-
amp", "reconcileID": "72cfd60c-a255-44a1-bfbd-88b0cbc4f90c", "datasource":
 "grafanadatasource-sample-amp", "grafana": "external-grafana", "error": "status:
 401, body: {\"message\":\"Expired API key\"}\n"}
github.com/grafana-operator/grafana-operator/controllers.
(*GrafanaDatasourceReconciler).Reconcile
```

In this case, create a new API key and deploy the solution again. If the problem persists, you can force synchronization by using the following command before redeploying:

```
kubectl delete externalsecret/external-secrets-sm -n grafana-operator
```

- Missing SSM parameter. If you see an error like the following, run `cdk bootstrap` and try again.

```
Deployment failed: Error: aws-observability-solution-jvm-eks-$EKS_CLUSTER_NAME: SSM
parameter /cdk-bootstrap/xxxxxxx/version not found. Has the environment been
bootstrapped? Please run 'cdk bootstrap' (see https://docs.aws.amazon.com/cdk/latest/
guide/bootstrapping.html)
```

Solution for monitoring Kafka applications with Amazon Managed Grafana

Applications built on top of Apache Kafka have specialized monitoring needs. This page describes a template that provides a solution for monitoring Kafka applications running in Java Virtual Machines on your Amazon EKS cluster. The solution can be installed using AWS Cloud Development Kit (AWS CDK).

> **Note**
>
> This solution does not support monitoring Amazon Managed Streaming for Apache Kafka applications. For information about monitoring Amazon MSK applications, see Monitor an Amazon MSK cluster in the *Amazon Managed Streaming for Apache Kafka Developer Guide*.

This solution configures:

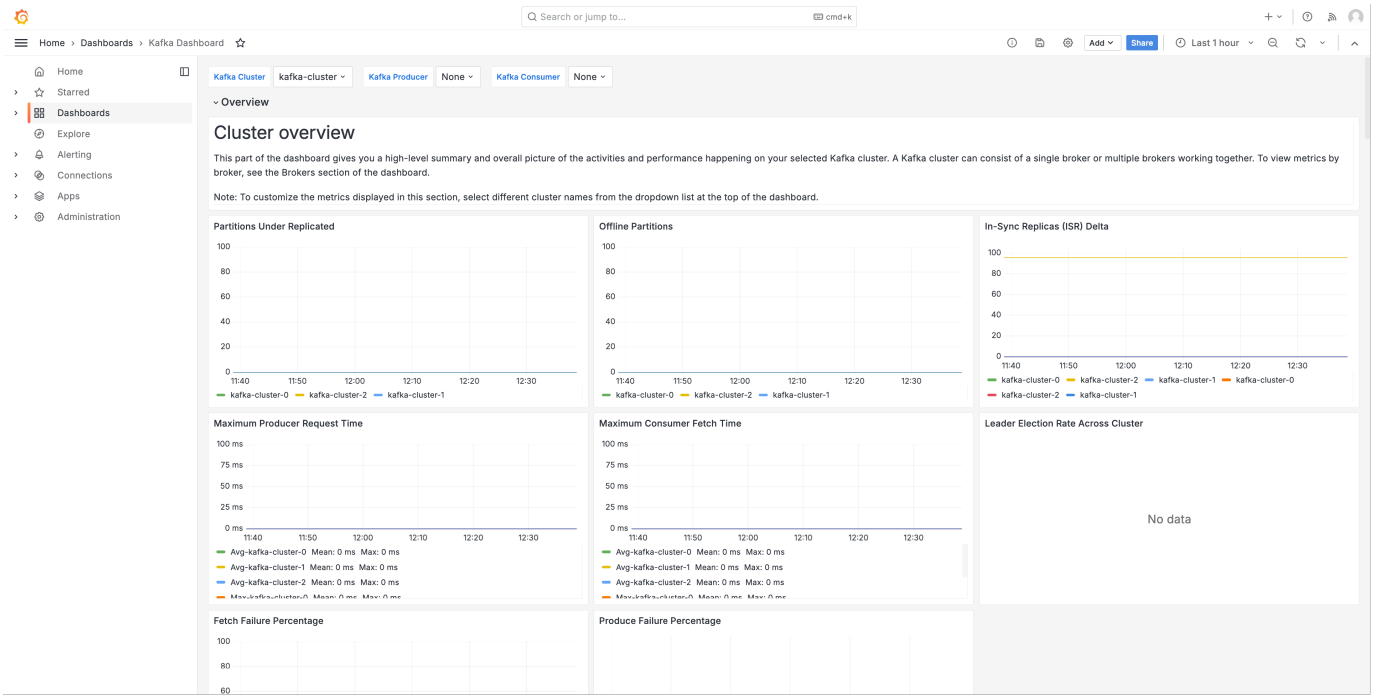
- Your Amazon Managed Service for Prometheus workspace to store Kafka and Java Virtual Machine (JVM) metrics from your Amazon EKS cluster.
- Gathering specific Kafka and JVM metrics using the CloudWatch agent, as well as a CloudWatch agent add-on. The metrics are configured to be sent to the Amazon Managed Service for Prometheus workspace.
- Your Amazon Managed Grafana workspace to pull those metrics, and create dashboards to help you monitor your cluster.

> **Note**
>
> This solution provides JVM and Kafka metrics for your application running on Amazon EKS, but does not include Amazon EKS metrics. You can use the Observability solution for monitoring Amazon EKS to see metrics and alerts for your Amazon EKS cluster.

About this solution

This solution configures an Amazon Managed Grafana workspace to provide metrics for your Apache Kafka application. The metrics are used to generate dashboards that help you to operate your application more effectively by providing insights into the performance and workload of the Kafka application.

The following image shows a sample of one of the dashboards created by this solution.



The metrics are scraped with a 1 minute scrape interval. The dashboards show metrics aggregated to 1 minute, 5 minutes, or more, based on the specific metric.

For a list of metrics tracked by this solution, see List of metrics tracked.

Costs

This solution creates and uses resources in your workspace. You will be charged for standard usage of the resources created, including:

- Amazon Managed Grafana workspace access by users. For more information about pricing, see Amazon Managed Grafana pricing.
- Amazon Managed Service for Prometheus metric ingestion and storage, and metric analysis (query sample processing). The number of metrics used by this solution depends on your application configuration and usage.

  You can view the ingestion and storage metrics in Amazon Managed Service for Prometheus using CloudWatch For more information, see CloudWatch metrics in the *Amazon Managed Service for Prometheus User Guide*.

  You can estimate the cost using the pricing calculator on the Amazon Managed Service for Prometheus pricing page. The number of metrics will depend on the number of nodes in your cluster, and the metrics your applications produce.

- Networking costs. You may incur standard AWS network charges for cross availability zone, Region, or other traffic.

The pricing calculators, available from the pricing page for each product, can help you understand potential costs for your solution. The following information can help get a base cost, for the solution running in the same availability zone as the Amazon EKS cluster.

Product	Calculator metric	Value
Amazon Managed Service for Prometheus	Active series	95 (per Kafka pod)
	Avg Collection Interval	60 (seconds)
Amazon Managed Grafana	Number of active editors/administrators	1 (or more, based on your users)

These numbers are the base numbers for a solution running Kafka on Amazon EKS. This will give you an estimate of the base costs. As you add Kafka pods to your application, the costs will grow, as shown. These costs leave out network usage costs, which will vary based on whether the Amazon Managed Grafana workspace, Amazon Managed Service for Prometheus workspace, and Amazon EKS cluster are in the same availability zone, AWS Region, and VPN.

Prerequisites

This solution requires that you have done the following before using the solution.

1. You must have or **create an Amazon Elastic Kubernetes Service cluster** that you wish to monitor, and the cluster must have at least one node. The cluster must have API server endpoint access set to include private access (it can also allow public access).

   The authentication mode must include API access (it can be set to either `API` or `API_AND_CONFIG_MAP`). This allows the solution deployment to use access entries.

   The following should be installed in the cluster (true by default when creating the cluster via the console, but must be added if you create the cluster using the AWS API or AWS CLI): Amazon EKS Pod Identity Agent, AWS CNI, CoreDNS, Kube-proxy and Amazon EBS CSI Driver AddOns

(the Amazon EBS CSI Driver AddOn is not technically required for the solution, but is required for most Kafka applications).

Save the Cluster name to specify later. This can be found in the cluster details in the Amazon EKS console.

> **Note**
>
> For details about how to create an Amazon EKS cluster, see Getting started with Amazon EKS.

2. You must be running an Apache Kafka application on Java Virtual Machines on your Amazon EKS cluster.
3. You must **create an Amazon Managed Service for Prometheus workspace** in the same AWS account as your Amazon EKS cluster. For details, see Create a workspace in the *Amazon Managed Service for Prometheus User Guide*.

   Save the Amazon Managed Service for Prometheus workspace ARN to specify later.
4. You must **create an Amazon Managed Grafana workspace** with Grafana version 9 or newer, in the same AWS Region as your Amazon EKS cluster. For details about creating a new workspace, see Create an Amazon Managed Grafana workspace.

   The workspace role must have permissions to access Amazon Managed Service for Prometheus and Amazon CloudWatch APIs. The easiest way to do this is to use Service-managed permissions and select Amazon Managed Service for Prometheus and CloudWatch. You can also manually add the AmazonPrometheusQueryAccess and AmazonGrafanaCloudWatchAccess policies to your workspace IAM role.

   Save the Amazon Managed Grafana workspace ID and endpoint to specify later. The ID is in the form `g-123example`. The ID and the endpoint can be found in the Amazon Managed Grafana console. The endpoint is the URL for the workspace, and includes the ID. For example, `https://g-123example.grafana-workspace.<region>.amazonaws.com/`.

> **Note**
>
> While not strictly required to set up the solution, you must set up user authentication in your Amazon Managed Grafana workspace before users can access the dashboards created. For more information, see Authenticate users in Amazon Managed Grafana workspaces.

Using this solution

This solution configures AWS infrastructure to support reporting and monitoring metrics from a Kafka application running in an Amazon EKS cluster. You can install it using AWS Cloud Development Kit (AWS CDK).

> **Note**
>
> The steps here assume that you have an environment with the AWS CLI, and AWS CDK, and both Node.js and NPM installed. You will use `make` and `brew` to simplify build and other common actions.

To use this solution to monitor an Amazon EKS cluster with AWS CDK

1. Make sure that you have completed all of the prerequisites steps.
2. Download all files for the solution from Amazon S3. The files are located at `s3://aws-observability-solutions/Kafka_EKS/OSS/CDK/v1.0.0/iac`, and you can download them with the following Amazon S3 command. Run this command from a folder in your command line environment.

   ```
   aws s3 sync s3://aws-observability-solutions/Kafka_EKS/OSS/CDK/v1.0.0/iac/ .
   ```

   You do not need to modify these files.
3. In your command line environment (from the folder where you downloaded the solution files), run the following commands.

   Set up the needed environment variables. Replace *`REGION`*, *`AMG_ENDPOINT`*, *`EKS_CLUSTER`*, and *`AMP_ARN`* with your AWS Region, Amazon Managed Grafana workspace endpoint (n the form `http://g-123example.grafana-workspace.us-east-1.amazonaws.com`), Amazon EKS cluster name, and Amazon Managed Service for Prometheus workspace ARN.

```
export AWS_REGION=REGION
export AMG_ENDPOINT=AMG_ENDPOINT
export EKS_CLUSTER_NAME=EKS_CLUSTER
export AMP_WS_ARN=AMP_ARN
```

4. You must create annotations that can be used by the deployment. You can choose to annotate a namespace, deployment, statefulset, daemonset, or your pods directly. The Kafka solution requires five annotations. You will use `kubectl` to annotation your resources with the following commands:

```
kubectl annotate <resource-type> <resource-value> instrumentation.opentelemetry.io/
inject-java=true
kubectl annotate <resource-type> <resource-value> cloudwatch.aws.amazon.com/inject-
jmx-jvm=true
kubectl annotate <resource-type> <resource-value> cloudwatch.aws.amazon.com/inject-
jmx-kafka=true
kubectl annotate <resource-type> <resource-value> cloudwatch.aws.amazon.com/inject-
jmx-kafka-producer=true
kubectl annotate <resource-type> <resource-value> cloudwatch.aws.amazon.com/inject-
jmx-kafka-consumer=true
```

   Replace *<resource-type>* and *<resource-value>* with the correct values for your system. For example, to annotate your `foo` deployment, your first command would be:

```
kubectl annotate deployment foo instrumentation.opentelemetry.io/inject-java=true
```

5. Create a service account token with ADMIN access for calling Grafana HTTP APIs. For details, see Using service accounts. You can use the AWS CLI with the following commands to create the token. You will need to replace the *GRAFANA_ID* with the ID of your Grafana workspace (it will be in the form `g-123example`). This key will expire after 7,200 seconds, or 2 hours. You can change the time (`seconds-to-live`), if you need to. The deployment takes under one hour.

```
# creates a new service account (optional: you can use an existing account)
GRAFANA_SA_ID=$(aws grafana create-workspace-service-account \
  --workspace-id GRAFANA_ID \
  --grafana-role ADMIN \
  --name grafana-operator-key \
  --query 'id' \
  --output text)
```

```
# creates a new token for calling APIs
export AMG_API_KEY=$(aws grafana create-workspace-service-account-token \
  --workspace-id $managed_grafana_workspace_id \
  --name "grafana-operator-key-$(date +%s)" \
  --seconds-to-live 7200 \
  --service-account-id $GRAFANA_SA_ID \
  --query 'serviceAccountToken.key' \
  --output text)
```

Make the API Key available to the AWS CDK by adding it to AWS Systems Manager with the following command. Replace *AWS_REGION* with the Region that your solution will run in (in the form `us-east-1`).

```
aws ssm put-parameter --name "/observability-aws-solution-kafka-eks/grafana-api-
key" \
  --type "SecureString" \
  --value $AMG_API_KEY \
  --region AWS_REGION \
  --overwrite
```

6. Run the following `make` command, which will install any other dependencies for the project.

