Add variables

- Enter general options for any variable
  - Variable best practices
- Add a query variable
  - Query type
- Add a custom variable
- Add a textbox variable {#add-a-text-box-variable}
- Add a constant variable
- Add a data source variable
- Add an interval variable
  - Interval variable examples
- Add a switch variable
  - Switch variable examples
- Variable selection options {#configure-variable-selection-options}
  - Multi-value variables
    - Multi-value variables with a Graphite data source
    - Multi-value variables with a Prometheus or InfluxDB data source
    - Multi-value variables with an Elastic data source
    - Variable indexing
    - Troubleshoot multi-value variables
  - Include All option
  - Custom all value

You can create your own variables within any dashboard, and these variables are specific to the dashboard in which you create them. For information on variables that you can use across dashboards, refer to the Cross-dashboard variables documentation.

The following table lists the types of variables you can create:

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.
Switch	Display a switch that allows you to toggle between two configurable values for enabled and disabled states. Add a switch variable.

{{< admonition type="note" >}} In Grafana 13.1, **Ad hoc filters** were renamed **Filter and Group by** and extended by adding grouping for Prometheus and Loki data sources. However, in the dashboard schema, they're still referred to as `"kind": "AdhocVariable"`.

For information on the **Filter and Group by** feature, refer to the documentation. {{< /admonition >}}

Enter general options for any variable

To create a variable, follow these steps:

{{< docs/list >}}

{{< shared id="add-variable" >}}

1. Navigate to the dashboard you want to update.
2. Click **Edit**.
3. Click the plus sign **Add new element** icon in the sidebar and select **Variable**.
   You can also click the **Add variable** plus sign icon at the top of the dashboard.
4. Choose a variable type from the list.
   You can update the variable type by clicking **Change type** in the sidebar.

5. Enter a **Name** for the variable.
6. (Optional) In the **Label** field, enter the display name for the variable drop-down list. If you leave this empty, then Grafana uses the variable name.
7. (Optional) In the **Description** field, enter a description of the variable. The description appears as an info icon tooltip next to the variable name on the dashboard. Descriptions support links. You can use Markdown-style links like `[link text](https://example.com)`, or paste bare URLs like `https://example.com`. Only `http` and `https` URLs render as clickable links—other protocols display as plain text.
8. Choose a **Display** option:
   - **Above dashboard** - The variable drop-down list displays above the dashboard with the variable **Name** or **Label** value. This is the default.
   - **Above dashboard, label hidden** - The variable drop-down list displays above the dashboard, but without showing the name of the variable.
   - **Controls menu** - The variable displays in the dashboard controls menu instead of above the dashboard. The dashboard controls menu appears as a button in the dashboard toolbar.
   - **Hidden** - No variable drop-down list displays on the dashboard.

{{< /shared >}}

1. Click one of the following links to complete the steps for adding your selected variable type:
   - Query
   - Custom
   - Textbox
   - Constant
   - Data source
   - Interval
   - Switch

{{< /docs/list >}}

Variable best practices

- Variable drop-down lists display in the order in which they're listed in the **Variables** section of the sidebar, so put the variables that you change often at the top, so they're shown first.
- By default, variables don't have a default value. This means that the topmost value in the drop-down list is always preselected. If you want to pre-populate a variable with an empty value, you can use the following workaround in the variable settings:
  1. Select the **Include All Option** checkbox.
  2. In the **Custom all value** field, enter a value like `.+`.

Add 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 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.

{{< admonition type="note" >}} Query expressions are different for each data source. For more information, refer to the documentation for your data source. {{< /admonition >}}

1. Enter general options.
2. Click **Open variable editor**.
3. Set the following options to configure your query in the **Query** tab:

Option	Description
Target data source	Select a data source in the drop-down list. You can also click **Open advanced data source picker** to see more options, including adding a data source (Admins only). For more information about data sources, refer to Add a data source.
Query type	If applicable, select an option in the drop-down list and fill in the query fields accordingly. For more information, refer to the Query type section following these steps.
Regex	(Optional) Type a regular expression in the field to filter or capture specific parts of the names returned by your data source query. To see examples, refer to Filter variables with a regular expression.
Apply regex to	Select **Variable value** or **Display text** to choose where the regular expression pattern is applied. The default is **Variable value**.
Sort	Select the sort order for values to display in the drop-down list. The default option, **Disabled**, means that the order of options returned by your data source query is used.

