+++ title = “Loki” description = “Guide for using Loki in Grafana” keywords = [“grafana”, “loki”, “logging”, “guide”] aliases = [“/docs/grafana/latest/features/datasources/loki”] weight = 800 +++

Using Loki in Grafana

Grafana ships with built-in support for Loki, an open source log aggregation system by Grafana Labs. This topic explains options, variables, querying, and other options specific to this data source.

Add it as a data source and you are ready to build dashboards or query your log data in [Explore]({{< relref “../explore” >}}). Refer to [Add a data source]({{< relref “add-a-data-source.md” >}}) for instructions on how to add a data source to Grafana. Only users with the organization admin role can add data sources.

Hosted Loki

You can run Loki on your own hardware or use Grafana Cloud. The free forever plan includes Grafana, 50 GB of Loki logs, 10K Prometheus series, and more. Create a free account to get started.

Loki settings

To access Loki settings, click the **Configuration** (gear) icon, then click **Data Sources**, and then click the Loki data source.

Name	Description
`Name`	The data source name. This is how you refer to the data source in panels, queries, and Explore.
`Default`	Default data source that is pre-selected for new panels.
`URL`	URL of the Loki instance, e.g., `http://localhost:3100`.
`Allowed cookies`	Grafana Proxy deletes forwarded cookies by default. Specify cookies by name that should be forwarded to the data source.
`Maximum lines`	Upper limit for the number of log lines returned by Loki (default is 1000). Lower this limit if your browser is sluggish when displaying logs in Explore.

> **Note:** To troubleshoot configuration and other issues, check the log file located at /var/log/grafana/grafana.log on Unix systems or in /data/log on other platforms and manual installations.

Derived fields

The Derived Fields configuration allows you to:

- Add fields parsed from the log message.

- Add a link that uses the value of the field.

For example, 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]({{< relref "../explore/logs-integration/#labels-and-detected-fields" >}}).

Each derived field consists of:

- **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.
- **URL Label -** (Optional) Set a custom display label for the link. The link label defaults to the full external URL or name of the linked internal data source and is overridden by this setting.
- **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. Click **Show example log message** to show the text area where you can enter a log message. {{< figure src="/static/img/docs/v75/loki_derived_fields_settings.png" class="docs-image–no-shadow" max-width="800px" caption="Screenshot of the derived fields debugging" >}}

The new field with the link shown in log details: {{< figure src="/static/img/docs/explore/detected-fields-link-7-4.png" max-width="800px" caption="Detected fields link in Explore" >}}

Loki query editor

You can use the Loki query editor to create log and metric queries.

Name	Description
`Query expression`	Loki query expression, refer to the LogQL documentation for more information.
`Query type`	Choose the type of query to run. The instant type queries against a single point in time. We are using "To" time from the time range. The range type queries over the selected range of time.
`Line limit`	Upper limit for number of log lines returned by query. The default is the Maximum lines limit set in Loki settings.

Name	Description
`Legend`	Available only in Dashboard. Controls the name of the time series, using name or pattern. For example `{{hostname}}` is replaced with the label value for the label `hostname`.

Log browser

With Loki log browser you can easily navigate through your list of labels and values and construct the query of your choice. Log browser has multi-step selection:

1. Choose the labels you would like to consider for your search.
2. Pick the values for selected labels. Log browser supports facetting and therefore it shows you only possible label combinations.
3. Choose the type of query - logs query or rate metrics query. Additionally, you can also validate selector.

{{< figure src="/static/img/docs/v75/loki_log_browser.png" class="docs-image–no-shadow" max-width="800px" caption="Screenshot of the log browser for Loki" >}}

Querying with Loki

There are two types of LogQL queries:

- Log queries - Return the contents of log lines.
- Metric queries - Extend log queries and calculate sample values based on the content of logs from a log query.

Log queries

Querying and displaying log data from Loki is available via [Explore]({{< relref "../explore" >}}), and with the [logs panel]({{< relref "../visualizations/logs-panel.md" >}}) in dashboards. Select the Loki data source, and then enter a LogQL query to display your logs.

A log query consists of two parts: log stream selector, and a log pipeline. For performance reasons begin by choosing a log stream by selecting a log label.

Log context

When using a search expression as detailed above, you can retrieve the context surrounding your filtered results. By clicking 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.

Live tailing

Loki supports Live tailing which displays logs in real-time. This feature is supported in [Explore]({{< relref "../explore/#loki-specific-features" >}}).