```
make deps
```

7. Finally, run the AWS CDK project:

```
make build && make pattern aws-observability-solution-kafka-eks-$EKS_CLUSTER_NAME
 deploy
```

8. [Optional] After the stack creation is complete, you may use the same environment to create more instances of the stack for other Kafka applications running on Amazon EKS clusters in the same region, as long as you complete the other prerequisites for each (including separate Amazon Managed Grafana and Amazon Managed Service for Prometheus workspaces). You will need to redefine the `export` commands with the new parameters.

When the stack creation is completed, your Amazon Managed Grafana workspace will be populated with a dashboard showing metrics for your application and Amazon EKS cluster. It will take a few minutes for metrics to be shown, as the metrics are collected.

List of metrics tracked

This solution collects metrics from your JVM-based Kafka application. Those metrics are stored in Amazon Managed Service for Prometheus, and then displayed in Amazon Managed Grafana dashboards.

The following metrics are tracked with this solution.

- jvm.classes.loaded
- jvm.gc.collections.count
- jvm.gc.collections.elapsed
- jvm.memory.heap.init
- jvm.memory.heap.max
- jvm.memory.heap.used
- jvm.memory.heap.committed
- jvm.memory.nonheap.init
- jvm.memory.nonheap.max
- jvm.memory.nonheap.used
- jvm.memory.nonheap.committed
- jvm.memory.pool.init
- jvm.memory.pool.max
- jvm.memory.pool.used
- jvm.memory.pool.committed
- jvm.threads.count
- kafka.message.count
- kafka.request.count
- kafka.request.failed
- kafka.request.time.total
- kafka.request.time.50p
- kafka.request.time.99p
- kafka.request.time.avg
- kafka.network.io
- kafka.purgatory.size

- kafka.partition.count
- kafka.partition.offline
- kafka.partition.under_replicated
- kafka.isr.operation.count
- kafka.max.lag
- kafka.controller.active.count
- kafka.leader.election.rate
- kafka.unclean.election.rate
- kafka.request.queue
- kafka.logs.flush.time.count
- kafka.logs.flush.time.median
- kafka.logs.flush.time.99p
- kafka.consumer.fetch-rate
- kafka.consumer.records-lag-max
- kafka.consumer.total.bytes-consumed-rate
- kafka.consumer.total.fetch-size-avg
- kafka.consumer.total.records-consumed-rate
- kafka.consumer.bytes-consumed-rate
- kafka.consumer.fetch-size-avg
- kafka.consumer.records-consumed-rate
- kafka.producer.io-wait-time-ns-avg
- kafka.producer.outgoing-byte-rate
- kafka.producer.request-latency-avg
- kafka.producer.request-rate
- kafka.producer.response-rate
- kafka.producer.byte-rate
- kafka.producer.compression-rate
- kafka.producer.record-error-rate
- kafka.producer.record-retry-rate
- kafka.producer.record-send-rate

Troubleshooting

There are a few things that can cause the setup of the project to fail. Be sure to check the following.

- You must complete all Prerequisites before installing the solution.
- The cluster must have at least one node in it before attempting to create the solution or access the metrics.
- Your Amazon EKS cluster must have the `AWS CNI`, `CoreDNS` and `kube-proxy` add-ons installed. If they are not installed, the solution will not work correctly. They are installed by default, when creating the cluster through the console. You may need to install them if the cluster was created through an AWS SDK.
- Amazon EKS pods installation timed out. This can happen if there is not enough node capacity available. There are multiple causes of these issues, including:
  - The Amazon EKS cluster was initialized with Fargate instead of Amazon EC2. This project requires Amazon EC2.
  - The nodes are tainted and therefore unavailable.

    You can use `kubectl describe node` *`NODENAME`* `| grep Taints` to check the taints. Then `kubectl taint node` *`NODENAME TAINT_NAME`*`-` to remove the taints. Make sure to include the `-` after the taint name.
  - The nodes have reached the capacity limit. In this case you can create a new node or increase the capacity.
- You do not see any dashboards in Grafana: using the incorrect Grafana workspace ID.

  Run the following command to get information about Grafana:

```
kubectl describe grafanas external-grafana -n grafana-operator
```

  You can check the results for the correct workspace URL. If it is not the one you are expecting, re-deploy with the correct workspace ID.

```
Spec:
  External:
    API Key:
      Key:   GF_SECURITY_ADMIN_APIKEY
      Name:  grafana-admin-credentials
```

```
    URL:     https://g-123example.grafana-workspace.aws-region.amazonaws.com
Status:
  Admin URL:  https://g-123example.grafana-workspace.aws-region.amazonaws.com
  Dashboards:
    ...
```

- You do not see any dashboards in Grafana: You are using an expired API key.

  To look for this case, you will need to get the grafana operator and check the logs for errors. Get the name of the Grafana operator with this command:

  ```
  kubectl get pods -n grafana-operator
  ```

  This will return the operator name, for example:

  ```
  NAME                                READY   STATUS    RESTARTS   AGE
  grafana-operator-1234abcd5678ef90   1/1     Running   0          1h2m
  ```

  Use the operator name in the following command:

  ```
  kubectl logs grafana-operator-1234abcd5678ef90 -n grafana-operator
  ```

  Error messages such as the following indicate an expired API key:

  ```
  ERROR   error reconciling datasource    {"controller": "grafanadatasource",
   "controllerGroup": "grafana.integreatly.org", "controllerKind": "GrafanaDatasource",
   "GrafanaDatasource": {"name":"grafanadatasource-sample-amp","namespace":"grafana-
  operator"}, "namespace": "grafana-operator", "name": "grafanadatasource-sample-
  amp", "reconcileID": "72cfd60c-a255-44a1-bfbd-88b0cbc4f90c", "datasource":
   "grafanadatasource-sample-amp", "grafana": "external-grafana", "error": "status:
   401, body: {\"message\":\"Expired API key\"}\n"}
  github.com/grafana-operator/grafana-operator/controllers.
  (*GrafanaDatasourceReconciler).Reconcile
  ```

  In this case, create a new API key and deploy the solution again. If the problem persists, you can force synchronization by using the following command before redeploying:

  ```
  kubectl delete externalsecret/external-secrets-sm -n grafana-operator
  ```

- Missing SSM parameter. If you see an error like the following, run `cdk bootstrap` and try again.

```
Deployment failed: Error: aws-observability-solution-kafka-eks-$EKS_CLUSTER_NAME:
 SSM
parameter /cdk-bootstrap/xxxxxxx/version not found. Has the environment been
bootstrapped? Please run 'cdk bootstrap' (see https://docs.aws.amazon.com/cdk/latest/
guide/bootstrapping.html)
```

Add tags to Amazon Managed Grafana resources

A *tag* is a custom attribute label that you or AWS assigns to an AWS resource. Each AWS tag has two parts:

- A *tag key* (for example, `CostCenter`, `Environment`, `Project`, or `Secret`). Tag keys are case sensitive.
- An optional field known as a *tag value* (for example, `111122223333`, `Production` , or a team name). Omitting the tag value is the same as using an empty string. Like tag keys, tag values are case sensitive.

Together these are known as key-value pairs. You can have as many as 50 tags assigned to each workspace.

Tags help you identify and organize your AWS resources. Many AWS services support tagging, so you can assign the same tag to resources from different services to indicate that the resources are related. For example, you can assign the same tag to Amazon Managed Grafana workspaces that are related. For more information about tagging strategies, see Tagging AWS resources in the AWS General Reference Guide.

Amazon Managed Grafana supports tagging workspaces. You can use the console, the AWS CLI, APIs, or SDKs to add, manage, and remove tags for these resources. In addition to identifying, organizing, and tracking your workspaces with tags, you can use tags in IAM policies to help control who can view and interact with your Amazon Managed Grafana resources. For more information on tag-based access control, see Controlling access to AWS resources using tags in the IAM User Guide.

Tag restrictions

The following basic restrictions apply to tags:

- Each resource can have a maximum of 50 tags.
- For each resource, each tag key must be unique, and each tag key can have only one value.
- The maximum tag key length is 128 Unicode characters in UTF-8.
- The maximum tag value length is 256 Unicode characters in UTF-8.
- If your tagging schema is used across multiple AWS services and resources, remember that other services might have restrictions on allowed characters. Generally allowed characters are letters, numbers, spaces representable in UTF-8, and the following characters: . : + = @ _ / - (hyphen).

- Tag keys and values are case sensitive. As a best practice, decide on a strategy for capitalizing tags and consistently implement that strategy across all resource types. For example, decide whether to use `Costcenter`, `costcenter`, or `CostCenter` and use the same convention for all tags. Avoid using similar tags with inconsistent case treatment.
- Do not use `aws:`, `AWS:`, or any combination of the upper or lower case of the keyword `AWS` as a prefix for either keys or values. These are reserved only for AWS use. You can't edit or delete tag keys or values with this prefix. Tags with this prefix do not count against your tags-per-resource limit.

For more information on tagging restrictions, see Tagging AWS resources in the AWS General Reference Guide.

Tag Amazon Managed Grafana workspaces

Use the procedures in this section to work with tags for Amazon Managed Grafana workspaces.

Add a tag to a workspace

Adding tags (key-value pairs) to an Amazon Managed Grafana workspace can help you identify and organize your AWS resources. First, you add one or more tags to a workspace, then you can create IAM policies to manage access to the workspace based on these tags. You can use the console or the AWS CLI to add tags to an Amazon Managed Grafana workspace.

> **Important**
>
> Adding tags to a workspace can impact access to that workspace. Before you add a tag to a workspace, make sure to review any IAM policies that might use tags to control access to resources.

For more information about adding tags to an Amazon Managed Grafana workspace when you create it, see Create an Amazon Managed Grafana workspace in Amazon Managed Grafana User Guide.

Console

You can use the console to add one or more tags to an Amazon Managed Grafana workspace.

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.
2. In the navigation pane on the left, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the workspace ID of the workspace that you want to manage.
5. Choose the **Tags** tab.
6. Choose **Manage tags**.
7. In the **Key** field, enter a name for the tag. You can add an optional value for the tag in the **Value** field.
8. (Optional) To add another tag, choose **Add tag**.
9. When you have finished adding tags, choose **Save changes**.

CLI

Follow these steps to use the AWS CLI to add one or more tags to an Amazon Managed Grafana workspace:

- At the terminal or command line, run the `tag-resource` command, specifying the Amazon Resource Name (ARN) of the workspace where you want to add tags to and the key and the value of the tag you want to add. You can add more than one tag to a workspace. For example, to tag the Grafana workspace *My-Workspace* with a tag key named *Name* with the tag value of *my_key_value*, run the following command:

```
aws grafana tag-resource --resource-arn arn:aws:grafana:us-west-2:123456789012:/
workspace/My-Workspace --tags "Name=my_key_value"
```

View tags for a workspace

You can view the tags associated with a Amazon Managed Grafana workspace. For more information about tagging strategies, see Tagging AWS resources in AWS General Reference.

Console

You can use the console to view the tags associated with an Amazon Managed Grafana workspace.

1. Open the Amazon Managed Grafana console at https://console.aws.amazon.com/grafana/.

2. In the navigation pane on the left, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the workspace ID of the workspace that you want to manage.
5. Choose the **Tags** tab.

CLI

Follow these steps to use the AWS CLI to view the AWS tags for an Amazon Managed Grafana workspace. If no tags have been added, the returned list is empty.

At the terminal or command line, run the **list-tags-for-resource** command. For example, to view a list of tag keys and tag values for a workspace, run the following command:

```
aws grafana list-tags-for-resource --resoure-arn arn:aws:grafana:us-west-2:/
workspace/workspace-IDstring
```

If successful, this command returns information similar to the following:

```
{
    "tags": {
        "Status": "Secret",
        "Team": "My-Team"
    }
}
```

Edit tags for a workspace

You can change the value of a tag associated with a workspace in a single call using `TagResource` API. To update the key of an existing tag, you must combine the `UntagResource` and `TagResource` APIs.

> **Important**
>
> Editing tags for an Amazon Managed Grafana workspace can impact access to that workspace. Before you edit a tag for a workspace, make sure to review any IAM policies that might use the key or value of a tag to control access to resources such as repositories.

Console

You can use the console to edit the tags associated with Amazon Managed Grafana workspace.

1. Open the Amazon Managed Grafana console at Grafana Console.
2. In the navigation pane on the left, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the workspace ID of the workspace that you want to manage.
5. Choose the **Tags** tab.
6. Choose **Manage tags**.
7. In the **Key** field, enter a name for the tag. You can add an optional value for the tag in the **value** field.
8. When you have finished editing tags, choose **Save changes**.

CLI

Follow these steps to use the AWS CLI to update a tag for a workspace. You can change the value of an existing key, or add another key.

At the terminal or command line, run the **tag-resource** command, specifying the Amazon Resource Name (ARN) of the Amazon Managed Grafana workspace where you want to update a tag and specify the tag key and tag value.

For example, to change a tag's value with a new value, *Key_value_new*, run the following command.