4. In the **Static options** tab, set the following options:
   - Select an option in the **Static options sort** drop-down list to control where in the list of variable values the static options appear.
   - Under **value**, click **+ Add new option** or **Paste from clipboard** to add static options.
5. Click **Preview** in the top of the dialog box to test the variable.
   Grafana displays a list of the current variable values.
6. In the **Preview of values** section, Review them to ensure they match what you expect.
7. Click **Apply** to close the query variable editor dialog box.
8. In the **Refresh** drop-down list, select an option to control when the variable should update:
   - **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 the dashboard can be initialized.
   - **On time range change** - Queries the data source every time the dashboard loads and when the dashboard time range changes. Use this option if your variable options query contains a time range filter or is dependent on the dashboard time range.
9. (Optional) Set Selection options if needed:
   - **Multi-value** - Enables the selection of select multiple values at the same time.
   - **Include All value** - Enables an option to include all variables.
   - **Allow custom values** - Enables users to add custom values to the list. Only applies to CSV custom values.
10. Click **Save** in the top-right corner.
11. Enter an optional description of your changes and click **Save**.
12. Click **Exit edit**.

Query type

The query section of the editor varies according to your data source. Some data sources have custom query editors.

Each data source defines how it extracts variable values. The typical implementation uses every string value returned from the data source response as a variable value. Make sure to double-check the documentation for the data source.

Some data sources let you provide custom "display names" for the values. For instance, the PostgreSQL, MySQL, and Microsoft SQL Server plugins handle this by looking for fields named `__text` and `__value` in the result. Other data sources may look for `text` and `value` or use a different approach. Always remember to double-check the documentation for the data source.

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.

Add a custom variable

Use a *custom* variable for a value that doesn't 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 don't change, you might use them in chained variables rather than other query variables. That would reduce the number of queries Grafana must send when you update chained variables.

1. Enter general options.
2. Click **Open variable editor**.
3. Configure one of the following options:
   - **CSV** - Enter a flat list of values for the 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`.
   - **JSON** - Provide a JSON array of objects where each object can have any number of properties that you can reference. For more information refer, to Configure multi-property variables.
4. Click **Preview** to test the variable.
5. In the **Preview of values** section, Grafana displays a list of the current variable values. If you've entered a JSON array, the preview is a table that includes all the value properties. Review them to ensure they match what you expect.
6. Click **Apply** to close the query editor dialog box.
7. (Optional) Set Selection options if needed:
   - **Multi-value** - Enables the selection of multiple values at the same time.
   - **Include All value** - Enables an option to include all variables.
   - **Allow custom values** - Enables users to add custom values to the list. Only applies to CSV custom values.
8. Click **Save** in the top-right corner.
9. Enter an optional description of your changes and click **Save**.
10. Click **Exit edit**.

Add a textbox variable {#add-a-text-box-variable}

Textbox 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.

For more information about cardinality, refer to What are cardinality spikes and why do they matter?

1. Enter general options.
2. (Optional) Under **Textbox options**, enter the default value for the variable in the **Value** field.
   If you don't enter anything in this field, then Grafana displays an empty text box for users to type text into.
3. Click **Save** in the top-right corner.
4. Enter an optional description of your changes and click **Save**.
5. Click **Exit edit**.

Add 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, Grafana converts constant variables to import options.

Constant variables are *not* flexible. Each constant variable only holds one value, and you can't update it 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`.

1. Enter general options.
2. Under **Constant options**, enter the variable value in the **Value** field.
   You can enter letters, numbers, and symbols. You can even use wildcards if you use raw format.
3. Click **Save** in the top-right corner.
4. Enter an optional description of your changes and click **Save**.
5. Click **Exit edit**.

Add a data source variable

Data source variables enable you to quickly change the data source for an entire dashboard. They're useful if you have multiple instances of a data source, perhaps in different environments.

1. Enter general options.

2. Configure the following **Data source options**:
   - **Type** - Select the target data source for the variable.
   - **Name filter** - (Optional) Enter a regular expression filter for which data source instances to choose from in the variable value drop-down list. Leave this field empty to display all instances.
3. (Optional) Set Selection options if needed:
   - **Multi-value** - Enables the selection of multiple values at the same time.
   - **Include All value** - Enables an option to include all variables.
   - **Allow custom values** - Enables users to add custom values to the list. Only applies to CSV custom values.
4. Click **Save** in the top-right corner.
5. Enter an optional description of your changes and click **Save**.
6. Click **Exit edit**.

Add an interval variable

Use an *interval* variable to represents 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 Elasticsearch), or as a summarize function parameter (for Graphite).

1. Enter general options.
2. Configure the following **Interval options**:

Option	Description
Values	Enter the time range intervals that you want to appear in the variable drop-down list. Grafana supports the following time units: `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`.
Auto option	(Optional) Select on the checkbox if you want to add the `auto` option to the list. This option allows you to specify how many times to divide the current time range to calculate the current `auto` time span.

Option	Description
Step count	Select the number of times to divide the current time range 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. This option only displays hen you select the **Auto option** checkbox.
Min interval	The minimum threshold below which the step count intervals doesn't 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. Only displayed when you select the **Auto option** checkbox.

3. Click **Save** in the top-right corner.
4. Enter an optional description of your changes and click **Save**.
5. Click **Exit edit**.