Note that Live Tailing relies on two Websocket connections: one between the browser and the Grafana server, and another between the Grafana server and the Loki server. If you run any reverse proxies, please configure them accordingly. The following example for Apache2 can be used for proxying between the browser and the Grafana server:

```
ProxyPassMatch "^/(api/datasources/proxy/\d+/loki/api/v1/tail)" "ws://127.0.0.1:3000/$1"
```

The following example shows basic NGINX proxy configuration. It assumes that the Grafana server is available at `http://localhost:3000/`, Loki server is running locally without proxy, and your external site uses HTTPS. If you also host Loki behind NGINX proxy, then you might want to repeat the following configuration for Loki as well.

In the `http` section of NGINX configuration, add the following map definition:

```
map $http_upgrade $connection_upgrade {
  default upgrade;
  '' close;
}
```

In your `server` section, add the following configuration:

```
location ~ /(api/datasources/proxy/\d+/loki/api/v1/tail) {
    proxy_pass          http://localhost:3000$request_uri;
    proxy_set_header    Host              $host;
    proxy_set_header    X-Real-IP         $remote_addr;
    proxy_set_header    X-Forwarded-for   $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Proto "https";
    proxy_set_header    Connection        $connection_upgrade;
    proxy_set_header    Upgrade           $http_upgrade;
}

location / {
    proxy_pass          http://localhost:3000/;
    proxy_set_header    Host              $host;
    proxy_set_header    X-Real-IP         $remote_addr;
    proxy_set_header    X-Forwarded-for   $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Proto "https";
}
```

Note: This feature is only available in Grafana v6.3+.

Metric queries

LogQL supports wrapping a log query with functions that allow for creating metrics out of the logs. See LogQL documentation on how to create and use metrics queries.

Templating

Instead of hard-coding things like server, application and sensor name in your metric queries, you can use variables in their place. Variables are shown as drop-down select boxes at the top of the dashboard. These drop-down boxes make it easy to change the data being displayed in your dashboard.

Check out the Templating({{< relref "../variables/_index.md" >}}) documentation for an introduction to the templating feature and the different types of template variables.

Query variable

Variable of the type *Query* allows you to query Loki for a list labels or label values. The Loki 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`.
`label_values(log stream selector, label)`	Returns a list of label values for the `label` in the specified `log stream selector`.

Ad hoc filters variable

Loki supports the special ad hoc filters variable type. It allows you to specify any number of label/value filters on the fly. These filters are automatically applied to all your Loki queries.

Using interval and range variables

You can use some global built-in variables in query variables; `$__interval`, `$__interval_ms`, `$__range`, `$__range_s` and `$__range_ms`. For more information, refer to [Global built-in variables]({{< relref "../variables/variable-types/global-variables.md" >}}).

Annotations

You can use any non-metric Loki query as a source for annotations({{< relref "../dashboards/annotations" >}}). Log content will be used as annotation text

and your log stream labels as tags, so there is no need for additional mapping.

Configure the data source with provisioning

You can set up the data source via config files with Grafana's provisioning system. You can read more about how it works and all the settings you can set for data sources on the [provisioning docs page]({{< relref "../administration/provisioning/#datasources" >}})

Here is an example:

```yaml
apiVersion: 1

datasources:
  - name: Loki
    type: loki
    access: proxy
    url: http://localhost:3100
    jsonData:
      maxLines: 1000
```

Here's another with basic auth and derived field. Keep in mind that `$` character needs to be escaped in YAML values as it is used to interpolate environment variables:

```yaml
apiVersion: 1

datasources:
  - name: Loki
    type: loki
    access: proxy
    url: http://localhost:3100
    basicAuth: true
    basicAuthUser: my_user
    basicAuthPassword: test_password
    jsonData:
      maxLines: 1000
      derivedFields:
        # Field with internal link pointing to data source in Grafana.
        # Right now, Grafana supports only Jaeger and Zipkin data sources as link targets.
        # datasourceUid value can be anything, but it should be unique across all defined d
        - datasourceUid: my_jaeger_uid
          matcherRegex: "traceID=(\\w+)"
          name: TraceID
          # url will be interpreted as query for the datasource
          url: '$${__value.raw}'

        # Field with external link.
```

```
      - matcherRegex: "traceID=(\\w+)"
        name: TraceID
        url: 'http://localhost:16686/trace/$${__value.raw}'
```

Here's an example of a Jaeger data source corresponding to the above example. Note that the Jaeger `uid` value does match the Loki `datasourceUid` value.

```
datasources:
    - name: Jaeger
      type: jaeger
      url: http://jaeger-tracing-query:16686/
      access: proxy
      # UID should match the datasourceUid in dervidedFields.
      uid: my_jaeger_uid
```