```
aws grafana tag-resource \
    --resource-arn arn:aws:grafana:us-west-2:123456789012:/workspace/workspace-
IDstring \
    --tags "Name=Key_value_new"
```

To change a tag's key with a new name, *Name_new*, run the following commands:

```
aws grafana untag-resource --resource-arn arn:aws:grafana:us-west-2:123456789012:/
workspace/workspace-IDstring --tag-keys Items=Name
aws grafana tag-resource --resource-arn arn:aws:grafana:us-west-2:123456789012:/
workspace/workspace-IDstring --tags "Name_new=Key_value_old"
```

Remove a tag from a workspace

You can remove one or more tags associated with a workspace. Removing a tag does not delete the tag from other AWS resources that are associated with that tag.

> **Important**
>
> Removing tags from an Amazon Managed Grafana workspace can impact access to that workspace. Before you remove a tag from a workspace, make sure to review any IAM policies that might use the key or value of a tag to control access to resources such as workspaces.

Console

You can use the console to remove the association between a tag and a workspace.

1. Open the Amazon Managed Grafana console at Grafana Console.
2. In the navigation pane on the left, choose the menu icon.
3. Choose **All workspaces**.
4. Choose the workspace ID of the workspace that you want to manage.
5. Choose the **Tags** tab.
6. Choose **Manage tags**.
7. Find the tag you want to delete, and choose **Remove**.
8. When you have finished removing tags, choose **Save changes**.

CLI

Follow these steps to use the AWS CLI to remove a tag from a workspace. Removing a tag does not delete the tag from other resources, but it simply removes the association between the tag and the workspace.

> **Note**
>
> If you delete an Amazon Managed Grafana workspace, all tag associations are removed from the deleted workspace. You do not have to remove tags before you delete a workspace.

At the terminal or command line, run the **untag-resource** command, specifying the Amazon Resource Name (ARN) of the workspace where you want to remove tags and the tag key of the tag you want to remove. For example, to remove a tag on a workspace named *workspace-IDstring* with the tag key *Name*, run the following command:

```
aws grafana untag-resource --resoure-arn arn:aws:grafana:us-west-2:/
workspaces/workspace-IDstring --tag-keys Items=Name
```

If successful, this command returns an empty response. To verify that the tags associated with the workspace have been removed, run the **list-tags-for-resource** command.

Security in Amazon Managed Grafana

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Amazon Managed Grafana, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon Managed Grafana. The following topics show you how to configure Amazon Managed Grafana to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon Managed Grafana resources.

Topics

- Data protection in AWS
- Identity and Access Management for Amazon Managed Grafana
- Amazon Managed Grafana permissions and policies for AWS data sources
- IAM permissions
- Compliance Validation for Amazon Managed Grafana
- Resilience in Amazon Managed Grafana
- Infrastructure Security in Amazon Managed Grafana
- Logging Amazon Managed Grafana API calls using AWS CloudTrail
- Security best practices
- Interface VPC endpoints

Data protection in AWS

The AWS shared responsibility model applies to data protection in Amazon Managed Grafana. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see Data Privacy FAQ. For information about data protection in Europe, see the General Data Protection Regulation (GDPR) Center.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with Amazon Managed Grafana or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Data protection in Amazon Managed Grafana

Amazon Managed Grafana collects and stores the following types of data:

- Customer-provided dashboard and alert configurations for Grafana workspaces.
- Grafana dashboard snapshots that you have saved to your workspace.
- A list of AWS IAM Identity Center users that have been granted access to the Grafana workspace, including the user names and email addresses of the users.

The data that Amazon Managed Grafana stores is encrypted with AWS Key Management Service. Data in transit is automatically encrypted with Secure Sockets Layer (SSL).

Identity and Access Management for Amazon Managed Grafana

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon Managed Grafana resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- How Amazon Managed Grafana works with IAM
- Identity-based policy examples for Amazon Managed Grafana
- AWS managed policies for Amazon Managed Grafana
- Troubleshooting Amazon Managed Grafana identity and access
- Cross-service confused deputy prevention
- Using service-linked roles for Amazon Managed Grafana

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting Amazon Managed Grafana identity and access)
- **Service administrator** - determine user access and submit permission requests (see How Amazon Managed Grafana works with IAM)
- **IAM administrator** - write policies to manage access (see Identity-based policy examples for Amazon Managed Grafana)

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Amazon Managed Grafana works with IAM

Before you use IAM to manage access to Amazon Managed Grafana, learn what IAM features are available to use with Amazon Managed Grafana.

IAM features you can use with Amazon Managed Grafana

IAM feature	Amazon Managed Grafana support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys	No
ACLs	No
ABAC (tags in policies)	Yes
Temporary credentials	Yes
Forward access sessions (FAS)	Yes
Service roles	Yes
Service-linked roles	Yes

To get a high-level view of how Amazon Managed Grafana and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for Amazon Managed Grafana

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can

perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for Amazon Managed Grafana

To view examples of Amazon Managed Grafana identity-based policies, see Identity-based policy examples for Amazon Managed Grafana.

Resource-based policies within Amazon Managed Grafana

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for Amazon Managed Grafana

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Amazon Managed Grafana actions, see Actions defined by Amazon Managed Grafana in the *Service Authorization Reference*.

Policy actions in Amazon Managed Grafana use the following prefix before the action:

```
grafana
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "grafana:action1",
      "grafana:action2"
         ]
```

To view examples of Amazon Managed Grafana identity-based policies, see Identity-based policy examples for Amazon Managed Grafana.

Policy resources for Amazon Managed Grafana

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of Amazon Managed Grafana resource types and their ARNs, see Resources defined by Amazon Managed Grafana in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by Amazon Managed Grafana.

To view examples of Amazon Managed Grafana identity-based policies, see Identity-based policy examples for Amazon Managed Grafana.

Policy condition keys for Amazon Managed Grafana

Supports service-specific policy condition keys: No

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element specifies when statements execute based on defined criteria. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

Access control lists (ACLs) in Amazon Managed Grafana

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Attribute-based access control (ABAC) with Amazon Managed Grafana

Supports ABAC (tags in policies): Yes

Amazon Managed Grafana supports resource and identity based tagging.

For more information about tagging Amazon Managed Grafana resources, see Tag Amazon Managed Grafana workspaces.

To view an example identity-based policy for limiting access to a resource based on the tags on that resource, see AWS managed policies for Amazon Managed Grafana.

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes called tags. You can attach tags to IAM entities and AWS resources, then design ABAC policies to allow operations when the principal's tag matches the tag on the resource.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using Temporary credentials with Amazon Managed Grafana

Supports temporary credentials: Yes

Temporary credentials provide short-term access to AWS resources and are automatically created when you use federation or switch roles. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM and AWS services that work with IAM in the *IAM User Guide*.

Forward access sessions for Amazon Managed Grafana

Supports forward access sessions (FAS): Yes

Forward access sessions (FAS) use the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. For policy details when making FAS requests, see Forward access sessions.

Service roles for Amazon Managed Grafana

Supports service roles: Yes

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break Amazon Managed Grafana functionality. Edit service roles only when Amazon Managed Grafana provides guidance to do so.

Service-linked roles for Amazon Managed Grafana

Supports service-linked roles: Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing Amazon Managed Grafana service-linked roles, see Using service-linked roles for Amazon Managed Grafana.

Identity-based policy examples for Amazon Managed Grafana

By default, users and roles don't have permission to create or modify Amazon Managed Grafana resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by Amazon Managed Grafana, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon Managed Grafana in the *Service Authorization Reference*.

Topics

- Policy best practices
- Using the Amazon Managed Grafana console
- Sample policies for Amazon Managed Grafana

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Amazon Managed Grafana resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies*

that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.

- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the Amazon Managed Grafana console

To access the console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

Sample policies for Amazon Managed Grafana

This section contains identity-based policies that are useful for several Amazon Managed Grafana scenarios.

Grafana administrator using SAML

If you use SAML for your user authentication, the administrator who creates and manages Amazon Managed Grafana needs the following policies:

- **AWSGrafanaAccountAdministrator** or the equivalent permissions to create and manage Amazon Managed Grafana workspaces.
- The **AWSMarketplaceManageSubscriptions** policy or equivalent permissions, if you want to upgrade an Amazon Managed Grafana workspace to Grafana Enterprise.

Grafana administrator in a management account using IAM Identity Center

To grant permissions to create and manage Amazon Managed Grafana workspaces across an entire organization, and to enable dependencies such as IAM Identity Center, assign the **AWSGrafanaAccountAdministrator**, **AWSSSOMasterAccountAdministrator** and the **AWSSSODirectoryAdministrator** policies to a user. Additionally, to upgrade an Amazon Managed Grafana workspace to Grafana Enterprise, a user must have the **AWSMarketplaceManageSubscriptions** IAM policy or the equivalent permissions.

If you want to use service-managed permissions when you create an Amazon Managed Grafana workspace, the user who creates the workspace must also have the `iam:CreateRole`, `iam:CreatePolicy`, and `iam:AttachRolePolicy` permissions. These are required to use CloudFormation StackSets to deploy policies that enable you to read data sources in the organization's accounts.

> **Important**
>
> Granting a user the `iam:CreateRole`, `iam:CreatePolicy`, and `iam:AttachRolePolicy` permissions gives that user full administrative access to your AWS account. For example, a user with these permissions can create a policy that has full permissions for all resources, and attach that policy to any role. Be very careful about who you grant these permissions to.

To see the permissions granted to **AWSGrafanaAccountAdministrator**, see AWS managed policy: AWSGrafanaAccountAdministrator

Grafana administrator in a member account using IAM Identity Center

To grant permissions to create and manage Amazon Managed Grafana workspaces in the member account of an organization, assign the **AWSGrafanaAccountAdministrator**, **AWSSSOMemberAccountAdministrator** and the **AWSSSODirectoryAdministrator** policies to a user. Additionally, to upgrade an Amazon Managed Grafana workspace to Grafana Enterprise, a user must have the **AWSMarketplaceManageSubscriptions** IAM policy or the equivalent permissions.

If you want to use service-managed permissions when you create an Amazon Managed Grafana workspace, the user who creates the workspace must also have the `iam:CreateRole`, `iam:CreatePolicy`, and `iam:AttachRolePolicy` permissions. These are required to enable the user to read data sources in the account.

> **Important**
>
> Granting a user the `iam:CreateRole`, `iam:CreatePolicy`, and `iam:AttachRolePolicy` permissions gives that user full administrative access to your AWS account. For example, a user with these permissions can create a policy that has full permissions for all resources, and attach that policy to any role. Be very careful about who you grant these permissions to.

To see the permissions granted to **AWSGrafanaAccountAdministrator**, see AWS managed policy: AWSGrafanaAccountAdministrator

Create and manage Amazon Managed Grafana workspaces and users in a single standalone account using IAM Identity Center

A standalone AWS account is an account that is not yet a member of an organization. For more information about organizations, see What is AWS Organizations?

To grant permissions to create and manage Amazon Managed Grafana workspaces and users in a standalone account, assign the **AWSGrafanaAccountAdministrator**, **AWSSSOMasterAccountAdministrator**, **AWSOrganizationsFullAccess** and **AWSSSODirectoryAdministrator** policies to a user. Additionally, to upgrade an

Amazon Managed Grafana workspace to Grafana Enterprise, a user must have the **AWSMarketplaceManageSubscriptions** IAM policy or the equivalent permissions.

> **Important**
>
> Granting a user the `iam:CreateRole`, `iam:CreatePolicy`, and `iam:AttachRolePolicy` permissions gives that user full administrative access to your AWS account. For example, a user with these permissions can create a policy that has full permissions for all resources, and attach that policy to any role. Be very careful about who you grant these permissions to.

To see the permissions granted to **AWSGrafanaAccountAdministrator**, see AWS managed policy: AWSGrafanaAccountAdministrator

Assign and unassign users access to Amazon Managed Grafana

To grant permissions to manage other users' access to Amazon Managed Grafana workspaces in the account, including granting Grafana admin permissions to those users for the workspaces,assign the **AWSGrafanaWorkspacePermissionManagementV2** policy to that user. If you are using IAM Identity Center to manage users in this workspace, the user also needs the **AWSSSOReadOnly** and **AWSSSODirectoryReadOnly** policies.

To see the permissions granted to **AWSGrafanaWorkspacePermissionManagementV2**, see AWS managed policy: AWSGrafanaWorkspacePermissionManagementV2

Amazon Managed Grafana read-only permissions

To grant permissions for read actions, such as listing and viewing workspaces and opening the Grafana workspace console, assign the **AWSGrafanaConsoleReadOnlyAccess**, **AWSSSOReadOnly** and **AWSSSODirectoryReadOnly** policies to a user or IAM role.

To see the permissions granted to **AWSGrafanaConsoleReadOnlyAccess**, see AWS managed policy: AWSGrafanaConsoleReadOnlyAccess.

AWS managed policies for Amazon Managed Grafana

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AWSGrafanaAccountAdministrator

AWSGrafanaAccountAdministrator policy provides access within Amazon Managed Grafana to create and manage accounts and workspaces for the entire organization.

You can attach AWSGrafanaAccountAdministrator to your IAM entities.

Permissions details

This policy includes the following permissions.

- `iam` – Allows principals to list and get IAM roles so that the administrator can associate a role with a workspace as well as pass roles to the Amazon Managed Grafana service.
- `Amazon Managed Grafana` – Allows principals read and write access to all Amazon Managed Grafana APIs.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AWSGrafanaOrganizationAdmin",
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Sid": "GrafanaIAMGetRolePermission",
            "Effect": "Allow",
            "Action": "iam:GetRole",
            "Resource": "arn:aws:iam::*:role/*"
        },
        {
            "Sid": "AWSGrafanaPermissions",
            "Effect": "Allow",
            "Action": [
                "grafana:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "GrafanaIAMPassRolePermission",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "grafana.amazonaws.com"
                }
            }
        }
    ]
}
```

AWS managed policy: AWSGrafanaWorkspacePermissionManagement (obsolete)

This policy is obsolete. This policy should not be attached to any new users, groups, or roles.

Amazon Managed Grafana added a new policy, AWSGrafanaWorkspacePermissionManagementV2 to replace this policy. This new managed policy improves security for your workspace by providing a more restrictive set of permissions.

AWS managed policy: AWSGrafanaWorkspacePermissionManagementV2

AWSGrafanaWorkspacePermissionManagementV2 policy provides only the ability to update user and group permissions for Amazon Managed Grafana workspaces.

You can attach AWSGrafanaWorkspacePermissionManagementV2 to your IAM entities.

Permissions details

This policy includes the following permissions.

- `Amazon Managed Grafana` – Allows principals to read and update user and group permissions for Amazon Managed Grafana workspaces.
- `IAM Identity Center` – Allows principals to read IAM Identity Center entities. This is a necessary part of associating principals with Amazon Managed Grafana applications, but that also requires an additional step, described after the policy listing that follows.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
            "Sid": "AWSGrafanaPermissions",
            "Effect": "Allow",
            "Action": [
                "grafana:DescribeWorkspace",
                "grafana:DescribeWorkspaceAuthentication",
                "grafana:UpdatePermissions",
                "grafana:ListPermissions",
                "grafana:ListWorkspaces"
            ],
            "Resource": "arn:aws:grafana:*:*:/workspaces*"
        },
        {
            "Sid": "IAMIdentityCenterPermissions",
            "Effect": "Allow",
```

```
            "Action": [
                "sso:DescribeRegisteredRegions",
                "sso:GetSharedSsoConfiguration",
                "sso:ListDirectoryAssociations",
                "sso:GetManagedApplicationInstance",
                "sso:ListProfiles",
                "sso:GetProfile",
                "sso:ListProfileAssociations",
                "sso-directory:DescribeUser",
                "sso-directory:DescribeGroup"
            ],
            "Resource": "*"
        }
    ]
}
```

Additional policy needed

To fully allow a user to assign permissions, in addition to the `AWSGrafanaWorkspacePermissionManagementV2` policy, you must also assign a policy to provide access to Application assignment in IAM Identity Center.

To create this policy, you must first collect the **Grafana application ARN** for your workspace

1. Open the IAM Identity Center console.
2. Choose **Applications** from the left menu.
3. Under the **AWS managed** tab, find the application called **Amazon Grafana-*workspace-name***, where `workspace-name` is the name of your workspace. Select the application name.
4. The IAM Identity Center application managed by Amazon Managed Grafana for the workspace is shown. This application's ARN is shown in the details page. It will be in the form: `arn:aws:sso::`*`owner-account-id`*`:application/ssoins-`*`unique-id`*`/apl-`*`unique-id`*.

The policy you create should look like the following. Replace *`grafana-application-arn`* with the ARN that you found in the previous step:

For information about how to create and apply policy to your roles or users, see Adding and removing IAM identity permissions in the *AWS Identity and Access Management User Guide*.

AWS managed policy: AWSGrafanaConsoleReadOnlyAccess

AWSGrafanaConsoleReadOnlyAccess policy grants access to read-only operations in Amazon Managed Grafana.

You can attach AWSGrafanaConsoleReadOnlyAccess to your IAM entities.

Permissions details

This policy includes the following permission.

- `Amazon Managed Grafana` – Allows principals read-only access to Amazon Managed Grafana APIs

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AWSGrafanaConsoleReadOnlyAccess",
      "Effect": "Allow",
      "Action": ["grafana:Describe*", "grafana:List*"],
      "Resource": "*"
    }
  ]
}
```

AWS managed policy: AmazonGrafanaRedshiftAccess

This policy grants scoped access to Amazon Redshift and the dependencies needed to use the Amazon Redshift plugin in Amazon Managed Grafana. AmazonGrafanaRedshiftAccess policy allows a user or an IAM role to use the Amazon Redshift data source plugin in Grafana. Temporary credentials for Amazon Redshift databases are scoped to the database user `redshift_data_api_user` and credentials from Secrets Manager can be retrieved if the secret is tagged with the key `RedshiftQueryOwner`. This policy allows access to Amazon Redshift clusters tagged with `GrafanaDataSource`. When creating a customer managed policy, the tag-based authentication is optional.