Interval variable examples

The following example shows a template variable `myinterval` in a Graphite function:

```
summarize($myinterval, sum, false)
```

The following example shows a more complex Graphite example, from the Graphite Template Nested Requests panel:

```
groupByNode(summarize(movingAverage(apps.$app.$server.counters.requests.count, 5),
'$interval', 'sum', false), 2, 'sum')
```

Add a switch variable

Switch variables display a switch with two configurable values representing enabled and disabled states. This variable type is useful when you need to:

- Toggle between different query conditions
- Enable or disable specific filters
- Switch between different visualization modes
- Control boolean parameters in your data sources

1. Enter general options.
2. Configure the following **Switch options**:

Option	Description
Value pair type	Select one of the following predefined options: • **True / False** - Uses boolean values `true` and `false`. • **1 / 0** - Uses numeric values `1` and `0`. • **Yes / No** - Uses string values `yes` and `no`. • **Custom** - Allows you to define custom values for both enabled and disabled states.
Enabled value	If you selected **Custom**, configure the custom values. Enter the value that represents the enabled state (for example, "on").
Disabled value	If you selected **Custom**, enter the value that represents the disabled state (for example, "off").

3. Click **Save** in the top-right corner.
4. Enter an optional description of your changes and click **Save**.
5. Click **Exit edit**.

Switch variable examples

The following example shows a switch variable `$debug_mode` used in a Prometheus query to conditionally include debug labels:

```
up{job="my-service"} and ($debug_mode == "true" or on() vector(0))
```

The following example shows a switch variable `$show_errors` used to filter log entries:

```
{job="application"} |= ($show_errors == "1" ? "ERROR" : "")
```

You can also use switch variables in panel titles and other dashboard elements:

```
{{#if debug_mode}}Debug Mode: {{/if}}Application Metrics
```

Variable selection options {#configure-variable-selection-

options}

Selection Options are a feature you can use to manage variable option selections for query and custom variables. All selection options are optional, and they're off by default.

Multi-value variables

Interpolating a variable with multiple values selected is tricky as it's not straight forward how to format the multiple values into a string that's valid in the given context where you use the variable. Grafana tries to solve this by allowing each data source plugin to inform the templating interpolation engine what format to use for multiple values.

{{< admonition type="note" >}} The **Custom all value** option on the variable must be blank for Grafana to format all values into a single string. If it's left blank, then Grafana concatenates (adds together) all the values in the query. Something like `value1,value2,value3`. If you use a custom `all` value, then instead the value is something like `*` or `all`. {{< /admonition >}}

Multi-value variables with a Graphite data source

Graphite uses glob expressions. In this case, Grafana interpolates a variable with multiple values as `{host1,host2,host3}` if the current variable value is *host1*, *host2*, and *host3*.

Multi-value variables with a Prometheus or InfluxDB data source

InfluxDB and Prometheus use regular expressions, so Grafana interpolates the same variable as `(host1|host2|host3)`. Grafana also escapes every value for regular expressions. If not, a value with a regular expression control character would break the regular expression.

Multi-value variables with an Elastic data source

Elasticsearch uses Lucene query syntax, so Grafana formats the same variable 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.

Variable indexing

If you have a multi-value variable that's formatted as an array, you can use array positions to reference the values rather than the actual values. You can use this functionality in dashboard panels to filter data, and maintain the array.

To reference variable values this way, use the following syntax:

```
${query0.0}
```

The preceding syntax references the first, or `0`, position in the array.

In the following example, there's an array of three values, `1t`, `2t`, and `3t`, and rather than referencing those values, the panel query references the second value in the array using the syntax `${query0.1}`:

{{< figure src="/media/docs/grafana/dashboards/screenshot-indexed-variables-v12.1.png" max-width="750px" alt="Panel query using variable indexing to reference a value" >}}

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 regular expression syntax requires you to use the variable in a regular expression operator context.

If you don't want Grafana to do this automatic regular expression 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 drop-down list. If a user selects this option, then all variable options are selected.

If you enable this option but also set a variable's **Display** option to **Hidden**, the variable defaults to selecting only the first value—not **All**—unless you explicitly select **All** while the variable is visible and then save the dashboard. To set **All** as the default for a hidden variable, complete the following steps:

1. Temporarily set **Display** to **Above dashboard**.
2. On the dashboard, select the **All** option from the variable drop-down list.
3. Save the dashboard to persist the selection.
4. Return to **Dashboard options** > **Variables**, select the variable, and set **Display** back to **Hidden**.
5. Save the dashboard again.

Custom all value

This option is only visible if you select the **Include All option**.

Enter regular expressions, 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 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 regular expression.

In order to have custom regular expression, globs, or Lucene syntax in the **Custom all value** option, it's never escaped so you have to think about what's a valid value for your data source.

□□□ #2
□ □□□ □□□ 2026-10-31 16:32:14 UTC
□ □□□ □□□ 2026-10-31 16:32:18 UTC