MySQL annotations

- Before you begin
- Create an annotation query
- Query columns
- Example queries
  - Basic annotation with epoch time
  - Annotation with a single tag
  - Range annotation with start and end time
  - Annotation with native SQL datetime
  - Deployment annotations
  - Maintenance window annotations
- Macros
- Best practices

Annotations overlay event data on your dashboard graphs, helping you correlate events with metrics. You can use MySQL as a data source for annotations to display events such as deployments, alerts, or other significant occurrences on your visualizations.

For general information about annotations, refer to Annotate visualizations.

Before you begin

Before creating MySQL annotations, ensure you have:

- A MySQL data source configured in Grafana.
- Tables containing event data with timestamp fields.
- Read access to the tables containing your events.

Create an annotation query

To add a MySQL annotation to your dashboard:

1. Navigate to the dashboard you want to update and click **Edit**.
2. Click the **Add new element** icon (blue plus sign).
3. Click **Annotation query**.
4. Enter a name for the annotation query.
5. If you don't want to use the annotation query right away, clear the **Enabled** checkbox.
6. Select a color for the annotation event markers.
7. Select an option in the **Show annotation controls in** drop-down list to control where on the dashboard the annotation is displayed.
8. Select an option in the **Show in** drop-down list to control the panels in which the annotation is displayed.
9. Click **Open query editor** to open the **Annotation Query** dialog box.
10. Select your **MySQL** data source from the **Data source** drop-down list.
11. Configure the annotation query and field mappings.
12. (Optional) Click **Test annotation query** to ensure that the query is working properly.
13. Click **Close** when you've completed the query setup.
14. Click **Save**.
15. (Optional) Enter a description of the changes you've made.
16. Click **Save**.
17. Click **Exit edit**.

Query columns

Your annotation query must return a `time` column and can optionally include `timeend`, `text`, and `tags` columns.

Column	Required	Description
`time`	Yes	The timestamp for the annotation. Can be a SQL datetime or UNIX epoch value.
`timeend`	No	The end timestamp for range annotations. Creates a shaded region instead of a vertical line.
`text`	No	The annotation description displayed when you hover over the annotation.
`tags`	No	Tags for the annotation as a comma-separated string. Helps categorize and filter annotations.

Example queries

The following examples show common annotation query patterns.

Basic annotation with epoch time

Display events using UNIX epoch timestamps:

```
SELECT
  epoch_time as time,
  description as text,
  CONCAT(tag1, ',', tag2) as tags
FROM events
WHERE $__unixEpochFilter(epoch_time)
```

Annotation with a single tag

Display events with a single tag value:

```
SELECT
  epoch_time as time,
  message as text,
  category as tags
FROM event_log
WHERE $__unixEpochFilter(epoch_time)
```

Range annotation with start and end time

Display events with duration as shaded regions:

```
SELECT
  start_time as time,
  end_time as timeend,
  description as text,
  CONCAT(type, ',', severity) as tags
FROM incidents
WHERE $__unixEpochFilter(start_time)
```

Annotation with native SQL datetime

Display events using native MySQL datetime columns:

```
SELECT
  event_date as time,
  message as text,
  CONCAT(category, ',', priority) as tags
FROM system_events
WHERE $__timeFilter(event_date)
```

Deployment annotations

Display deployment events:

```
SELECT
  deployed_at as time,
  CONCAT('Deployed ', version, ' to ', environment) as text,
  environment as tags
FROM deployments
WHERE $__timeFilter(deployed_at)
```

Maintenance window annotations

Display maintenance windows as range annotations:

```
SELECT
  start_time as time,
  end_time as timeend,
  CONCAT('Maintenance: ', description) as text,
  'maintenance' as tags
FROM maintenance_windows
WHERE $__timeFilter(start_time)
```

Macros

Use these macros in your annotation queries to filter by the dashboard time range:

Macro	Description
`$__timeFilter(column)`	Filters by time range using a native SQL datetime column.
`$__unixEpochFilter(column)`	Filters by time range using a column with UNIX epoch timestamps.

Best practices

Follow these best practices when creating MySQL annotations:

- **Use time filters:** Always include `$__timeFilter()` or `$__unixEpochFilter()` to limit results to the dashboard time range.
- **Keep queries efficient:** Add indexes on time columns and filter columns to improve query performance.
- **Use meaningful text:** Include descriptive information in the `text` column to make annotations useful.
- **Organize with tags:** Use consistent tag values to categorize annotations and enable filtering.
- **Test queries first:** Verify your query returns expected results in Explore before adding it as an annotation.

□□□ #2
□ □□□ □□□ 2026-10-31 15:59:46 UTC
□ □□□ □□□ 2026-10-31 15:59:50 UTC