You can attach AmazonGrafanaRedshiftAccess to your IAM entities. Amazon Managed Grafana also attaches this policy to a service role that allows Amazon Managed Grafana to perform actions on your behalf.

Permissions details

This policy includes the following permission.

- `Amazon Redshift` – Allows principals to describe clusters and obtain temporary credentials for a database user named `redshift_data_api_user`.
- `Amazon Redshift-data` – Allows principals to execute queries on clusters tagged as `GrafanaDataSource`.
- `Secrets Manager` – Allows principals to list secrets and read secret values for secrets tagged as `RedshiftQueryOwner`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "redshift:DescribeClusters",
        "redshift-data:GetStatementResult",
        "redshift-data:DescribeStatement",
        "secretsmanager:ListSecrets"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "redshift-data:DescribeTable",
        "redshift-data:ExecuteStatement",
        "redshift-data:ListTables",
        "redshift-data:ListSchemas"
      ],
      "Resource": "*",
      "Condition": {
```

```
        "Null": {
          "aws:ResourceTag/GrafanaDataSource": "false"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "redshift:GetClusterCredentials",
      "Resource": [
        "arn:aws:redshift:*:*:dbname:*/*",
        "arn:aws:redshift:*:*:dbuser:*/redshift_data_api_user"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": "*",
      "Condition": {
        "Null": {
          "secretsmanager:ResourceTag/RedshiftQueryOwner": "false"
        }
      }
    }
  ]
}
```

AWS managed policy: AmazonGrafanaAthenaAccess

This policy grants access to Athena and the dependencies needed to enable querying and writing results to Amazon S3 from the Athena plugin in Amazon Managed Grafana. AmazonGrafanaAthenaAccess policy allows a user or an IAM role to use the Athena data source plugin in Grafana. Athena workgroups must be tagged with `GrafanaDataSource` to be accessible. This policy contains permissions for writing query results in an Amazon S3 bucket with a name prefixed with `grafana-athena-query-results-`. Amazon S3 permissions for accessing the underlying data source of an Athena query are not included in this policy.

You can attach AWSGrafanaAthenaAccess policy to your IAM entities. Amazon Managed Grafana also attaches this policy to a service role that allows Amazon Managed Grafana to perform actions on your behalf.

Permissions details

This policy includes the following permission.

- `Athena` – Allows principals to run queries on Athena resources in workgroups tagged as `GrafanaDataSource`.
- `Amazon S3` – Allows principals to read and write query results to a bucket prefixed with `grafana-athena-query-results-`.
- `AWS Glue` – Allows principals access to AWS Glue databases, tables, and partitions. This is required so that the principal can use the AWS Glue Data Catalog with Athena.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
     {
         "Effect": "Allow",
         "Action": [
             "athena:GetDatabase",
             "athena:GetDataCatalog",
             "athena:GetTableMetadata",
             "athena:ListDatabases",
             "athena:ListDataCatalogs",
             "athena:ListTableMetadata",
             "athena:ListWorkGroups"
         ],
         "Resource": [
             "*"
         ]
     },
     {
         "Effect": "Allow",
         "Action": [
             "athena:GetQueryExecution",
             "athena:GetQueryResults",
             "athena:GetWorkGroup",
             "athena:StartQueryExecution",
             "athena:StopQueryExecution"
         ],
         "Resource": [
```

```
                "*"
            ],
            "Condition": {
                "Null": {
                    "aws:ResourceTag/GrafanaDataSource": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:GetTable",
                "glue:GetTables",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject",
                "s3:PutBucketPublicAccessBlock"
            ],
            "Resource": [
                "arn:aws:s3:::grafana-athena-query-results-*"
            ]
        }
    ]
}
```

AWS managed policy: AmazonGrafanaCloudWatchAccess

This policy grants access to Amazon CloudWatch and the dependencies needed to use CloudWatch as a datasource within Amazon Managed Grafana.

You can attach AWSGrafanaCloudWatchAccess policy to your IAM entities. Amazon Managed Grafana also attaches this policy to a service role that allows Amazon Managed Grafana to perform actions on your behalf.

Permissions details

This policy includes the following permissions.

- `CloudWatch` – Allows principals to list and get metric data and logs from Amazon CloudWatch. It also allows viewing data shared from source accounts in CloudWatch cross-account observability.
- `Amazon EC2` – Allows principals to get details regarding resources that are being monitored.
- `Tags` – Allows principals to access tags on resources, to allow filtering the CloudWatch metric queries.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:GetMetricData",
        "cloudwatch:GetInsightRuleReport"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "logs:DescribeLogGroups",
        "logs:GetLogGroupFields",
        "logs:StartQuery",
        "logs:StopQuery",
        "logs:GetQueryResults",
        "logs:GetLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeTags",
        "ec2:DescribeInstances",
        "ec2:DescribeRegions"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "tag:GetResources",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "oam:ListSinks",
        "oam:ListAttachedLinks"
      ],
      "Resource": "*"
    }
  ]
}
```

Amazon Managed Grafana updates to AWS managed policies

View details about updates to AWS managed policies for Amazon Managed Grafana since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Amazon Managed Grafana document history page.

Change	Description	Date
AWSGrafanaWorkspacePermissionManagement – obsolete	This policy has been replaced by **AWSGrafanaWorkspacePermissionManagementV2**. This policy is considered obsolete, and will no longer be updated. The new policy improves security for your workspace by providing a more restrictive set of permissions.	January 5, 2024
AWSGrafanaWorkspacePermissionManagementV2 – New policy	Amazon Managed Grafana added a new policy, **AWSGrafanaWorkspacePermissionManagementV2** to replace the obsolete **AWSGrafanaWorkspacePermissionManagement** policy. This new managed policy improves security for your workspace by providing a more restrictive set of permissions.	January 5, 2024
AmazonGrafanaCloudWatchAccess – New policy	Amazon Managed Grafana added a new policy **AmazonGrafanaCloudWatchAccess**.	March 24, 2023
AWSGrafanaWorkspacePermissionManagement – Update to an existing policy	Amazon Managed Grafana added new permissions to **AWSGrafanaWorkspacePermissionManagement** so that IAM Identity Center	March 14, 2023

Change	Description	Date
	users and groups in Active Directory can be associated with Grafana workspaces. The following permissions were added: `sso-directory:DescribeUser` , and `sso-directory:DescribeGroup`	
AWSGrafanaWorkspacePermissionManagement – Update to an existing policy	Amazon Managed Grafana added new permissions to **AWSGrafanaWorkspacePermissionManagement** so that IAM Identity Center users and groups can be associated with Grafana workspaces. The following permissions were added: `sso:DescribeRegisteredRegions` , `sso:GetSharedSsoConfiguration` , `sso:ListDirectoryAssociations` , `sso:GetManagedApplicationInstance` , `sso:ListProfiles` , `sso:AssociateProfile` , `sso:DisassociateProfile` , `sso:GetProfile` , and `sso:ListProfileAssociations` .	December 20, 2022

Change	Description	Date
AmazonGrafanaServiceLinkedRolePolicy – New SLR policy	Amazon Managed Grafana added a new policy for the Grafana service-linked role, **AmazonGrafanaServiceLinkedRolePolicy**.	November 18, 2022
AWSGrafanaAccountAdministrator, AWSGrafanaConsoleReadOnlyAccess	Allow access to all Amazon Managed Grafana resources	February 17, 2022
AmazonGrafanaRedshiftAccess – New policy	Amazon Managed Grafana added a new policy **AmazonGrafanaRedshiftAccess**.	November 26, 2021
AmazonGrafanaAthenaAccess – New policy	Amazon Managed Grafana added a new policy **AmazonGrafanaAthenaAccess**.	November 22, 2021
AWSGrafanaAccountAdministrator – Update to an existing policy	Amazon Managed Grafana removed permissions from **AWSGrafanaAccountAdministrator**. The `iam:CreateServiceLinkedRole` permission scoped to the `sso.amazonaws.com` service was removed, and instead we recommend that you attach the **AWSSSOMasterAccountAdministrator** policy to grant this permission to a user.	October 13, 2021

Change	Description	Date
AWSGrafanaWorkspacePermissionManagement – Update to an existing policy	Amazon Managed Grafana added new permissions to **AWSGrafanaWorkspacePermissionManagement** so that users with this policy can see the authentication methods associated with workspaces. The `grafana:DescribeWorkspaceAuthentication` permission was added.	September 21, 2021
AWSGrafanaConsoleReadOnlyAccess – Update to an existing policy	Amazon Managed Grafana added new permissions to **AWSGrafanaConsoleReadOnlyAccess** so that users with this policy can see the authentication methods associated with workspaces. The `grafana:Describe*` and `grafana:List*` permissions were added to the policy, and they replace the previous narrower permissions `grafana:DescribeWorkspace`, `grafana:ListPermissions`, and `grafana:ListWorkspaces`.	September 21, 2021
Amazon Managed Grafana started tracking changes	Amazon Managed Grafana started tracking changes for its AWS managed policies.	September 9, 2021

Troubleshooting Amazon Managed Grafana identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon Managed Grafana and IAM.

Topics

- I am not authorized to perform an action in Amazon Managed Grafana
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my Amazon Managed Grafana resources

I am not authorized to perform an action in Amazon Managed Grafana

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `grafana:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 grafana:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `grafana:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Amazon Managed Grafana.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Amazon Managed Grafana. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Amazon Managed Grafana resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon Managed Grafana supports these features, see How Amazon Managed Grafana works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service

impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in resource policies to limit the permissions that Amazon Managed Grafana gives another service to the resource. If the `aws:SourceArn` value does not contain the account ID, such as an Amazon S3 bucket ARN, you must use both global condition context keys to limit permissions. If you use both global condition context keys and the `aws:SourceArn` value contains the account ID, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement. Use `aws:SourceArn` if you want only one resource to be associated with the cross-service access. Use `aws:SourceAccount` if you want to allow any resource in that account to be associated with the cross-service use.

The value of `aws:SourceArn` must be the ARN of your Amazon Managed Grafana workspace.

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` global condition context key with the full ARN of the resource. If you don't know the full ARN of the resource or if you are specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcards (*) for the unknown portions of the ARN. For example, `arn:aws:grafana:*:123456789012:*`.

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in Amazon Managed Grafana Workspace IAM role trust policies to prevent the confused deputy problem.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
   {
     "Effect": "Allow",
     "Principal": {
       "Service": "grafana.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "accountId",
          "aws:SourceArn": "arn:aws:grafana:region:accountId:/
workspaces/workspaceId"
        }
      }
    }
  ]
}
```

Using service-linked roles for Amazon Managed Grafana

Amazon Managed Grafana uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to Amazon Managed Grafana. Service-linked roles are predefined by Amazon Managed Grafana and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Amazon Managed Grafana easier because you don't have to manually add the necessary permissions. Amazon Managed Grafana defines the permissions of its service-linked roles, and unless defined otherwise, only Amazon Managed Grafana can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting their related resources. This protects your Amazon Managed Grafana resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have **Yes** in the **Service-linked roles** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for Amazon Managed Grafana

Amazon Managed Grafana uses the service-linked role named **AmazonManagedGrafana** – Amazon Managed Grafana uses this role to create and configure resources, such as ENIs or Secrets Manager

secrets, within customer accounts. The AmazonManagedGrafana service-linked role trusts the following services to assume the role:

- `grafana.amazonaws.com`

The AmazonManagedGrafana service-linked role is attached to the `AmazonGrafanaServiceLinkedRolePolicy` policy. For updates to this policy, see Amazon Managed Grafana updates to AWS managed policies.

