+++ title = "Chained variables" keywords = ["grafana", "templating", "variable", "nested", "chained", "linked"] aliases = ["/docs/grafana/latest/variables/chained-variables.md"] weight = 800 +++

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 page 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.

Grafana Play dashboard examples

The following Grafana Play dashboards contain fairly simple chained variables, only two layers deep. To view the variables and their settings, click **Dashboard settings** (gear icon) and then click **Variables**. Both examples are expanded in the following section.

- Graphite Templated Nested
- InfluxDB Templated

Examples explained

Variables are useful to reuse dashboards, dynamically change what is shown in dashboards. Chained variables are especially useful to filter what you see.

Create parent/child relationship in variable, sort of a tree structure where you can select different levels of filters.

The following sections explain the linked examples in the dashboards above in depth and builds on them. While the examples are data source-specific, the concepts can be applied broadly.

Graphite example

In this example, you have several applications. Each application has a different subset of servers. It is based on the Graphite Templated Nested.

Now, you could make separate variables for each metric source, but then you have to know which server goes with which app. A better solution is to use one variable to filter another. In this example, when the user changes the value of the `app` variable, it changes the dropdown options returned by the `server` variable. Both variables use the **Multi-value** option and **Include all option**, allowing users to select some or all options presented at any time.

app variable

The query for this variable basically says, "Give me all the applications that exist."

```
apps.*
```

The values returned are `backend`, `country`, `fakesite`, and `All`.

server variable

The query for this variable basically says, "Give me all servers for the currently chosen application."

```
apps.$app.*
```

If the user selects `backend`, then the query changes to:

```
apps.backend.*
```

The query returns all servers associated with `backend`, including `backend_01`, `backend_02`, and so on.

If the user selects `fakesite`, then the query changes to:

```
apps.fakesite.*
```

The query returns all servers associated with `fakesite`, including `web_server_01`, `web_server_02`, and so on.

More variables

> ***Note:*** *This example is theoretical. The Graphite server used in the example does not contain CPU metrics.*

The dashboard stops at two levels, but you could keep going. For example, if you wanted to get CPU metrics for selected servers, you could copy the `server` variable and extend the query so that it reads:

```
apps.$app.$server.cpu.*
```

This query basically says, "Show me the CPU metrics for the selected server."

Depending on what variable options the user selects, you could get queries like:

```
apps.backend.backend_01.cpu.*
apps.{backend.backend_02,backend_03}.cpu.*
apps.fakesite.web_server_01.cpu.*
```

InfluxDB example

In this example, you have several data centers. Each data center has a different subset of hosts. It is based on the InfluxDB Templated.

In this example, when the user changes the value of the `datacenter` variable, it changes the dropdown options returned by the `host` variable. The `host` variable uses the **Multi-value** option and **Include all option**, allowing users to select some or all options presented at any time. The `datacenter` does not use either option, so you can only select one data center at a time.

datacenter variable

The query for this variable basically says, "Give me all the data centers that exist."

```
SHOW TAG VALUES  WITH KEY = "datacenter"
```

The values returned are `America`, `Africa`, `Asia`, and `Europe`.

host variable

The query for this variable basically says, "Give me all hosts for the currently chosen data center."

```
SHOW TAG VALUES WITH KEY = "hostname" WHERE "datacenter" =~ /^$datacenter$/
```

If the user selects `America`, then the query changes to:

```
SHOW TAG VALUES WITH KEY = "hostname" WHERE "datacenter" =~ /^America/
```

The query returns all servers associated with `America`, including `server1`, `server2`, and so on.

If the user selects `Europe`, then the query changes to:

```
SHOW TAG VALUES WITH KEY = "hostname" WHERE "datacenter" =~ /^Europe/
```

The query returns all servers associated with `Europe`, including `server3`, `server4`, and so on.

More variables

> ***Note:*** *This example is theoretical. The InfluxDB server used in the example does not contain CPU metrics.*

The dashboard stops at two levels, but you could keep going. For example, if you wanted to get CPU metrics for selected hosts, you could copy the `host` variable and extend the query so that it reads:

```
SHOW TAG VALUES WITH KEY = "cpu" WHERE "datacenter" =~ /^$datacenter$/ AND "host" =~
/^$host$/
```

This query basically says, "Show me the CPU metrics for the selected host."

Depending on what variable options the user selects, you could get queries like:

```
SHOW TAG VALUES WITH KEY = "cpu" WHERE "datacenter" =~ /^America/ AND "host" =~
/^server2/
SHOW TAG VALUES WITH KEY = "cpu" WHERE "datacenter" =~ /^Africa/ AND "host" =~
/^server/7/
SHOW TAG VALUES WITH KEY = "cpu" WHERE "datacenter" =~ /^Europe/ AND "host" =~
/^server3+server4/
```

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.