The role permissions policy allows Amazon Managed Grafana to complete the following actions on the specified resources.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterface",
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "AmazonGrafanaManaged"
                    ]
                }
            }
        },
        {
```

```
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
                "Condition": {
                    "StringEquals": {
                    "ec2:CreateAction": "CreateNetworkInterface"
                },
                "Null": {
                    "aws:RequestTag/AmazonGrafanaManaged": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DeleteNetworkInterface",
            "Resource": "*",
            "Condition": {
                "Null": {
                    "ec2:ResourceTag/AmazonGrafanaManaged": "false"
                }
            }
        }
    ]
}
```

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-linked role permissions in the *IAM User Guide*.

Creating a service-linked role for Amazon Managed Grafana

You don't need to manually create a service-linked role. When you call CreateWorkspace with a VpcConfiguration in the AWS Management Console, the AWS CLI, or the AWS API, Amazon Managed Grafana creates the service-linked role for you.

> **Important**
>
> This service-linked role can appear in your account if you completed an action in another service that uses the features supported by this role. Also, if you were using the Amazon Managed Grafana service before November 30, 2022, when it began supporting service-

linked roles, then Amazon Managed Grafana created the AmazonManagedGrafana role in your account. To learn more, see A new role appeared in my IAM account.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you call CreateWorkspace with a VpcConfiguration, Amazon Managed Grafana creates the service-linked role for you again.

You can also use the IAM console to create a service-linked role with the **Grafana** use case. In the AWS CLI or the AWS API, create a service-linked role with the `grafana.amazonaws.com` service name. For more information, see Creating a service-linked role in the *IAM User Guide*. If you delete this service-linked role, you can use this same process to create the role again.

Editing a service-linked role for Amazon Managed Grafana

Amazon Managed Grafana does not allow you to edit the AmazonManagedGrafana service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

Deleting a service-linked role for Amazon Managed Grafana

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

Note

If the Amazon Managed Grafana service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To delete Amazon Managed Grafana resources used by the AmazonManagedGrafana

1. Navigate to the **All workspaces** view in your `Region` in the AWS console.
2. Delete all the workspaces in the `Region`. You have to check the radio button for each workspace and choose the **delete** button in the upper right side of the **All workspaces** view.

Repeat deleting each workspace until all the workspaces are deleted from the `Region`. For more information about deleting a workspace in Amazon Managed Grafana, see Deleting a workspace topic in this user guide.

> **Note**
>
> Repeat the procedure for each AWS Region where you have workspaces. You must delete all workspaces *in all Regions* before you can delete the service-linked role.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the AmazonManagedGrafana service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Supported regions for Amazon Managed Grafana service-linked roles

Amazon Managed Grafana supports using service-linked roles in all of the regions where the service is available. For more information, see AWS regions and endpoints.

Amazon Managed Grafana permissions and policies for AWS data sources

Amazon Managed Grafana offers three permission modes:

- Service-managed permissions for current account
- Service-managed permissions for organizations
- Customer-managed permissions

When you create a workspace, you choose which permission mode to use. You can also change this later if you want.

In either of the service-managed permission modes, Amazon Managed Grafana creates roles and policies that are needed to access and discover AWS data sources in your account or organization. You can then edit these policies in the IAM console if you choose.

Service-managed permissions for a single account

In this mode, Amazon Managed Grafana creates a role called **AmazonGrafanaServiceRole-*random-id***. Amazon Managed Grafana then attaches a policy to this role for each AWS service that you select to access from the Amazon Managed Grafana workspace.

CloudWatch

Amazon Managed Grafana attaches the AWS managed policy **AmazonGrafanaCloudWatchAccess**.

> **Note**
>
> For workspaces that used CloudWatch before the **AmazonGrafanaCloudWatchAccess** managed policy was created, Amazon Managed Grafana created a customer-managed policy with the name **AmazonGrafanaCloudWatchPolicy-*random-id***.

Amazon OpenSearch Service

Amazon Managed Grafana creates a customer-managed policy with the name **AmazonGrafanaOpenSearchPolicy-*random-id***. The Get/Post permissions are needed for data source access. The List/Describe permissions are used by Amazon Managed Grafana for data source discovery, but they aren't required for the data source plugin to work. The contents of the policy are as follows:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "es:ESHttpGet",
                "es:DescribeElasticsearchDomains",
                "es:ListDomainNames"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": "es:ESHttpPost",
            "Resource": [
                "arn:aws:es:*:*:domain/*/_msearch*",
                "arn:aws:es:*:*:domain/*/_opendistro/_ppl"
            ]
        }
    ]
}
```

AWS IoT SiteWise

Amazon Managed Grafana attaches the AWS managed policy **AWSIoTSiteWiseReadOnlyAccess**.

Amazon Redshift

Amazon Managed Grafana attaches the AWS managed policy **AmazonGrafanaRedshiftAccess**.

Amazon Athena

Amazon Managed Grafana attaches the AWS managed policy **AmazonGrafanaAthenaAccess**.

Amazon Managed Service for Prometheus

Amazon Managed Grafana creates a customer-managed policy with the name **AmazonGrafanaPrometheusPolicy-*random-id***. The List/Describe permissions are used by Amazon Managed Grafana for data source discovery, they aren't required for the plugin to work. The contents of the policy are as follows:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "aps:ListWorkspaces",
                "aps:DescribeWorkspace",
                "aps:QueryMetrics",
                "aps:GetLabels",
                "aps:GetSeries",
                "aps:GetMetricMetadata"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

Amazon SNS

Amazon Managed Grafana creates a customer-managed policy with the name **AmazonGrafanaSNSPolicy-*random-id***. The policy restricts you to only using SNS topics in your account that start with the string `grafana`. This is not necessary if you create your own policy. The contents of the policy are as follows:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:*:111122223333:grafana*"
            ]
        }
    ]
}
```

Timestream

Amazon Managed Grafana attaches the AWS managed policy **AmazonTimestreamReadOnlyAccess**.

X-Ray

Amazon Managed Grafana attaches the AWS managed policy **AWSXrayReadOnlyAccess**.

Service-managed permissions for an organization

This mode is supported only for workspaces created in management accounts or delegated administrator accounts in an organization. Delegated administrator accounts can create and

administer stack sets for the organization. For more information about delegated administrator accounts, see Register a delegated administrator.

> **Note**
>
> Creating resources such as Amazon Managed Grafana workspaces in the management account of an organization is against AWS security best practices.

In this mode, Amazon Managed Grafana creates all the IAM roles that are necessary to access AWS resources in other accounts in your AWS organization. In each account in the Organizational Units that you select, Amazon Managed Grafana creates a role called **AmazonGrafanaOrgMemberRole-*random-id***. This role creation is performed through an integration with AWS CloudFormation StackSets.

This role has a policy attached for each AWS data source that you select to use in the workspace. For the contents of these data policies, see Service-managed permissions for a single account.

Amazon Managed Grafana also creates a role called **AmazonGrafanaOrgAdminRole-*random-id*** in the organization's management account. This role allows the Amazon Managed Grafana workspace permission to access other accounts in the organization. AWS service notification channel policies also get attached to this role. Use the **AWS Data Source** menu in your workspace to quickly provision data sources for each account that your workspace can access

To use this mode, you must enable CloudFormation Stacksets as a trusted service in your AWS organization. For more information, see Enable trusted access with AWS Organizations.

Here is the content of the **AmazonGrafanaStackSet-*random-id*** stack set:

```
Parameters:
  IncludePrometheusPolicy:
    Description: Whether to include Amazon Prometheus access in the role
    Type: String
    AllowedValues:
      - true
      - false
    Default: false
  IncludeAESPolicy:
    Description: Whether to include Amazon Elasticsearch access in the role
    Type: String
    AllowedValues:
```

```
      - true
      - false
    Default: false
  IncludeCloudWatchPolicy:
    Description: Whether to include CloudWatch access in the role
    Type: String
    AllowedValues:
      - true
      - false
    Default: false
  IncludeTimestreamPolicy:
    Description: Whether to include Amazon Timestream access in the role
    Type: String
    AllowedValues:
      - true
      - false
    Default: false
  IncludeXrayPolicy:
    Description: Whether to include AWS X-Ray access in the role
    Type: String
    AllowedValues:
      - true
      - false
    Default: false
  IncludeSitewisePolicy:
    Description: Whether to include AWS IoT SiteWise access in the role
    Type: String
    AllowedValues:
      - true
      - false
    Default: false
  IncludeRedshiftPolicy:
    Description: Whether to include Amazon Redshift access in the role
    Type: String
    AllowedValues:
      - true
      - false
    Default: false
  IncludeAthenaPolicy:
    Description: Whether to include Amazon Athena access in the role
    Type: String
    AllowedValues:
      - true
      - false
```

```
    Default: false
  RoleName:
    Description: Name of the role to create
    Type: String
  AdminAccountId:
    Description: Account ID of the Amazon Grafana org admin
    Type: String
Conditions:
  addPrometheus: !Equals [!Ref IncludePrometheusPolicy, true]
  addAES: !Equals [!Ref IncludeAESPolicy, true]
  addCloudWatch: !Equals [!Ref IncludeCloudWatchPolicy, true]
  addTimestream: !Equals [!Ref IncludeTimestreamPolicy, true]
  addXray: !Equals [!Ref IncludeXrayPolicy, true]
  addSitewise: !Equals [!Ref IncludeSitewisePolicy, true]
  addRedshift: !Equals [!Ref IncludeRedshiftPolicy, true]
  addAthena: !Equals [!Ref IncludeAthenaPolicy, true]

Resources:
  PrometheusPolicy:
    Type: AWS::IAM::Policy
    Condition: addPrometheus
    Properties:
      Roles:
       - !Ref GrafanaMemberServiceRole
      PolicyName: AmazonGrafanaPrometheusPolicy
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action:
              - aps:QueryMetrics
              - aps:GetLabels
              - aps:GetSeries
              - aps:GetMetricMetadata
              - aps:ListWorkspaces
              - aps:DescribeWorkspace
            Resource: '*'

  AESPolicy:
    Type: AWS::IAM::Policy
    Condition: addAES
    Properties:
      Roles:
       - !Ref GrafanaMemberServiceRole
```

```
      PolicyName: AmazonGrafanaElasticsearchPolicy
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Sid: AllowReadingESDomains
            Effect: Allow
            Action:
              - es:ESHttpGet
              - es:ESHttpPost
              - es:ListDomainNames
              - es:DescribeElasticsearchDomains
            Resource: '*'

  CloudWatchPolicy:
    Type: AWS::IAM::Policy
    Condition: addCloudWatch
    Properties:
      Roles:
       - !Ref GrafanaMemberServiceRole
      PolicyName: AmazonGrafanaCloudWatchPolicy
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Sid: AllowReadingMetricsFromCloudWatch
            Effect: Allow
            Action:
              - cloudwatch:DescribeAlarmsForMetric
              - cloudwatch:DescribeAlarmHistory
              - cloudwatch:DescribeAlarms
              - cloudwatch:ListMetrics
              - cloudwatch:GetMetricStatistics
              - cloudwatch:GetMetricData
              - cloudwatch:GetInsightRuleReport
            Resource: "*"
          - Sid: AllowReadingLogsFromCloudWatch
            Effect: Allow
            Action:
              - logs:DescribeLogGroups
              - logs:GetLogGroupFields
              - logs:StartQuery
              - logs:StopQuery
              - logs:GetQueryResults
              - logs:GetLogEvents
            Resource: "*"
```

```
          - Sid: AllowReadingTagsInstancesRegionsFromEC2
            Effect: Allow
            Action:
              - ec2:DescribeTags
              - ec2:DescribeInstances
              - ec2:DescribeRegions
            Resource: "*"
          - Sid: AllowReadingResourcesForTags
            Effect: Allow
            Action:
              - tag:GetResources
            Resource: "*"
  GrafanaMemberServiceRole:
    Type: 'AWS::IAM::Role'
    Properties:
      RoleName: !Ref RoleName
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              AWS: !Sub arn:aws:iam::${AdminAccountId}:root
            Action:
              - 'sts:AssumeRole'
      Path: /service-role/
      ManagedPolicyArns:
        - !If [addTimestream, arn:aws:iam::aws:policy/AmazonTimestreamReadOnlyAccess, !
Ref AWS::NoValue]
        - !If [addXray, arn:aws:iam::aws:policy/AWSXrayReadOnlyAccess, !Ref
 AWS::NoValue]
        - !If [addSitewise, arn:aws:iam::aws:policy/AWSIoTSiteWiseReadOnlyAccess, !Ref
 AWS::NoValue]
        - !If [addRedshift, arn:aws:iam::aws:policy/service-role/
AmazonGrafanaRedshiftAccess, !Ref AWS::NoValue]
        - !If [addAthena, arn:aws:iam::aws:policy/service-role/
AmazonGrafanaAthenaAccess, !Ref AWS::NoValue]
```

Here is the content of **AmazonGrafanaOrgAdminPolicy-*random-id***.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "organizations:ListAccountsForParent",
            "organizations:ListOrganizationalUnitsForParent"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:PrincipalOrgID": "o-organizationId"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "sts:AssumeRole"
        ],
        "Resource": "arn:aws:iam::*:role/service-role/
AmazonGrafanaOrgMemberRole-random-Id"
    }]
}
```

Customer-managed permissions

If you choose to use customer-managed permissions, you specify an existing IAM role in your account when you create an Amazon Managed Grafana workspace. The role must have a trust policy which trusts `grafana.amazonaws.com`.

The following is an example of such a policy:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "grafana.amazonaws.com"
      },
```

```
      "Action": "sts:AssumeRole"
    }
  ]
}
```

For that role to access AWS data sources or notification channels in that account, it must have the permissions in the policies listed earlier in this section. For example, to use the CloudWatch data source, it must have the permissions in the CloudWatch policy listed in Service-managed permissions for a single account.

The `List` and `Describe` permissions in the policies for Amazon OpenSearch Service and Amazon Managed Service for Prometheus shown in Service-managed permissions for a single account are only needed for the data source discovery and provisioning to work correctly. They aren't needed if you just want to set up these data sources manually.

Cross-account access

When a workspace is created in account 111111111111, a role in account 1111111111111 must be supplied. For this example, call this role *WorkspaceRole*. To access data in account 999999999999, you must create a role in account 999999999999. Call that *DataSourceRole*. You must then establish a trust relationship between *WorkspaceRole* and *DataSourceRole*. For more information about establishing trust between two roles, see IAM Tutorial: Delegate access across AWS accounts using IAM roles.

DataSourceRole needs to contain the policy statements listed earlier in this section for each data source that you want to use. After the trust relationship is established, you can specify the ARN of *DataSourceRole* (arn:aws:iam::999999999999:role:DataSourceRole) in the **Assume Role ARN** field on the data source configuration page of any AWS data source in your workspace. The data source then accesses account 999999999999 with the permissions that are defined in *DataSourceRole*.

IAM permissions

Access to Amazon Managed Grafana actions and data requires credentials. Those credentials must have permissions to perform the actions and to access the AWS resources, such as retrieving Amazon Managed Grafana data about your cloud resources. The following sections provide details about how you can use AWS Identity and Access Management and Amazon Managed Grafana to help secure your resources, by controlling who can access them. For more information, see Policies and permissions in IAM.

Amazon Managed Grafana permissions

The following table displays possible Amazon Managed Grafana actions and their required permissions:

Action	Required permission
Create an Amazon Managed Grafana workspace. A workspace is a logically isolated Grafana server used to create and visualize metrics, logs, and traces.	`grafana:CreateWorkspace`
Delete an Amazon Managed Grafana workspace.	`grafana:DeleteWorkspace`
Retrieve detailed information about an Amazon Managed Grafana workspace.	`grafana:DescribeWorkspace`
Retrieve the authentication configuration associated with a workspace.	`grafana:DescribeWorkspaceAuthentication`
Retrieve a list of permissions associated with workspace users and groups.	`grafana:ListPermissions`
Retrieve a list of the Amazon Managed Grafana workspaces that exist in the account.	`grafana:ListWorkspaces`
Update the permissions associated with workspace users and groups.	`grafana:UpdatePermissions`
Update Amazon Managed Grafana workspaces.	`grafana:UpdateWorkspace`
Update the authentication configuration associated with a workspace.	`grafana:UpdateWorkspaceAuthentication`
Associate a Grafana enterprise license with a workspace.	`grafana:AssociateLicense`

Compliance Validation for Amazon Managed Grafana

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. For more information about your compliance responsibility when using AWS services, see AWS Security Documentation.

Resilience in Amazon Managed Grafana

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, Amazon Managed Grafana offers several features to help support your data resiliency and backup needs.

Infrastructure Security in Amazon Managed Grafana

As a managed service, Amazon Managed Grafana is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access Amazon Managed Grafana through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.

- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Logging Amazon Managed Grafana API calls using AWS CloudTrail

Amazon Managed Grafana is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service. CloudTrail captures all API calls for Amazon Managed Grafana as events. The calls captured include calls from the Amazon Managed Grafana console and code calls to the Amazon Managed Grafana API operations.

Amazon Managed Grafana also captures some calls that use Grafana APIs. The calls captured are those that change data, such as calls that create, update, or delete resources. For more information about Grafana APIs that are supported in Amazon Managed Grafana, see Using Grafana HTTP APIs.

Using the information collected by CloudTrail, you can determine the request that was made to Amazon Managed Grafana, the IP address from which the request was made, when it was made, and additional details.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root user or user credentials.
- Whether the request was made on behalf of an IAM Identity Center user.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

CloudTrail is active in your AWS account when you create the account and you automatically have access to the CloudTrail **Event history**. The CloudTrail **Event history** provides a viewable, searchable, downloadable, and immutable record of the past 90 days of recorded management events in an AWS Region. For more information, see Working with CloudTrail Event history in the *AWS CloudTrail User Guide*. There are no CloudTrail charges for viewing the **Event history**.

For an ongoing record of events in your AWS account past 90 days, create a trail or a CloudTrail Lake event data store.

CloudTrail trails

A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. All trails created using the AWS Management Console are multi-Region. You can create a single-Region or a multi-Region trail by using the AWS CLI. Creating a multi-Region trail is recommended because you capture activity in all AWS Regions in your account. If you create a single-Region trail, you can view only the events logged in the trail's AWS Region. For more information about trails, see Creating a trail for your AWS account and Creating a trail for an organization in the *AWS CloudTrail User Guide*.

You can deliver one copy of your ongoing management events to your Amazon S3 bucket at no charge from CloudTrail by creating a trail, however, there are Amazon S3 storage charges. For more information about CloudTrail pricing, see AWS CloudTrail Pricing. For information about Amazon S3 pricing, see Amazon S3 Pricing.

CloudTrail Lake event data stores

CloudTrail Lake lets you run SQL-based queries on your events. CloudTrail Lake converts existing events in row-based JSON format to Apache ORC format. ORC is a columnar storage format that is optimized for fast retrieval of data. Events are aggregated into *event data stores*, which are immutable collections of events based on criteria that you select by applying advanced event selectors. The selectors that you apply to an event data store control which events persist and are available for you to query. For more information about CloudTrail Lake, see Working with AWS CloudTrail Lake in the *AWS CloudTrail User Guide*.

CloudTrail Lake event data stores and queries incur costs. When you create an event data store, you choose the pricing option you want to use for the event data store. The pricing option determines the cost for ingesting and storing events, and the default and maximum retention period for the event data store. For more information about CloudTrail pricing, see AWS CloudTrail Pricing.

Amazon Managed Grafana management events in CloudTrail

Management events provide information about management operations that are performed on resources in your AWS account. These are also known as control plane operations. By default, CloudTrail logs management events.

Amazon Managed Grafana logs all Amazon Managed Grafana control plane operations as management events. For a list of the Amazon Managed Grafana control plane operations that Amazon Managed Grafana logs to CloudTrail, see the Amazon Managed Grafana API Reference.

Amazon Managed Grafana event examples

An event represents a single request from any source and includes information about the requested API operation, the date and time of the operation, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so events don't appear in any specific order.

The following example shows a CloudTrail log entry for a CreateWorkspace action.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "ANPAJ2UCCR6DPCEXAMPLE:sdbt-example",
        "arn": "arn:aws:sts::123456789012:assumed-role/Admin/sdbt-example",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "ANPAJ2UCCR6DPCEXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-11-26T20:59:21Z"
            }
        }
    },
    "eventTime": "2020-11-26T21:10:48Z",
    "eventSource": "grafana.amazonaws.com",
    "eventName": "CreateWorkspace",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "205.251.233.179",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:82.0) Gecko/20100101
 Firefox/82.0",
```

```
    "requestParameters": {
        "permissionType": "Service Managed",
        "workspaceNotificationDestinations": [
            "SNS"
        ],
        "workspaceDescription": "",
        "clientToken": "12345678-abcd-1234-5678-111122223333",
        "workspaceDataSources": [
            "SITEWISE",
            "XRAY",
            "CLOUDWATCH",
            "ELASTICSEARCH",
            "PROMETHEUS",
            "TIMESTREAM"
        ],
        "accountAccessType": "CURRENT_ACCOUNT",
        "workspaceName": "CloudTrailTest",
        "workspaceRoleArn": "arn:aws:iam::123456789012:role/service-role/
AmazonGrafanaServiceRole-2705976ol"
    },
    "responseElements": {
        "Access-Control-Expose-Headers": "x-amzn-RequestId,x-amzn-ErrorType,x-amzn-
ErrorMessage,Date",
        "workspace": {
            "accountAccessType": "CURRENT_ACCOUNT",
            "created": 1606425045.22,
            "dataSources": [
                "SITEWISE",
                "XRAY",
                "CLOUDWATCH",
                "ELASTICSEARCH",
                "PROMETHEUS",
                "TIMESTREAM"
            ],
            "description": "",
            "grafanaVersion": "7.3.1",
            "id": "g-a187c473d3",
            "modified": 1606425045.22,
            "name": "CloudTrailTest",
            "notificationDestinations": [
                "SNS"
            ],
            "permissionType": "Service Managed",
            "status": "CREATING",
```

```
            "workspaceRoleArn": "arn:aws:iam::123456789012:role/service-role/
AmazonGrafanaServiceRole-2705976ol"
        }
    },
    "requestID": "12345678-5533-4e10-b486-e9c7b219f2fd",
    "eventID": "12345678-2710-4359-ad90-b902dbfb606b",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123456789012"
}
```

The following example shows a CloudTrail log entry for an UpdateWorkspaceAuthentication action.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId":
 "AROAU2UJBF3NRO35YZ3GV:CODETEST_Series_GrafanaApiTestHydraCanary12-
o6aeXqaXS_1090259374",
        "arn": "arn:aws:sts::332073610971:assumed-role/
HydraInvocationRole-4912743f1277b7c3c67cb29518f8bc413ae/
CODETEST_Series_GrafanaApiTestHydraCanary12-o6aeXqaXS_1090259374",
        "accountId": "111122223333",
        "accessKeyId": "AIDACKCEVSQ6C2EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAU2UJBF3NRO35YZ3GV",
                "arn": "arn:aws:iam::111122223333:role/
HydraInvocationRole-4912743f1277b7c3c67cb29518f8bc413ae",
                "accountId": "332073610971",
                "userName": "TestInvocationRole-4912743f1277b7c3c67cb29518f8bc413ae"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2021-08-04T20:50:24Z",
                "mfaAuthenticated": "false"
            }
        }
    },
```

```
    "eventTime": "2021-08-04T21:29:25Z",
    "eventSource": "gamma-grafana.amazonaws.com",
    "eventName": "UpdateWorkspaceAuthentication",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "34.215.72.249",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.1030
 Linux/4.14.231-180.360.amzn2.x86_64 OpenJDK_64-Bit_Server_VM/11.0.11+9-LTS
 java/11.0.11 vendor/Amazon.com_Inc. cfg/retry-mode/legacy exec-env/AWS_Lambda_java11",
    "requestParameters": {
        "authenticationProviders": [
            "AWS_SSO",
            "SAML"
        ],
        "samlConfiguration": {
            "idpMetadata": {
                "url": "https://portal.sso.us-east-1.amazonaws.com/saml/metadata/
NjMwMDg2NDc4OTA3X2lucy1jY2E2ZGU3ZDlmYjdiM2Vh"
            }
        },
        "workspaceId": "g-84ea23c1b4"
    },
    "responseElements": {
        "authentication": {
            "awsSso": {
                "ssoClientId": "gAROcWGs9-LoqCMIQ56XyEXAMPLE"
            },
            "providers": [
                "AWS_SSO",
                "SAML"
            ],
            "saml": {
                "configuration": {
                    "idpMetadata": {
                        "url": "https://portal.sso.us-east-1.amazonaws.com/saml/
metadata/NjMwMDg2NDc4OTA3X2lucy1jY2E2ZGU3ZDlmYjdiM2Vh"
                    },
                    "loginValidityDuration": 60
                },
                "status": "CONFIGURED"
            }
        }
    },
    "requestID": "96adb1de-7fa5-487e-b6c6-6b0d4495cb71",
    "eventID": "406bc825-bc52-475c-9c91-4c0d8a07c1fa",
```

```
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

For information about CloudTrail record contents, see CloudTrail record contents in the *AWS CloudTrail User Guide*.

Grafana API event examples

Amazon Managed Grafana also logs some Grafana API calls in CloudTrail. The calls captured are those that change data, such as calls that create, update, or delete resources. For more information about Grafana APIs that are supported in Amazon Managed Grafana, see Using Grafana HTTP APIs.

User signs in to Amazon Managed Grafana workspace using AWS IAM Identity Center

```
{
    "Records": [
        {
            "eventVersion": "1.08",
            "userIdentity": {
                "type": "SAMLUser",
                "userName": "johndoe"
            },
            "eventTime": "2021-07-09T02:31:59Z",
            "eventSource": "grafana.amazonaws.com",
            "eventName": "login-auth.sso",
            "awsRegion": "us-west-2",
            "sourceIPAddress": "192.0.2.0,198.51.100.0",
            "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36",
            "requestParameters": null,
            "responseElements": null,
            "eventID": "176bf326-0302-4190-8dbf-dfdf481d8198",
            "readOnly": false,
            "eventType": "AwsServiceEvent",
            "managementEvent": true,
            "eventCategory": "Management",
            "recipientAccountId": "111122223333",
            "serviceEventDetails": {
                "timestamp": "2021-07-09T02:31:59.045984031Z",
```

```
                "user": {
                    "userId": 1,
                    "orgId": 1,
                    "name": "johndoe",
                    "isAnonymous": false
                },
                "action": "login-auth.sso",
                "requestUri": "",
                "request": {
                    "query": {
                        "code": [

 "eyJraWQiOiJrZXktMTU2Njk2ODEyMSIsImFsZyI6IkhTMzg0In0.eyJwbGFpbnRleHQiOiJZUzEwYWtaWHpBZUowTDlQc
                        ],
                        "state": [

 "QUFBQURtdGxlUzB4TlRZNE9UVTFOekkyM2RUWUFUaHZHYXcyOU9ULUVaWHhNUXAwX184N25RVGVWMmd0enFpVE1iWlRPV
                        ]
                    }
                },
                "result": {
                    "statusType": "failure"
                },
                "ipAddress": "192.0.2.0,198.51.100.0",
                "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36",
                "grafanaVersion": "7.5.7",
                "additionalData": {
                    "GiraffeCustomerAccount": "111122223333",
                    "GiraffeWorkspaceId": "g-123EXAMPLE",
                    "extUserInfo": "{\"OAuthToken\":null,\"AuthModule\":\"auth.sso\",
\"AuthId\":\"92670be4c1-e524608b-82f2-452d-a707-161c1e5f4706\",\"UserId\":0,\"Email\":
\"\",\"Login\":\"johndoe\",\"Name\":\"johndoe\",\"Groups\":null,\"OrgRoles\":{\"1\":
\"Admin\"},\"IsGrafanaAdmin\":false,\"IsDisabled\":false}"
                }
            }
        }
    ]
}
```


Grafana API POST /api/auth/keys


```
{
```

```
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown",
        "userName": "api_key"
    },
    "eventTime": "2021-07-09T02:16:32Z",
    "eventSource": "grafana.amazonaws.com",
    "eventName": "create",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0,198.51.100.1",
    "userAgent": "python-requests/2.24.0",
    "errorCode": "200",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "157bbf19-6ba4-4704-bc3b-d3e334b3a2b8",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "timestamp": "2021-07-09T02:16:32.419795511Z",
        "user": {
            "orgId": 1,
            "orgRole": "Admin",
            "name": "api_key",
            "apiKeyId": "23",
            "isAnonymous": false
        },
        "action": "create",
        "resources": [
            {
                "ID": 0,
                "type": "api-key"
            }
        ],
        "requestUri": "",
        "request": {
            "body": "{\"name\":\"keyname\",\"role\":\"Admin\",\"secondsToLive\":60}"
        },
        "result": {
            "statusType": "success",
            "statusCode": "200"
        },
```

```
        "ipAddress": "192.0.2.0,198.51.100.1",
        "userAgent": "python-requests/2.24.0",
        "grafanaVersion": "7.5.7",
        "additionalData": {
            "GiraffeCustomerAccount": "111122223333",
            "GiraffeWorkspaceId": "g-123EXAMPLE"
        }
    }
}
```

Grafana API DELETE /api/auth/keys/:id

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown",
        "userName": "api_key"
    },
    "eventTime": "2021-07-09T02:16:33Z",
    "eventSource": "grafana.amazonaws.com",
    "eventName": "delete",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0,198.51.100.2",
    "userAgent": "python-requests/2.24.0",
    "errorCode": "200",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "df1aafb3-28c6-4836-a64b-4d34538edc51",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "timestamp": "2021-07-09T02:16:33.045041594Z",
        "user": {
            "orgId": 1,
            "orgRole": "Admin",
            "name": "api_key",
            "apiKeyId": "23",
            "isAnonymous": false
        },
        "action": "delete",
```

```
        "resources": [
            {
                "ID": 0,
                "type": "api-key"
            }
        ],
        "requestUri": "",
        "request": {
            "params": {
                ":id": "24"
            }
        },
        "result": {
            "statusType": "success",
            "statusCode": "200"
        },
        "ipAddress": "192.0.2.0,198.51.100.2",
        "userAgent": "python-requests/2.24.0",
        "grafanaVersion": "7.5.7",
        "additionalData": {
            "GiraffeCustomerAccount": "111122223333",
            "GiraffeWorkspaceId": "g-123EXAMPLE"
        }
    }
}
```

Grafana API POST /api/alerts/:id/pause

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown",
        "userName": "api_key"
    },
    "eventTime": "2021-07-09T02:16:40Z",
    "eventSource": "grafana.amazonaws.com",
    "eventName": "pause",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0,198.51.100.3",
    "userAgent": "python-requests/2.24.0",
    "errorCode": "200",
    "requestParameters": null,
    "responseElements": null,
```

```
    "eventID": "d533a7ba-f193-45ac-a88c-75ed0594509b",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "timestamp": "2021-07-09T02:16:40.261226856Z",
        "user": {
            "orgId": 1,
            "orgRole": "Admin",
            "name": "api_key",
            "apiKeyId": "23",
            "isAnonymous": false
        },
        "action": "pause",
        "resources": [
            {
                "ID": 0,
                "type": "alert"
            }
        ],
        "requestUri": "",
        "request": {
            "params": {
                ":alertId": "1"
            },
            "body": "{\"paused\":true}"
        },
        "result": {
            "statusType": "success",
            "statusCode": "200"
        },
        "ipAddress": "192.0.2.0,198.51.100.3",
        "userAgent": "python-requests/2.24.0",
        "grafanaVersion": "7.5.7",
        "additionalData": {
            "GiraffeCustomerAccount": "111122223333",
            "GiraffeWorkspaceId": "g-123EXAMPLE"
        }
    }
}
```

Grafana POST /api/alerts/test


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown",
        "userName": "api_key"
    },
    "eventTime": "2021-07-09T02:16:39Z",
    "eventSource": "grafana.amazonaws.com",
    "eventName": "test",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0,10.0.42.208",
    "userAgent": "python-requests/2.24.0",
    "errorCode": "400",
    "errorMessage": "The dashboard needs to be saved at least once before you can test
 an alert rule",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "7094644d-8230-4774-a092-8a128eb6dec9",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "timestamp": "2021-07-09T02:16:39.622607860Z",
        "user": {
            "orgId": 1,
            "orgRole": "Admin",
            "name": "api_key",
            "apiKeyId": "23",
            "isAnonymous": false
        },
        "action": "test",
        "resources": [
            {
                "ID": 0,
                "type": "panel"
            }
        ],
        "requestUri": "",
        "request": {},
        "result": {
```

```
            "statusType": "failure",
            "statusCode": "400",
            "failureMessage": "The dashboard needs to be saved at least once before you
 test an alert rule"
        },
        "ipAddress": "192.0.2.0, 10.0.42.208",
        "userAgent": "python-requests/2.24.0",
        "grafanaVersion": "7.5.7",
        "additionalData": {
            "GiraffeCustomerAccount": "111122223333",
            "GiraffeWorkspaceId": "g-123EXAMPLE"
        }
    }
}
```

Grafana API POST /api/alert-notifications

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown",
        "userName": "api_key"
    },
    "eventTime": "2021-07-09T02:16:40Z",
    "eventSource": "grafana.amazonaws.com",
    "eventName": "create",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0,198.51.100.0",
    "userAgent": "python-requests/2.24.0",
    "errorCode": "200",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "1ce099b3-c427-4338-9f42-d38d1ef64efe",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "timestamp": "2021-07-09T02:16:40.888295790Z",
        "user": {
            "orgId": 1,
            "orgRole": "Admin",
```

```
            "name": "api_key",
            "apiKeyId": "23",
            "isAnonymous": false
        },
        "action": "create",
        "resources": [
            {
                "ID": 0,
                "type": "alert-notification"
            }
        ],
        "requestUri": "",
        "request": {
            "body": "{\"name\":\"alert notification name\",\"type\":\"Slack\"}"
        },
        "result": {
            "statusType": "success",
            "statusCode": "200"
        },
        "ipAddress": "192.0.2.0,198.51.100.0",
        "userAgent": "python-requests/2.24.0",
        "grafanaVersion": "7.5.7",
        "additionalData": {
            "GiraffeCustomerAccount": "111122223333",
            "GiraffeWorkspaceId": "g-123EXAMPLE"
        }
    }
}
```

Grafana API PUT /api/alert-notifications/uid/:uid

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown",
        "userName": "api_key"
    },
    "eventTime": "2021-07-09T02:16:42Z",
    "eventSource": "grafana.amazonaws.com",
    "eventName": "update",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0,198.51.100.3",
    "userAgent": "python-requests/2.24.0",
```

```
    "errorCode": "200",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "cebfeb38-5007-495c-bd29-c8077797acac",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "timestamp": "2021-07-09T02:16:42.792652648Z",
        "user": {
            "orgId": 1,
            "orgRole": "Admin",
            "name": "api_key",
            "apiKeyId": "23",
            "isAnonymous": false
        },
        "action": "update",
        "resources": [
            {
                "ID": 0,
                "type": "alert-notification"
            }
        ],
        "requestUri": "",
        "request": {
            "params": {
                ":uid": "WvDWDSinz"
            },
            "body": "{\"name\":\"DIFFERENT alert notification name\",\"type\":\"AWS SNS
\"}"
        },
        "result": {
            "statusType": "success",
            "statusCode": "200"
        },
        "ipAddress": "192.0.2.0,198.51.100.3",
        "userAgent": "python-requests/2.24.0",
        "grafanaVersion": "7.5.7",
        "additionalData": {
            "GiraffeCustomerAccount": "111122223333",
            "GiraffeWorkspaceId": "g-123EXAMPLE"
        }
```

```
    }
}
```

Grafana API POST /api/annotations

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown",
        "userName": "api_key"
    },
    "eventTime": "2021-07-09T02:16:45Z",
    "eventSource": "grafana.amazonaws.com",
    "eventName": "create",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0,198.51.100.1",
    "userAgent": "python-requests/2.24.0",
    "errorCode": "200",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "13bf3bef-966c-4913-a760-ade365a4a08f",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "timestamp": "2021-07-09T02:16:45.394513179Z",
        "user": {
            "orgId": 1,
            "orgRole": "Admin",
            "name": "api_key",
            "apiKeyId": "23",
            "isAnonymous": false
        },
        "action": "create",
        "resources": [
            {
                "ID": 0,
                "type": "annotation"
            }
        ],
        "requestUri": "",
```

```
        "request": {
            "body": "{\"dashboardId\":36,\"panelId\":2,\"tags\":[\"tag1\",\"tag2\"],
\"what\":\"Event Name\"}"
        },
        "result": {
            "statusType": "success",
            "statusCode": "200"
        },
        "ipAddress": "192.0.2.0,198.51.100.1",
        "userAgent": "python-requests/2.24.0",
        "grafanaVersion": "7.5.7",
        "additionalData": {
            "GiraffeCustomerAccount": "111122223333",
            "GiraffeWorkspaceId": "g-123EXAMPLE"
        }
    }
}
```


Grafana API DELETE /api/dashboards/uid/:uid


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown",
        "userName": "api_key"
    },
    "eventTime": "2021-07-09T02:17:09Z",
    "eventSource": "grafana.amazonaws.com",
    "eventName": "delete",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0,198.51.100.7",
    "userAgent": "python-requests/2.24.0",
    "errorCode": "200",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "d6ad9134-5fbc-403c-a76d-4ed9a81065b6",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "timestamp": "2021-07-09T02:17:09.200112003Z",
```

```
        "user": {
            "orgId": 1,
            "orgRole": "Admin",
            "name": "api_key",
            "apiKeyId": "23",
            "isAnonymous": false
        },
        "action": "delete",
        "resources": [
            {
                "ID": 0,
                "type": "dashboard"
            }
        ],
        "requestUri": "",
        "request": {
            "params": {
                ":uid": "GLzWvIi7z"
            }
        },
        "result": {
            "statusType": "success",
            "statusCode": "200"
        },
        "ipAddress": "192.0.2.0,198.51.100.7",
        "userAgent": "python-requests/2.24.0",
        "grafanaVersion": "7.5.7",
        "additionalData": {
            "GiraffeCustomerAccount": "111122223333",
            "GiraffeWorkspaceId": "g-123EXAMPLE"
        }
    }
}
```

Grafana API PUT /api/datasources/:datasourceId

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown",
        "userName": "api_key"
    },
    "eventTime": "2021-07-09T02:16:36Z",
```

```
    "eventSource": "grafana.amazonaws.com",
    "eventName": "update",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0,10.0.108.94",
    "userAgent": "python-requests/2.24.0",
    "errorCode": "200",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "92877483-bdf6-44f5-803e-1ac8ad997113",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "timestamp": "2021-07-09T02:16:36.918660585Z",
        "user": {
            "orgId": 1,
            "orgRole": "Admin",
            "name": "api_key",
            "apiKeyId": "23",
            "isAnonymous": false
        },
        "action": "update",
        "resources": [
            {
                "ID": 0,
                "type": "datasource"
            }
        ],
        "requestUri": "",
        "request": {
            "params": {
                ":id": "108"
            },
            "body": "{\"access\":\"proxy\",\"basicAuth\":false,\"name\":
\"test_amp_datasource_NEW_name\",\"type\":\"Amazon Managed Prometheus\",\"url\":
\"http://amp.amazonaws.com\"}"
        },
        "result": {
            "statusType": "success",
            "statusCode": "200"
        },
        "ipAddress": "192.0.2.0,10.0.108.94",
```

```
        "userAgent": "python-requests/2.24.0",
        "grafanaVersion": "7.5.7",
        "additionalData": {
            "GiraffeCustomerAccount": "111122223333",
            "GiraffeWorkspaceId": "g-123EXAMPLE"
        }
    }
}
```

Grafana API DELETE /api/teams/:teamId/groups/:groupId

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown",
        "userName": "api_key"
    },
    "eventTime": "2021-07-09T02:17:07Z",
    "eventSource": "grafana.amazonaws.com",
    "eventName": "delete",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0,198.51.100.2",
    "userAgent": "python-requests/2.24.0",
    "errorCode": "200",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "b41d3967-daab-44d1-994a-a437556add82",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "timestamp": "2021-07-09T02:17:07.296142539Z",
        "user": {
            "orgId": 1,
            "orgRole": "Admin",
            "name": "api_key",
            "apiKeyId": "23",
            "isAnonymous": false
        },
        "action": "delete",
        "resources": [
```

```
            {
                "ID": 0,
                "type": "team"
            }
        ],
        "requestUri": "",
        "request": {
            "params": {
                ":groupId": "cn=editors,ou=groups,dc=grafana,dc=org",
                ":teamId": "35"
            }
        },
        "result": {
            "statusType": "success",
            "statusCode": "200"
        },
        "ipAddress": "192.0.2.0,198.51.100.2",
        "userAgent": "python-requests/2.24.0",
        "grafanaVersion": "7.5.7",
        "additionalData": {
            "GiraffeCustomerAccount": "111122223333",
            "GiraffeWorkspaceId": "g-123EXAMPLE"
        }
    }
}
```

Grafana API PUT /api/folders/:uid

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown",
        "userName": "api_key"
    },
    "eventTime": "2021-07-09T02:16:56Z",
    "eventSource": "grafana.amazonaws.com",
    "eventName": "update",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0,198.51.100.1",
    "userAgent": "python-requests/2.24.0",
    "errorCode": "412",
    "errorMessage": "the folder has been changed by someone else",
    "requestParameters": null,
```

```
    "responseElements": null,
    "eventID": "414c98c8-aa53-45e4-940d-bea55716eaf6",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "timestamp": "2021-07-09T02:16:56.382646826Z",
        "user": {
            "orgId": 1,
            "orgRole": "Admin",
            "name": "api_key",
            "apiKeyId": "23",
            "isAnonymous": false
        },
        "action": "update",
        "resources": [
            {
                "ID": 0,
                "type": "folder"
            }
        ],
        "requestUri": "",
        "request": {
            "params": {
                ":uid": "lnsZvSi7z"
            },
            "body": "{\"title\":\"NEW Folder Name\"}"
        },
        "result": {
            "statusType": "failure",
            "statusCode": "412",
            "failureMessage": "the folder has been changed by someone else"
        },
        "ipAddress": "192.0.2.0,198.51.100.1",
        "userAgent": "python-requests/2.24.0",
        "grafanaVersion": "7.5.7",
        "additionalData": {
            "GiraffeCustomerAccount": "111122223333",
            "GiraffeWorkspaceId": "g-123EXAMPLE"
        }
    }
```

```
}
```

Grafana API POST /api/teams

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown",
        "userName": "api_key"
    },
    "eventTime": "2021-07-09T02:17:02Z",
    "eventSource": "grafana.amazonaws.com",
    "eventName": "create",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0,10.0.40.206",
    "userAgent": "python-requests/2.24.0",
    "errorCode": "200",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "8d40bd79-76a8-490c-b7bb-74205253b707",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
        "timestamp": "2021-07-09T02:17:02.845022379Z",
        "user": {
            "orgId": 1,
            "orgRole": "Admin",
            "name": "api_key",
            "apiKeyId": "23",
            "isAnonymous": false
        },
        "action": "create",
        "resources": [
            {
                "ID": 0,
                "type": "team"
            }
        ],
        "requestUri": "",
        "request": {
```

```
            "body": "{\"name\":\"TeamName\"}"
        },
        "result": {
            "statusType": "success",
            "statusCode": "200"
        },
        "ipAddress": "192.0.2.0,10.0.40.206",
        "userAgent": "python-requests/2.24.0",
        "grafanaVersion": "7.5.7",
        "additionalData": {
            "GiraffeCustomerAccount": "111122223333",
            "GiraffeWorkspaceId": "g-123EXAMPLE"
        }
    }
}
```

Security best practices

The topics in this section explain the best practices to follow to best maintain security in your Amazon Managed Grafana deployment.

Use short-lived API keys

To use Grafana APIs in an Amazon Managed Grafana workspace, you must first create an API key to use for authorization. When you create the key, you specify the **Time to live** for the key, which defines how long the key is valid, up to a maximum of 30 days. We strongly recommend that you set the key's time to live for a shorter time, such as a few hours or less. This creates much less risk than having API keys that are valid for a long time.

We also recommend that you treat API keys as passwords, in terms of securing them. For example, do not store them in plain text.

Migrating from self-managed Grafana

This section is relevant for you if you are migrating an existing self-managed Grafana or Grafana Enterprise deployment to Amazon Managed Grafana. This applies to both on-premises Grafana and to a Grafana deployment on AWS, in your own account.

If you are running Grafana on-premises or in your own AWS account, you have likely defined users and teams and potentially organization roles to manage access. In Amazon Managed Grafana,

users and groups are managed outside of Amazon Managed Grafana, using IAM Identity Center or directly from your identity provider (IdP) via SAML 2.0 integration. With Amazon Managed Grafana, you can assign certain permissions as necessary for carrying out a task— for example viewing dashboards. For more information about user management in Amazon Managed Grafana, see Manage workspaces, users, and policies in Amazon Managed Grafana.

Additionally, when you run on-premises Grafana you're using long-lived keys or secret credentials to access data sources. We strongly recommend that when you migrate to Amazon Managed Grafana, you replace these IAM users with IAM roles. For an example, see Manually add CloudWatch as a data source.

Interface VPC endpoints

We provide AWS PrivateLink support between Amazon VPC and Amazon Managed Grafana. You can control access to the Amazon Managed Grafana service from the virtual private cloud (VPC) endpoints by attaching an IAM resource policy for Amazon VPC endpoints.

Amazon Managed Grafana supports two different kinds of VPC endpoints. You can connect to the Amazon Managed Grafana service, providing access to the Amazon Managed Grafana APIs to manage workspaces. Or you can create a VPC endpoint to a specific workspace.

Using Amazon Managed Grafana with interface VPC endpoints

There are two ways to use interface VPC endpoints with Amazon Managed Grafana. You can use a VPC endpoint to allow AWS resources such as Amazon EC2 instances to access the Amazon Managed Grafana API to manage resources, or you can use a VPC endpoint as part of limiting network access to your Amazon Managed Grafana workspaces.

- If you are using Amazon VPC to host your AWS resources, you can establish a private connection between your VPC and the Amazon Managed Grafana API using the `com.amazonaws.region.grafana` service name endpoint.
- If you are trying to use network access control to add security to your Amazon Managed Grafana workspace, you can establish a private connection between your VPC and the Grafana workspaces endpoint, using the `com.amazonaws.region.grafana-workspace` service name endpoint.

Amazon VPC is an AWS service that you can use to launch AWS resources in a virtual network that you define. With a VPC, you have control over your network settings, such as the IP address range,

subnets, route tables, and network gateways. To connect your VPC to your Amazon Managed Grafana API, you define an *interface VPC endpoint* . The endpoint provides reliable, scalable connectivity to Amazon Managed Grafana without requiring an internet gateway, a network address translation (NAT) instance, or a VPN connection. For more information, see What is Amazon VPC? in the *Amazon VPC User Guide.*

Interface VPC endpoints are powered by AWS PrivateLink, an AWS technology that enables private communication between AWS services using an elastic network interface with private IP addresses. For more information, see New – AWS PrivateLink for AWS Services.

For information about how to get started with Amazon VPC, see Get started in the *Amazon VPC User Guide*.

Creating a VPC endpoint to make an AWS PrivateLink connection to Amazon Managed Grafana

Create an interface VPC endpoint to Amazon Managed Grafana with one of the following service name endpoints:

- To connect to the Amazon Managed Grafana API for managing workspaces, choose:

  `com.amazonaws.region.grafana`.
- To connect to a Amazon Managed Grafana workspace (for example, to use the Grafana API), choose:

  `com.amazonaws.region.grafana-workspace`

For the details about creating an interface VPC endpoint, see Create an interface endpoint in the *Amazon VPC User Guide.*

For calling Grafana APIs, you must also enable private DNS for your VPC endpoint, by following the instructions in the Amazon VPC User Guide. This enables local resolution of URLs in the form `*.grafana-workspace.region.amazonaws.com`

Using network access control to limit access to your Grafana workspace

If you want to limit what IP addresses or VPC endpoints can be used to access a specific Grafana workspace, you can configure network access control to that workspace.

For VPC endpoints that you give access to your workspace, you can further limit their access by configuring security groups for the endpoints. To learn more, see Associate security groups and Security group rules in the *Amazon VPC documentation*.

Controlling access to your Amazon Managed Grafana API VPC endpoint with an endpoint policy

For VPC endpoints that are connected the Amazon Managed Grafana API (using `com.amazonaws.region.grafana`), you can add a VPC endpoint policy to limit access to the service.

> **Note**
>
> VPC endpoints connected to workspaces (using `com.amazonaws.region.grafana-workspace`) do not support VPC endpoint policies.

A VPC endpoint policy is an IAM resource policy that you attach to an endpoint when you create or modify the endpoint. If you don't attach a policy when you create an endpoint, Amazon VPC attaches a default policy for you that allows full access to the service. An endpoint policy doesn't override or replace IAM identity-based policies or service-specific policies. It's a separate policy for controlling access from the endpoint to the specified service.

Endpoint policies must be written in JSON format.

For more information, see Control access to service with VPC endpoints in the *Amazon VPC User Guide*.

The following is an example of an endpoint policy for Amazon Managed Grafana. This policy allows users connecting to Amazon Managed Grafana through the VPC to send data to the Amazon Managed Grafana service. It also prevents them from performing other Amazon Managed Grafana actions.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "AWSGrafanaPermissions",
            "Effect": "Allow",
            "Action": [
                "grafana:DescribeWorkspace",
                "grafana:UpdatePermissions",
                "grafana:ListPermissions",
                "grafana:ListWorkspaces"
            ],
            "Resource": "arn:aws:grafana:*:*:/workspaces*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
            }
        }
    ]
}
```

To edit the VPC endpoint policy for Grafana

1. Open the Amazon VPC console at VPC console.
2. In the navigation pane, choose **Endpoints**.
3. If you have not already created endpoints, choose **Create Endpoint**.
4. Select the `com.amazonaws.region.grafana` endpoint, and then choose the **Policy** tab.
5. Choose **Edit Policy**, and then make your changes.

Amazon Managed Grafana service quotas

Amazon Managed Grafana has the following quotas. You can request a quota increase for the number of workspaces.

Name	Default	Adjustable	Description
Number of workspaces	Each supported Region: 5	Yes	The maximum number of workspaces that you can have in this account in the current region.
Rate of AssociateLicense requests	Each supported Region: 1 per second	No	The maximum number of AssociateLicense requests that you can make, per second, in this account in the current region.
Rate of CreateWorkspace requests	Each supported Region: 1 per second	No	The maximum number of CreateWorkspace requests that you can make, per second, in this account in the current region.
Rate of DeleteWorkspace requests	Each supported Region: 1 per second	No	The maximum number of DeleteWorkspace requests that you can make, per second, in this account in the current region.
Rate of DescribeWorkspace requests	Each supported Region: 5 per second	No	The maximum number of DescribeWorkspace requests that you can

Name	Default	Adjustable	Description
			make, per second, in this account in the current region.
Rate of DescribeWorkspaceAuthentication requests	Each supported Region: 1 per second	No	The maximum number of DescribeWorkspaceAuthentication requests that you can make, per second, in this account in the current region.
Rate of DisassociateLicense requests	Each supported Region: 1 per second	No	The maximum number of DisassociateLicense requests that you can make, per second, in this account in the current region.
Rate of ListPermissions requests	Each supported Region: 10 per second	No	The maximum number of ListPermissions requests that you can make, per second, in this account in the current region.
Rate of ListWorkspaces requests	Each supported Region: 5 per second	No	The maximum number of ListWorkspaces requests that you can make, per second, in this account in the current region.

Name	Default	Adjustable	Description
Rate of UpdatePermissions requests	Each supported Region: 10 per second	No	The maximum number of UpdatePermissions requests that you can make, per second, in this account in the current region.
Rate of UpdateWorkspace requests	Each supported Region: 10 per second	No	The maximum number of UpdateWorkspace requests that you can make, per second, in this account in the current region.
Rate of UpdateWorkspaceAuthentication requests	Each supported Region: 1 per second	No	The maximum number of UpdateWorkspaceAuthentication requests that you can make, per second, in this account in the current region.

Additionally, Amazon Managed Grafana has the following quotas within each workspace

Resource	Adjustable	Default Quota
Alerts The number of rules per workspace in classic alerting, or the number of rule instances per workspace in Grafana alerting.	No	100 per workspace.

Resource	Adjustable	Default Quota
Dashboards	No	2,000 per workspace.
Data sources	No	2,000 per workspace.
Users	No	10,000 provisioned, 500 concurrent per workspace.
API keys	No	100 per workspace.
Service accounts	No	100 per workspace.
Service account tokens Active tokens and expired tokens count toward this quota. Delete tokens to remove them from the quota.	No	100 per workspace.
Network access control: Prefix lists	No	5 per workspace.
Network access control: IP address ranges	No	100 per prefix list.
Network access control: VPC endpoints	No	5 per workspace.
Annotations Applies to workspaces running Grafana version 12 and later.	No	3,000,000 per workspace.

Resource	Adjustable	Default Quota
Alert rule evaluation results Applies to workspaces running Grafana version 12 and later.	No	500 per alert rule evaluation.

Document history for User Guide

The following table describes the important changes to the documentation since the last release of Amazon Managed Grafana. For notification about updates to this documentation, you can subscribe to an RSS feed.

Change	Description	Date
Amazon Managed Grafana adds support for troublesh ooting degraded workspaces	Amazon Managed Grafana adds support for troublesh ooting degraded workspaces.	May 19, 2026
Amazon Managed Grafana adds support for in-place upgrade to Grafana version 12.4	Amazon Managed Grafana adds support for in-place upgrade to Grafana version 12.4.	May 15, 2026
Amazon Managed Grafana adds support for Grafana version 12	Amazon Managed Grafana adds support for Grafana version 12. Grafana versions 8, 9, and 10 are also still available for use.	April 2, 2026
Amazon Managed Grafana updates the solution for monitoring Amazon EKS clusters	Amazon Managed Grafana updates the solution to automatically monitor Amazon EKS clusters with a dashboard, using AWS CDK, simplifying deployment.	June 10, 2024
Amazon Managed Grafana adds support for Grafana version 10	Amazon Managed Grafana adds support for Grafana version 10. Grafana versions 8 and 9 are also still available for use.	May 15, 2024

Amazon Managed Grafana adds a solution for monitoring Amazon EKS clusters	Amazon Managed Grafana adds support for a solution to automatically monitor Amazon EKS clusters with a dashboard created via CloudFormation.	April 30, 2024
Amazon Managed Grafana replaces an obsolete managed policy	Amazon Managed Grafana adds a new managed policy, **AWSGrafanaWorkspacePermissionManagementV2** to replace the obsolete **AWSGrafanaWorkspacePermissionManagement** managed policy. This new managed policy improves security for your workspace by providing a more restrictive set of permissions.	January 5, 2024
Amazon Managed Grafana adds support for using community plugins	Amazon Managed Grafana adds support for using community plugins in workspaces that are compatible with Grafana version 9.	November 15, 2023
Amazon Managed Grafana adds support for updating the version of an existing workspace	Amazon Managed Grafana adds support for updating existing workspaces to a more recent version. For example, you can update a workspace that is compatible with Grafana version 8 to be compatible with version 9.	July 19, 2023

Amazon Managed Grafana adds support for trace analytics in OpenSearch data sources	Amazon Managed Grafana adds support for viewing a list of traces in OpenSearch data sources, when using workspaces that support version 9 or later.	June 22, 2023
Amazon Managed Grafana adds support for Grafana version 9	Amazon Managed Grafana adds support for Grafana version 9. Grafana version 8 is also still available for use.	April 28, 2023
Amazon Managed Grafana adds a new managed policy	Amazon Managed Grafana added a new managed policy, **AmazonGrafanaCloudWatchAccess**, that allows Amazon Managed Grafana to access metrics in CloudWatch.	March 24, 2023
Amazon Managed Grafana adds new permissions	Amazon Managed Grafana added new permissions to **AWSGrafanaWorkspacePermissionManagement** so that IAM Identity Center users and groups that are managed in Microsoft Active Directory or Active Directory Connector can be associated with or disassociated from Amazon Managed Grafana workspaces.	March 22, 2023

Clarifying alert instances	With Grafana alerts, a single alert rule can create multiple instances of alerts, which affects how quickly quotas are reached. The documentation is updated to provide additional details.	March 20, 2023
Amazon Managed Grafana adds new data sources	Amazon Managed Grafana adds new data source plugins for Databricks and Google BigQuery.	March 14, 2023
Amazon Managed Grafana adds network access control to workspaces	Amazon Managed Grafana adds network access control to allow only specified IP addresses or VPC endpoints to access a workspace.	February 16, 2023
Amazon Managed Grafana adds new permissions	Amazon Managed Grafana added new permissions to **AWSGrafanaWorkspacePermissionManagement** so that IAM Identity Center users and groups can be associated with Grafana workspaces.	December 20, 2022
New SLR policy for Grafana service	Added service-linked role,`AmazonManagedGrafana` which receives permissions from the `AmazonGrafanaServiceLinkedRolePolicy` AWS managed policy.	November 23, 2022
Connection with Amazon VPC data sources	Added connections to data sources in Amazon VPC.	November 23, 2022

Amazon Managed Grafana added workspace configuration	Amazon Managed Grafana added support for making configuration changes per Grafana workspace instance	November 23, 2022
Amazon Managed Grafana added support for Grafana alerting	Amazon Managed Grafana added support for using the updated Grafana alerting feature, including integrating alerts from Amazon Managed Service for Prometheus and Prometheus instances.	November 23, 2022
Amazon Managed Grafana adds 3 new visualizations	Amazon Managed Grafana adds the Plotly, Sankey, and Scatter panel visualizations.	November 17, 2022
AWS SSO rebranding to IAM Identity Center	AWS SSO is rebranded to IAM Identity Center.	July 26, 2022
Amazon Managed Grafana feature enhancements	Amazon Managed Grafana adds support for Grafana version 8.4, Pixie, GitHub and Moogsoft datasources, Recorded queries and the WindRose panel visualization.	May 13, 2022
Amazon Managed Grafana feature enhancements	Amazon Managed Grafana adds support for tagging.	March 31, 2022
Amazon Managed Grafana feature enhancements	Amazon Managed Grafana adds support for VPC Privatelink.	January 7, 2022

Amazon Managed Grafana feature enhancements	Amazon Managed Grafana adds support for Amazon Redshift data source, Amazon Athena data source, Zabbix, Cloudflare. Amazon Managed Grafana supports Geomap panel visualization and Grafana version 8.2.	November 24, 2021
Amazon Managed Grafana preview feature enhancements	The preview of Amazon Managed Grafana supports Grafana version 7.5 and supports upgrade to Grafana Enterprise via AWS Marketplace integration. The Amazon Elasticsearch Service data source has also been upgraded to support Open Distro for Elasticsearch.	April 16, 2021
Amazon Managed Grafana preview released.	The preview of Amazon Managed Grafana is released.	December 15, 2020